NodeWorthy

THE ACTION REFERENCE

Every action and check that ships with NodeWorthy, what it does, why you would use it and what each of its settings means.

Version 0.1, 26 September 2026. 293 actions and checks, and 19 nodes.

NodeWorthy © 2026 StateCraft Games. All rights reserved.

How to use this book

Every action in NodeWorthy is one thing a state does. You add one to a state with the **Add Action** button, and it runs when the state is entered, top to bottom with the other actions in that state.

This book has a page for each action and each check. The page shows these things.

- **The name**, exactly as the **Add Action** list shows it, with the group it is in at the top of the page.
- **One or two lines** saying what it does, so you can tell at a glance whether it is the one you want.
- **A picture** of the action as it appears in the **Actions** tab the moment you add it, before you change anything. What you see on your screen should look like the picture.
- **When to use it**, which says why you would reach for this one, with a few things it is good for.
- **Its settings**, with every box on the card, what it does, and what it starts as. Some settings only appear once another setting is changed. Those are listed at the end of the table and say what brings them out. A setting measured in metres starts at a different value when the action is put in a 2D game, where it is measured in pixels, 32 to a metre. The **Starts as** column gives that value too, marked **in 2D**. The pictures are taken in a 3D game.
- **See also**, which names the actions people often want when they land on this one by mistake.

The last group, **Nodes To Add**, is different. These 19 are not things a state does but ready-made nodes that go into your scene, such as health bars, meters and globes, a map, an inventory and its grid, quest lists and a quest log, a dialogue box, a skill bar, stats and experience, hit and hurt boxes, an aiming mark, a character's Perception and more. They are at the end of the **Add Action** list, where **Add To Scene** puts one in the scene for you, and in Godot's own **Add Child Node** window. Their pages show the node in a running game. Picking one in the Scene dock shows its settings in the Inspector.

Three ways to find what you need

- **Contents** lists the groups, in the same order as the **Add Action** list, so the book and the panel agree.
- **I want to…** holds plain descriptions of what you are trying to do, in alphabetical order, each pointing at the action that does it. Start here if you do not know what the thing you want is called.
- **A to Z** at the back lists every action and check by name.

The main **NodeWorthy Manual** explains states, events, transitions and variables. This book assumes you have met those and only explains the actions themselves.

Contents

Actions are grouped exactly as they are in the **Add Action** list, and are in the same order there and here.

State Machine Events, states and other machines.

Cancel Delayed Events 17
Enable Machine 18
Forward Events 19
Get Event Data 20
Get Last Event 21
Get Previous State 22
Get State 23
Go To Previous State 24
Go To State 25
In State 26
Machine Variable 27
Next Frame Event 28
Random Event 29
Run Template 30
Send Event 31
Sub State Finished 32

Logic True and false, choices, and doing something more than once.

Bool Operator 33
Chance 34
Compare 35
For Each 36
Repeat 37
Sequence 38
Switch 39
Variable 40
Variable Changed 41

Math Numbers and their sums, formulas, limits and chance.

Clamp 42
Ease Along Curve 43
Ease Variable 44
Expression 45
Math 46
Random Number 47
Random Seed 48
Set Variable 49

Vector Vector2 and Vector3, for places and directions.

Get Vector Components 50
Random Vector 51
Set Vector Components 52
Vector 53

String Building text, formatting it and comparing it.

Format String 54
Format Time 55
String 56
String Compare 57

Array Many values in one variable.

Array 58
Dictionary 59
High Score Table 60

Data Records and tables of your game's things, such as weapons, enemy types and levels.

Data 61
Find Row 62
Get Data 63
Set Data 64

Time Waiting, timers and the speed of time.

Get Time 65
Set Time Scale 66
Timeout 67
Timer 68
Wait 69

Node Finding, making, freeing and changing nodes.

Add Node 70
Call Method 71
Call Static Method 72
Duplicate 73
Emit Signal 74
Enable Node 75
Event On Signal 76
Find Child 77
Get Children 78
Get Parent 79
Get Property 80
Group 81
Group Switch 82
Has Child Of Type 83
Instantiate 84
Is Child Of 85
Node 86
Property 87
Queue Free 88
Reparent 89
Set Process Mode 90
Set Property 91

Transform Moving and turning things, paths and distances.

Carry 92
Distance 93
Follow Path 94
In Range 95
Look At 96
Move To 97
Path Progress 98
Punch 99

Rotate 100
Rotate To 101
Scale To 102
To Global 103
To Local 104
Translate 105
Tween Property 106

Physics Forces, velocity, gravity, raycasts and collisions.

Apply Force 107
Drive 108
Event On Collision 109
Explosion 110
Freeze 111
Get Contact 112
Get Overlapping 113
Get Velocity 114
On Screen 115
Raycast 116
Raycast All 117
Set Body Properties 118
Set Collision 119
Set Gravity 120
Set Joint 121
Set Velocity 122
Shape Cast 123
Swing 124
Touching 125

Character A character that walks, jumps, dashes, swims and climbs.

Click To Move 126
Crouch 127
Dash 128
Get Last Collision 129
Get Water Depth 130
In Water 131
Jump 132
Ledge Grab 133
Move Character 134
Stamina 135
Step Up 136
Swim 137
Wall Cling 138

Input Keys, buttons, sticks, the mouse and touch.

Device Motion 139
Drag 140
Event On Input 141
Get Action Strength 142
Get Input Vector 143
Input 144
Input Combo 145
Joypad Connected 146
Last Device 147
Mouse Motion 148
Mouse Pick 149
Mouse Position 150
Pinch 151
Rebind Input 152
Screen Orientation 153
Set Mouse Mode 154
Touch Count 155
Touch Gesture 156
Vibrate 157
Virtual Joystick 158
Warp Mouse 159

Animation AnimationPlayer, AnimationTree and animated sprites.

Animation Event 160
Flip Sprite 161
Play Animation 162
Play Sprite Animation 163
Playing 164
Seek Animation 165
Set Animation Speed 166
Set Tree Parameter 167
Stop Animation 168
Stop Sprite Animation 169
Travel To 170

Audio Sounds, music, volume and speech.

Mute 171
Pause Sound 172
Play Music 173
Play Random Sound 174
Play Sound 175
Play Sound At Position 176
Set Bus Effect 177
Set Pitch 178
Set Sound Loop 179
Set Volume 180
Stop Music 181
Stop Sound 182
Text To Speech 183

Camera Cameras that follow, shake, zoom and look around.

Camera Follow 184
Camera Limits 185
Camera Shake 186
Camera Zoom 187
Fade Screen 188
Get Current Camera 189
Head Bob 190
Mouse Look 191
Orbit Camera 192
Recoil 193
Screen To World 194
Set Background Color 195
Switch Camera 196
Weapon Sway 197
World To Screen 198

Effects Particles, marks, screen effects and shaders.

Emit Particles 199
Flicker 200
Leave Marks 201
Screen Effect 202
Set Environment 203
Set Shader Parameter 204

UI Labels, textures, values and choices on the screen.

Get Focus 205
Get Text 206
Get Value 207
Grab Focus 208
Off Screen Indicator 209
Option Button 210
Pointer Over UI 211
Set Disabled 212
Set Text 213
Set Texture 214
Set Value 215
Set Visible 216
Show Choices 217
Toggle Button 218

Scene Changing scenes, pausing, quitting and the game window.

Build Dungeon 219
Change Scene 220
Get Scene 221
Pause Game 222
Procedural Placement 223
Quit Game 224
Screen Settings 225
Set Language 226
System 227

Save & Files Saving the game, and files and folders on the computer.

Change Text In File 228
Choose Folder 229
Copy Folder 230
File Exists 231
Move To Bin 232
Read File 233
Run Program 234
Save Game 235
Write File 236

Navigation Finding a way round walls on the navigation mesh.

Get Navigation Point 237
Grid Path 238
Navigate To 239
Navigation Reachable 240

TileMap Reading and changing tiles and grid cells.

Get Cell 241
Grid Map Cell 242
Paint Terrain 243
Set Cell 244
Tile 245

Debug Messages, checks and drawings for finding what is wrong.

Assert 246
Draw Debug 247
Print 248

AI Guards, hunters and things that run away.

Alert Allies 249
Can See 250
Chase 251
Flee 252
Interaction 253
Keep Apart 254
Keep Distance 255
Look For Target 256
Make Noise 257
Patrol 258
Search Last Known 259
Strafe Around 260
Take Cover 261
Target Lost 262
Wander 263

Combat Attacks, bullets, health and stats.

Attack 264
Bullet Pattern 265
Heal 266
Hurt 267
Modify Stat 268

Color Tinting colours, fading them and taking them apart.

Get Color Components 269
Random Color 270
Sample Gradient 271
Set Color Components 272
Set Modulate 273

Lights Turning lights on and off, and their colour and brightness.

Set Light 274

Material The materials 3D things are drawn with.

Set Material 275
Set Material Color 276

Set Material Texture 277

Convert Turning one kind of value into another.

String To Number 278

Application & Display The game window, the frame rate and screenshots.

Display Settings 279
Get Screen Size 280
Run When Unfocused 281
Set UI Scale 282
Take Screenshot 283

Video Playing videos.

Play Video 284

Inventory Finding items, carrying them, wearing them and using them.

Add Item 285
Count Item 286
Drop Item 287
Drop Loot 288
Equip Item 289
Give Item 290
Inventory Has 291
Move Item 292
New Run 293
Offer Item 294
Pick Item 295
Remove Item 296
Show Item 297
Sort Inventory 298
Unequip Item 299
Use Item 300

Quests Things to do, ticked off as they are done.

Abandon Quest 301
Complete Quest 302
Fail Quest 303
Quest Is 304
Quest Step Is 305
Start Quest 306
Track Quest 307

Dialogue Characters talking, and the answers the player picks.

Dialogue Line Said 308
Start Dialogue 309

Nodes To Add Ready-made nodes you add to a scene, not to a state.

Bar 310
Crosshair 311
Data Node 312
Dialogue Box 313
Experience 314
Globe 315
Hit Box 316
Hurt Box 317
Icon Meter 318
Inventory 319
Inventory Grid 320
Item Collection 321
Minimap 322
Perception 323
Performance Overlay 324
Quest List 325
Quest Log 326
Skill Bar 327
Stat Panel 328

I want to…

What you are trying to do, in plain words, and the action that does it. If what you want is not here, look through the contents, where the groups say what they are for.

A

act on a score reaching a number ... Variable 40
act when a timer runs out ... Event On Signal 76
add a double jump ... Jump 132
add a fullscreen tick box to an options menu ... Screen Settings 225
add a hit stop when a blow lands ... Set Time Scale 66
add a light when a switch is thrown ... Add Node 70
add a little kick to a shot ... Recoil 193
add a pickup to the collection ... Give Item 290
add a text size option to a menu ... Set UI Scale 282
add a timer while the game runs ... Add Node 70
add an echo inside a cave ... Set Bus Effect 177
add an on-screen joystick ... Virtual Joystick 158
add juice to a pickup ... Punch 99
add one to a count by name ... Dictionary 59
add up a score ... Math 46
aim a gun at the mouse pointer ... Mouse Position 150
aim a speed at a target in one step ... Expression 45
aim with a second thumb stick ... Virtual Joystick 158
aim with the mouse ... Mouse Look 191
aim with the mouse while it is hidden ... Mouse Motion 148
aim with the right stick ... Look At 96
aim with the right stick ... Get Input Vector 143
air dash to a far ledge ... Dash 128
allow a jump only when on the ground ... Touching 125
allow one air dash per jump ... Dash 128
answer one question in many places ... Bool Operator 33
ask a yes or no question ... Show Choices 217
ask for a name only when a score makes the table ... High Score Table 60
ask what a weapon's rarity is ... Data 61
ask where to save something ... Choose Folder 229
ask whether a quest step is done ... Quest Step Is 305
attack once an enemy is in position ... Keep Distance 255
attack when an enemy catches up ... Chase 251
attack when the enemy is close enough ... In Range 95

B

back away when the player gets too near ... In Range 95
back up the player's saves ... Copy Folder 230
begin a challenge or a tutorial ... Start Quest 306
blend a character from walking to running ... Set Tree Parameter 167
blow up a barrel ... Explosion 110
bob up to the surface ... Swim 137
branch on ranges of health ... Switch 39
break a chain ... Set Joint 121
break a loop of states that run in one frame ... Next Frame Event 28
bring a beaten character back to life ... Heal 266
bring a door back when a lever is pulled ... Enable Node 75
build a branching dialogue ... Show Choices 217
build a bridge as the player walks ... Set Cell 244
build a message out of several values ... Format String 54
build a random dungeon floor ... Build Dungeon 219
build a wall jump ... Wall Cling 138
build a world one piece at a time ... Grid Map Cell 242
build walls that join up at the edges ... Paint Terrain 243
buy an item from a shop ... Move Item 292
buzz the pad when the player is hit ... Vibrate 157

C

call nearby guards for help ... Alert Allies 249
call out a corner before the car reaches it ... Path Progress 98
cancel a timed message ... Cancel Delayed Events 17
cap a score at a maximum ... Clamp 42
carry a key to a door ... Carry 92
catch a mistake while testing ... Assert 246
change a character's portrait ... Set Texture 214
change a colour as a number rises ... Sample Gradient 271
change a node's colour ... Set Property 91
change a setting no action covers ... Set Property 91
change a value in a data record ... Set Data 64
change just one part of a place ... Set Vector Components 52
change the background colour ... Set Background Color 195
change the colour of one enemy ... Set Material Color 276
change the colour of the sky ... Set Environment 203
change the hue of a colour ... Set Color Components 272
change the music for a boss fight ... Play Music 173
change the picture on a 3d screen ... Set Material Texture 277
change the size of the game window ... Screen Settings 225
change the speed of the music ... Set Pitch 178
change what a character's stats are while geared ... Equip Item 289
change words inside a file ... Change Text In File 228
charge a shot while a button is held ... Event On Input 141
chase whoever the guard has spotted ... Chase 251
check a folder name is free ... File Exists 231
check a spot is clear before spawning ... Shape Cast 123
check a typed-in password ... String Compare 57
check a value in a data record ... Data 61
check an enemy's own health ... Data 61
check if the mouse is over a button ... Pointer Over UI 211
check the player can afford something ... Count Item 286
check the player can pay ... Inventory Has 291
check what another character is doing ... In State 26
check what kind of thing a variable is holding ... Node 86
check what surface you are standing on ... Tile 245
check whether a character is against a wall ... Touching 125
check whether a file exists ... File Exists 231
check whether a file is there ... Call Static Method 72
check whether a save is there ... File Exists 231
check whether a sound is still playing ... Playing 164
check whether a state is reached ... Print 248
check whether a timer is still under a door ... Has Child Of Type 83
check whether a zone is empty ... Get Overlapping 113
check whether health has run out ... Variable 40
check whether something has a part ... Has Child Of Type 83
check whether something is in view ... World To Screen 198
check whether something is riding on a platform ... Is Child Of 85
check whether something is visible ... Property 87
check whether the player is holding an item ... Is Child Of 85
check whether the score beat the high score ... Compare 35
check whether there is ground ahead ... Raycast 116
check whether what was hit is part of the player ... Is Child Of 85
choose a screen size from a list ... Option Button 210
choose a state by a weapon number ... Switch 39
choose by a word ... Switch 39
choose what to do next from where you came from ... Get Previous State 22
choose where a new project goes ... Choose Folder 229
clear a picture from the screen ... Set Texture 214

clear away bullets that leave the screen ... On Screen 115
clear away every pickup on a floor ... Queue Free 88
clear what the player is holding ... New Run 293
click a chest to walk over and open it ... Click To Move 126
click a tile to walk there round the walls ... Grid Path 238
click an enemy to walk up and fight it ... Click To Move 126
click on something in the world ... Mouse Pick 149
climb a rope ... Swing 124
climb a shaft ... Wall Cling 138
climb a staircase without jumping ... Step Up 136
climb onto a ledge after a short jump ... Ledge Grab 133
close the game from a menu ... Quit Game 224
collect a coin ... Event On Collision 109
colour a health bar by how much health is left ... Sample Gradient 271
combine two true-or-false variables ... Bool Operator 33
compare two variables ... Compare 35
control one object through another ... Forward Events 19
copy a code for the player to share ... System 227
copy a folder ... Copy Folder 230
copy a value off another node ... Get Property 80
copy something already in the level ... Duplicate 73
count how many a blast caught ... Explosion 110
count how many coins are collected ... Set Variable 49
count how many gamepads are plugged in ... Joypad Connected 146
count how many steps away something is ... Grid Path 238
count something on one thing ... Set Data 64
count the coins left to collect ... Group 81
count the enemies left in a wave ... Get Children 78
count the fingers on the screen ... Touch Count 155
count the walls between two places ... Raycast All 117
count what is standing on a pressure plate ... Get Overlapping 113
crawl through a low gap ... Crouch 127
crouch behind cover ... Crouch 127
cut a fuse before the bomb goes off ... Cancel Delayed Events 17
cut to a cutscene camera ... Switch Camera 196

D

damage everything near an explosion ... Explosion 110
darken the shadows over time ... Set Environment 203
dash across a gap ... Dash 128
delete a file ... Move To Bin 232
delete a folder ... Move To Bin 232
delete a game from a launcher ... Move To Bin 232
dig a tile out of the ground ... Set Cell 244
dig a tunnel while the game runs ... Paint Terrain 243
dim a light gradually ... Ease Variable 44
dip the view when the player lands ... Head Bob 190
disable the continue button when there is no save ... Set Disabled 212
dissolve an enemy as it dies ... Set Shader Parameter 204
dive to the bottom of a lake ... Swim 137
do a special move from a run of presses ... Input Combo 145
do something else when there is nowhere to hide ... Take Cover 261
do something only while in a certain state ... In State 26
do something to every enemy in a room ... For Each 36
do something to every enemy in a room ... Group 81
do something when a slider moves ... Get Value 207
do something when two fingers touch the screen ... Touch Count 155
do something while the player stands on a plate ... Touching 125
dodge on a double tap ... Event On Input 141
dodge roll out of the way ... Dash 128
drag a card with the mouse ... Drag 140
drain the colour out when health is low ... Screen Effect 202
draw a trail on the ground behind something ... Leave Marks 201
drift a cloud across the sky ... Translate 105
drink a potion from the inventory with a key ... Use Item 300
drive a car with the keys ... Drive 108
drive a car's accelerator ... Get Action Strength 142
drive a character with a stick ... Get Input Vector 143
drive an animated character from its speed ... Set Tree Parameter 167
drop a coin when an enemy is defeated ... Instantiate 84
drop a door shut smoothly ... Tween Property 106
drop an item back into the level ... Reparent 89
drop gold when an enemy dies ... Drop Loot 288
drop loot on the floor ... Drop Item 287
drop through a platform ... Set Collision 119
duck the music under some speech ... Set Volume 180
duck under a swing ... Crouch 127

E

empty a chest into the player's pack ... Move Item 292
empty a pack onto the ground ... Drop Item 287
empty a slot on purpose ... Unequip Item 299
end a multi-step sequence ... Sub State Finished 32
end a round when time runs out ... Timer 68

F

face a character where the mouse points ... Look At 96
fade a colour to see-through ... Set Color Components 272
fade a level in from black ... Fade Screen 188
fade a menu in ... Set Visible 216
fade a sound down ... Set Volume 180
fade a sound out ... Stop Sound 182
fade a value over time ... Ease Variable 44
fade one animation layer over another ... Set Tree Parameter 167
fade out when the player dies ... Fade Screen 188
fade something out ... Tween Property 106
fade something out slowly ... Set Modulate 273
fade something out with a shader ... Set Shader Parameter 204
fade the music out ... Stop Music 181
fade the screen to black ... Fade Screen 188
fade the sky to orange at sunset ... Set Background Color 195
fail a quest ... Fail Quest 303
fill a chest as the level starts ... Add Item 285
fill a chest with loot ... Inventory 319
fill a drop-down list ... Option Button 210
fill a health bar ... Set Value 215
fill an area with copies of a scene ... Procedural Placement 223
fill in a shop's price and name ... Show Item 297
find a word in some text ... String 56
find every enemy in a sword swing ... Shape Cast 123
find everything along a laser ... Raycast All 117
find how high something is ... Get Vector Components 50
find out how big the screen is ... Get Screen Size 280
find out how fast the player is moving ... Get Velocity 114
find out how hard two things hit ... Get Contact 112
find out how long the player has been in a level ... Get Time 65
find out how much damage a hit did ... Get Event Data 20
find out the hue of a colour ... Get Color Components 269
find out what another character is doing ... Get State 23
find out what day it is ... System 227
find out what is under a character's feet in 3d ... Grid Map Cell 242
find out what the player is standing on ... Get Cell 241
find out what the player ran into ... Get Last Collision 129
find out whether something is visible ... Get Property 80

find out whether the machines are slowing the game — Performance Overlay 324
find out whether there is a save to continue — Save Game 235
find out which event led to a state — Get Last Event 21
find the angle between two directions — Vector 53
find the camera the player is looking through — Get Current Camera 189
find the enemy a hurtbox belongs to — Get Parent 79
find the exit door — Find Child 77
find the nearest enemy — Distance 93
find the nearest enemy — Get Overlapping 113
find the nearest spot a character can stand — Get Navigation Point 237
find the place under the middle of the screen — Screen To World 194
find the player's documents folder — System 227
find the shortest turn between two angles — Math 46
find the spot in front of a gun — To Global 103
find the strongest weapon — Find Row 62
find the way through a maze of tiles — Grid Path 238
find what is carrying an object — Get Parent 79
find where a click landed compared with a node — To Local 104
find where an enemy is on the screen — World To Screen 198
find which button is highlighted — Get Focus 205
finish a quest — Complete Quest 302
finish a timer early — Emit Signal 74
fire a bullet as wide as it looks — Shape Cast 123
fire a bullet from a gun — Instantiate 84
fire a bullet in a straight line — Set Velocity 122
fire a ring of bullets from a boss — Bullet Pattern 265
fire three shots in a row — Repeat 37
flash a character red when it is hit — Set Modulate 273
flash a health bar when health moves — Variable Changed 41
flash a sprite white when it is hit — Set Shader Parameter 204
flash something white when it is hit — Set Material 275
flash the background red — Set Background Color 195
flash the screen twice — Repeat 37
flash the screen white — Fade Screen 188
flick a light switch on and off — Set Light 274
flip a picture upside down — Flip Sprite 161
flip a true-or-false variable — Bool Operator 33
flip a true-or-false variable — Set Variable 49
flip gravity with a button — Set Gravity 120
fly a camera through a level — Follow Path 94
follow a moving target — Chase 251
follow a moving target through a maze — Navigate To 239
follow one quest out of many — Track Quest 307
freeze a moving part when the game is paused — Stop Animation 168
freeze everything for a moment — Set Time Scale 66
freeze something with an ice spell — Freeze 111

G

get a different game every time — Random Seed 48
get hold of the player — Find Child 77
get the level number out of a name — String To Number 278
get through a doorway with a lip — Step Up 136
give a boss three attacks in turn — Sequence 38
give a chest a chance of holding treasure — Chance 34
give a chest a random reward — Random Event 29
give a chest its own contents — Data Node 312
give a dodge a cooldown — Dash 128
give a gun a fire rate — Bullet Pattern 265
give a level a seed players can share — Random Seed 48
give a quest's reward — Complete Quest 302
give a random amount of damage — Random Number 47
give a sprint a stamina bar — Stamina 135
give a top-down 3d ship a gun — Bullet Pattern 265
give an enemy eyes and ears — Perception 323
give each enemy its own health — Data Node 312
give every run a different layout — Build Dungeon 219
give health back slowly over time — Heal 266
give idle characters something to do — Wander 263
give something a new part — Add Node 70
give something a random colour — Random Color 270
give the hero skills on number keys — Skill Bar 327
give the player a ghostly look — Set Material 275
give the player a pack — Inventory 319
give the player a quest — Start Quest 306
give the player a starter project — Copy Folder 230
give the player a temporary speed boost — Modify Stat 268
give the player an item they picked up — Give Item 290
give the player answers to choose — Dialogue Box 313
give the player experience — Experience 314
give up a chase — Target Lost 262
give up a quest — Abandon Quest 301
give up the chase after a look around — Search Last Known 259
give up when the player cannot be reached — Navigation Reachable 240
go back to the camera from before a cutscene — Get Current Camera 189
go back to the previous state — Go To Previous State 24
go to the next level — Change Scene 220
greet the player differently the second time — Dialogue Line Said 308
grey out a button — Set Disabled 212
grey out a button — Set Modulate 273
group weapons together — Sort Inventory 298

H

halve damage until a curse is cured — Modify Stat 268
hand a quest item over — Move Item 292
have a co-driver call the corners — Text To Speech 183
have a dialogue with a character — Start Dialogue 309
have an animal drift around a field — Wander 263
have the camera watch a thrown ball — Camera Follow 184
heal the player with a potion — Heal 266
hide a health bar — Set Visible 216
hide and freeze part of a level — Enable Node 75
hide behind a crate — Take Cover 261
hide the mouse in a first-person game — Set Mouse Mode 154
hold a crate still until it is hit — Freeze 111
hold a finger down to open a menu — Touch Gesture 156
hold a title card on screen — Wait 69
hold a value steady while a state runs — Set Property 91
hold an archer at a distance — In Range 95
hold an engine sound while a car is stopped — Pause Sound 172
hold on only while a button is held — Wall Cling 138
hold your breath underwater — Stamina 135
hurt a character in a cut scene — Hurt 267
hurt enemies with a sword — Hit Box 316
hurt the player for falling too far — Get Contact 112
hurt the player when they fall too far — Hurt 267

I

ignore clicks on the menu or inventory — Pointer Over UI 211

J

jump an animation to a moment — Seek Animation 165
jump straight to another state — Go To State 25

jump when a button is pressed ... Input 144
jump when a key is pressed ... Event On Input 141

K

keep a crowd from bunching up ... Keep Apart 254
keep a list of doors already opened ... Array 58
keep a log of each run ... Write File 236
keep a menu working while the game is paused ... Set Process Mode 90
keep a pickup turning ... Rotate 100
keep a score ... Set Variable 49
keep a table of best lap times ... High Score Table 60
keep a top ten of high scores ... High Score Table 60
keep an alarm ringing until it is switched off ... Set Sound Loop 179
keep count of each item the player carries ... Dictionary 59
keep pixel art the same size on every screen ... Camera Zoom 187
keep something moving during a pause ... Set Process Mode 90
keep something moving forward ... Translate 105
keep the camera inside the whole level ... Camera Limits 185
keep the camera on the player ... Camera Follow 184
keep the music going when the scene changes ... Play Music 173
keep the player's health and items in the next level ... Change Scene 220
keep the pointer inside the window ... Set Mouse Mode 154
keep the view inside one room ... Camera Limits 185
keep up when the window is resized ... Get Screen Size 280
knock a block out of a wall ... Grid Map Cell 242
knock an enemy back ... Apply Force 107
know how far round a track a car is ... Path Progress 98
know what the cursor is hovering over ... Mouse Pick 149
know when an enemy has lost you ... Look For Target 256
know when something dies ... Hurt Box 317
know when something leaves a zone ... Event On Collision 109
know when the player is in the lake ... In Water 131
know whether the phone is held upright or sideways ... Screen Orientation 153
know who attacked me ... Get Event Data 20

L

land a hit part way through a swing ... Playing 164
lay out a shop's shelf ... Inventory Grid 320
lean a car into its corners ... Drive 108
leave a state after a few seconds ... Timeout 67
leave tracks in the snow ... Leave Marks 201
leave tyre marks behind a sliding car ... Leave Marks 201
let a character remember where it last saw the player ... Perception 323
let a chest roll its own loot ... Inventory 319
let a frame pass before moving on ... Next Frame Event 28
let a held sound carry on ... Pause Sound 172
let a menu be used with a gamepad ... Grab Focus 208
let a vehicle take over the player's events ... Forward Events 19
let a wall be climbed instead of avoided ... Ledge Grab 133
let an animationtree move on with a condition ... Set Tree Parameter 167
let an enemy be beaten ... Hurt Box 317
let every item turn up again ... New Run 293
let go of a rope ... Set Joint 121
let go of a swing and fly ... Swing 124
let go when a button is released ... Input 144
let players invert the look controls ... Mouse Look 191
let players turn screen shake down ... Camera Shake 186
let the camera go anywhere again ... Camera Limits 185
let the game run again ... Pause Game 222
let the player abandon a mission ... Abandon Quest 301
let the player be hurt ... Hurt Box 317
let the player choose a folder ... Choose Folder 229
let the player choose a language ... Set Language 226
let the player choose their own keys ... Rebind Input 152
let the player give up a quest ... Quest Log 326
let the player pick from a list that changes ... Show Choices 217
let the player place tiles with the mouse ... Set Cell 244
let the player track a quest ... Quest Log 326
let the player type their name ... Get Text 206
level up ... Experience 314
limit how long a shield holds ... Stamina 135
limit the frame rate in an options menu ... Display Settings 279
list every coin within reach ... Get Overlapping 113
load a list of names from a file ... Read File 233
lock the screen to landscape ... Screen Orientation 153
look around in first person ... Mouse Look 191
look up a node by its name ... Find Child 77
look up a value in a table ... Get Data 63
look up an item by name ... Find Row 62
lose a mission when time runs out ... Fail Quest 303
lower the 3d resolution for slow computers ... Display Settings 279

M

make a back button ... Go To Previous State 24
make a ball bouncy ... Set Body Properties 118
make a balloon float ... Freeze 111
make a bar fill up smoothly ... Ease Variable 44
make a barrel that breaks ... Hurt Box 317
make a broken lamp cut out ... Flicker 200
make a bulb glow while it is on ... Set Material Color 276
make a bullet fly ... Translate 105
make a bullet go off when it hits ... Event On Collision 109
make a bullet go through several enemies ... Raycast All 117
make a button bounce when it is pressed ... Punch 99
make a button grow under the pointer ... Scale To 102
make a character bounce off a wall ... Get Last Collision 129
make a character face the way it walks ... Flip Sprite 161
make a character heavier underwater ... Set Gravity 120
make a character speak a line at a time ... Set Text 213
make a character weightless only while a button is held ... Set Gravity 120
make a coin squash and stretch ... Punch 99
make a collection page ... Item Collection 321
make a colour from numbers ... Set Color Components 272
make a conveyor belt move ... Set Material Texture 277
make a crate heavier ... Set Body Properties 118
make a crystal glow ... Set Material Color 276
make a difficulty menu ... Option Button 210
make a door react to a switch's state ... Get State 23
make a fast car need braking for corners ... Drive 108
make a forest with clearings ... Procedural Placement 223
make a fuse burn down ... Timer 68
make a ghost see-through ... Set Modulate 273
make a grappling hook ... Swing 124
make a guard investigate a noise ... Search Last Known 259
make a guard react to seeing you ... Look For Target 256
make a guard see the player ... Perception 323
make a guard walk a beat ... Patrol 258
make a guard wander around a room ... Get Navigation Point 237
make a gun kick when it fires ... Recoil 193
make a health bar drain smoothly ... Set Value 215
make a held gun swing as the player looks around ... Weapon Sway 197
make a hit feel heavy ... Camera Shake 186
make a lift go up ... Move To 97
make a light flicker ... Flicker 200
make a node do something no action covers ... Call Method 71

make a photo mode ... Take Screenshot 283
make a pickup grow as it appears ... Scale To 102
make a pickup grow as it appears ... Tween Property 106
make a pickup sound go up a step each time ... Set Pitch 178
make a place from two numbers ... Set Vector Components 52
make a platform disappear ... Enable Node 75
make a platform move along a track ... Follow Path 94
make a power-up that lasts eight seconds ... Modify Stat 268
make a puncture pull a car to one side ... Drive 108
make a quest journal ... Quest List 325
make a quit button work ... Quit Game 224
make a run repeatable ... Random Seed 48
make a shop's buy and sell buttons ... Show Choices 217
make a shotgun spread ... Bullet Pattern 265
make a skill cost mana ... Skill Bar 327
make a sound come out different each time ... Play Random Sound 174
make a sound finish after the node has gone ... Play Sound At Position 176
make a sound loop ... Set Sound Loop 179
make a state move on ... Send Event 31
make a switch drawn as two frames ... Stop Sprite Animation 169
make a tap a small hop ... Jump 132
make a tick box on an options menu work ... Toggle Button 218
make a torch fade out ... Set Light 274
make a trap hurt whoever steps on it ... Hurt 267
make a turret aim at the player ... Look At 96
make a value pulse up and back down ... Ease Along Curve 43
make a villager run from the player ... Flee 252
make a volume slider work ... Set Volume 180
make a volume slider work ... Get Value 207
make a wall jump ... Jump 132
make a wall too slippery to rest on ... Wall Cling 138
make an animated character start running ... Travel To 170
make an animation play faster ... Set Animation Speed 166
make an archer keep its distance ... Keep Distance 255
make an effect tidy itself up ... Queue Free 88
make an enemy act unpredictably ... Chance 34
make an enemy behave differently each time ... Random Event 29
make an enemy duck into cover ... Take Cover 261
make an enemy find its way to the player ... Navigate To 239
make an enemy glow red when it is angry ... Set Material 275
make an enemy hurt to touch ... Hit Box 316
make an enemy run at the player ... Chase 251
make an enemy shoot at the player ... Bullet Pattern 265
make an enemy shudder when it is hit ... Punch 99
make an enemy swing at the player ... Attack 264
make an enemy turn round at the edge of a platform ... Touching 125
make an enemy walk towards the player ... Move Character 134
make an open door stop blocking the way ... Set Collision 119
make another enemy like this one ... Duplicate 73
make confetti in many colours ... Random Color 270
make enemies appear anywhere in an arena ... Instantiate 84
make enemies arrive from spawn points ... Instantiate 84
make enemies circle the player ... Strafe Around 260
make enemies drop loot ... Drop Loot 288
make enemies investigate a noise ... Make Noise 257
make gravity lighter on the moon ... Set Gravity 120
make guards come when a gun is fired ... Make Noise 257
make held fire climb up the screen ... Recoil 193
make ice and lava behave differently ... Get Cell 241
make one enemy spotting you alert the room ... Alert Allies 249
make rare items come up less often ... Pick Item 295
make rare items drop less often ... Drop Loot 288
make smoke while a car is sliding ... Drive 108
make something blink in and out of sight ... Flicker 200
make something glow hotter as it heats up ... Sample Gradient 271
make something move at a set speed ... Set Velocity 122
make something patrol a fixed route ... Follow Path 94
make something ride on a moving platform ... Reparent 89
make something spring past its end and settle ... Ease Along Curve 43
make sparks where something hits ... Get Contact 112
make spikes ... Hit Box 316
make stats grow with levels ... Experience 314
make sure a value never goes out of range ... Assert 246
make the camera chase a car ... Camera Follow 184
make the crosshair open when the player runs or fires ... Crosshair 311
make the floor slippery ... Set Body Properties 118
make the game window borderless ... Display Settings 279
make the interface bigger or smaller ... Set UI Scale 282
make the player attack ... Attack 264
make the player jump ... Jump 132
make the world glow ... Set Environment 203
make villagers mill about ... Wander 263
make walking feel like walking in first person ... Head Bob 190
make water or lava flow ... Set Material Texture 277
make water ripple faster ... Set Shader Parameter 204
mark a door as open ... Group 81
mark a quest on the map ... Track Quest 307
mark quests on a map ... Minimap 322
match a run animation to how fast something moves ... Play Animation 162
measure a place from a node's point of view ... To Local 104
measure how deep the lake is where the player stands ... Get Water Depth 130
measure the distance between two nodes ... Vector 53
move a character to where you click ... Mouse Position 150
move a number along a shape i draw ... Ease Along Curve 43
move a platform across a gap ... Move To 97
move a platform between markers ... Patrol 258
move a unit across a tactics board ... Grid Path 238
move furniture around a room ... Drag 140
move on when a character arrives ... Navigation Reachable 240
move on when everything inside a state is done ... Sub State Finished 32
move the game window or put it in the middle of the screen ... Display Settings 279
move the pointer onto a button ... Warp Mouse 159
muffle the sound under water ... Set Bus Effect 177
mute the game from an options menu ... Mute 171

N

name a new project ... Change Text In File 228
narrate the story between levels ... Text To Speech 183
notice a car crashing ... Drive 108
notice an empty chest ... Inventory Has 291
notice the player at a distance ... Distance 93
notice a target has died ... Node 86
notice when a target is replaced ... Variable Changed 41
notice when the phone is shaken ... Device Motion 139
notice when the player comes into view ... Look For Target 256
notice when the player turns the phone ... Screen Orientation 153

O

offer a quest only once ... Quest Is 304
offer a reward the player may refuse ... Offer Item 294
offer the player a choice ... Show Choices 217
open a chest by pressing a key beside it ... Interaction 253
open a chest on screen ... Inventory Grid 320
open a door from a switch ... Send Event 31
open a door once a quest is done ... Quest Is 304
open a door only while a switch is on ... In State 26

open a door only with the key ... Inventory Has 291
open a door when every enemy is defeated ... Group 81
open a door when the player walks in ... Event On Collision 109
open a door when the player walks up ... Distance 93
open a file in a particular program ... Run Program 234
open a game from a launcher ... Run Program 234
open a hole under the player ... Set Cell 244
open a menu with a key ... Event On Input 141
open a sliding door ... Move To 97
open a web page from a menu ... System 227
open and close an inventory with one key ... Set Visible 216
open the manual or a folder from a menu ... System 227

P

paint a road or a path of tiles ... Paint Terrain 243
pass events from one machine to another ... Forward Events 19
pass straight through a state ... Next Frame Event 28
pause a sprite animation ... Stop Sprite Animation 169
pause a video ... Play Video 284
pause an animation and carry on later ... Stop Animation 168
pause before the next step ... Wait 69
pause the game when the player switches to another window ... Run When Unfocused 281
pause the game with a menu ... Pause Game 222
pause the music while a menu is up ... Pause Sound 172
pause when a controller is unplugged ... Joypad Connected 146
pay a shop price ... Remove Item 296
pick a colour at random from a few ... Random Color 270
pick a patrol point it can walk to ... Navigation Reachable 240
pick a random direction ... Random Vector 51
pick a random enemy type ... Find Row 62
pick a random place to walk to ... Get Navigation Point 237
pick a random spawn point ... Array 58
pick a random spot near something ... Random Vector 51
pick one of several outcomes at random ... Random Event 29
pick the first item on a shelf ... Get Children 78
pick up a crate ... Carry 92
pick up a dropped pile, whatever is in it ... Move Item 292
pick up and drop a puzzle piece ... Drag 140
pinch to zoom on a phone ... Pinch 151
place a block where the player points ... Grid Map Cell 242
place enemies at random across an arena ... Procedural Placement 223
place something at the edge of the screen ... Get Screen Size 280
play a cutscene and carry on when it ends ... Play Video 284
play a dialogue with answers to choose ... Start Dialogue 309
play a door opening animation ... Play Animation 162
play a footstep ... Play Sound 175
play a footstep sound on each step ... Animation Event 160
play a sound on a coin as it disappears ... Play Sound At Position 176
play a sound when a crate lands ... Get Contact 112
play a sound when a door opens ... Play Sound 175
play a splash as feet go into water ... In Water 131
play a taunt now and then ... Chance 34
play a walk animation ... Play Sprite Animation 163
play an animation backwards ... Play Sprite Animation 163
play an attack and move on when it ends ... Play Sprite Animation 163
play an attack over the walk with a oneshot ... Set Tree Parameter 167
play an explosion where something was destroyed ... Play Sound At Position 176
play an intro video ... Play Video 284
play footstep sounds in time with the walk ... Head Bob 190
play idle standing and run while moving ... Play Sprite Animation 163
play music on a level ... Play Music 173
play one of several hit sounds ... Play Random Sound 174
play the same floor again from a number ... Build Dungeon 219
point at the next objective ... Off Screen Indicator 209
point the mouse at a character in a tutorial ... Warp Mouse 159
poison the player a little every second ... Hurt 267
press a button from a machine ... Emit Signal 74
puff out smoke when something breaks ... Emit Particles 199
pull a lever or talk to someone when close ... Interaction 253
pull the camera back for a boss fight ... Camera Zoom 187
pull up onto a rooftop ... Ledge Grab 133
punch ... Attack 264
push boxes along a conveyor ... Translate 105
push things with wind ... Apply Force 107
put a boss far from the start ... Build Dungeon 219
put a bullet mark on a wall ... Raycast 116
put a car back on the road ... Path Progress 98
put a countdown on screen as minutes and seconds ... Format Time 55
put a delete button beside each answer ... Show Choices 217
put a gap between shots ... Wait 69
put a health bar over an enemy's head ... Bar 310
put a name into a line of text ... Format String 54
put a name over a character's head ... World To Screen 198
put a random item on a pedestal ... Pick Item 295
put a sword in the player's hand ... Reparent 89
put a weapon away ... Unequip Item 299
put an icon in an inventory slot ... Set Texture 214
put an item on a pedestal the player can walk past ... Offer Item 294
put down what the player is holding ... Carry 92
put every enemy of a wave in a group ... Group 81
put loot in the player's pack ... Add Item 285
put on armour found in a chest ... Equip Item 289
put one thing at each spawn point ... For Each 36
put race results or high scores in order ... Dictionary 59
put something just above a character's head ... To Global 103
put the controls back to how they started ... Rebind Input 152
put the keyboard on the first button ... Grab Focus 208
put the mouse pointer in the middle of the screen ... Warp Mouse 159
put the player's name into a file ... Change Text In File 228
put words on a label ... Set Text 213

Q

quit after a sound finishes ... Quit Game 224

R

raise an alarm when a shot is heard ... Alert Allies 249
react differently to different ways in ... Get Last Event 21
react differently to what a bullet hit ... Group Switch 82
react to a menu button being pressed ... Event On Signal 76
react to part of a quest ... Quest Step Is 305
react when a score goes up ... Variable Changed 41
read a code typed into a keypad ... Get Text 206
read a menu aloud ... Text To Speech 183
read a node's position into a variable ... Get Property 80
read a number the player typed ... String To Number 278
read a slider on an options menu ... Get Value 207
read a slider's value into a variable ... Event On Signal 76
read a text file into a variable ... Read File 233
read a value off another node ... Property 87
read a weapon's damage ... Get Data 63
read an enemy's own stats ... Get Data 63
read how far a trigger is pulled ... Get Action Strength 142
read how fast something falls ... Get Vector Components 50

read how fast the mouse is moving ... Mouse Motion 148
read how full a bar is ... Get Value 207
read how see-through something is ... Get Color Components 269
read the player's score from another node ... Machine Variable 27
read the tile under the mouse ... Get Cell 241
read the time of day ... Call Static Method 72
read the value an event brought with it ... Get Event Data 20
read what a label or button says ... Get Text 206
read words out loud ... Text To Speech 183
recover from being stunned ... Timeout 67
remember a high score between runs ... Save Game 235
remember several things at once ... Array 58
remember the best time on each level ... Dictionary 59
remember the chosen language for next time ... Set Language 226
remember what a character was doing before ... Get Previous State 22
remember what was said ... Dialogue Line Said 308
remember where the menu focus was ... Get Focus 205
remember which camera is in use ... Get Current Camera 189

S

save a picture beside each save slot ... Take Screenshot 283
save and load the player's progress ... Save Game 235
save battery when the game is in the background ... Run When Unfocused 281
save what the player typed ... Write File 236
say when there are no potions left ... Use Item 300
scatter animals when you come near ... Flee 252
scatter trees or rocks over the ground ... Procedural Placement 223
score a coin that is worth more than one point ... Get Event Data 20
search where the player was last seen ... Search Last Known 259
see a line between two things while testing ... Draw Debug 247
see how fast the game runs ... Performance Overlay 324
see if a node has moved past a point ... Property 87
see if health dropped below a number ... Compare 35
see if the player has the key ... Variable 40
see if there is a gap ahead ... Tile 245
see what a gun is pointing at ... Raycast 116
see what a variable holds ... Print 248
see whether a character fits through a gap ... Shape Cast 123
see whether something is on screen ... Node 86
see which room the player is in ... String Compare 57
select a unit with the mouse ... Mouse Pick 149
sell something for gold in a dialogue ... Start Dialogue 309
send a ball back the other way ... Set Velocity 122
send a car round a circuit ... Follow Path 94
send a guard back on patrol ... Target Lost 262
send a lift to its floors in order ... Sequence 38
send a machine back to the start ... Go To State 25
send an event on a particular animation frame ... Animation Event 160
send an event when a bar is full ... Clamp 42
set a volume slider when a menu opens ... Set Value 215
set how high something is ... Set Vector Components 52
set off the player's active item with a button ... Use Item 300
shake the screen when a bomb goes off ... Camera Shake 186
shoot only while a guard can see the player ... Can See 250
shove a crate across the room ... Apply Force 107
show a bar when the player walks into a room ... Bar 310
show a boss's health bar ... Bar 310
show a chest drawn open ... Set Texture 214
show a door already open ... Seek Animation 165
show a guard's state while testing ... Get State 23
show a lap time to a hundredth ... Format Time 55
show a loading bar ... Set Value 215
show a loading screen while a level loads ... Change Scene 220
show a map of the rooms ... Minimap 322
show a minimap in the corner of the screen ... Minimap 322
remember which level the player is in ... Get Scene 221
remove a coin when it is collected ... Queue Free 88
restart an enemy's behaviour ... Enable Machine 18
restart the current level ... Get Scene 221
restart the level after losing ... Change Scene 220
restart the level with the score it began with ... Change Scene 220
return from a pause menu to where you were ... Go To Previous State 24
reuse one behaviour in several templates ... Run Template 30
rewind an animation ... Seek Animation 165
roll a ball by tilting the screen ... Device Motion 139
roll a dice ... Random Number 47
roll fog in ... Set Environment 203
rumble on an explosion ... Vibrate 157
run a boost tank down and back up ... Stamina 135
run a round timer ... Timer 68
run a shared take-a-hit routine ... Run Template 30
run away when health is low ... Flee 252
run while a key is held ... Input 144
show a name in capitals ... String 56
show a name over a character ... Off Screen Indicator 209
show a potion on a hotkey ... Skill Bar 327
show a press e prompt near things ... Interaction 253
show a progress bar along a route ... Path Progress 98
show a quest log ... Quest Log 326
show a saved setting on a tick box ... Toggle Button 218
show a score on screen ... Set Text 213
show a score with a label in front of it ... Format String 54
show a security camera view ... Switch Camera 196
show a speaker's picture beside their words ... Dialogue Box 313
show a speedometer ... Drive 108
show a stage time ... Format Time 55
show a timer counting down ... Set Text 213
show an aiming mark in the middle of the screen ... Crosshair 311
show an arrow count on the hud ... Count Item 286
show an arrow to something off screen ... Off Screen Indicator 209
show an item on a pedestal ... Show Item 297
show every item the player has ever found ... Item Collection 321
show frames a second ... Performance Overlay 324
show half a heart ... Icon Meter 318
show health as a row of hearts ... Icon Meter 318
show health in a red globe ... Globe 315
show health out of its maximum ... Stat Panel 328
show how deep a diver is ... Get Water Depth 130
show how far a quest has got ... Quest List 325
show how far something reaches ... Draw Debug 247
show how full a pack is ... Count Item 286
show level up! ... Experience 314
show lives or ammunition as icons ... Icon Meter 318
show mana in a blue globe ... Globe 315
show one frame of a sprite ... Stop Sprite Animation 169
show that a shot hit ... Crosshair 311
show the best loot first ... Sort Inventory 298
show the current language on a button ... Set Language 226
show the current screen settings in a menu ... Screen Settings 225
show the gap to a best time ... Format Time 55
show the high-score table on the title screen ... High Score Table 60
show the last event while testing ... Get Last Event 21
show the level name on the screen ... Get Scene 221
show the level on the minimap ... Minimap 322
show the mouse for a menu ... Set Mouse Mode 154
show the player from a little way above ... Camera Follow 184
show the player's pack ... Inventory Grid 320
show the player's quests ... Quest List 325

show the player's saved choice in a list ... Option Button 210
show the player's stats on the screen ... Stat Panel 328
show the previous state while testing ... Get Previous State 22
show the real time of day ... Get Time 65
show the right button prompt ... Last Device 147
show the saves or recent files to choose from ... Show Choices 217
show touch controls only on a phone ... System 227
show what a character is wearing ... Inventory Grid 320
show what a character says ... Dialogue Box 313
show what a sword or a potion adds to a stat ... Stat Panel 328
show what the player has just picked up ... Show Item 297
show what the player is carrying on a pause menu ... Item Collection 321
show what the player wrote earlier ... Read File 233
show what the selected menu item does ... Get Focus 205
show where a shot came from ... Off Screen Indicator 209
show where to go next ... Minimap 322
show which key a control is on ... Rebind Input 152
show which rooms the player has visited ... Minimap 322
show which way something is aiming ... Draw Debug 247
shrink an enemy away when it dies ... Scale To 102
shuffle a deck of cards ... Array 58
sidestep around a target ... Strafe Around 260
silence a sound when a state is left ... Stop Sound 182
silence the music ... Mute 171
silence the music for a cutscene ... Stop Music 181
skip to the end of a sequence ... Go To State 25
slide a menu panel onto the screen ... Move To 97
slide a menu panel onto the screen ... Tween Property 106
slide down a wall ... Wall Cling 138
slow a player down in deep water ... In Water 131
slow a player the deeper they wade ... Get Water Depth 130
slow a voice to a growl ... Set Pitch 178
slow an animation down ... Set Animation Speed 166
slow the game down for a finishing move ... Set Time Scale 66
smooth one animation into the next ... Play Animation 162
sort things by what they are ... Group Switch 82
spawn five coins one at a time ... Repeat 37
spawn something where the crosshair points ... Screen To World 194
spawn things at random places ... Random Vector 51
speed up an animation as something happens ... Set Animation Speed 166
spend a key on a door ... Remove Item 296
spill what an enemy carried ... Drop Item 287
spin a barrel ... Apply Force 107
spin a coin ... Rotate 100
splash into water and out again ... Swim 137
split a boss in two ... Duplicate 73
split a line of text into a list ... String 56
spread a pack out around the player ... Keep Apart 254
stagger a group of enemies ... Timeout 67
start a bullet at the end of a barrel ... To Global 103
start a chase when an enemy spots you ... Can See 250
start a drawbridge motor ... Set Joint 121
start a dust trail while running ... Emit Particles 199
start a fresh run after dying ... New Run 293
start a mission ... Start Quest 306
start a new game and throw the old save away ... Save Game 235
start a pause menu on the continue button ... Grab Focus 208
start an air bar when the head goes under ... In Water 131
start another program ... Run Program 234
start searching when the player gets away ... Target Lost 262
start the game from a menu ... Change Scene 220
steer by tilting the phone ... Device Motion 139
step a speed toward a top speed ... Math 46
step a tutorial through its stages ... Sequence 38
step over a low crate ... Step Up 136
step through a list of dialogue lines ... For Each 36
step through the languages with one button ... Set Language 226
stock a shop shelf ... Add Item 285
stock a shop shelf ... Pick Item 295
stock a shop shelf ... Inventory 319
stock a shop shelf without spending the stock ... Offer Item 294
stop a barrel from rolling ... Set Body Properties 118
stop a body dead ... Set Velocity 122
stop a boulder rolling until the player passes ... Freeze 111
stop a character standing up inside a ceiling ... Crouch 127
stop a click on a button from moving the player ... Pointer Over UI 211
stop a delayed event from arriving ... Cancel Delayed Events 17
stop a fight looking static ... Strafe Around 260
stop a guard patrolling during a cutscene ... Enable Machine 18
stop a hidden menu being clicked ... Set Visible 216
stop a looping engine sound ... Stop Sound 182
stop a looping sound after it finishes ... Set Sound Loop 179
stop a pickup being collected twice ... Set Collision 119
stop a sprite animation ... Stop Sprite Animation 169
stop a walker at the edge of a ledge ... Get Cell 241
stop a weapon looking glued to the screen ... Weapon Sway 197
stop an animation ... Stop Animation 168
stop an item being used before it has charged ... Use Item 300
stop chasing when the player breaks line of sight ... Can See 250
stop enemies crowding the player ... Keep Distance 255
stop enemies standing inside each other ... Keep Apart 254
stop everything when the level is won ... Pause Game 222
stop footsteps sounding the same every step ... Play Random Sound 174
stop health going below zero ... Clamp 42
stop part of a level from running ... Set Process Mode 90
stop the camera seeing through walls ... Orbit Camera 192
stop the camera showing past the edge of the level ... Camera Limits 185
stop the game when something goes wrong ... Assert 246
stop the music at the end of a level ... Stop Music 181
stop the pad buzzing ... Vibrate 157
stop the player buying what they cannot afford ... Set Disabled 212
store how hard a stick is pushed ... Get Action Strength 142
store the answer a node gives back ... Call Method 71
store values on one thing ... Data Node 312
store who an enemy is chasing ... Look For Target 256
strip armour before a swim ... Unequip Item 299
swap a car's paint for a burnt look ... Set Material 275
swap icons when a gamepad is picked up ... Last Device 147
swap one weapon for another ... Equip Item 289
swim across a pool ... Swim 137
swing a door open on its hinge ... Rotate To 101
swing a sword ... Attack 264
swing from a vine ... Swing 124
swing on a rope ... Swing 124
swing the camera round the player with the mouse ... Orbit Camera 192
swipe to change weapon ... Touch Gesture 156
switch a character into its attack pose ... Travel To 170
switch a lamp on with a lever ... Set Light 274
switch a sound effect on the mixer on and off ... Set Bus Effect 177
switch an enemy on when the player arrives ... Enable Node 75
switch back to the player's camera ... Switch Camera 196
switch between walking and running animations ... Get Velocity 114
switch on a hit box partway through a swing ... Animation Event 160

T

take a life away ... Set Variable 49
take a screenshot ... Take Screenshot 283

take an item away when it is sold ... Give Item 290
take damage off an enemy's own health ... Set Data 64
take damage off health ... Math 46
take everything a defeated enemy was carrying ... Move Item 292
take health off the enemy a bullet hit ... Machine Variable 27
take one part out of a place ... Get Vector Components 50
take something back off a thief ... Remove Item 296
take the focus off a menu when it closes ... Grab Focus 208
take the spaces off something typed in ... String 56
talk to someone ... Start Dialogue 309
tap the screen to shoot ... Touch Gesture 156
tear a door off its hinges ... Set Joint 121
tell a coin from a key ... Group Switch 82
tell a whole pack who to chase ... Alert Allies 249
tell another machine something happened ... Send Event 31
tell every enemy the player was spotted ... Send Event 31
tell every minion what the boss knows ... Machine Variable 27
tell other scripts that something happened ... Emit Signal 74
tell the player their pack is full ... Add Item 285
tell when a ball has stopped rolling ... Get Velocity 114
tell when the player lifts every finger ... Touch Count 155
tell whether an enemy is on the screen ... On Screen 115
tell whether something is in front of the player ... To Local 104
tell whether the player is on a pad ... Last Device 147
tell which enemy the player bumped ... Get Last Collision 129
test whether a guard has a clear view ... Raycast 116
test whether a variable is empty ... String Compare 57
thank the player for a quest ... Quest Is 304
throw a ball ... Carry 92
throw a car sideways into a corner ... Drive 108
throw a weapon away ... Drop Item 287
throw away a save ... Move To Bin 232
throw crates away from a grenade ... Explosion 110
throw something to lure a guard away ... Make Noise 257
tidy the pack when the panel opens ... Sort Inventory 298
time how long a level takes ... Timer 68
time how long a race takes ... Get Time 65
tip a platform over ... Rotate To 101
toggle pause with one key ... Pause Game 222
trigger a slide kick from two buttons ... Input Combo 145
turn a character to face the way it moves ... Move Character 134
turn a colour into numbers ... Get Color Components 269
turn a door panel green when it unlocks ... Set Material Color 276
turn a fan or a radar dish ... Rotate 100
turn a health into a share of its bar ... Expression 45
turn a key in a lock ... Rotate To 101
turn a light up or down ... Set Property 91
turn a setting on and off with a checkbox ... Toggle Button 218
turn a sprite into an on-screen button ... Touch Gesture 156
turn a sprite round for art drawn facing left ... Flip Sprite 161
turn a touch on the screen into a place in the level ... Screen To World 194
turn an enemy round at the edge of a platform ... Tile 245
turn off a particle effect ... Emit Particles 199
turn something to an angle kept in a variable ... Rotate To 101
turn something to look at a target ... Look At 96
turn something with the mouse ... Mouse Motion 148
turn text into a number ... String To Number 278
turn the lights on in the evening ... Get Time 65
turn the lights red for an alarm ... Set Light 274
turn the map with the player ... Minimap 322
turn the sound back on ... Mute 171
turn the whole picture black and white ... Screen Effect 202
turn v-sync on or off ... Screen Settings 225

U

undo an upgrade when it runs out ... Modify Stat 268
unlock a secret with a code ... Input Combo 145
use a node that came with another plugin ... Call Method 71
use a sum from a script i wrote ... Call Static Method 72
use crosshairs instead of the arrow ... Set Mouse Mode 154
use up an arrow ... Remove Item 296

V

vault a low wall ... Ledge Grab 133

W

wait a random amount of time ... Random Number 47
wait a random moment ... Wait 69
wait for a second player's pad ... Joypad Connected 146
wait for a sound to finish before carrying on ... Play Sound 175
wait for all parallel states to finish ... Sub State Finished 32
wait for an animation to finish ... Playing 164
wait for an animation to finish before moving on ... Play Animation 162
wait for another template to finish ... Run Template 30
wait until a spot is out of view before spawning ... On Screen 115
wait until a weapon has been drawn ... Travel To 170
wake a trap when a lever is pulled ... Enable Machine 18
walk a character on a phone ... Virtual Joystick 158
walk a character with the arrow keys ... Get Input Vector 143
walk a patrol through a set of markers ... Get Children 78
walk a player with the keys ... Move Character 134
walk a route and come back again ... Patrol 258
walk a villager across a town round the walls ... Navigate To 239
walk on the ceiling ... Set Gravity 120
walk through walls for a moment ... Set Collision 119
walk to where the mouse was clicked ... Move Character 134
walk up a kerb ... Step Up 136
walk where the mouse is clicked ... Click To Move 126
warn myself when something goes wrong ... Print 248
warn that the pack is full ... Inventory Has 291
warn the player before something appears ... Instantiate 84
wash the screen red when hit ... Screen Effect 202
watch the player from behind ... Orbit Camera 192
wear a sword from the pack ... Equip Item 289
work out a percentage for a bar ... Math 46
work out damage from several numbers ... Expression 45
work out the direction from one thing to another ... Vector 53
work out whether a door can be opened ... Bool Operator 33
write text to a file ... Write File 236

Z

zoom a map with two fingers ... Pinch 151
zoom in and out with the wheel ... Orbit Camera 192
zoom in for a dialogue ... Camera Zoom 187
zoom the view with the mouse wheel ... Pinch 151
zoom with the mouse wheel ... Mouse Position 150

The settings every action has

Five settings are on nearly every action, and are not repeated on the pages that follow. Run Mode is left off the actions that take time of their own. The other four are on every one.

SETTING	WHAT IT DOES	STARTS AS
Run Mode	**Once** does the work once when the state is entered. **Every Frame** does it over and over for as long as the state is active. Once suits setting a value, doing a sum or sending an event, and Every Frame suits anything that has to keep up with something as it changes, such as reading a stick that is being held. An action on Every Frame is never finished, so its state never sends FINISHED. Leave that state on another event. Actions that take time of their own, such as **Wait** or **Move To**, do not show this setting, because they always run until they are done.	Once, or Every Frame on some (see below)
Update Interval	Seconds between repeats. On Every Frame, how often the work is done again. On an action that takes time, it is how often it moves along, so a large number makes movement jump in steps. Zero repeats every frame, which is what you want unless the work is expensive.	0
Wait for Previous	Hold this action back until the action above it in the list has finished. Off, every action in the state starts together.	off
Disabled	Skip this action without deleting it, for trying something out.	off
Display Name	The words shown for this action on the state's list. Leave it empty and the action's own name is used.	empty

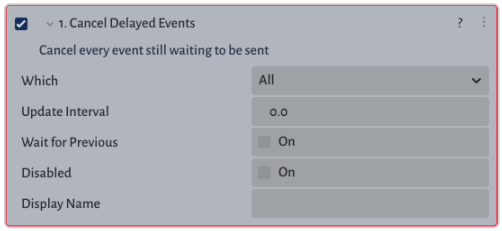
Most actions start on Once. These start on Every Frame, because what they do has to keep up with something as it changes. They are Assert, Clamp, Click To Move, Device Motion, Draw Debug, Event On Input, Flip Sprite, Format Time, Get Action Strength, Get Contact, Get Input Vector, Get Last Collision, Input Combo, Interaction, Last Device, Mouse Motion, Mouse Pick, Mouse Position, Path Progress, Raycast, Raycast All, Screen To World, Set Text, Set Tree Parameter, Set Value, Touch Count, World To Screen. Set any of them to Once to do the work a single time.

The Var button

Most settings have a small **Var** button beside them. Press it and instead of the value you typed, the action reads a variable while the game runs, so the same action can do something different each time. Press it again to go back to the typed value, which is kept where it was. Each page says which kind of variable a setting's Var list offers, in the words the Variables tab uses, which are float, int, bool, String, Vector2, Vector3, Color, Node, Resource, Array and Dictionary.

Cancel Delayed Events

Calls off events that a Send Event with a Delay is still waiting to send, before they arrive. It can call off every one of them, or only the event you name.



Cancel Delayed Events as it appears in the Actions tab.

When to use it

A lit fuse has a Send Event waiting to send BOOM in 3 seconds. When the player cuts the fuse first, this calls BOOM off. Leaving the state that sent an event calls it off anyway. This is only needed while that state still runs, from a state inside it or a parallel state beside it.

> A Send Event whose event is called off counts as finished, so the state it is in can still finish and send FINISHED.

See also
Send Event 31, **Wait** 69, **Next Frame Event** 28, **Random Event** 29

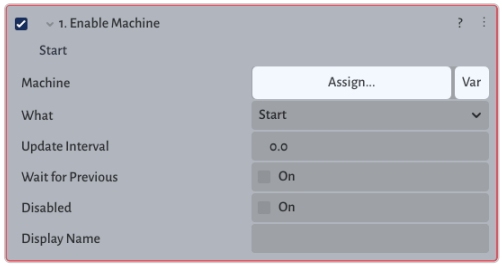
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/state_machine/cancel_delayed_events/cancel_delayed_events.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Which	All calls off every event this machine is still waiting to send. One Event calls off only the event named below and lets the others arrive as planned. Choices: All, One Event.	All
Event Name	Only when Which is One Event. The event to call off, spelled as the Send Event names it. Left empty, no event is called off and a warning is printed.	—

Enable Machine

Starts, stops, restarts, pauses or resumes another machine.



Enable Machine as it appears in the Actions tab.

When to use it

A cutscene can Pause a guard's patrol machine and Resume it afterwards, holding the guard where it was, timers and all. A lever can Start a trap that sat idle, and a boss's second phase can Restart its attack machine from the top.

> Resuming also puts back what the other machine's Every Frame actions were showing, such as a label it writes, in case something else wrote over it while it was held.

See also

Run Template 30, **Machine Variable** 27, **Go To State** 25, **Pause Game** 222

Try it

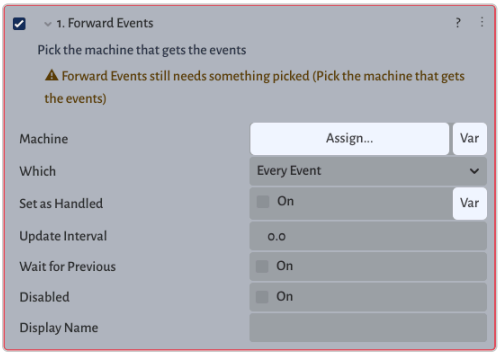
See the action work on its own with **Open Example** in the Add Action window, or open `examples/state_machine/enable_machine/enable_machine.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Machine	The machine to act on, as a Machine node or a node with one under it, such as the guard itself. It has to be filled in. Left empty, nothing happens and a warning says Machine points at no machine. Var: a Node.	empty
What	Start sets it running from its first state, and does nothing if it is already running. Stop stops it where it is, leaving its states properly, so starting again begins from the top. Restart does both in one go. Pause holds it where it is (its actions stop, its timers stop, and nothing is lost), and Resume lets it carry on from there. Start is the default. Choices: Start, Stop, Restart, Pause, Resume.	Start

Forward Events

While its state runs, passes the events this machine gets on to another machine too. That can be all of them, only the ones you list, or all but those.



Forward Events as it appears in the Actions tab.

When to use it

For one thing taking charge of another for a while. A player sitting in a car can hand HIT and REPAIRED on to the car's machine from a Driving state, and the handing on stops as soon as that state is left. A single event for another machine is simpler with Send Event.

> FINISHED is never passed on, and an event two machines pass back and forth to each other stops after both have had it once.

See also

Send Event 31, **Enable Machine** 18, **Machine Variable** 27, **Get Event Data** 20

Try it

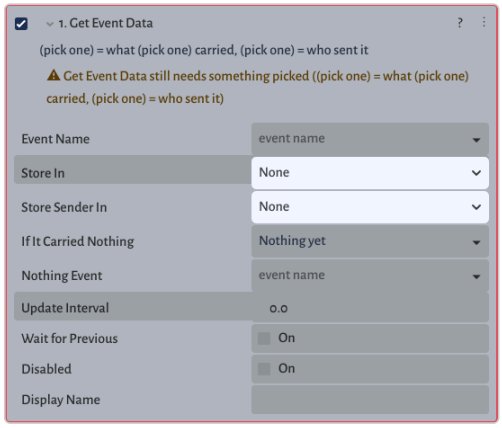
See the action work on its own with **Open Example** in the Add Action window, or open `examples/state_machine/forward_events/forward_events.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Machine	The machine that gets the events. You can pick the Machine node itself, a node with a machine under it, or a node inside something that has one, such as an enemy's hurtbox. Left empty, no events are passed on and a warning is printed. Var: a Node.	empty
Which	Every Event passes on everything this machine gets. Only These passes on just the events listed below. All But These passes on everything except the events listed. Choices: Every Event, Only These, All But These.	Every Event
Set as Handled	Off, the default, both machines act on the events. On, the events go to the other machine only, so this machine's transitions waiting for them do nothing while the state runs. With Every Event, only FINISHED can then move it on. Var: a bool.	off
Events	Only when Which is Only These or All But These. The events to pass on, or to keep back, picked from the events the design uses.	—

Get Event Data

Reads the value an event arrived carrying and puts it in a variable. It is how the state that catches an event gets hold of what came with it, such as how much damage a hit did or who pressed the button.



Get Event Data as it appears in the Actions tab.

When to use it

A coin worth 5 sends its event carrying 5, and the score's machine reads the 5 here before adding it. Put it first in the state the transition leads to. When only the fact of the event matters, the transition alone is enough.

> The value is only there for the frame the event arrives, so this has to be the first action of the state the transition leads to, not a state further on.

See also

Send Event 31, **Set Variable** 49, **Variable** 40, **Print** 248

Try it

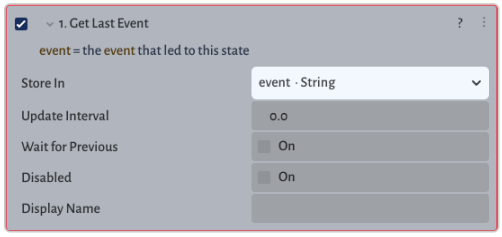
See the action work on its own with **Open Example** in the Add Action window, or open `examples/state_machine/get_event_data/get_event_data.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Event Name	Which event to read. Left empty, as it comes, it reads whichever event brought the machine into this state. Fill it in only to read one particular name whatever else arrived.	empty
Store In	The variable the value goes into. Left empty, the value is not kept, for when only Store Sender In matters. Nothing Event is still sent when there was nothing to read.	empty
Store Sender In	Also puts the node that sent the event into a variable, such as the guard whose shot hit you. Empty, as it comes, keeps nobody. It keeps whoever last sent an event of that name, so read it on the frame the event arrives.	empty
If It Carried Nothing	What goes into Store In when the event came empty-handed, so the variable is never left holding the last event's value. Empty as it comes, and the variable is then left empty. Set it to 0, or to 1, where the rest of your template reads a number.	empty
Nothing Event	An event sent when there was nothing to read, because the event carried no value or no event was waiting this frame. Empty, as it comes, sends nothing. It is sent whether or not Store In is filled in.	empty

Get Last Event

Puts the name of the event that brought the machine into this state into a text variable.



Get Last Event as it appears in the Actions tab.

When to use it

Several transitions can lead into one state. A Hurt state reached by HIT_BY_ARROW or HIT_BY_SWORD can store which it was, then play a different sound for each through Switch. Get Event Data reads what an event carried, not its name.

> A state that no event led to gives an empty name, such as the first state the machine starts in, or one reached by Go To State. A state reached because the one before finished gives FINISHED.

See also

Get Event Data 20, **Switch** 39, **String Compare** 57, **Go To State** 25

Try it

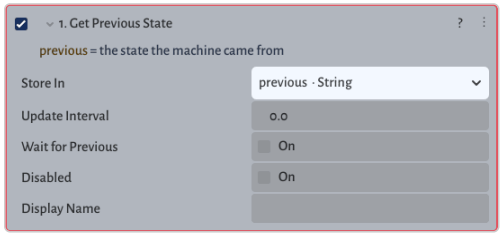
See the action work on its own with **Open Example** in the Add Action window, or open `examples/state_machine/get_last_event/get_last_event.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Store In	The text variable (a String) the event's name goes into. It starts as event. A variable that holds a number, or anything else that is not text, is left alone, and a warning names it and what it holds.	event

Get Previous State

Puts the name of the state the machine came from into a text variable.



Get Previous State as it appears in the Actions tab.

When to use it

A Stunned state can find out whether the guard was Patrolling or Chasing before it was hit. To go straight back there, Go To Previous State does it in one step.

> The first state the machine starts in came from nowhere and gives an empty name. Leaving a state that has states inside it gives the innermost one that was running.

See also

Go To State 25, **Get Last Event** 21, **String Compare** 57, **Switch** 39

Try it

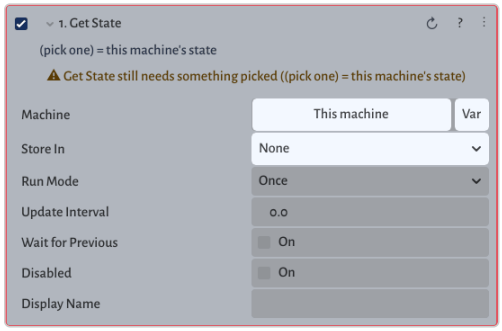
See the action work on its own with **Open Example** in the Add Action window, or open `examples/state_machine/get_previous_state/get_previous_state.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Store In	The text variable (a String) the state's name goes into. It starts as previous. A variable that is not text, a number for instance, is left as it was and a warning names it.	previous

Get State

Puts the name of the state a machine is in into a text variable. It can ask another machine, or this one when Machine is left empty.



Get State as it appears in the Actions tab.

When to use it

A boss fight can hand the boss's state name to Switch, which then sends a different event for each state. While testing, a label over a guard's head can show the state it is in. For a plain yes or no about one state, the In State check is simpler and needs no variable.

> A variable holding a number, or anything else that cannot hold text, is left as it was, and a warning asks for a String variable.

See also

In State 26, **Machine Variable** 27, **Switch** 39, **Enable Machine** 18

Try it

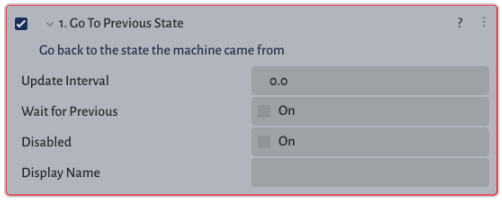
See the action work on its own with **Open Example** in the Add Action window, or open `examples/state_machine/get_state/get_state.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Machine	The machine to ask. Pick a Machine node, the node it sits under, or something inside that node, such as an enemy's hurtbox, and the machine above it is found. Left empty, the default, it asks this machine about itself. Var: a Node.	empty
Store In	The text variable the state's name goes into, as the state is named in the graph, such as Hunting. A machine in a state inside another state gives the inner one, so in Combat and, inside that, Attacking, it gives Attacking. A machine that is not running gives empty text.	empty

Go To Previous State

Goes straight back to the state the machine came from, without a transition that names it.



Go To Previous State as it appears in the Actions tab.

When to use it

A Stunned state entered from Patrolling or from Guarding can end with this, and the guard goes back to whichever it was doing. A pause menu and a Back button on an options screen return the same way.

> In the first state the machine starts in there is nowhere to go back to, so the machine stays where it is and prints a warning. Going back to a state inside another enters the one holding it on the way.

See also

Go To State 25, **Get Previous State** 22, **Pause Game** 222, **Wait** 69

Try it

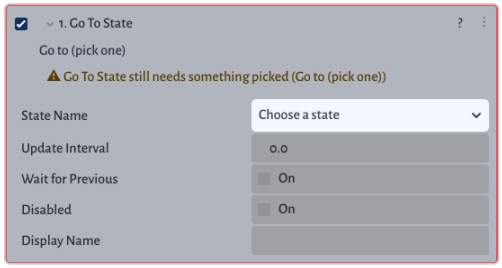
See the action work on its own with **Open Example** in the Add Action window, or open `examples/state_machine/go_to_previous_state/go_to_previous_state.tscn`.

Its settings

This one has no settings of its own beyond the four every action has.

Go To State

Jumps the machine to another state by name, with no transition. Any action listed below this one never runs.



Go To State as it appears in the Actions tab.

When to use it

For a state that has worked out where to go next, or several endings that would need a tangle of transitions. A transition is usually clearer, because the state's row reads "on FINISHED goes to Open" and the graph shows the arrow.

> The jump leaves no arrow in the graph, so someone reading the template later cannot see where that state leads.

See also

Send Event 31, **Next Frame Event** 28, **Random Event** 29, **Run Template** 30

Try it

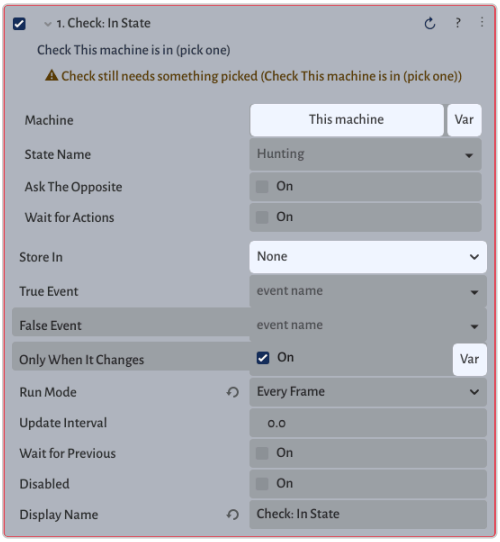
See the action work on its own with **Open Example** in the Add Action window, or open `examples/state_machine/go_to_state/go_to_state.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
State Name	The state to jump to, picked from the list of the template's states. A name no state has leaves the machine where it is, and a warning names the missing state. The Problems button lists it before the game runs. Pointing it at the state the machine is already in does nothing, and a warning says a transition from that state back to itself starts it again from the top.	empty

In State CHECK

Asks whether a machine is in a state. It can ask about another machine, or this one when Machine is left empty, and the answer is yes for as long as that state is running.



In State as it appears in the Actions tab.

When to use it

A door that opens only while the switch's machine is in On asks this, and so does an enemy that attacks only while the player is in Vulnerable. A state counts while a state inside it is running too, so asking about Combat says yes while the machine is in Combat's Attacking.

> A machine that is not running is in no state, so the answer is no until it starts.

See also

Get State 23, **Machine Variable** 27, **Sub State Finished** 32, **Enable Machine** 18

Try it

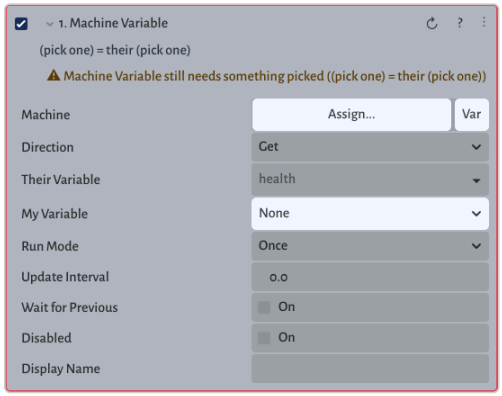
See the action work on its own with **Open Example** in the Add Action window, or open `examples/state_machine/in_state/in_state.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Machine	The machine to ask about. Pick a Machine node, the node it sits under, or something inside that node, such as an enemy's hurtbox, and the machine above it is found. Left empty, the default, it asks about this machine. Var: a Node.	empty
State Name	The state asked about, picked from the states that machine's design has. A name the machine does not have gets a warning, and the answer is then always no.	empty
Ask The Opposite	Turns the answer round, so yes becomes no. Ticked, the question becomes "it is not in this state". It turns round the answer the Check keeps in a variable as well.	off
Wait for Actions	Holds the question back until every other action in the state has finished. Left off, the default, it is asked as soon as the state runs.	off
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Machine Variable

Reads a variable from another machine, writes one to it, or adds an amount to one of its numbers. You name the variable on the other machine, the variable on this one, and which way the value travels.



Machine Variable as it appears in the Actions tab.

When to use it

The commonest use is a bullet taking health off the enemy it hit. That is Add, with Their Variable set to health and Amount to -10. Get goes the other way, as when a scoreboard reads the player's score. To tell another machine that something happened, with no number to hand over, use Send Event.

> A list or a dictionary is copied, not shared, so changing it on one machine never changes the other's.

See also

Set Variable 49, **Send Event** 31, **Enable Machine** 18, **Get Property** 80

Try it

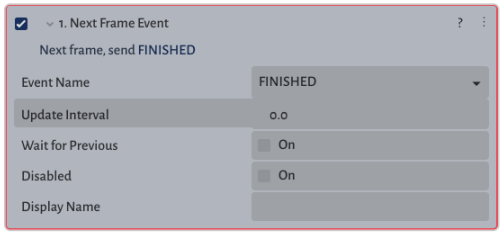
See the action work on its own with **Open Example** in the Add Action window, or open `examples/state_machine/machine_variable/machine_variable.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Machine	The other machine. It can be a Machine node, a node with one under it, or a node sitting inside something that has one. Point it at an enemy's hurtbox and the enemy's machine is found by looking upwards. Left empty, nothing happens and a warning says Machine points at no machine. Var: a Node.	empty
Direction	Get copies their variable into one of yours. Set copies yours into theirs. Add adds a number to theirs without reading it back first, as a bullet's damage does. Get is the default. Choices: Get, Set, Add.	Get
Their Variable	The name of the variable on the other machine (health, keys, score), spelled as it is there. Every Direction needs it. Left empty, nothing is read or written and a warning asks for it. The other machine needs no setting up beyond a variable by that name.	empty
My Variable	The name of the variable on this machine, the one being read into or written from. Get puts their value in here, and Set sends this value over to them. Leave it empty and nothing moves. It is taken off the panel for Add, which adds a plain amount and needs no variable of yours.	empty
Amount	Only when Direction is Add. How much to add to their number. Take some away with a negative number, so a bullet doing ten damage uses -10. It starts at -1. Press Var to read it from one of this machine's variables, so a weapon's damage can change with an upgrade.	—

Next Frame Event

Sends an event on the next frame. It comes set to FINISHED, the event a state sends when its work is done, and a state holding only this action passes straight through.



Next Frame Event as it appears in the Actions tab.

When to use it

Something has to settle first. A scene you have just added needs a frame before it is properly in the game, and a value written this frame is read next frame. It also breaks a ring of states that each lead straight into the next and would go round and round inside a single frame.

See also
Wait 69, **Send Event** 31, **Go To State** 25, **Instantiate** 84

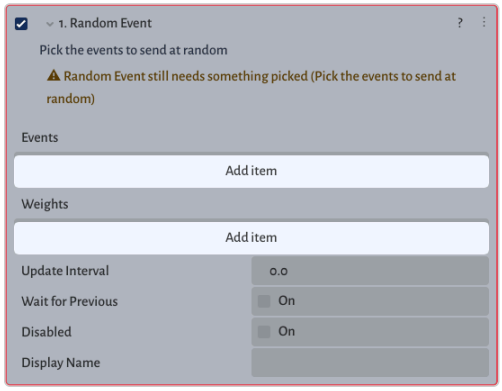
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/state_machine/next_frame_event/next_frame_event.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Event Name	The event sent on the next frame, FINISHED to begin with. Left as FINISHED it finishes the state it sits in and nothing more, so a state inside another does not finish its parent too. Any other name needs a transition waiting for it, or nothing moves.	FINISHED

Random Event

Picks one event from a list at random and sends it. The same state can lead somewhere different each time it runs.



Random Event as it appears in the Actions tab.

When to use it

Dodge or Charge, picked fresh on each visit. List both, give each a transition on the state, and the one sent decides where the machine goes. Weights let one outcome come up more often than another.

> Set every weight to 0 and the events are equally likely again. The first one does not always win.

See also

Chance 34, **Send Event** 31, **Random Number** 47, **Next Frame Event** 28

Try it

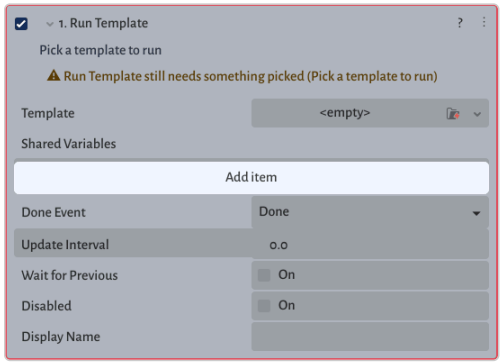
See the action work on its own with **Open Example** in the Add Action window, or open `examples/state_machine/random_event/random_event.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Events	The events to choose between, one to a row, each with a transition waiting for it. An empty list sends nothing, ends the action straight away and prints a warning, so fill in at least two. A row left without a name sends nothing when it is picked, with a warning to fill it in or remove it.	empty
Weights	How likely each event is, one number a row in the same order as the events. Leave it empty, as it comes, and every event is equally likely. The numbers are shares, not percentages, so 3 and 1 means the first is picked three times in four. A weight always lines up with its event. A row short counts as 1, and a spare row is ignored.	empty

Run Template

Runs another template inside this state, on the same node, for as long as the state lasts. Build a behaviour once, such as "open a door" or "take a hit", and use it from as many templates as you like.



Run Template as it appears in the Actions tab.

When to use it

When the same handful of states turns up in several templates, build them once as a separate template and run it from each. The state carries on once the other template reaches a state marked as an end state, or sends its Done event, and leaving the state stops the copy and takes it away.

> If the other template never reaches an end state and never sends the Done event, this state never completes. Give the template an end state, or another way out of the state.

See also

Enable Machine 18, **Machine Variable** 27, **Go To State** 25, **Send Event** 31

Try it

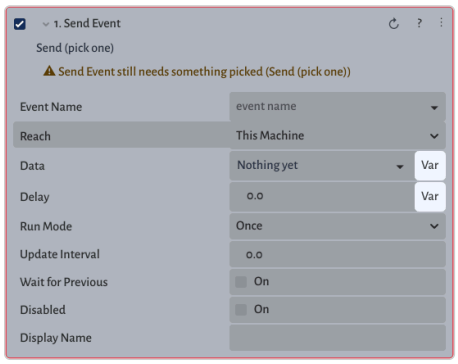
See the action work on its own with **Open Example** in the Add Action window, or open `examples/state_machine/run_template/run_template.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Template	The template to run. With none picked, nothing runs, the state carries on straight away, and a warning asks for a template.	empty
Shared Variables	Variables handed to the other template as it starts, and taken back from it when it ends, under the same name in both. Add a row for each name. Leave it empty and the template still runs, but the two have no way of passing anything between them.	empty
Done Event	An event the other template can send to say it has finished, as well as reaching a state marked as an end state. It starts at "Done". Empty waits for an end state only, so the other template has to have one.	Done

Send Event

Sends a message with a name, such as Opened or PlayerSpotted. An event moves a machine on only when the state that machine is in has a transition waiting for that same name. That transition is then taken straight away, and any action listed below this one never runs.



Send Event as it appears in the Actions tab.

When to use it

A switch tells a door to open by sending Opened to the door's machine. Inside one machine, letting the state finish on FINISHED is usually enough. Send Event is for a message that has to travel, or a state with more than one way out.

> The transition is taken the instant the event is sent, so anything listed below this action in the same state never gets to run.

See also
Get Event Data 20, **Random Event** 29, **Next Frame Event** 28, **Event On Signal** 76

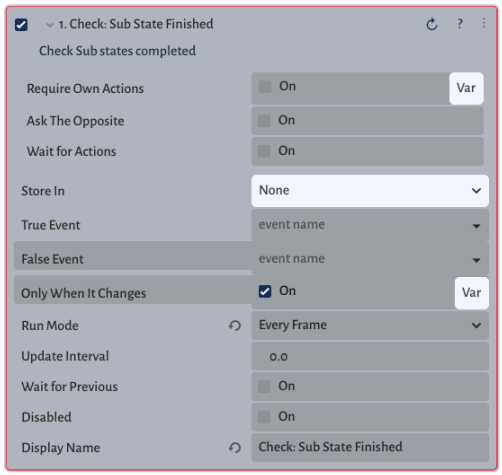
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/state_machine/send_event/send_event.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Event Name	The name of the message, such as Opened or PlayerSpotted. Names must match letter for letter, so Opened and opened are two different events. Pick from the list beside the box. An event nobody is waiting for is thrown away. Left empty, nothing is sent and a warning asks for a name.	empty
Reach	Who hears it. This Machine, as it comes, keeps it at home. A Group tells every machine in a named group, Every Machine tells everything in the game, for a message like GameOver, and A Node tells one particular machine. An event the template has marked global goes everywhere whatever you pick here. Choices: This Machine, A Group, Every Machine, A Node.	This Machine
Data	A value carried along with the event, such as how much damage, how many points, or which node. Empty, as it comes, sends the message on its own. Whoever catches it reads the value with Get Event Data. Var: any kind.	empty
Delay	Seconds to wait before the event goes, 0 for straight away. While it waits the action is not finished and its state holds back FINISHED. Leave that state before the delay is up and the event is never sent. Var: a float.	0.0
To Group	Only when Reach is A Group. Who hears it, such as every copy of the guard. Press the box and pick it in the scene. You can pick one thing by its name, Every Coin for every copy of the coin scene (including ones made while playing), or Everything under a node for a set of markers. NodeWorthy does the rest.	—
To Node	Only when Reach is A Node. The node whose machine hears it, left empty for the thing this machine itself works on. If that node has no machine, the first machine on a node above it hears it, so pointing at an enemy's hurt box reaches the enemy. With no machine on it or above it, nobody hears, and a warning names the node.	—

Sub State Finished CHECK

Asks whether the work inside this state has finished. That means the sub state marked as an end state has been reached and completed, or, when the state holds parallel states, every one of them has reached an end state.



Sub State Finished as it appears in the Actions tab.

When to use it

Put it on a parent state, one with a little machine inside it, to move on once everything in there is done. For example, a boss fight moves on after its last phase. Tick Require Own Actions when the parent's own actions must finish too. To leave when one particular thing happens inside, send an event from that sub state and wait for it.

> On a plain state with no sub states the answer stays no, because there is nothing inside to complete.

See also

Go To State 25, **Send Event** 31, **Sequence** 38, **Run Template** 30

Try it

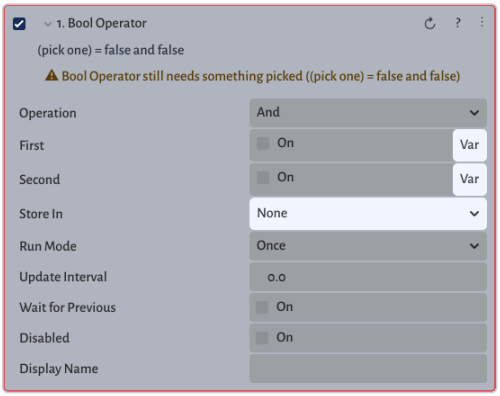
See the action work on its own with **Open Example** in the Add Action window, or open `examples/state_machine/sub_state_finished/sub_state_finished.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Require Own Actions	Ticked, the answer waits for the parent state's own actions to finish as well as everything inside it. The question never counts as one of those actions, so it cannot end up waiting on itself. Leave it off, the default, and only the states inside matter. Var: a bool.	off
Ask The Opposite	Turns the answer round into "there is still work going on inside", to keep something running while the inner machine is busy. It turns round the true-or-false answer the Check stores as well, which swapping the True Event and False Event over cannot do.	off
Wait for Actions	Holds the question back until every other action in the state has finished. Require Own Actions usually covers that, so leave this off, the default, unless one particular action must run first.	off
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Bool Operator

Works out a true-or-false sum from two values and puts the answer in a variable. It can ask for both of these, either of these, not this, or one of these but not both.



Bool Operator as it appears in the Actions tab.

When to use it

The same two or three questions asked all over a template. Work the answer out once here, into a variable such as can_open, and every door, button and transition reads can_open. A question asked once on a transition, with no answer kept, belongs to the Variable or Compare check. This action is for keeping the answer.

> Nothing is worked out while Store In is empty.

See also

Compare 35, **Variable** 40, **Set Variable** 49, **Math** 46

Try it

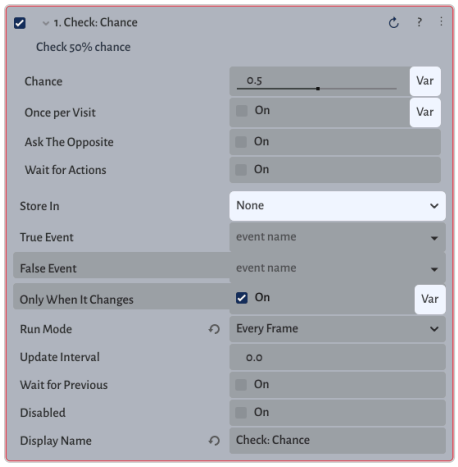
See the action work on its own with **Open Example** in the Add Action window, or open `examples/logic/bool_operator/bool_operator.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Operation	Which sum to work out. And is true only when both sides are true, Or is true when either one is, Not gives the opposite of the left-hand side and ignores the right, and Xor is true when one side is true and the other false. It starts on And. Choices: And, Or, Not, Xor.	And
First	The value on the left. Tick the box for a plain true, or press Var beside it and pick a variable, such as has_key, for the sum to follow as the game goes. Anything empty counts as false. That covers empty text, 0, and a variable that has never been set. Var: a bool.	off
Second	The value on the right, filled in the same way as First, by ticking it or pressing Var. Not ignores it. Var: a bool.	off
Store In	The name of the variable the answer is written into, such as can_open. Fill this box in. Left empty, the answer is lost, with a warning asking for a bool variable beside Store In.	empty

Chance CHECK

A roll of the dice. It is a question that comes out yes a set share of the time, for behaviour the player cannot predict.



Chance as it appears in the Actions tab.

When to use it

A guard at Chance 0.3 with Once per Visit ticked taunts on about three visits in ten and attacks straight away on the rest. Give the Check an Update Interval so the dice are thrown at a sensible pace. Asked every frame, it rolls dozens of times a second and even a small chance comes up within moments.

> With Once per Visit off and no Update Interval it rolls every frame, so even a one-in-a-hundred chance happens within a second or so.

See also

Random Event 29, **Random Number** 47, **Timeout** 67, **Random Seed** 48

Try it

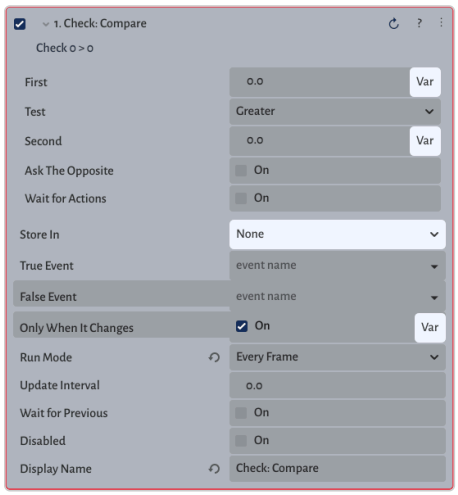
See the action work on its own with **Open Example** in the Add Action window, or open `examples/logic/chance/chance.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Chance	How likely yes is, from 0 for never to 1 for always. The default 0.5 is a coin toss and 0.1 is one time in ten. How that looks on screen depends on how often the question is asked, so set an interval to go with it. Var: a float.	0.5
Once per Visit	Ticked, the dice are thrown once when the state is entered and the same answer stands for the rest of the visit. Leave it off, the default, and every ask is a fresh roll. Tick it when the state is deciding something once, such as whether this guard taunts this time round. Var: a bool.	off
Ask The Opposite	Turns the answer round, so a chance of 0.1 becomes a nine-in-ten yes. It turns round the true-or-false answer the Check stores as well, which swapping the True Event and False Event over cannot do.	off
Wait for Actions	Holds the roll back until every other action in the state has finished, so the dice are thrown after the state's work and not on its first frame. Left off, the default, it is asked as soon as the state runs.	off
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Compare CHECK

Asks how one number compares with another. Is the first greater, smaller, or the same? Either number can be typed in or read from one of the template's variables.



Compare as it appears in the Actions tab.

When to use it

Made for two numbers that both change, such as the score against the high score, or ammo left against what a shot needs. One variable against a fixed number you typed is shorter with Variable.

> The answer comes from numbers and variables alone, so with Only When It Changes ticked it is asked again only when one of those variables is written, and costs nothing in between.

See also
Variable 40, **Property** 87, **Math** 46, **In Range** 95

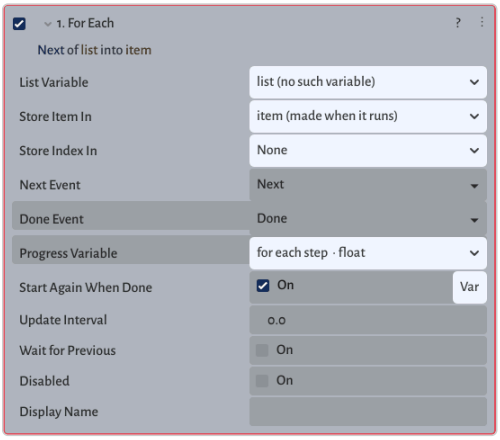
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/logic/compare/compare.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
First	The number on the left of the comparison. Type a number here, or take it from one of the template's variables. Left at 0 with no variable behind it, the question compares zero with the Second number. Var: a float.	0.0
Test	How the two are compared. The choices are Equals, Not Equals, Greater, Greater Or Equal, Less or Less Or Equal. The default is Greater, so the answer is yes while First is bigger than Second. Equals and Not Equals allow for the tiny rounding that fractional numbers pick up, so two numbers that ought to match do match. Choices: Equals, Not Equals, Greater, Greater Or Equal, Less, Less Or Equal.	Greater
Second	The number on the right, the one compared against. Type a number, or take it from a variable. Left at 0, Greater asks whether the first number is above zero. Var: a float.	0.0
Ask The Opposite	Turns the answer round, so Greater becomes Less Or Equal without changing the test. It turns round the true-or-false answer the Check stores as well, which swapping the True Event and False Event over cannot do.	off
Wait for Actions	Holds the question back until every other action in the state has finished, so it reads numbers the state's own actions have just worked out. Left off, the default, it is asked as soon as the state runs.	off
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

For Each

Walks through a list one entry at a time. Each time the state is reached it puts the next entry into a variable and sends Next. Once the list runs out it sends Done instead.



For Each as it appears in the Actions tab.

When to use it

To hurt every enemy in the room, build two states in a ring. Point this one's Next at the state that does the work, and point that state's FINISHED back here. Done leads out of the ring.

> It hands out one entry each time the state is reached, not one each frame, so there has to be a way back into this state or the loop stops after the first one.

See also

Repeat 37, **Sequence** 38, **Array** 58, **Go To State** 25

Try it

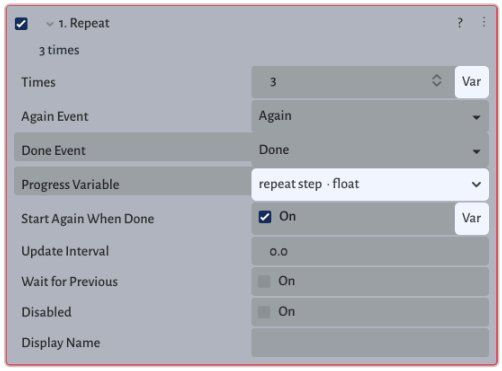
See the action work on its own with **Open Example** in the Add Action window, or open `examples/logic/for_each/for_each.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
List Variable	The variable holding the list to walk through. It starts at "list". Fill it first with the Array action, which can gather every enemy in a group, every child of a node, or a set of things you type in. If the list is empty, or the variable holds no list at all, Done is sent on the first visit. A variable that is missing or holds something other than a list also brings a warning naming it.	list
Store Item In	The variable the current entry is put into, ready for the next state to use. It starts at "item", and the state that does the work reads it by this name.	item
Store Index In	The variable the entry's place in the list is put into, counting the first as 0. Empty, the default, stores nothing. It is useful for spacing things out, so the third coin is put three steps further along.	empty
Next Event	Sent when there was another entry to hand out. It starts at "Next". Point a transition out of this state at the state that does the work.	Next
Done Event	Sent once the list has run out. It starts at "Done". Point a transition at whatever comes after the loop, or the machine stays in this state.	Done
Progress Variable	The variable that remembers how far down the list it has got, so the walk survives leaving this state and coming back. It starts at "for each step". You can watch it go up in the Variables tab, and set it back to 0 yourself to start the list again. Two For Each actions in one machine must have different names here, or each will use up the other's place.	for each step
Start Again When Done	On, the default, the count goes back to the first entry as soon as the list runs out, so coming back to this state later walks the list again from the top. Off, it stays finished and every later visit sends Done. Var: a bool.	on

Repeat

Goes round a set number of times. Each time the state is reached it sends Again, until it has been reached that many times, and then it sends Done.



Repeat as it appears in the Actions tab.

When to use it

For a burst of three shots, set Times to 3, point Again at the Fire state, and point Fire's FINISHED back here. Every loop built on it has that shape, two states in a ring, with Done leading out.

> It sends one event each time the state is reached, not one each frame, so there has to be a way back into this state or nothing goes round.

See also

For Each 36, **Sequence** 38, **Wait** 69, **Bullet Pattern** 265

Try it

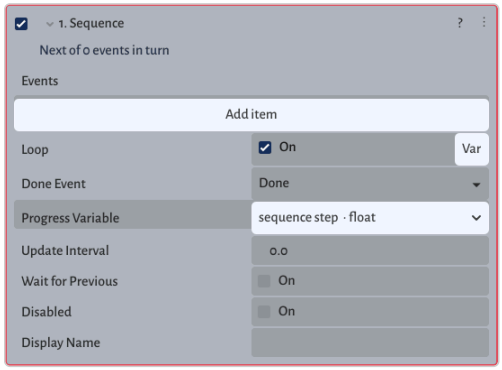
See the action work on its own with **Open Example** in the Add Action window, or open `examples/logic/repeat/repeat.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Times	How many times to go round, starting at 3. Press Var to read the number from a variable, so a weapon's burst can grow with an upgrade. At 0 it sends Done straight away and the work never happens. Var: an int.	3
Again Event	Sent while there are goes left. It starts at "Again". Point a transition out of this state at the state that does the work.	Again
Done Event	Sent once the goes are used up. It starts at "Done". Point a transition at whatever comes after the loop, or the machine sits in this state.	Done
Progress Variable	The variable that remembers how many goes have been used, so the count survives leaving this state and coming back. It starts at "repeat step". You can watch it climb in the Variables tab, and set it back to 0 yourself to start the count over. Two Repeat actions in one machine must have different names here, or one will use up the other's goes.	repeat step
Start Again When Done	On, the default, the count starts afresh once it is done, so the next visit goes round all over again. Off, it stays used up and every later visit sends Done straight away. Var: a bool.	on

Sequence

Sends a different event each time the state is reached, working down a list in order.



Sequence as it appears in the Actions tab.

When to use it

A boss with three attacks in turn lists SLAM, SPIN and CHARGE under Events, with a transition out of this state for each, and every visit takes the next one. A tutorial's steps and a lift's floors work the same way.

> With Loop off, every visit after the last event sends Done again, so a transition on Done is the only way out.

See also

Random Event 29, **Repeat** 37, **For Each** 36, **Send Event** 31

Try it

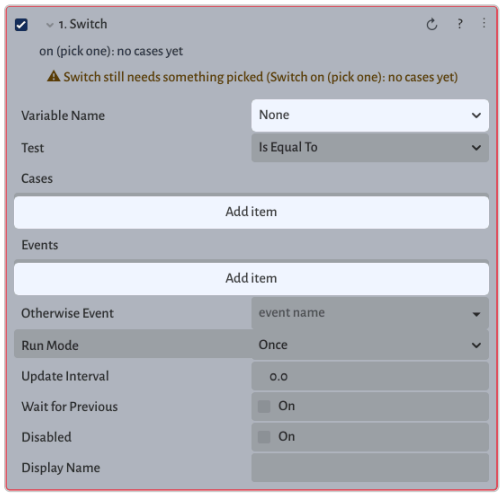
See the action work on its own with **Open Example** in the Add Action window, or open `examples/logic/sequence/sequence.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Events	The events sent, one per visit, in the order you write them. Add a row for each. With the list empty, no event is ever sent and the state completes.	empty
Loop	On, the default, it goes back to the first event after the last one, so a boss cycles through its attacks for as long as the fight lasts. Off, it stops at the end of the list and sends the Done event. Var: a bool.	on
Done Event	Sent once the list is used up, and only when Loop is off. It starts at "Done". Point a transition at what comes next, such as the end of the tutorial or the boss's second phase. With Loop on it is never sent.	Done
Progress Variable	The variable that remembers which event is next, so the place in the list survives leaving this state and coming back. It starts at "sequence step". You can watch it in the Variables tab, and set it back to 0 yourself to start the list again from the top. Two Sequence actions in one machine must have different names here, or each will move the other along.	sequence step

Switch

Sends one of several events, picked by what a variable holds. The first case that matches sends its event, and when none does, Otherwise is sent.



Switch as it appears in the Actions tab.

When to use it

One Switch does the work of a Check for every possibility, such as the word picked from a menu choosing where a door leads. Each event needs its own transition on the state. A single yes-or-no question is simpler with a Check.

> Cases without an event, a word among Is Below's numbers, and ranges out of order are listed by the Problems button.

See also

Variable 40, **Random Event** 29, **Compare** 35, **Send Event** 31

Try it

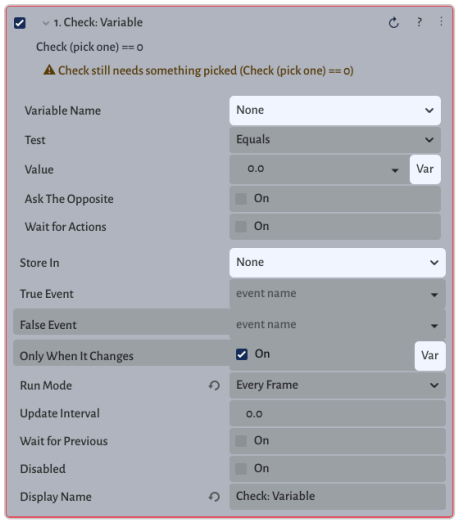
See the action work on its own with **Open Example** in the Add Action window, or open `examples/logic/switch/switch.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Variable Name	The variable to look at, which can be a float, an int, a bool or a String. Left empty, it sends nothing and a warning asks you to pick one.	empty
Test	Is Equal To, as it comes, matches the case equal to what the variable holds, such as 0, 1 and 2 for weapons, or words such as sword and bow for a String variable. Is Below matches the first case the number is under, for ranges. List them from lowest to highest, such as 25 for Critical and 50 for Hurt, and anything at 50 or above goes to Otherwise. Choices: Is Equal To, Is Below.	Is Equal To
Cases	The values to match, one to a row, in order. Numbers for a number variable, words for a Text one.	empty
Events	The event each case sends, one to a row in the same order as the cases. Each needs a transition waiting for it. A case that matches with no event beside it sends nothing, and a warning names the case.	empty
Otherwise Event	Sent when no case matches, such as Fine above every range. Empty sends nothing.	empty
Only When It Changes	Only while it runs on Every Frame. On, as it comes, an event is sent only when the matching case changes, and a transition is not taken again and again while it stays the same.	—

Variable CHECK

Asks how one of the template's variables compares with a value you type. Is the score above ten, is a flag on, is the fuel at 0?



Variable as it appears in the Actions tab.

When to use it

The everyday question about anything the machine keeps count of, such as lives reaching 0 for a game over.

> Only that variable can change the answer. With Only When It Changes ticked it is asked only when the variable is written, and a machine waiting on it costs nothing in between.

See also

Compare 35, **Set Variable** 49, **Variable Changed** 41, **Property** 87

Try it

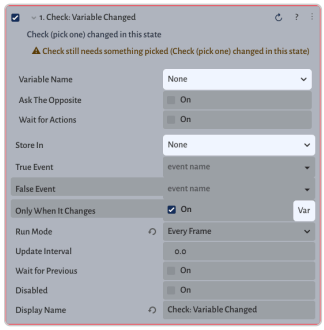
See the action work on its own with **Open Example** in the Add Action window, or open `examples/logic/variable/variable.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Variable Name	The variable to test. It should be one the template lists or one another action writes. A name nothing ever writes reads as empty, and the comparison will then hardly ever say yes.	empty
Test	How to compare. Equals, the default, and Not Equals compare the whole value, so they work for text and nodes as well as numbers. Greater, Greater Or Equal, Less and Less Or Equal read both sides as numbers. Is True and Is False ask whether the variable is on or off and ignore the Value box. Choices: Equals, Not Equals, Greater, Greater Or Equal, Less, Less Or Equal, Is True, Is False.	Equals
Value	What the variable is compared with. It takes whatever you need, such as a number, some text, or on or off. Is True and Is False ignore it. Var: any kind.	0.0
Ask The Opposite	Turns the answer round, for the other side of the same question with no second check. It turns round the true-or-false answer the Check stores as well, which swapping the True Event and False Event over cannot do.	off
Wait for Actions	Holds the question back until every other action in the state has finished, so it reads a variable the state's own actions have just written. Left off, the default, it is asked as soon as the state runs.	off
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Variable Changed CHECK

Asks whether a variable's value has changed since the last time the question was asked. It says yes once, on the first ask after the change, and then goes quiet again.



When to use it

Reacts to a change, whatever the new value. A health bar can flash each time health moves, or a chase aims again when the variable holding its target is swapped for another. A list that is added to counts as changed too, because a copy of it is kept to compare against.

> The very first ask never says yes. There is nothing to compare against yet, so it remembers the value and waits for the next ask. It watches only while its state is running. Coming into the state it takes the value as it is then, and a change made while the machine was in another state is not seen. To catch every change, ask it in a state that stays running, such as a parent state holding the others.

See also

Variable 40, **Compare** 35, **Set Variable** 49, **Ease Variable** 44

Try it

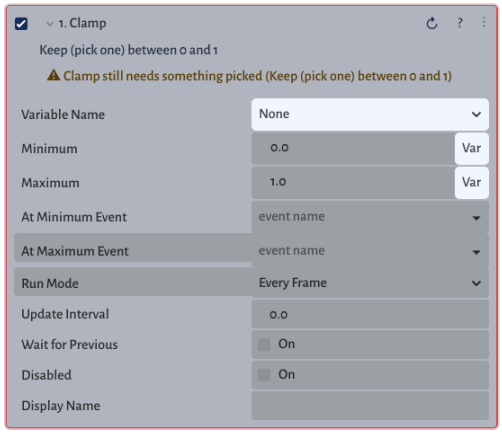
See the action work on its own with **Open Example** in the Add Action window, or open `examples/logic/variable_changed/variable_changed.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Variable Name	The variable to watch. Any change of its value counts, in either direction and whatever kind of thing it holds. A name nothing ever writes never changes, so the answer stays no.	empty
Ask The Opposite	Turns the answer round into "nothing has changed", true on nearly every frame and seldom useful here. It turns round the true-or-false answer the Check stores as well, which swapping the True Event and False Event over cannot do.	off
Wait for Actions	Holds the question back until every other action in the state has finished, so a change made by this state's own actions is seen in the same frame. Left off, the default, it is asked as soon as the state runs.	off
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Clamp

Keeps a number variable inside a range, and can send an event as it reaches either end. A variable is a name the machine remembers a value under, such as health or charge.



Clamp as it appears in the Actions tab.

When to use it

Point it at health, set 0 and 100, and give the bottom end an event such as "died" and the top one "fully healed". Health then stops at 0 and never goes negative, and the rest of the template can wait for those two events. To hear when a number passes a mark without changing it, ask with the Compare check.

> A variable nothing has written yet is left alone, so it cannot report hitting the bottom before anything has set it.

See also

Math 46, **Set Variable** 49, **Compare** 35, **Modify Stat** 268

Try it

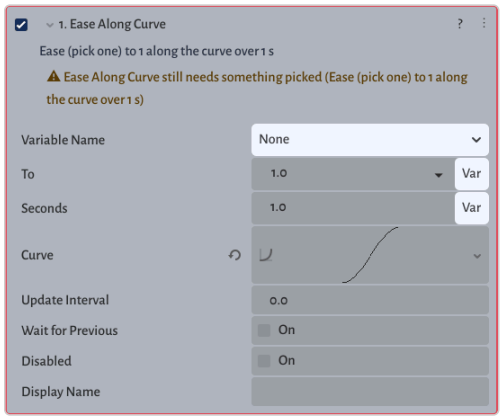
See the action work on its own with **Open Example** in the Add Action window, or open `examples/math/clamp/clamp.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Variable Name	The name of the number to keep in range. Left empty nothing happens, and a warning is printed. A number kept as a whole number stays whole after the action has been through it.	empty
Minimum	The lowest it may be. It starts at 0. Anything lower is pulled up to it. Var: a float.	0.0
Maximum	The highest it may be, starting at 1. Leave it at 1 for a bar measured from 0 to 1, or set it to 100 for health counted the usual way. Keep it above the minimum. The other way round, anything below the minimum is set to the minimum and anything else to the maximum. Var: a float.	1.0
At Minimum Event	The event sent as the number arrives at or below the lowest, such as "died" when health reaches 0. Left empty nothing is sent. It is sent once on arrival and not again while the number sits there. Leave the end and come back, and it is sent again.	empty
At Maximum Event	The event sent as the number arrives at or above the highest, such as "fully charged" or "level up". Left empty nothing is sent, and it follows the same once-on-arrival rule as the bottom end.	empty

Ease Along Curve

Moves a variable toward a value over a number of seconds, along a Curve you draw, and then finishes. Where the Curve is at 0 the variable is where it started. At 1 it has arrived, and above 1 it goes past.



Ease Along Curve as it appears in the Actions tab.

When to use it

When one of Ease Variable's ready-made shapes will do, that is quicker to set up. Draw a Curve here for anything else, such as a jump that rises fast and hangs at the top. A Curve that ends back at 0 returns the variable to where it began, for a bounce or a pulse.

> An int variable stays whole all the way, rounded at every step.

See also

Ease Variable 44, **Tween Property** 106, **Set Variable** 49, **Math** 46

Try it

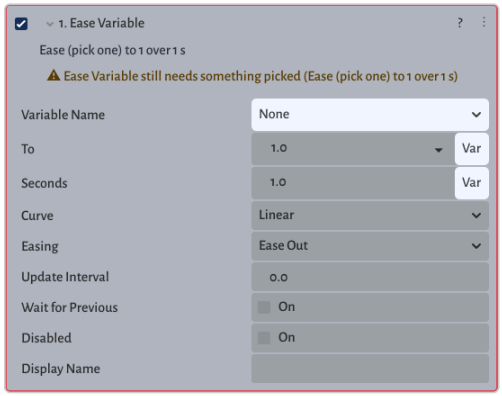
See the action work on its own with **Open Example** in the Add Action window, or open `examples/math/ease_along_curve/ease_along_curve.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Variable Name	The variable to move. It can be a number, a Vector or a colour. It is read once when the state starts and written every frame after that.	empty
To	The value it arrives at where the Curve reaches 1, of the same kind as the variable. Press Var to read it from another variable. Var: any kind.	1.0
Seconds	How long the journey takes, from the Curve's left end to its right end. At 0 it goes straight to where the Curve ends. Var: a float.	1.0
Curve	The shape of the journey. Left to right is the time going by, and the height is how far along the way it is, 0 where it started and 1 arrived. It starts as a smooth rise from 0 to 1. Click on it to add a point and drag a point's handles to bend it.	a smooth rise from 0 to 1

Ease Variable

Moves a variable's value gradually to a new one over a number of seconds, and finishes when it arrives. A variable is a name the machine remembers a value under.



Ease Variable as it appears in the Actions tab.

When to use it

A fuel gauge can settle to its new level over 0.4 seconds, and a colour can fade from red back to white. Because it finishes on arrival, a transition on FINISHED carries straight on from there. It moves variables only. A setting on a node in the scene is moved by Tween Property.

> A variable holding a whole number stays a whole number throughout, so a slow journey from 0 to 1 jumps with no values in between.

See also

Set Variable 49, **Tween Property** 106, **Math** 46, **Screen Effect** 202

Try it

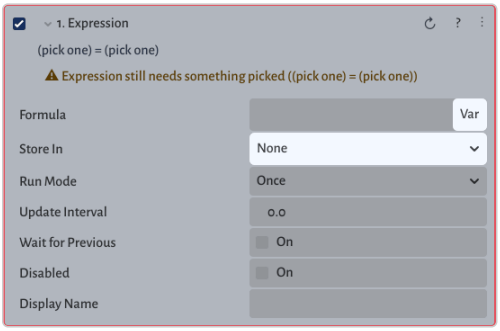
See the action work on its own with **Open Example** in the Add Action window, or open `examples/math/ease_variable/ease_variable.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Variable Name	The name of the value to move. It is read once as the state is entered, and that reading is where the journey starts. After that it is written every frame. It can hold a number, a Vector2 or a Vector3, or a colour. Left empty nothing moves.	empty
To	The value to arrive at, of the same kind as the variable, whether a number, a Vector or a colour. Picking the variable makes it that kind, so a place eases to a place and a colour to a colour. Press Var to arrive at another variable's value. Var: any kind.	1.0
Seconds	How long the journey takes, starting at 1. At 0 it puts the value there and finishes straight away. That is a quick way to turn the easing off without unpicking anything. Var: a float.	1.0
Curve	The shape of the journey, the same curves Tween Property uses. Linear, as it comes, keeps the same speed all the way. Sine and Quad settle softly, and Cubic and Expo more sharply. Elastic, Back and Bounce go past the end and back, or bounce there. Choices: Linear, Sine, Quad, Cubic, Expo, Elastic, Back, Bounce.	Linear
Easing	Where the journey is slow. Ease Out, as it comes, arrives softly and suits most things. Ease In sets off softly, Ease In Out does both, and Ease Out In is quick at both ends. Linear takes no notice of it. Choices: Ease Out, Ease In, Ease In Out, Ease Out In.	Ease Out

Expression

Works out a formula written in one line, such as clamp(health - damage, 0, 100), and keeps the answer in a variable. The machine's variables are written by their names.



Expression as it appears in the Actions tab.

When to use it

Damage from strength and armour, strength * 2 - armour, takes one line here and two Math actions otherwise. If formulas are new to you, start with Math, Vector and Set Variable, one step at a time. Between them they can do everything this does. On Every Frame the formula is worked out again each frame and the answer follows the variables it reads.

> Godot's own sin() and cos() take radians rather than degrees, so use sine() and cosine() to stay in degrees.

See also

Math 46, **Vector** 53, **Set Variable** 49, **Set Text** 213

Try it

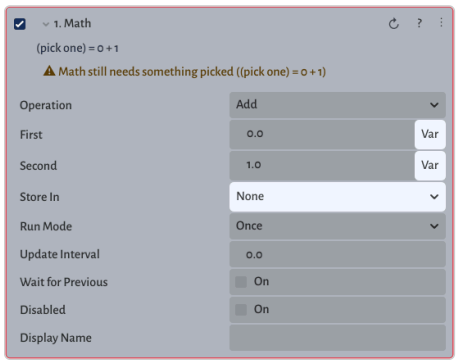
See the action work on its own with **Open Example** in the Add Action window, or open `examples/math/expression/expression.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Formula	The formula. Write a variable by its name, such as speed, or in braces as Set Text writes them, {time left}, which a name with a space in it needs. A variable named like one of Godot's functions, such as seed or min, needs braces too. + - * / and % (what is left after dividing) and brackets work as on paper. < > <= >= == != compare, while and, or and not combine true and false. choose(condition, a, b) gives a when the condition holds and b when it does not, because a formula cannot choose with if and else. delta is the seconds since the last frame, for anything that should grow by the second. In degrees there are sine(), cosine(), tangent(), asine(), acosine(), atangent(), angle_of(x, y) and turn_between(a, b). Godot's own work too, such as min, max, clamp, lerp, inverse_lerp, remap, smoothstep, move_toward, abs, sign, sqrt, pow, floor, ceil, round, snapped, pingpong, log, exp, randf_range, randi_range. float(text) and int(text) read a number from text and str() writes one. A Node variable's properties can be read with a dot, such as player.position or player.position.y. A Vector's can be read too, as in (target - here).length(). Var: a String.	empty
Store In	The variable the answer goes into. It keeps its kind, so an answer with a fraction going into an int is rounded, and an answer it cannot hold, such as text into a float, is not stored and a warning is printed. Left empty the answer is lost, with a warning.	empty

Math

Does one sum and keeps the answer in a variable, a name the machine remembers a value under. Either side of the sum can be typed in or read from another variable while the game runs.



Math as it appears in the Actions tab.

When to use it

A health bar's share is health Divide max health, kept in a separate variable for the bar to read. Adding a fixed amount to a value you already keep is one step with Set Variable. This is for when the two sides are different things, or the answer belongs somewhere else.

> Two whole numbers give a whole answer, and a score stays 110, not 110.0. Divide, Sqrt, Lerp and the angle sums are the exceptions and can give a number with a fraction.

See also

Set Variable 49, **Vector** 53, **Clamp** 42, **Random Number** 47

Try it

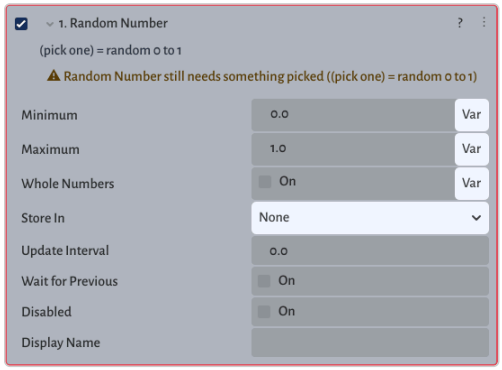
See the action work on its own with **Open Example** in the Add Action window, or open `examples/math/math/math.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Operation	What to do. Add, Subtract, Multiply and Divide are the four everyday sums. Min and Max give the smaller or the larger of the two. Pow raises the first to the second. Modulo gives what is left over after dividing, so a count Modulo 3 is 0 every third time. Lerp goes part of the way from the first to the second, and Move Toward steps the first toward the second without going past it. Wrap brings the first back into the range from 0 up to the second, so 370 in 360 is 10. Angle Difference is the shortest turn from one angle to another, in degrees. Sine, Cosine and Tangent take an angle in degrees, and Angle Of Direction gives the angle a direction points, with First as its X and Second as its Y. In a 2D game that is the same angle Rotate To's Angle takes, 0 to the right and 90 straight down the screen. Sqrt, Abs, Round, Floor, Ceil, Sign, Sine, Cosine and Tangent use only the first number. It starts at Add. Choices: Add, Subtract, Multiply, Divide, Min, Max, Pow, Abs, Round, Floor, Ceil, Modulo, Sqrt, Sign, Lerp, Move Toward, Wrap, Angle Difference, Sine, Cosine, Tangent, Angle Of Direction.	Add
First	The number on the left of the sum. Type one in, or press Var beside it and pick a variable to read while the game runs. Var: a float.	0.0
Second	The number on the right of the sum. Not shown for the sums that use only the first number. Dividing by zero, or taking the remainder with zero, gives 0 and the game goes on, with a warning printed once. Var: a float.	1.0
Store In	The name the answer is kept under. Left empty the answer is lost and a warning is printed. An empty Store In is the usual reason a sum seems not to run.	empty
Amount	Only for Lerp and Move Toward. For Lerp, how far along from the first to the second, where 0 is the first, 1 the second and 0.5 half way. For Move Toward, the size of the step.	—
Per Second	Only for Add, Subtract, Lerp and Move Toward on Every Frame. Tick it to work by the second, not by the frame, and a count, a speed or an easing runs the same on any computer. Add and Subtract add Second each second, Move Toward steps Amount each second, and Lerp closes that share of the gap each second.	—

Random Number

Puts a random number, between a lowest and a highest, into a variable.



Random Number as it appears in the Actions tab.

When to use it

A dice roll is Minimum 1 and Maximum 6 with Whole Numbers on. A plain one-in-so-many chance needs no number at all, as the Chance check answers yes or no by itself.

> Use Random Seed first if a run has to come out the same way twice.

See also

Random Seed 48, **Chance** 34, **Random Event** 29, **Array** 58

Try it

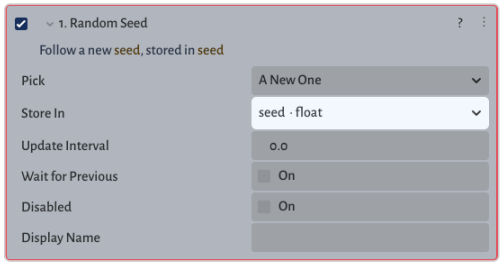
See the action work on its own with **Open Example** in the Add Action window, or open `examples/math/random_number/random_number.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Minimum	The lowest it can be. It starts at 0. Var: a float.	0.0
Maximum	The highest it can be, starting at 1. With Whole Numbers on, both ends can come out, so 1 to 6 really can give a 1 or a 6. Var: a float.	1.0
Whole Numbers	Off, where it starts, any number in between can come out, fractions and all. That is right for a wait or a speed. On, only whole numbers come out, each from the lowest to the highest as likely as the others, for a dice roll or a count of coins. Var: a bool.	off
Store In	The name the number is kept under. Left empty the number is lost, with a warning.	empty

Random Seed

Decides how chance comes out from here on. A seed is a starting number for chance, and the same seed makes every random roll, pick and chance in the game fall the same way, in the same order.



Random Seed as it appears in the Actions tab.

When to use it

A daily challenge gives everybody the same dungeon. This One with the day's seed, at the start of the run, builds it the same on every computer. A New One at the start of a normal game gives each player a different run. A seed kept from a run also lets you see a bug a second time.

> A new seed is picked between 1 and 999999, so a run code is always short enough for somebody to read out.

See also

Random Number 47, **Chance** 34, **Random Event** 29, **New Run** 293

Try it

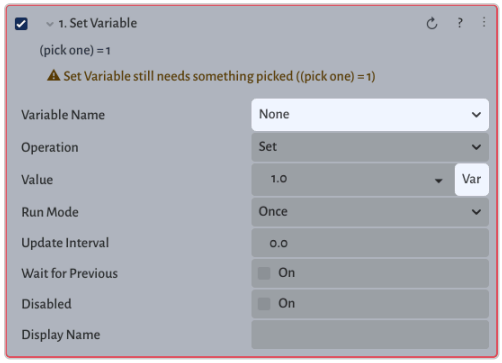
See the action work on its own with **Open Example** in the Add Action window, or open `examples/math/random_seed/random_seed.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Pick	A New One picks a fresh seed, different every time, for a normal new game. This One follows the seed given below, and the run comes out as it did before. It starts at A New One. Choices: A New One, This One.	A New One
Store In	The name the seed actually used is kept under. It starts as "seed". Show it on the screen so a player can write it down, or hand it to whatever builds the level. Left empty, the seed is still used but not kept.	seed
Use Seed	Only when Pick is This One. The whole number to follow. The same number always gives the same run, so keep the seed a player wants to replay and put it here.	—

Set Variable

Sets one of the machine's variables, adds to it, takes away from it, multiplies it, or flips it between true and false.



Set Variable as it appears in the Actions tab.

When to use it

How a machine keeps count, such as Add 1 to coins at each pickup or Subtract 1 from lives on a fall. Dividing, squaring and other sums beyond adding and multiplying need Math. On Every Frame with Per Second it grows a value steadily, such as an angle a Rotate To follows.

> An int stays an int. Adding a fraction to one rounds the answer, and a warning, given once, says what it was rounded to. Make the variable a float to keep fractions.

See also

Math 46, **Variable** 40, **Ease Variable** 44, **Get Event Data** 20

Try it

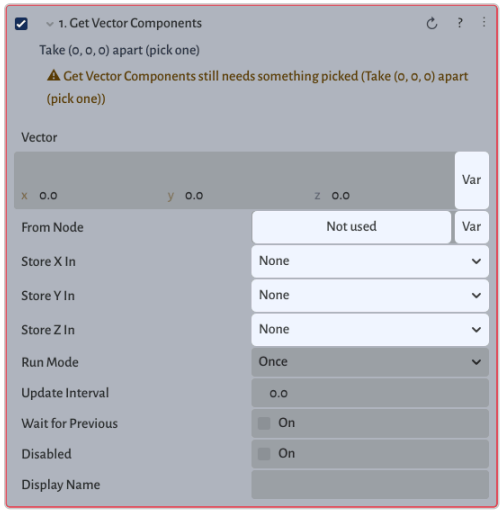
See the action work on its own with **Open Example** in the Add Action window, or open `examples/math/set_variable/set_variable.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Variable Name	The variable to change, picked from the template's list. A name nothing else knows about does no harm, but nothing will read it either. Left empty, nothing changes and a warning asks you to pick one.	empty
Operation	Set, as it comes, puts the value in. Add adds it, and a value below zero takes away. Subtract takes it away, and health loses the damage read from a variable this way. Multiply multiplies. Toggle flips true to false and back and takes no notice of Value at all. Choices: Set, Add, Multiply, Toggle, Subtract.	Set
Value	The value used by Set, Add, Subtract and Multiply, 1 to begin with. It is not only for numbers. A place, a direction and a colour can all be added or multiplied, and Set will hold text, a node, or a list. Var: any kind.	1.0
Per Second	Only for Add or Subtract on Every Frame. Ticked, Value is added, or taken away, per second. 90 adds ninety a second, and an angle, a timer or a speed changes at the same rate on any computer, however fast it draws.	—

Get Vector Components

Takes a Vector apart into plain numbers. Its X, its Y and, in a 3D game, its Z each go into a separate float variable.



Get Vector Components as it appears in the Actions tab.

When to use it

In a 3D game, how high the player stands is the Y of their position. Taken out here it becomes a plain number, ready for Math, a Check or a label.

See also

Set Vector Components 52, **Vector** 53, **Math** 46, **Get Property** 80

Try it

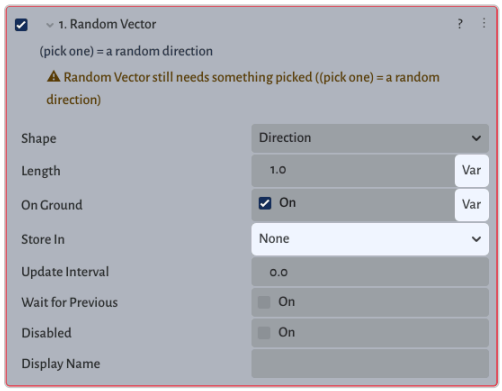
See the action work on its own with **Open Example** in the Add Action window, or open `examples/vector/get_vector_components/get_vector_components.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Vector	The Vector to take apart, two numbers in a 2D game and three in a 3D one. Most often you press Var beside it and pick a Vector variable. Var: a Vector2 or a Vector3, or a Node for where it stands.	0, 0, 0 in 2D: 0, 0
From Node	Point it at a node to take apart where that node stands. Left empty, as it comes, the Vector above is used. Var: a Node.	empty
Store X In	The float variable its X goes into. X is how far across. Empty keeps nothing.	empty
Store Y In	The float variable its Y goes into. Y is how far up in a 3D game, and how far down the screen in a 2D one. Empty keeps nothing.	empty
Store Z In	Only in a 3D game. The float variable its Z goes into. Z is how far toward the camera. Empty keeps nothing.	empty

Random Vector

Puts a random direction, or a random place inside a box or a circle, into a Vector variable. That is two numbers in a 2D game or three in a 3D one, going by the machine's object.



Random Vector as it appears in the Actions tab.

When to use it

A coin spilling from a chest can land In Circle around it with a Radius of 40, and Instantiate reads the place with Var. Move To, Apply Force and Set Velocity read the variable the same way, for an animal setting off to wander or a random push.

> A variable holding something other than a Vector is left as it was, and a warning says what it holds.

See also

Random Number 47, **Vector** 53, **Instantiate** 84, **Get Navigation Point** 237

Try it

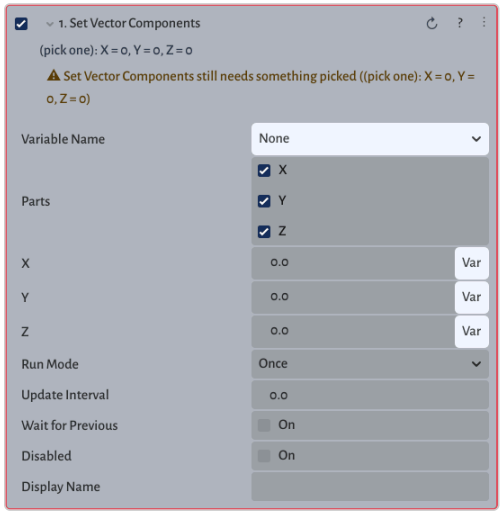
See the action work on its own with **Open Example** in the Add Action window, or open `examples/vector/random_vector/random_vector.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Shape	Direction gives a way to go. In Box gives a place anywhere in a box around Center. In Circle gives a place anywhere in a circle around Center. Choices: Direction, In Box, In Circle.	Direction
Length	Only with Direction. How long the direction is, such as 1 for a direction on its own, or a speed for a random push or a random velocity. Var: a float.	1.0
On Ground	Only in a 3D game, with Direction or In Circle. On, it stays flat across the ground, for a way to walk or a spot to stand on. Off, a direction can point up and down too, and a place is anywhere inside a ball. Var: a bool.	on
Store In	The Vector variable it goes into. A variable that already holds a Vector2 or a Vector3 keeps its kind, and a new one is made the kind the game needs.	empty
Center	Only with In Box and In Circle. The middle of the box or circle. Press Var to read it from a variable.	—
Center Node	Only with In Box and In Circle. Around this node, such as the thing itself or a spawn point, wherever it is when the action runs. Left empty, Center is used.	—
Size	Only with In Box. How big the box is from side to side, so every part of the place is up to half of it either way from the middle. In a 3D game a Size of 0 up and down keeps the place on the ground.	—
Radius	Only with In Circle. How far from the middle the place can be, at most. Places are spread evenly over the circle, not crowded in the middle.	—

Set Vector Components

Changes one or more parts of a Vector variable and leaves the rest as they are, or builds a Vector from plain numbers.



Set Vector Components as it appears in the Actions tab.

When to use it

To build a random place, fill two float variables with Random Number, then read them into X and Y here with Var. Each number can also be typed in. Adding to a whole Vector is done with Set Variable's Add.

See also

Get Vector Components 50, **Set Variable** 49, **Vector** 53, **Random Number** 47

Try it

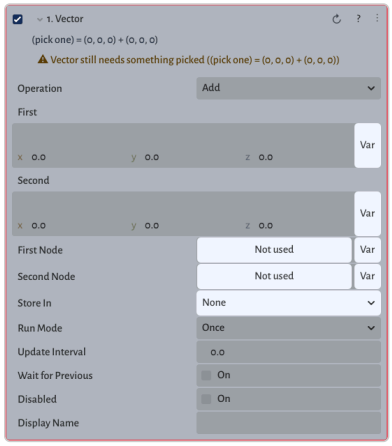
See the action work on its own with **Open Example** in the Add Action window, or open `examples/vector/set_vector_components/set_vector_components.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Variable Name	The Vector variable to change. A name no variable has yet is made for you, as a Vector2 in a 2D game and a Vector3 in a 3D one.	empty
Parts	Which parts to change. Tick X, Y and Z as you need them. The parts left unticked keep what they already hold. All three are ticked to begin with, which builds a whole Vector from the numbers below.	X, Y and Z
X	Only when X is ticked. The number X becomes (how far across). Var: a float.	0.0
Y	Only when Y is ticked. The number Y becomes (up in a 3D game, down the screen in a 2D one). Var: a float.	0.0
Z	Only when Z is ticked, in a 3D game. The number Z becomes (toward the camera). Var: a float.	0.0

Vector

Does sums with places and directions and keeps the answer in a variable. A place in a flat game is two numbers, across and down the screen, and in a 3D game it is three.



Vector as it appears in the Actions tab.

When to use it

A guard chasing the player needs the direction to them. Direction To, with Second Node on the player, gives it at a length of one, ready to hand to a movement action. Whether two things are within a certain distance is less work with the Distance check.

> If either side is a flat, two-number value the sum is worked out flat. Mixing a 2D place with a 3D one quietly throws the height away.

See also

Math 46, **Get Vector Components** 50, **Set Vector Components** 52, **Distance** 93

Try it

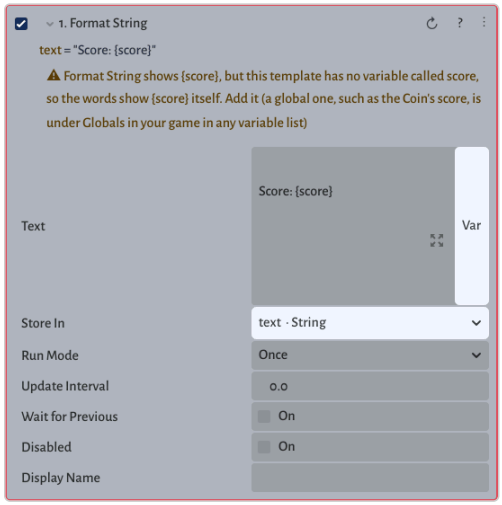
See the action work on its own with **Open Example** in the Add Action window, or open `examples/vector/vector/vector.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Operation	What to work out. Add puts two together and Subtract takes the second from the first. Scale makes the first longer or shorter by Amount, and Multiply, Divide, Min and Max work part by part. Normalized cuts the first down to a length of one. Length, Distance To, Angle To and Dot give a plain number for how long, how far apart, the angle between them from 0 to 180, and how much they point the same way. Direction To gives the direction from the first to the second. Cross gives the direction at right angles to both. Lerp goes part way from one to the other and Move Toward steps toward it. Limit Length makes it no longer than Amount, to hold a speed to its top. Project gives the part of the first that lies along the second, Reflect bounces it off a surface, Negate turns it round, and Rotated turns it Amount degrees. It starts at Add. Signed Angle To gives the turn from the first direction to the second, from -180 to 180 degrees, so you know which way. Direction To always gives a length of one. Choices: Add, Subtract, Scale, Normalized, Length, Distance To, Angle To, Direction To, Multiply, Divide, Min, Max, Dot, Cross, Lerp, Move Toward, Limit Length, Project, Reflect, Negate, Rotated, Signed Angle To.	Add
First	The value on the left. Type two numbers for a flat game or three for a 3D one, or press Var beside it and pick a variable holding a place. Var: a Vector2 or a Vector3, or a Node for where it stands.	0, 0, 0 in 2D: 0, 0
Second	The value on the right. Not shown for the sums that use only the first value. Var: a Vector2 or a Vector3, or a Node for where it stands.	0, 0, 0 in 2D: 0, 0
First Node	Point at a node here and the left-hand value is taken from wherever that node is standing, read afresh each time the sum runs, and the answer follows it as it moves. It wins over anything typed into First or read from a variable. Var: a Node.	empty
Second Node	The same for the right-hand value. Point it at the player, and a guard works out the direction to the player wherever the player has got to. Var: a Node.	empty
Store In	The name the answer is kept under. Left empty the answer is lost, with a warning. Length, Distance To and Angle To give a plain number, and the rest give a place or a direction.	empty
Amount	How much. It is the number to multiply by for Scale, 0 to 1 for Lerp, the size of the step for Move Toward, the longest it may be for Limit Length, and degrees for Rotated. Only shown for those. Press Var beside it to read it from a float variable.	—
Per Second	Only for Lerp, Move Toward and Rotated on Every Frame. Tick it to work by the second, not by the frame, and it runs the same on any computer. Move Toward steps Amount each second, Rotated turns Amount degrees each second, and Lerp closes that share of the gap each second.	—

Format String

Builds a line of text and keeps it in a variable, a name the machine remembers a value under. Anything written as {name} in the text is swapped for whatever that variable holds.



Format String as it appears in the Actions tab.

When to use it

"You found the {item}" becomes "You found the Brass Key", ready to show in a message box and to write into the player's diary. A line that only goes straight onto a label needs no separate step, since Set Text fills in {name} by itself. Build it here when the finished line is needed in more than one place or kept for later.

See also

Set Text 213, **Set Variable** 49, **Math** 46, **Get Property** 80

Try it

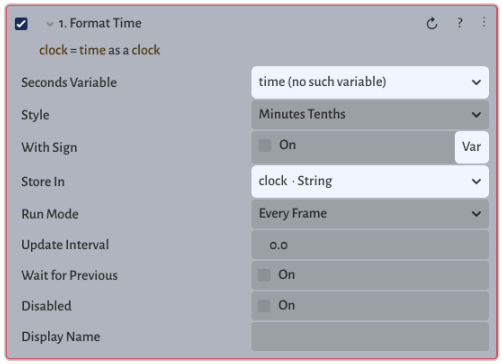
See the action work on its own with **Open Example** in the Add Action window, or open `examples/string/format_string/format_string.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Text	The text to build. It starts as "Score: {score}". Plain words come out as written, and {name} is swapped for whatever is kept under that name. After a colon you can say how a number should look, such as {time:mm:ss} as minutes and seconds, {time:h:mm:ss} with hours, {score:0000} padded out with zeros, {speed:0.0} with one decimal, {coins:0} as a whole number, and {volume:%} turns a share from 0 to 1 into a percentage. A Vector shows each of its parts that way, so {place:0} gives (100, 20). A list comes out as its items with commas between, and a table as each name with its value, such as potion: 2, key: 1. {target} gives the name of the node the Machine works on. A name that is no variable is left on the screen as you wrote it, braces and all, and a typo shows itself that way. Var: a String.	Score: {score}
Store In	The name the finished text is kept under. It starts as "text". Pick it from the list of the template's names, or make a new one from the bottom of that list. Left empty the text is lost, with a warning.	text

Format Time

Writes a number of seconds the way a clock shows it, 1:03.4 for 63.4, into a text variable for Set Text to put on screen.



Format Time as it appears in the Actions tab.

When to use it

A Timer counts in plain seconds, and 63.4321 on the screen means little to a player, so this turns it into 1:03.4. It starts on Every Frame and keeps a running clock up to date. The fraction is cut, not rounded, and the screen never shows a tenth that has not gone by yet.

> If the variable does not hold a number, it writes nothing and warns once.

See also

Timer 68, **Format String** 54, **Set Text** 213

Try it

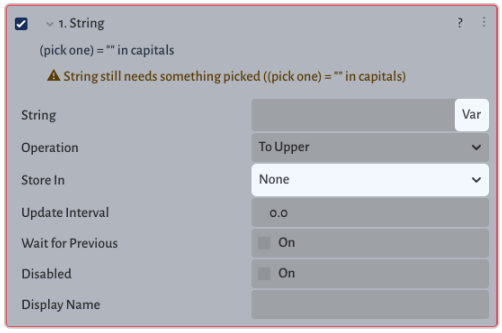
See the action work on its own with **Open Example** in the Add Action window, or open `examples/string/format_time/format_time.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Seconds Variable	The variable holding the seconds, such as the one a Timer counts in, or any number.	time
Style	How the time is written. Minutes Seconds gives 1:03. Minutes Tenths, the default, gives 1:03.4, as a rally clock reads, and Minutes Hundredths gives 1:03.45 for a lap timer. Seconds Tenths and Seconds Hundredths leave the minutes off and count every second, so 63 seconds is 63.4 or 63.45. Hours Minutes Seconds gives 0:01:03 for something long. Choices: Minutes Seconds, Minutes Tenths, Minutes Hundredths, Seconds Tenths, Seconds Hundredths, Hours Minutes Seconds.	Minutes Tenths
With Sign	Put + or - in front, for a gap to a rival's time or a split against the best. Off writes the time plain, and a time below nought still gets its minus. Var: a bool.	off
Store In	The text variable the clock goes into. Set Text shows it with {clock}, or whatever the variable is called. Left empty there is nowhere to put the time, and a warning is printed.	clock

String

Works with text. It counts its characters, takes the first or last few, cuts a piece out, swaps one piece for another, splits it into a list, joins a list into one text, puts it all in capitals or small letters, trims the spaces off its ends, or finds where a piece is in it. The answer goes into a variable.



String as it appears in the Actions tab.

When to use it

A line such as sword,shield,potion becomes a list of three with Split, and a code the player typed can be checked once Strip Edges has taken off its stray spaces. Putting text together from variables, such as Score: 10, is simpler with Format String. Asking whether text matches, begins or ends with something is a String Compare check.

See also
Format String 54, **String Compare** 57, **Array** 58, **Set Text** 213

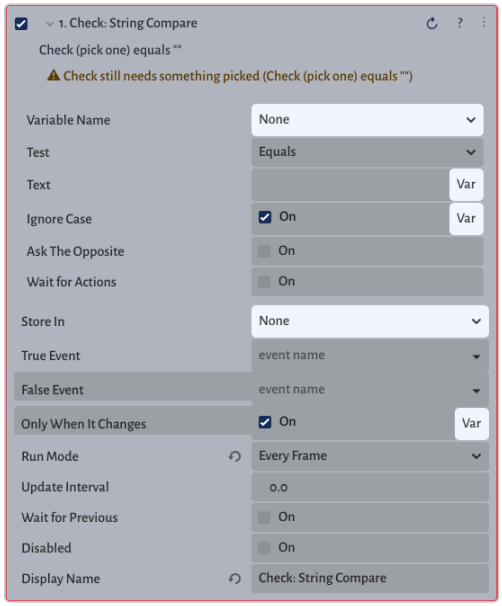
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/string/string/string.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
String	Not shown for Join. The text to work on, typed in. Press Var to read it from a variable, such as the name the player typed, as most uses do. Var: a String.	empty
Operation	What to do. Length counts its characters. Left and Right take the first or last few. Substr takes a piece from a place. Replace swaps every What for For What. Split cuts it into a list at every Delimiter, and Join puts a list together with Delimiter between the items. To Upper and To Lower put it in capitals or small letters. Strip Edges takes spaces, tabs and new lines off both ends. Find says where What first is. It starts at To Upper, and only the settings the one picked uses are shown. Choices: Length, Left, Right, Substr, Replace, Split, Join, To Upper, To Lower, Strip Edges, Find.	To Upper
Store In	The variable the answer goes into. It gets text, a whole number for Length and Find, or a list for Split. It may be the same variable the text came from, to change it where it is. Left empty the answer has nowhere to go, and a warning says so.	empty
Array Variable	Only when Operation is Join. The variable holding the list to put together. Each item is written as a line shows it, so 3, not 3.0, and a node by its name. A variable holding no list gets a warning in the Output panel.	—
Length	Only when Operation is Left, Right or Substr. How many characters, starting at 3. For Left and Right a negative number takes all but that many from the other end. For Substr it is how long the piece is, and -1 runs to the end.	—
From	Only when Operation is Substr or Find. Where to start, counting from 0, so 0 is the first character. For Find it is where to start looking.	—
What	Only when Operation is Replace or Find. The text looked for. Left empty there is nothing to look for, and the Output panel warns about it.	—
For What	Only when Operation is Replace. What goes in its place. Left empty the text looked for is taken out.	—
Ignore Case	Only when Operation is Replace or Find. On, A and a count as the same letter. Off, the default, they are different.	—
Delimiter	Only when Operation is Split or Join. The text that separates the items, starting as a comma. For Split, empty cuts between every character.	—
Allow Empty	Only when Operation is Split. On, the default, two delimiters side by side give an empty item between them, so a,,b gives three items. Off leaves the empty ones out and gives two.	—
Not Found Event	Only when Operation is Find. The event sent when What is not in the text, while Store In is given -1. Left empty nothing is sent.	—

String Compare CHECK

Asks how the text in a variable compares with something you type. Is it the same, does it contain a word, does it start or end with one, or is there nothing in it at all?



String Compare as it appears in the Actions tab.

When to use it

For words, not numbers, such as the name of the room the player is in, or a password typed into a box. Text that lives on a node, not in a variable, is read with Property.

> The answer comes from a variable and typed-in text alone, so with Only When It Changes ticked it is asked again only when that variable is written.

See also

Variable 40, **Compare** 35, **Format String** 54, **Set Variable** 49

Try it

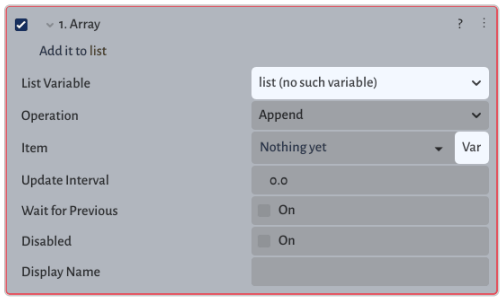
See the action work on its own with **Open Example** in the Add Action window, or open `examples/string/string_compare/string_compare.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Variable Name	The variable holding the text. A variable holding a number or a node is read as the text you would see written down, and one holding nothing at all reads as empty text.	empty
Test	How to compare. Equals, the default, and Not Equals compare the whole text. Contains looks for your text anywhere inside it, and Starts With and Ends With at one end. Is Empty asks whether the variable is empty and ignores the Text box. Choices: Equals, Not Equals, Contains, Starts With, Ends With, Is Empty.	Equals
Text	What the variable is compared with. Is Empty ignores it. Left empty with Contains, Starts With or Ends With, everything matches. Var: a String.	empty
Ignore Case	On by default, so "Red", "red" and "RED" all count as the same word. Untick it where capitals matter, such as a code that has to be typed letter for letter. Var: a bool.	on
Ask The Opposite	Turns the answer round into "it is not that", with no second check. It turns round the true-or-false answer the Check stores as well, which swapping the True Event and False Event over cannot do.	off
Wait for Actions	Holds the question back until every other action in the state has finished, so it reads text the state's own actions have just written. Left off, the default, it is asked as soon as the state runs.	off
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Array

Works with a list (several values kept in order in one variable). You can add to it, read from it, sort it, shuffle it, deal from it and more. An empty list is made for you if the variable holds nothing yet.



Array as it appears in the Actions tab.

When to use it

A card game keeps its deck as a list. Shuffle it at the start, then Move Item deals from the deck into a hand. For Each walks through a list one item at a time. A value kept against each of several names, such as how many of each item the player carries, belongs in a Dictionary.

> Sorting a mixed list is safe. Numbers come before words, and numbers go by size, so 2 comes before 10, not after it. Shuffle follows Random Seed, and the same seed deals the same order again.

See also
Dictionary 59, **For Each** 36, **Set Variable** 49, **Random Seed** 48

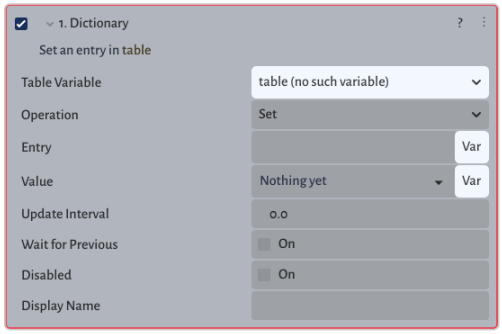
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/array/array/array.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
List Variable	The name the list is kept under. It starts as "list". Everything this action does, it does to that one list. Pick a name from the template's list of names, or make a new one from the bottom of that list. A name that holds nothing yet becomes an empty list the first time this runs, ready to add to. A name holding something that is not a list, such as a number, is started again as an empty list too, with a warning.	list
Operation	What to do. Append puts the item on the end. Erase takes out the first one that matches. Get reads the one at a given place. Size counts them. Pick Random reads one at random. Clear empties the list. Has says whether it is in there. Insert puts an item in at a place and moves the rest along. Set changes the one at a place. Remove At takes out the one at a place. Find says where an item is. Sort puts them in order smallest or A first, and Sort Descending largest or Z first. Shuffle puts them in a random order, as a deck of cards is shuffled. Append Array puts every item of another list on the end. Move Item takes the one at a place out and puts it on the end of another list, as a card is dealt from the deck into a hand. Resize makes the list a given length. Reverse turns the order round, so the last is first. It starts at Append, and the settings that mean nothing for the one you pick are taken off the panel. Choices: Append, Erase, Get, Size, Pick Random, Clear, Has, Insert, Set, Remove At, Find, Sort, Sort Descending, Shuffle, Append Array, Move Item, Resize, Reverse.	Append
Item	The item for Append, Erase, Has, Insert, Set and Find. It can be a number, a tick, some words, a place or a node. Press Var beside it to take the item from a variable, such as the enemy that was just spotted. Var: any kind.	empty
Index	Only when the Operation is Get, Insert, Set, Remove At or Move Item. Which place in the line, counting from 0, so the first item is 0 and the third is 2. A negative number counts back from the end, so -1 is the last one. A place past either end finds no item and sends the Empty Event. Insert one past the last is the exception and puts the item on the end.	—
Other Array Variable	Only when the Operation is Append Array. The variable holding the other list, whose items are all put on the end of this one, in their order, and the other list is left as it was. A variable holding no list adds nothing and prints a warning.	—
Store Moved Item In	Only when the Operation is Move Item. The variable holding the list the item is put on the end of, such as the hand a card is dealt into. It is made for you if it holds nothing yet. Picking the same list as List Variable moves the item to the back of that same list. Left empty, or holding something that is not a list, no item is moved and a warning is printed.	—
New Size	Only when the Operation is Resize. How many items the list is to hold, starting at 0. A bigger number adds empty places on the end, which stay empty until something is put there, and a smaller one takes items off the end. Less than 0 empties it, with a warning.	—
Store In	Only when the Operation is Get, Size, Pick Random, Has or Find. The name the answer is kept under. It holds the item itself for Get and Pick Random, how many for Size, a yes or no for Has, and for Find the place the item sits at, or -1 when it is not there. Left empty the answer is lost and a warning is printed, unless Empty Event is filled in (that event may be all you need). Size always warns.	—
Empty Event	Only when the Operation is Get, Pick Random, Has, Find, Set, Remove At or Move Item. The event sent when what you asked for is not there, such as a place past either end, an empty list to pick from, or an item the list does not hold. Left empty nothing is sent, and that is fine when the answer in Store In is enough by itself.	—

Dictionary

Works with a Dictionary, a table of entries kept in one variable. Each entry is a name with a value beside it, such as potion: 3.



Dictionary as it appears in the Actions tab.

When to use it

The player's bag can be a Dictionary under "carrying". Add with potion and 1 gives one more potion, and works even before the first potion is there. Things with an order and no names belong in an Array. A high-score table, where one name can be on it twice and only the best ten are kept, is High Score Table's job.

> Add keeps a count whole, so a stack of potions reads 3, not 3.0.

See also

Array 58, **High Score Table** 60, **For Each** 36, **Set Variable** 49, **Give Item** 290

Try it

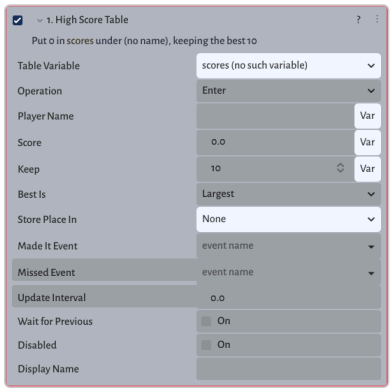
See the action work on its own with **Open Example** in the Add Action window, or open `examples/array/dictionary/dictionary.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Table Variable	The name the table is kept under. It starts as "table". Everything this action does, it does to that one table, so an inventory kept under "carrying" is reached by every action that names it. Pick a name from the template's list of names, or make a new one from the bottom of that list. A name that holds nothing yet is turned into an empty table for you the first time this runs.	table
Operation	What to do. Set gives the named entry a value, putting the entry in if it was not there. Get reads one. Has asks whether it is there. Erase takes it out. Size counts the entries. Keys makes a list of all the entries' names, ready for For Each to walk through. Keys By Value Ascending and Keys By Value Descending make that list in order of the entries' values, the smallest or the largest at the top, for a race's results with the fastest time first, or a high-score table with the best score first. Clear empties the table. Add adds a number to the entry, starting from 0 when it is not there yet. It starts at Set, and the settings that mean nothing for the one you pick are taken off the panel. Choices: Set, Get, Has, Erase, Size, Keys, Clear, Add, Keys By Value Ascending, Keys By Value Descending.	Set
Entry	The name of the entry, for Set, Get, Has, Erase and Add, such as coins, red_key or level_3. Spell it the same way every time, because a different spelling is a different entry and the table will quietly grow two of them. Var: a String.	empty
Value	The value Set gives the entry, or the number Add adds to it. A negative number takes some away, so -10 is ten fewer coins. Press Var beside it to read the value from a variable. Var: any kind.	empty
Store In	Only when the Operation is Get, Has, Size, Keys or Add. The name the answer is kept under. It holds the entry's value for Get, a yes or no for Has, how many entries for Size, a list of their names for Keys, and for Add the entry's new total, saving a read back. Left empty the answer is lost and a warning is printed, except for Add, and for a Get or a Has with Missing Event filled in, where that event may be all you need.	—
Missing Event	Only when the Operation is Get, Has or Erase. The event sent when there is no entry of that name. For example, a door asking Has red_key hears it when the player has no key. Left empty nothing is sent.	—
Remove at Nothing	Only when the Operation is Add. Ticked, the entry is taken out when its total reaches 0 or less, so a potion whose count has run out is no longer listed or counted. The new total still goes into Store In. Off, as it starts, the entry stays at 0.	—

High Score Table

Keeps a table of best scores in a variable, like an arcade game's top ten, with each run's name and score, best first.



High Score Table as it appears in the Actions tab.

When to use it

At the end of a run, ask Would Enter with the run's score. If it makes the table, show a text box for a name, Enter it once it is typed, then Write Out into a variable and show it with Set Text. Two runs under the same name are two lines, as on any arcade table, and only the best few are kept, so nothing needs tidying afterwards.

> The table is a list of names and scores, so Save Game writes it into its file like any other variable, and Save Game's Load brings it back. Anything else found under the table's name is left out and does not break it.

See also

Dictionary 59, **Save Game** 235, **Set Text** 213, **Array** 58

Try it

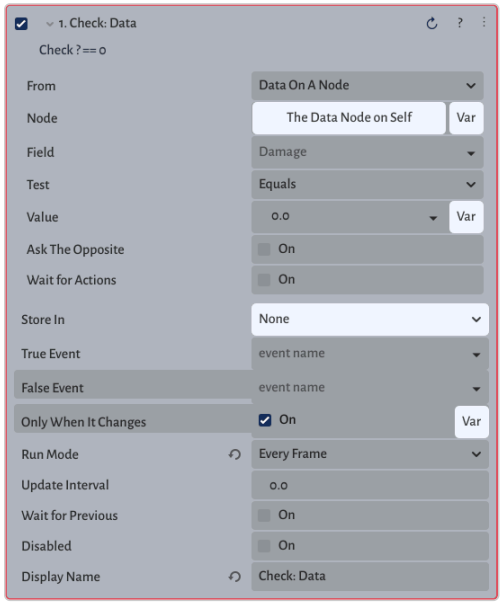
See the action work on its own with **Open Example** in the Add Action window, or open `examples/array/high_score_table/high_score_table.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Table Variable	The name the table is kept under. It starts as "scores". A game with several levels or arenas can keep one table for each under a different name. A name that holds nothing yet is taken as an empty table.	scores
Operation	What to do. Enter puts the name and score in, in its place, and anything pushed off the bottom goes. Would Enter asks whether a score would make the table without putting it in, to decide whether to ask for a name. Write Out writes the table as text, a line for each place. Clear empties it. It starts at Enter, and the settings that mean nothing for the one you pick are taken off the panel. Choices: Enter, Would Enter, Write Out, Clear.	Enter
Player Name	Only for Enter. The name the score goes in under. Press Var to read it from a variable, such as the text the player typed. Var: a String.	empty
Score	Only for Enter and Would Enter. The score itself. Press Var to read it from a variable such as the run's score. A whole number is kept whole, so the table reads 1200, not 1200.0. Var: a float.	0.0
Keep	Only for Enter and Would Enter. How many places the table keeps, starting at 10. A score that does not beat the last of them does not go in. Var: an int.	10
Best Is	Only for Enter and Would Enter. Largest for points, where more is better, and Smallest for times, where quicker is better. It starts at Largest. A score that ties one already there goes in below it, since a new score only takes a place from one it beats. Choices: Largest, Smallest.	Largest
Store Place In	Only for Enter and Would Enter. The name the place is kept under, counting the best as 1. It is where the score went in, or would go, ready for a banner saying New high score, third place. It is 0 when the score does not make the table. Left empty the place is not kept.	empty
Made It Event	Only for Enter and Would Enter. The event sent when the score makes the table. Left empty nothing is sent.	empty
Missed Event	Only for Enter and Would Enter. The event sent when it does not. Left empty nothing is sent.	empty
Line	Only for Write Out. How each line is written. {place}, {name} and {score} are filled in, so the usual "{place}. {name} {score}" reads 1. ANNA 12400.	—
When Empty	Only for Write Out. The text written when the table has nothing in it yet. It starts as "No scores yet".	—
Store Text In	Only for Write Out. The name the text is kept under, ready for Set Text to show with {that name} in its words. Left empty the text is lost and a warning is printed.	—

Data CHECK

Asks a question about one field of a Data record, such as whether a weapon's Rarity is Legendary.



Data as it appears in the Actions tab.

When to use it

Goes in a transition when the answer lives in Data, such as going to Dead once the enemy's Health is 0 or less. Questions about a variable belong to Compare or Variable.

> With no record found, or no such field, the answer is no and a warning gives the reason.

See also
Get Data 63, **Set Data** 64, **Find Row** 62

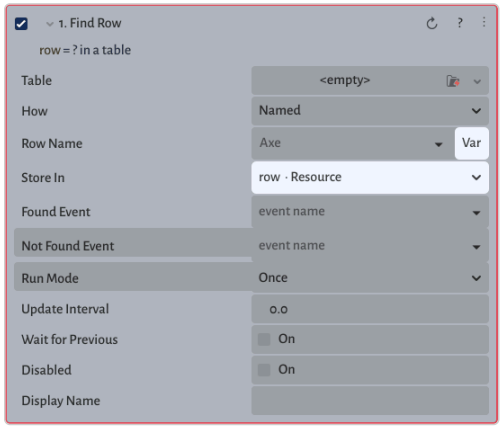
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/data/data/data.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
From	The record can be in A Data File, in A Variable holding one (such as the row Find Row found), or in Data On A Node (the Data node under a node). Data On A Node is the default. Choices: A Data File, A Variable, Data On A Node.	Data On A Node
Node	Only when From is Data On A Node. The node whose Data node is asked about. Leave it empty for the node this machine works on. Var: a Node.	Self
Field	The field to ask about, picked from the list of fields the record has.	empty
Test	How to compare. The choices are Equals, Not Equals, Greater, Greater Or Equal, Less, Less Or Equal, or Is True and Is False for an On/Off field. Choices: Equals, Not Equals, Greater, Greater Or Equal, Less, Less Or Equal, Is True, Is False.	Equals
Value	What the field is compared with. Not shown for Is True and Is False, which need nothing to compare. Var: any kind.	0.0
Ask The Opposite	Turns the answer round, so Equals becomes "it is not that". The true-or-false answer the Check stores is turned round too, and swapping the True Event and False Event over cannot do that.	off
Wait for Actions	Holds the question back until every other action in the state has finished, so it reads a field the state's own actions have just set or found. Left off, the default, it is asked as soon as the state runs.	off
Data	Only when From is A Data File. The Data record to ask about.	—
Data Variable	Only when From is A Variable. The variable holding the record, such as the row Find Row found.	—
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Find Row

Picks one row of a Data Table and keeps it in a variable for Get Data, Set Data and the Data check to work on.



Find Row as it appears in the Actions tab.

When to use it

Many things of one kind, such as weapons, levels or enemy types, sit best in a table. Pick the row once (a Random one for the next enemy to spawn, say), keep it in a variable and read its fields with Get Data. A single record needs no table. Point Get Data at the Data file itself.

> When no row is found the variable keeps whatever it held before. If a missing row matters, send Not Found Event to a state that deals with it.

See also
Get Data 63, **Set Data** 64, **Data** 61

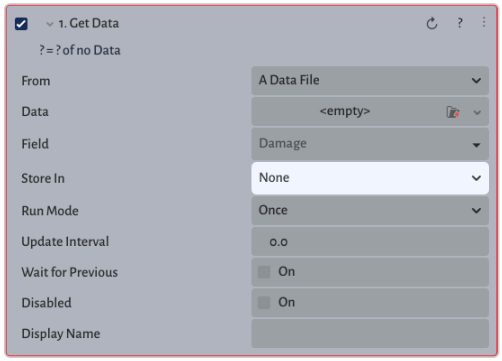
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/data/find_row/find_row.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Table	The Data Table to look in, made with Create New in the FileSystem dock and filled in as a grid in the NodeWorthy panel. With none picked, the action prints a warning.	empty
How	How the row is picked. Named finds the row with that name. At Place takes the row at that place, counting from 1. Random takes any row. Where Field Is takes the first row whose field holds a value. Highest and Lowest take the row with the biggest or smallest number in a field. Choices: Named, At Place, Random, Where Field Is, Highest, Lowest.	Named
Row Name	Only when How is Named. The row's name as it is written in the table's Name column. Capitals count. Var: a String.	empty
Store In	The variable the row goes into, which Get Data, Set Data and the Data check then read with From set to A Variable.	row
Found Event	Sent when a row is found, carrying the row. Leave it empty if nothing needs to know.	empty
Not Found Event	Sent when no row is found. That can be no row by that name, a place past the end, or no row holding that value. Leave it empty if nothing needs to know.	empty
Place	Only when How is At Place. The row's place in the table, 1 for the top row. A place past the last row finds nothing.	—
Field	Only when How is Where Field Is, Highest or Lowest. The field the rows are compared by, picked from the table's fields.	—
Equals	Only when How is Where Field Is. What the field must hold. The first row from the top that holds it is picked.	—

Get Data

Copies one field of a Data record into a variable, where the actions after it can use it.



Get Data as it appears in the Actions tab.

When to use it

Handy when the numbers and names about your game's things live in Data and not in each template. Just before a sword hits, read its Damage into a number variable. When a level starts, read its Time Limit.

> When the record or the field cannot be found, the variable keeps its old value. If a value never seems to change, look in the Output panel for a warning.

See also

Set Data 64, **Find Row** 62, **Data** 61

Try it

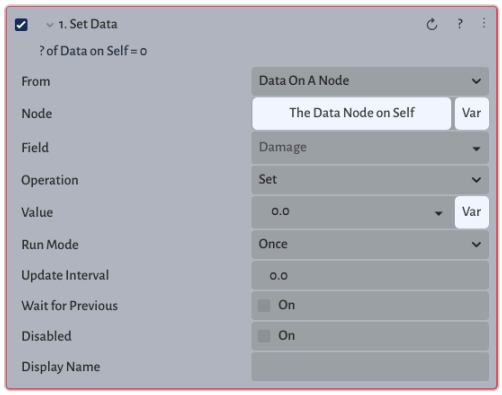
See the action work on its own with **Open Example** in the Add Action window, or open `examples/data/get_data/get_data.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
From	Where the record is. A Data File is one Data record made with Create New in the FileSystem dock. A Variable is a variable holding a record, such as the row Find Row found. Data On A Node is the Data node under a node, added with NodeWorthy > Give It > Data Node on its right-click menu. Choices: A Data File, A Variable, Data On A Node.	A Data File
Data	Only when From is A Data File. The Data record to read, picked from the FileSystem. With none picked, the action changes nothing and prints a warning.	empty
Field	The field to read, picked from the list of fields the record's Data Definition has. Ask for a name the record has not got and the warning lists the fields it does have.	empty
Store In	The variable the field's value goes into. Pick one of the template's variables of the same kind as the field, so a number variable for a number field and a text variable for text.	empty
Data Variable	Only when From is A Variable. The variable holding the record, most often the one Find Row keeps its row in. If it is still empty, or holds something other than a Data record, the action warns and changes nothing.	—
Node	Only when From is Data On A Node. The node whose Data node is read. Leave it empty for the node this machine works on, as a thing reading its own Data does.	—

Set Data

Changes one field of a Data record while the game runs, either setting it to a value or adding to a number. Each hit can take 1 off this enemy's Health.



Set Data as it appears in the Actions tab.

When to use it

Suits values that belong to one thing and change during play, like the number of times a chest has been opened. A value every part of the game shares is simpler as a global variable changed with Set Variable.

> Changes made to a Data file while playing are not saved back into it, so the next run starts from the values written in the file. Give each thing a separate Data node for values that change.

See also

Get Data 63, **Data** 61, **Find Row** 62

Try it

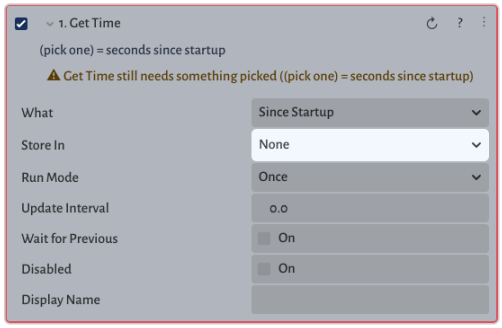
See the action work on its own with **Open Example** in the Add Action window, or open `examples/data/set_data/set_data.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
From	The record can be in A Data File, in A Variable holding one (such as the row Find Row found), or in Data On A Node (the Data node under a node). Data On A Node is the default, since most often a thing changes its own values. Choices: A Data File, A Variable, Data On A Node.	Data On A Node
Node	Only when From is Data On A Node. The node whose Data node is changed. Leave it empty for the node this machine works on. Var: a Node.	Self
Field	The field to change, picked from the list of fields the record has.	empty
Operation	Set makes the field the value. Add adds the value to a number field, and a negative value takes away. Add on a field that is not a number changes nothing and prints a warning. Choices: Set, Add.	Set
Value	What the field becomes, or what is added to it. It must suit the field, for example a number for a number field or one of its choices for a Choice field. A value that does not suit it leaves the field as it was, with a warning. Var: any kind.	0.0
Data	Only when From is A Data File. The Data record to change. Every machine using that file sees the change until the game stops.	—
Data Variable	Only when From is A Variable. The variable holding the record to change, such as the row Find Row found.	—

Get Time

Puts a time into a variable. It can be how long the game has been running, how long this scene or this state has been going, the time of day on the player's own clock, or how long the last frame took.



Get Time as it appears in the Actions tab.

When to use it

To time a race, store Since Startup as the race starts and again at the line, and take one from the other. Format Time can then show the seconds as minutes and seconds. A countdown or a stopwatch that must stop while the game is paused belongs to Timer.

> Since Scene Started counts from when the scene was added. A design loaded by name only after its scene began cannot know that, so it warns about it and counts from when it was loaded.

See also

Timer 68, **Format Time** 55, **Timeout** 67, **Set Time Scale** 66

Try it

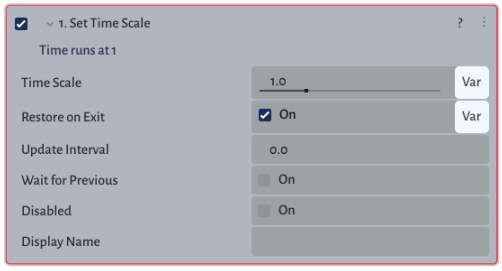
See the action work on its own with **Open Example** in the Add Action window, or open `examples/time/get_time/get_time.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
What	Which time to read. Since Startup is the seconds since the game started, and Since Scene Started the seconds since the scene this machine is in started. Both are real seconds, which carry on while the game is paused and are not slowed by Set Time Scale. In This State is the seconds this state has been running, by the machine's own clock, which stops while the game is paused and follows Set Time Scale. System Time is the time of day on the player's computer. Delta is the seconds the last frame took, which is 0 while Set Time Scale has time stopped. Choices: Since Startup, Since Scene Started, In This State, System Time, Delta.	Since Startup
Store In	The variable the time goes into. That is a float of seconds, or for System Time the time of day as text, such as 14:05:09. An int variable keeps whole seconds. With System Time it may be left empty when only the hour, minute or second below is wanted.	empty
Store Hour In	Only when What is System Time. The int variable the hour goes into, from 0 to 23, and a day and night that follows the player's clock reads this. Empty, the default, keeps nothing.	—
Store Minute In	Only when What is System Time. The int variable the minute goes into, from 0 to 59. Empty, the default, keeps nothing.	—
Store Second In	Only when What is System Time. The int variable the second goes into, from 0 to 59. Empty, the default, keeps nothing.	—

Set Time Scale

Speeds the game up or slows it down. 1 is normal speed, 0.5 is slow motion, and 0 freezes everything that follows the clock.



Set Time Scale as it appears in the Actions tab.

When to use it

For a moment of drama, drop it to 0.05 for a hit stop as a blow lands, or to 0.3 as the last enemy falls. A pause with a menu on top is Pause Game's job, because it stops the game outright and keeps the menu working.

> It changes the game's clock, not one node. A slow-motion state slows the menus, the music timing and every waiting action along with the fight.

See also

Pause Game 222, **Wait** 69, **Timer** 68, **Freeze** 111

Try it

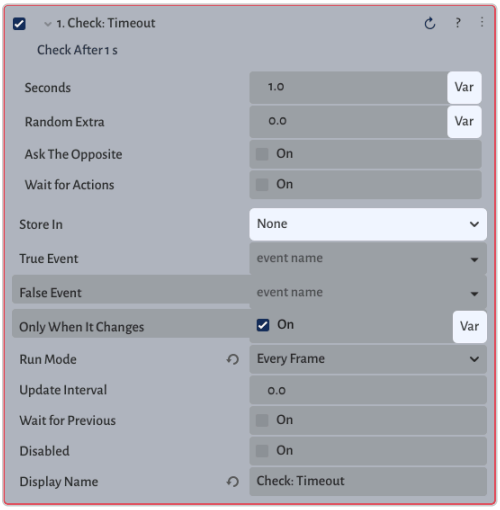
See the action work on its own with **Open Example** in the Add Action window, or open `examples/time/set_time_scale/set_time_scale.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Time Scale	How fast time runs, from 0 to 4. 1 is normal speed. 0.3 is a clear slow motion, 0.05 is an almost-frozen hit stop, and 2 runs everything at double speed. 0 freezes everything that follows the clock. Press Var to read it from a variable, so a slider or an ability can set it. Var: a float.	1.0
Restore on Exit	On, the speed goes back as soon as the state is left, and there is no undo to forget. If two slowed states overlap, the game runs at the speed of the newest one still in force and goes back to normal only once the last has ended. Off leaves the new speed in place until something else changes it, and the game can easily be left stuck at the wrong speed. Var: a bool.	on

Timeout CHECK

Asks whether the state has been running long enough. It says yes once so many seconds have passed since the state was entered.



Timeout as it appears in the Actions tab.

When to use it

A stunned enemy recovers after 2 seconds, and a door closes again a few seconds after opening. It moves the machine out of the state. To hold up one action and stay, use Wait.

> It measures time in the state, not time in the game. Leave the state and come back, and the count starts again from 0.

See also

Wait 69, **Timer** 68, **Chance** 34, **Repeat** 37

Try it

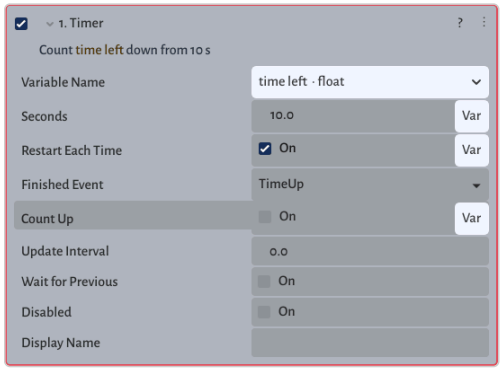
See the action work on its own with **Open Example** in the Add Action window, or open `examples/time/timeout/timeout.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Seconds	How long the state must have been running before the answer is yes. The default is 1. The clock starts when the state is entered, and starts afresh every time the state is entered again. Var: a float.	1.0
Random Extra	Up to this many extra seconds on top, picked anew each time the state is entered. At the default 0, every visit lasts the same time. With 1 or 2 here, a row of enemies entering the same state stop marching in step. Var: a float.	0.0
Ask The Opposite	Turns the answer round into "it has not been running that long yet", true from when the state is entered until the time is up. It turns round the true-or-false answer the Check stores as well, which swapping the True Event and False Event over cannot do.	off
Wait for Actions	Holds the question back until every other action in the state has finished, so a state does not time out in the middle of its own work. Left off, the default, it is asked as soon as the state runs.	off
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Timer

Counts a variable down in seconds and sends an event when it reaches 0. With Count Up ticked it counts up from 0, like a stopwatch. The number is kept in a variable for the screen to show and checks to ask about.



Timer as it appears in the Actions tab.

When to use it

Time as part of the rules. A bomb's fuse of 5 seconds sends TimeUp at 0, and a Set Text can show the seconds left as it burns. It follows the game's speed. Slow motion slows it, and a paused game holds it still.

> Counting up with Seconds at 0 there is no end, so the action never finishes and Finished Event never goes. Give that state another way out.

See also

Wait 69, **Timeout** 67, **Set Text** 213, **Send Event** 31

Try it

See the action work on its own with **Open Example** in the Add Action window, or open `examples/time/timer/timer.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Variable Name	The variable the time is kept in, time left to begin with. Two counts running together need a name each. It is an ordinary variable, so a Set Text can show it and a check can ask whether it is low. Left empty it still counts, and a warning says the time is kept nowhere to show or check.	time left
Seconds	How many seconds it starts at, 10 to begin with. Counting up, it is where the count stops and the event goes. At 0 a count up never stops. Var: a float.	10.0
Restart Each Time	On to begin with, so the clock starts afresh at Seconds every time the state is entered. Turn it off for a fuse that must keep burning while the machine moves through several states. It then carries on from the time that was left, and starts at Seconds only when there is none. Var: a bool.	on
Finished Event	The event sent when the count runs out, TimeUp to begin with. It is sent once, not every frame afterwards. Nothing happens unless the state you are in, or a state it sits inside, has a transition waiting for that same name.	TimeUp
Count Up	Off to begin with, counting down. Ticked, it counts up from 0, for how long something has taken, and stops at Seconds with Finished Event. Var: a bool.	off

Wait

Waits a number of seconds and then finishes. Alone in a state with a transition on FINISHED, it makes the simplest delay.



Wait as it appears in the Actions tab.

When to use it

Half a second between one shot and the next, or three seconds on a title card before the menu appears. A countdown the player can see needs Timer, which keeps the time in a variable. Wait costs nothing while it counts, so a hundred machines waiting are as cheap as none.

> The wait holds still whenever its machine does, so a wait inside a machine that is paused or switched off carries on when it starts again.

See also
Timer 68, **Next Frame Event** 28, **Timeout** 67, **Set Time Scale** 66

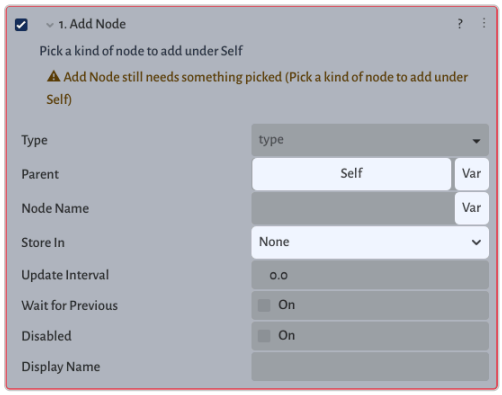
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/time/wait/wait.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Seconds	How long to wait, 1 second to begin with. It counts the game's own time, so it stands still while the game is paused and carries on afterwards from where it was. Var: a float.	1.0
Random Extra	Up to this many extra seconds on top, drawn afresh each time, 0 to begin with. Half a second here stops a row of enemies firing in step with one another. Var: a float.	0.0
Real Time	Off to begin with, counting the game's time. Tick it to count real seconds whatever speed the game is running at. You need it for a hit stop, where the game is frozen with Set Time Scale at 0, because a wait on the game's own time would never end there. Var: a bool.	off

Add Node

Adds a new node of one of Godot's kinds, such as a Timer, under a node you pick while the game runs. It arrives with Godot's default settings and can be kept in a variable so the actions after it can set it up.



Add Node as it appears in the Actions tab.

When to use it

Give a thing a part it did not start with, such as a Timer to count down a fuse, or a light that comes on once a switch is thrown. This adds one plain node. A whole scene you have made, such as a bullet or an enemy, is added with Instantiate.

> A Timer added this way is not counting yet, so call its start with Call Method on the node kept in Store In.

See also

Instantiate 84, **Queue Free** 88, **Set Property** 91, **Call Method** 71

Try it

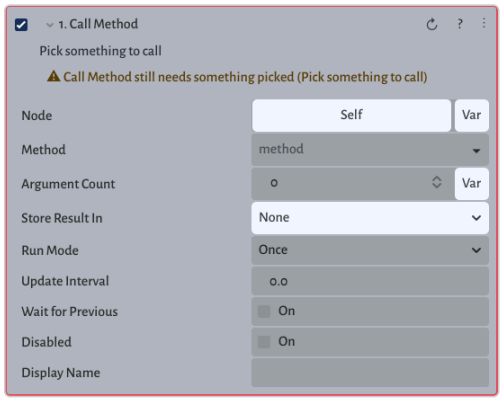
See the action work on its own with **Open Example** in the Add Action window, or open `examples/node/add_node/add_node.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Type	The kind of node to add, picked from the list of Godot's kinds, such as Timer, Area2D, OmniLight3D or AudioStreamPlayer. A class from your project works too. Spaces and capitals are forgiven, so area 2d means Area2D. Left empty, nothing is added and a warning asks for a kind, such as Timer.	empty
Parent	The node the new one goes under, as its child. Leave it empty for Self, the node this Machine works on. A 2D or 3D node added this way starts at its parent's position. Var: a Node.	Self
Node Name	What the new node is called in the scene, so other actions can find it by name, such as Fuse. Left empty it is called after its kind, such as Timer. When the name is already taken by a node beside it, a number is added, as in Timer2 or Timer3. Var: a String.	empty
Store In	The Node variable the new node goes into, so the next actions can change it with Set Property or Call Method, or listen to it with Event On Signal. Leave it empty to keep nothing, and the node is still added.	empty

Call Method

Makes a node do something it already knows how to do, named in a box, with up to two values handed to it. Nearly everything a game needs has a named action, and this one reaches what is left over, such as a node that came with another plugin.



Call Method as it appears in the Actions tab.

When to use it

A node from another plugin, or a script somebody wrote for you, can do something no action in this book names. Changing or reading a value on a node is a different job. Set Property and Get Property read more clearly on the graph, and Set Property warns when the value does not fit.

> A node that can do nothing by the name you typed does nothing, and a warning names the node and the name it was given. Pick from the list wherever you can.

See also

Set Property 91, **Get Property** 80, **Send Event** 31

Try it

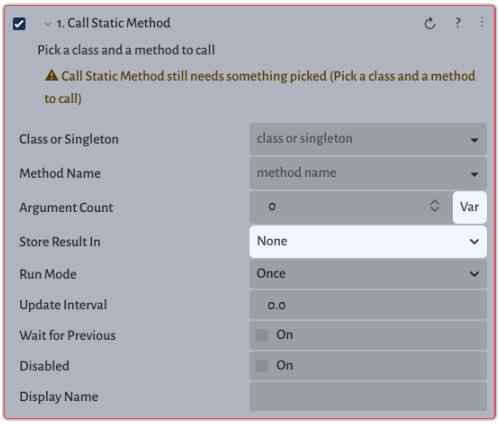
See the action work on its own with **Open Example** in the Add Action window, or open `examples/node/call_method/call_method.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node that is asked to do the thing. Leave it empty for Self, the node this Machine works on. Press Var beside the box and pick a Node variable to work on whatever that variable is holding, such as the thing a raycast just found. While the variable holds no node, nothing is called and a warning says Node points at nothing. Var: a Node.	Self
Method	The name of the thing the node can do. Press the little arrow beside the box and pick from the list. It looks at the node named in Node and offers what it can do, with a line under each saying what it is. You may also type a name, spelled as it is on the node, with no brackets after it.	empty
Argument Count	How many values are handed over. It can be none, one or two. It starts at 0, and raising it brings out that many value boxes below, which are the only ones handed over. Match it to the number of values the method takes, because a mismatch calls nothing, and the warning gives the right count. Var: an int.	0
Store Result In	If the node hands a value back, this is the variable it goes into, ready for a later action or a check to use. Leave it empty to throw the answer away, as most calls need nothing kept.	empty
Argument 1	Only when Argument Count is 1 or 2. The first value handed over. Press the button beside the box to say what kind of value it is, such as a number, text, a place, true or false, or a node, and the right sort of box appears. Or press Var to read it from a variable.	—
Argument 2	Only when Argument Count is 2. The second value, filled in the same way as the first.	—

Call Static Method

Calls a method that belongs to a class, not to a node in the scene, handing it up to two values, and can keep what it gives back. Godot's Time, OS and Input are reached this way, and so is a static func in a class_name script you wrote.



Call Static Method as it appears in the Actions tab.

When to use it

Time's get_time_string_from_system gives the time of day as text, ready for a label, and FileAccess can say whether a file is there. A small sum that several machines share can live as a static func in your script. An autoload, a node Godot keeps loaded while the game runs, can be named here as well.

See also

Call Method 71, **Get Property** 80, **Format Time** 55

Try it

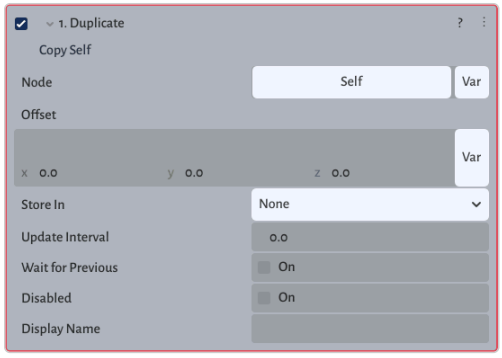
See the action work on its own with **Open Example** in the Add Action window, or open `examples/node/call_static_method/call_static_method.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Class or Singleton	The name of the class to call, with its capitals as Godot writes them. It can be a singleton such as Time, OS or Input, one of Godot's classes such as FileAccess, or the class_name of a script you wrote. A name that matches no class, singleton or autoload calls nothing, and a warning reminds you that capitals count.	empty
Method Name	The method to call, spelled as the class spells it, with no brackets after it, such as get_time_string_from_system. On one of Godot's classes or a script of yours, only a static method can be called, since there is no object to call it on. A func that is not static is not called, and a warning suggests Call Method or marking the func static.	empty
Argument Count	How many values are handed over, from 0 to 2. Raising it brings out that many value boxes below. It must suit the method. When the method takes a different number, nothing is called and a warning says how many it takes. Var: an int.	0
Store Result In	The variable what the method gives back goes into, such as the time as text. Leave it empty to throw the answer away.	empty
Argument 1	Only when Argument Count is 1 or 2. The first value handed over. Press the button beside the box to say what kind of value it is, such as a number, text or true or false, or press Var to read it from a variable.	—
Argument 2	Only when Argument Count is 2. The second value, filled in the same way as the first.	—

Duplicate

Makes a copy of a node that is already standing in the game, beside the original, such as another enemy like this one, a second coin or a clone. The copy brings everything the original has with it, including any Machine underneath it, and that machine starts afresh from its first state.



Duplicate as it appears in the Actions tab.

When to use it

Another of something already in the level, set up the same way, such as a boss that splits in two when it is hurt. New things usually come in with Instantiate, from a saved scene in the FileSystem dock. Duplicate copies a node already standing in the level.

> A copy of Self made in a machine's first state copies itself again as soon as it starts, on and on. Put Duplicate in a state the copy does not run first. It stops at a hundred copies in one frame, with a warning saying why.

See also

Instantiate 84, **Queue Free** 88, **Find Child** 77, **Group** 81

Try it

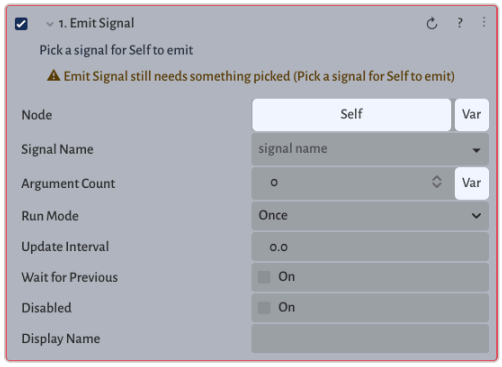
See the action work on its own with **Open Example** in the Add Action window, or open `examples/node/duplicate/duplicate.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node to copy. Leave it empty to copy Self, the node this Machine works on. The level's top node cannot be copied. Pointed there, nothing is copied, and a warning explains it has nothing to sit beside. Var: a Node.	Self
Offset	How far from the original the copy appears, so the two are not standing inside one another. 0 puts the copy where the original is. It is two numbers in a 2D game and three in a 3D one. Var: a Vector2 or a Vector3, or a Node for where it stands.	0, 0, 0 in 2D: 0, 0
Store In	The variable the new copy goes into, so the actions after it can move it, change it or send it an event. Leave it empty if you have no further use for it.	empty

Emit Signal

Makes a node send one of its signals, as though the node had sent it itself, with up to two values. A signal is the message a node sends when something happens to it, such as a button's pressed, and everything connected to that signal hears it.



Emit Signal as it appears in the Actions tab.

When to use it

Sending a button's pressed signal makes everything listening act as though it had been clicked, and sending a Timer's timeout finishes it early. A signal from your own script, such as died or opened, reaches plain scripts and other plugins that listen for it. Between two machines, Send Event is simpler.

> A button's pressed and a Timer's timeout carry no values, so Argument Count stays at 0 for them.

See also
Event On Signal 76, **Send Event** 31, **Call Method** 71

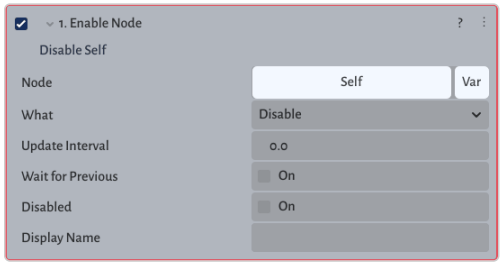
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/node/emit_signal/emit_signal.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node that sends the signal. Leave it empty for Self, the node this Machine works on. Press Var to use the node held in a variable. Var: a Node.	Self
Signal Name	The signal to send. Press the little arrow beside the box and pick from the signals that node really has, such as pressed or timeout. A name the node does not have sends nothing, and a warning names the node and the signal.	empty
Argument Count	How many values the signal hands over, from 0 to 2. It has to match the number the signal carries, or nothing is sent and a warning gives the right count. Var: an int.	0
Argument 1	Only when Argument Count is 1 or 2. The first value handed over with the signal. Press the button beside the box to say what kind of value it is, or press Var to read it from a variable.	—
Argument 2	Only when Argument Count is 2. The second value, filled in the same way as the first.	—

Enable Node

Switches a node off completely, or on again, with everything under it. Switched off, it is hidden, it stops running, and nothing can touch it. Switched on, it is put back as it was.



Enable Node as it appears in the Actions tab.

When to use it

An enemy in a room the player has not reached yet can wait here switched off until the player walks in, and a platform can be gone until a lever is pulled. Set Visible only hides a node, which still blocks and still runs, and Set Collision only stops it touching things, but this action does both and stops it running too.

> Switching off the node this Machine works on switches this Machine off with it, and another Machine has to switch it on again. A node enabled under one that is switched off still does nothing, and a warning points this out.

See also
Set Visible 216, **Set Collision** 119, **Set Process Mode** 90, **Enable Machine** 18

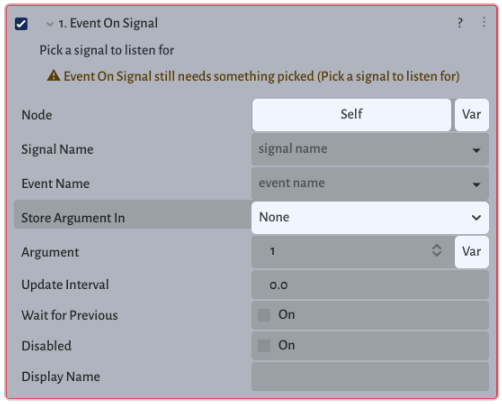
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/node/enable_node/enable_node.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node to switch. Everything under it goes with it. Left empty, the default, it switches Self, the node this Machine works on. Var: a Node.	Self
What	Disable hides it, stops it running (its own machine, its animations and its timers all hold where they are) and takes its bodies and areas out of the physics world, so things fall and walk through them and areas stop noticing anything. Enable puts it back as it was before it was switched off, or, for a node set up switched off in the scene, switches it on and shows it. Toggle does whichever of the two it is not already. Choices: Disable, Enable, Toggle.	Disable

Event On Signal

Listens to a node's signal while this state runs and sends an event when it fires. A signal is a node's way of announcing that something has just happened to it. A button announces pressed, an area announces that a body walked in, and a timer announces that it ran out.



Event On Signal as it appears in the Actions tab.

When to use it

This is how the rest of the scene talks to a machine. A menu's Play button announcing pressed can move the machine to its playing state, and a slider can hand its new value to a variable. It listens without holding the state up, and the state can get on with other things meanwhile. Two physics bodies touching are easier with Event On Collision, which names the groups for you.

> A node with no signal of that name is never heard, and a warning says the signal is missing. Pick the signal from the list beside Signal Name to avoid a typing slip.

See also
Event On Collision 109, **Get Event Data** 20, **Get Property** 80, **Send Event** 31

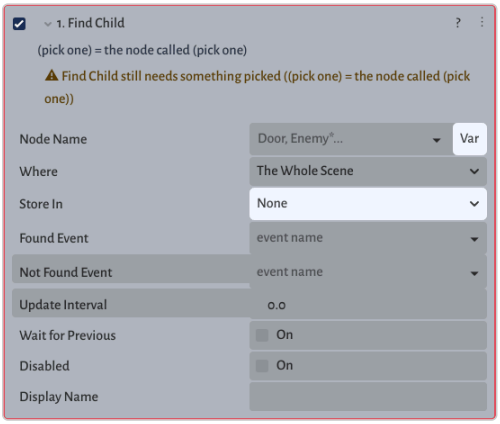
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/node/event_on_signal/event_on_signal.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node whose signal you are listening to. Leave it empty for the node the machine works on. Point it at another node, such as a Reset button somewhere in the menu, to hear that node's signal. Var: a Node.	Self
Signal Name	The signal's name, spelled as the engine spells it, such as pressed for a button, body_entered for an area, or timeout for a timer. The Node dock beside the Inspector lists every signal the selected node has, so copy the name from there, not from memory.	empty
Event Name	The event sent each time the signal fires. Leave it empty when you only want the value the signal carries kept in a variable, such as a slider's new position, and nothing needs to change state.	empty
Store Argument In	Some signals hand a value over with them, such as the body that walked in or the slider's new number. Name a variable here and that value is kept in it, ready for another action to use. Leave it empty and nothing is kept.	empty
Argument	Which of the signal's values to keep, counting from 1 for the first. Most signals hand over one value or none, and the default of 1 fits them. A 3D body's input event is the exception, because it hands over the camera first and the click second. Var: an int.	1

Find Child

Looks for a node by its name and puts it in a variable, so the actions after it can work on it, such as the player, the exit door or a boss that was spawned. It sends one event when it finds something and another when it does not.



Find Child as it appears in the Actions tab.

When to use it

At the start of a machine, to get hold of something it needs but is not joined to, such as a label on the screen. Once the node is in a variable, press Var on any Node box and that action works on it.

See also

Group 81, **Event On Collision** 109, **Look For Target** 256, **Raycast** 116

Try it

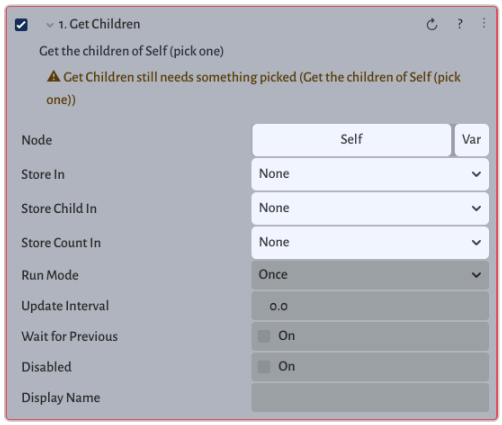
See the action work on its own with **Open Example** in the Add Action window, or open `examples/node/find_child/find_child.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node Name	The name as it is written in the Scene dock, such as Player. A * stands for any letters, so Enemy* matches Enemy, Enemy2 and EnemyBoss, and the first one found is taken. Left empty, nothing is ever found. Capital letters count, so Player and player are two different names. Var: a String.	empty
Where	The Whole Scene looks anywhere in the level being played, including things that appeared while the game was running, and can match the level's top node too. Under Self looks only under the node this Machine works on. It is quicker, and safer when one name is used in several places. Choices: The Whole Scene, Under Self.	The Whole Scene
Store In	The variable the node goes into. Leave it empty and what was found is thrown away straight after the search.	empty
Found Event	Sent when a node of that name did turn up, so a transition can move on to the next state knowing the variable is filled. Leave it empty if nothing needs telling.	empty
Not Found Event	Sent when nothing of that name was there, so the machine can go somewhere else and not sit there doing nothing. The variable is emptied too.	empty

Get Children

Looks at the nodes directly under a node, its children. It can give all of them as a list, one of them by its place, or how many there are, each into a separate variable.



Get Children as it appears in the Actions tab.

When to use it

A wave is cleared when the count of children under its Wave node reaches 0, and a Check can watch for that. A patrol can be handed the markers under a Waypoints node as a list, and For Each works through it one child at a time. Only the nodes one step down count, in the order the Scene dock lists them, and any Machine among them is left out.

> Pick at least one of the three variables. With none picked there is nothing to keep, and the action warns about it.

See also

For Each 36, **Get Parent** 79, **Find Child** 77, **Array** 58

Try it

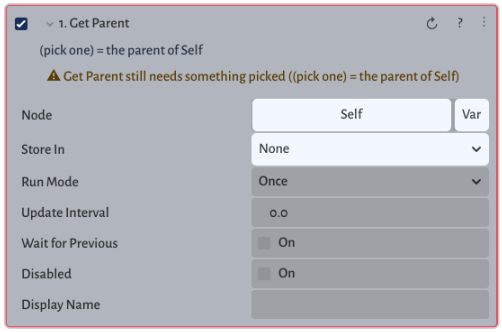
See the action work on its own with **Open Example** in the Add Action window, or open `examples/node/get_children/get_children.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node whose children are looked at. Left empty, the default, it looks under Self, the node this Machine works on. A Machine under it is left out, being how the node behaves and not one of the things under it, so the enemies left in a wave do not count the wave's own Machine. Var: a Node.	Self
Store In	The Array variable every child goes into, as a list in the Scene dock's order. Empty, the default, keeps nothing.	empty
Store Child In	The Node variable the child at Index goes into. Empty, the default, keeps nothing.	empty
Store Count In	The int variable how many children there are goes into. Empty, the default, keeps nothing.	empty
Index	Only once Store Child In names a variable. Which child to take, counting from 0 for the first, so the third is 2. A number below 0 counts from the end, so -1 is the last. A child that is not there empties the variable, and a warning gives how many children there are.	—

Get Parent

Puts the node a node sits directly under, its parent, into a Node variable, ready for other actions to work on.



Get Parent as it appears in the Actions tab.

When to use it

When a sword strikes a hurtbox, the enemy is usually the hurtbox's parent, and this puts that enemy in a variable for the damage to go to. It also finds the hand carrying a crate right now, or the room a lamp is in.

See also

Get Children 78, **Reparent** 89, **Find Child** 77, **Machine Variable** 27

Try it

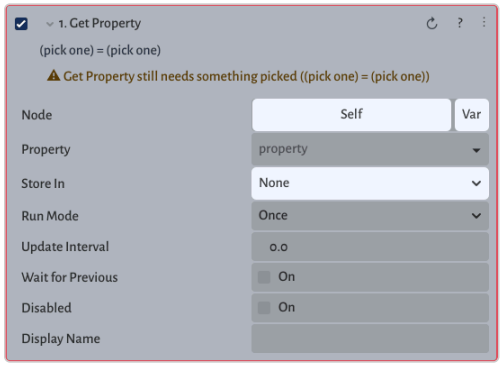
See the action work on its own with **Open Example** in the Add Action window, or open `examples/node/get_parent/get_parent.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node whose parent is wanted. Left empty, the default, it uses Self, the node this Machine works on, and gives the node Self sits under. Var: a Node.	Self
Store In	The Node variable the parent goes into. A variable holding a number or anything else that cannot hold a node is left as it was, and a warning asks for a Node variable.	empty

Get Property

Reads any setting of any node in the scene into a variable. A node is one of the pieces listed in the Scene dock. A variable is a name the machine keeps a value under.



Get Property as it appears in the Actions tab.

When to use it

A lift's machine can read where its moving platform is each frame and stop it at the right floor. A value kept by another machine is read with Machine Variable, not with this.

> Position is measured from the node's parent, as the Inspector shows it. Global Position, where it is in the scene overall, is in the list too. A property the node does not have leaves the variable as it was, with a warning.

See also

Set Property 91, **Machine Variable** 27, **Property** 87, **Set Variable** 49

Try it

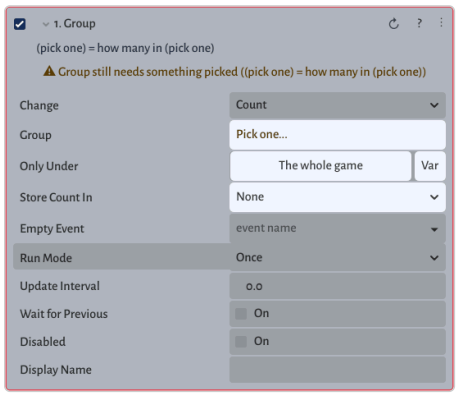
See the action work on its own with **Open Example** in the Add Action window, or open `examples/node/get_property/get_property.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node to read. Leave it empty for the node this Machine works on. Press Var beside it and pick a Node name to read whichever node that name is holding at the time, such as the enemy a raycast found or the coin that was just touched. Var: a Node.	Self
Property	Which setting to read, picked from the list of what that node has, so there is nothing to spell. A colon reaches inside a setting made of several parts, so position:y reads only the height.	empty
Store In	The name the value is kept under. Pick one from the list, or make a new one from the bottom of that list. Left empty nothing is kept, and a warning is printed.	empty

Group

Works with a group, Godot's way of sorting nodes into a named set. It can put a node in a group or take it out, put every child of a node in or take them all out in one go, count how many are in one, or make a list of them to hand to For Each.



Group as it appears in the Actions tab.

When to use it

Count, with Empty Event, opens the door when the last enemy falls. Add and Remove mark things as the game goes along, such as the doors now open, and Add Children sorts a set in one go, such as every enemy of a wave standing under a Wave node. List hands every one of them to For Each. A question about one node belongs to the Node check.

> Anything already on its way out of the game is left out of a count or a list, so an enemy destroyed this frame is not counted as still standing.

See also

For Each 36, **Find Child** 77, **Node** 86, **Queue Free** 88

Try it

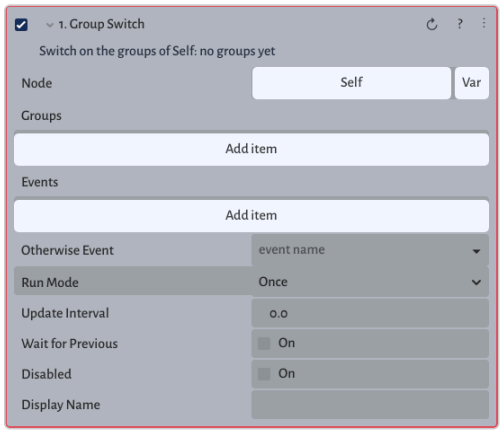
See the action work on its own with **Open Example** in the Add Action window, or open `examples/node/group/group.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Change	What this one does. Add puts a node in the group, Remove takes it out, Count works out how many are in it, and List makes a list of them. Add Children puts every node straight under a node in the group in one go, and Remove Children takes them all out again. It starts on Count, and the boxes below change with it. Choices: Add, Remove, Count, List, Add Children, Remove Children.	Count
Group	Which ones, such as every coin. Press the box and pick it in the scene. It can be one thing by its name, Every Coin for every copy of the coin scene (including ones made while playing), or Everything under a node for a set of markers. NodeWorthy does the rest. Add and Remove put a node in with them or take it out, and Add Children and Remove Children do that to every child of a node. Left empty, the action does nothing whatever Change says, and a warning says no group is named.	empty
Only Under	Only while Change is Count or List. Count or list only the ones inside this node, such as the enemies in this room, not the whole game's. Leave it empty to look at the whole game. If it names a node that is not there, nothing is counted, Empty Event is not sent, and a warning says the Only Under node was not found. Var: a Node.	empty
Store Count In	Only while Change is Count. The variable the number goes into, ready for a check such as whether any are left or a label showing coins collected. Leave it empty and the count is only used to decide about Empty Event.	empty
Empty Event	Only while Change is Count. Sent straight after counting when there is nobody left in the group at all, such as when the last enemy is defeated or the last coin collected. Leave it empty if nothing needs telling.	empty
Node	Only while Change is Add, Remove, Add Children or Remove Children. For Add and Remove, the node put into the group or taken out of it. For Add Children and Remove Children, the node whose children are, such as a Wave node with the enemies under it. Only the nodes straight under it count, not what is under them, and a Machine sitting under it is left out. Leave it empty for Self, the node this Machine works on. A door marks itself as open this way. If it names a node that is not there, nothing changes and a warning asks you to pick one under Node.	—
Store List In	Only while Change is List. The variable the list goes into, ready to hand to For Each. Leave it empty and nothing is kept, and the action then does nothing.	—

Group Switch

Sends one of several events, picked by which group a node is in. The groups are looked at in the order listed, and the first one the node is in sends its event. When it is in none of them, Otherwise is sent.



Group Switch as it appears in the Actions tab.

When to use it

After a bullet hits something, one Group Switch can send HIT_ENEMY for an enemy, HIT_WALL for a wall and HIT_GLASS for a window, doing the work of a separate Check for each group. Point Node at what was hit with Var.

See also
Switch 39, **Group** 81, **Node** 86, **Raycast** 116

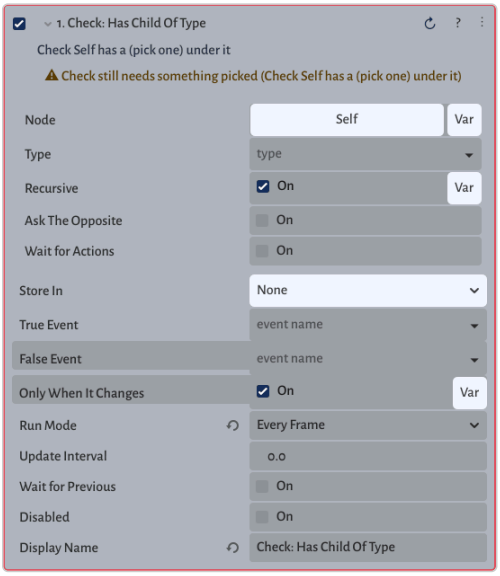
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/node/group_switch/group_switch.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node whose groups are looked at. Leave it empty for Self, the node this Machine works on. Press Var to look at the node held in a variable, such as what a Raycast just hit. Var: a Node.	Self
Groups	The groups to look for, one row each, in order. The first one the node is in wins, so list the more particular ones first.	empty
Events	The event each group sends, in the same order as the groups, so the first event goes with the first group. The Problems button says so when the two lists are not the same length.	empty
Otherwise Event	Sent when the node is in none of the groups, or when there is no node there at all. Leave it empty to send nothing.	empty
Only When It Changes	Only while Run Mode is Every Frame. On, the default, an event is sent only when the group that matches changes, not every frame it stays the same.	—

Has Child Of Type CHECK

Asks whether a node has a node of a particular kind under it, such as an AnimationPlayer, a Hurt Box or an Inventory. A kind built on the one asked about counts too.



Has Child Of Type as it appears in the Actions tab.

When to use it

What a thing can do often depends on the parts it has. A chest can work as a shop only while it has an Inventory under it, and a door can stay shut while a Timer is still under it.

> A node already on its way out of the game is not counted, so the answer turns to no as soon as one is removed.

See also

Is Child Of 85, **Node** 86, **Get Children** 78

Try it

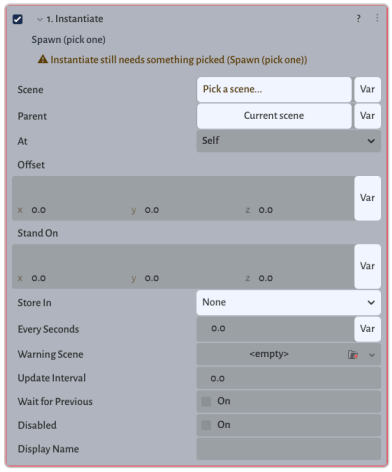
See the action work on its own with **Open Example** in the Add Action window, or open `examples/node/has_child_of_type/has_child_of_type.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node to look under. Leave it empty for Self, the node this Machine works on. Press Var to look under the node held in a variable. Var: a Node.	Self
Type	The kind of node to look for, picked from the list. It can be one of Godot's own, such as AnimationPlayer or Area2D, one of NodeWorthy's nodes, such as Hurt Box or Inventory, or a class from your project. Asking for a general kind finds the kinds built on it, so PhysicsBody2D finds a CharacterBody2D. A name that is no kind of node always gives no, and a warning names it.	empty
Recursive	On, the default, it looks all the way down, under its children too. Off, only the nodes straight under it count. Var: a bool.	on
Ask The Opposite	Turns the answer round into "it has none of those under it". It turns round the true-or-false answer the Check stores as well, which swapping the True Event and False Event over cannot do.	off
Wait for Actions	Holds the question back until every other action in the state has finished, so it looks after the state's own actions have added or removed nodes. Left off, the default, it is asked as soon as the state runs.	off
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Instantiate

Adds a saved scene to the running game, such as an effect, a pickup, a bullet or an enemy. You say where it appears and, if you like, keep the new node in a variable for the actions that follow.



Instantiate as it appears in the Actions tab.

When to use it

New things come into a level this way, such as a coin dropped where an enemy fell. For a great many 2D bullets, Bullet Pattern is faster and does the hitting for you.

> Every Seconds only keeps going while the state is still running, so a state that moves straight on spawns just the one.

See also

Duplicate 73, **Bullet Pattern** 265, **Queue Free** 88, **Change Scene** 220

Try it

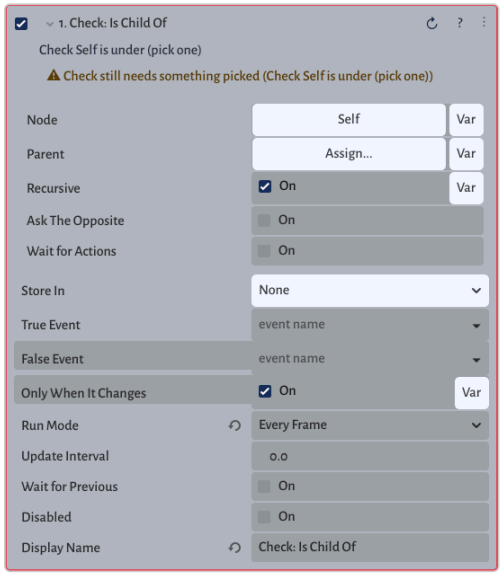
See the action work on its own with **Open Example** in the Add Action window, or open `examples/node/instantiate/instantiate.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Scene	Normally empty, and hidden while it is. It holds the scene itself, as Godot's Inspector sets it. Pick the scene in Scene below.	empty
Scene	The scene that appears, picked from your own scenes by name, or any scene with Pick a file.... It keeps the scene's path, so a scene can make copies of itself, as a splitter breaking into smaller ones does. It can be read from a variable, letting each node spawn a different scene. With nothing picked, nothing appears, the field shows in warning colour, and a warning asks you to pick one under Scene. Var: a String.	empty
Parent	Which node the new scene is added underneath. Empty adds it under the level being played. Point it at a node, or press Var and use a node from a variable such as the thing a raycast hit, and the new scene moves with that node, so a scorch mark sticks to a moving lift. Var: a Node.	empty
At	Where it appears. Self is where the node this Machine works on stands, turned the way it faces. A Node is somewhere else, such as a marker at a gun's muzzle. A Place is a spot typed in or read from a variable, such as where the mouse points. Where The Scene Says leaves it wherever it was saved. One Of The Nodes Under picks one of another node's children at random, for enemies arriving from a set of spawn points. Somewhere Inside picks a place at random anywhere inside another node's collision shape, such as a box drawn over the arena, so enemies can come from anywhere. Whichever you choose, the new scene keeps its own size. Choices: Self, A Node, A Place, Where The Scene Says, One Of The Nodes Under, Somewhere Inside.	Self
Offset	How far from that place it appears, measured in the directions that node faces, so a bullet started a little in front of a gun stays in front however the gun turns. 0 puts it at the place. For A Place it is in the world's directions. Two numbers in 2D, three in 3D. The box is hidden for Where The Scene Says, which does not move the scene at all. Var: a Vector2 or a Vector3, or a Node for where it stands.	0, 0, 0 in 2D: 0, 0
Stand On	Turns the new scene so its up points this way, for a mark on a wall or a spray of sparks off one. Raycast's Store Normal In gives the way a surface faces, so press Var and use that. 0, the default, leaves the turn alone. It is 3D only, and adds a random spin about that up, so several marks on one wall do not all look the same. Var: a Vector2 or a Vector3, or a Node for where it stands.	0, 0, 0
Store In	The variable the new node goes into, so the actions after it can move it, send it an event or destroy it later. Empty stores nothing.	empty
Every Seconds	Keep adding another one this often, in seconds, for as long as the state runs, like a gun firing while the trigger is held or a spawner dropping enemies. 0.25 is four a second. 0, the default, adds just the one and stops. Var: a float.	0.0
Warning Scene	A scene shown first, where the real one is going to appear, such as a flashing mark on the ground, so the player sees it coming and can step aside. Leave it empty for no warning. When the time is up the warning goes and the real scene appears where it stood.	empty
At Node	Only when At is A Node, One Of The Nodes Under or Somewhere Inside. For A Node it is the node the scene appears at, such as a marker at a gun's muzzle. For One Of The Nodes Under it is the node whose children are the places to choose between, so put a handful of markers under one empty node and point this at that node. For Somewhere Inside it is a collision shape, or a node with one under it such as an Area3D, marking out the area. It can be a box, sphere, cylinder or capsule in 3D, where the scene appears at the height of the shape's middle, or a rectangle, circle or capsule in 2D.	—
Away From	Only when At is One Of The Nodes Under or Somewhere Inside. Places closer than At Least to this node are not picked, so pointing it at the player keeps enemies from appearing on top of them. Leave it empty and any of the places may be picked.	—
At Least	Only when At is One Of The Nodes Under or Somewhere Inside. How far from Away From a place has to be, in metres in 3D or pixels in 2D. 0 ignores the distance. If every place is closer than this, the farthest of them is used.	—
Place	Only when At is A Place. The spot the scene appears at, as two numbers in a 2D game or three in a 3D one. Press Var to read it from a variable, such as the one Mouse Position fills in, to drop something where the player clicked.	—
Warning Seconds	Only when a Warning Scene is picked. How long the warning shows before the real scene takes its place. It starts at 0.8 seconds, about long enough for a player to step out of the way.	—

Is Child Of CHECK

Asks whether a node is somewhere under another node, however many steps down. Untick Recursive to ask only whether it is straight under it.



Is Child Of as it appears in the Actions tab.

When to use it

Whether a crate is riding on the lift, or whether the body a bullet hit is part of the player, depends on where the node sits in the scene now. Reparent moves a node under another, and this asks where it is.

> With Parent left empty, or pointing at a node that is not there, the answer is no and a warning says which setting points at nothing.

See also

Reparent 89, **Has Child Of Type** 83, **Get Parent** 79, **Node** 86

Try it

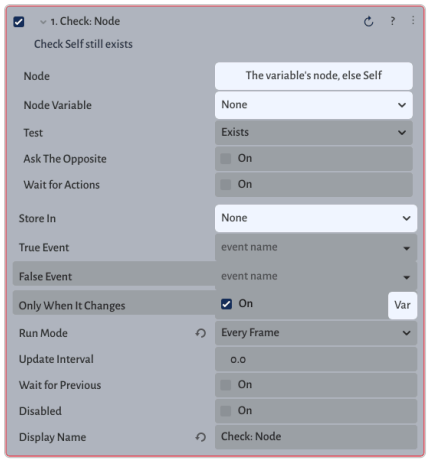
See the action work on its own with **Open Example** in the Add Action window, or open `examples/node/is_child_of/is_child_of.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node asked about. Leave it empty for Self, the node this Machine works on. Press Var to ask about the node held in a variable, such as what was hit. Var: a Node.	Self
Parent	The node it may be under, such as the lift, the hand or the player. It must be picked. Pick Self to ask whether something is part of this very thing. A node is never under itself. Var: a Node.	empty
Recursive	On, the default, it counts however many steps down the node is. Off, only a node straight under Parent, its child, counts. Var: a bool.	on
Ask The Opposite	Turns the answer round into "it is not under that node", such as a sword no longer in the hand. It turns round the true-or-false answer the Check stores as well, which swapping the True Event and False Event over cannot do.	off
Wait for Actions	Holds the question back until every other action in the state has finished, so it asks after the state's own actions have moved the node. Left off, the default, it is asked as soon as the state runs.	off
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Node CHECK

Asks a question about a node, such as whether it is still there, carries a group's label, or is shown. It notices that a target has died or a pickup has been taken.



Node as it appears in the Actions tab.

When to use it

Any state built round something that can disappear. For example, Exists with Ask The Opposite ticked ends a chase when the thing being chased is gone. In Group tells you what sort of thing a variable is holding, such as whether it is in the "enemy" group.

> A variable holding something that is not a node, or a node that has already been removed, answers no and the game carries on.

See also

Property 87, **Find Child** 77, **Group** 81, **On Screen** 115

Try it

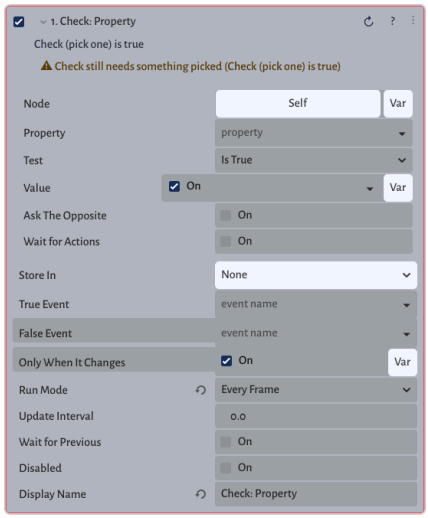
See the action work on its own with **Open Example** in the Add Action window, or open `examples/node/node.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node to ask about. Leave it empty and the node held in the variable below is used when you have named one, otherwise the node the machine works on, which always exists.	empty
Node Variable	Name a variable holding a node and that node is asked about. It wins over the box above even when it holds no node, and then the answer is no, so fill in one or the other. "Is the thing I am chasing still there" is asked here, since the chase stored it in a variable in the first place.	empty
Test	Exists, the default, says yes while the node is still in the game. In Group asks whether it carries the label named below. Is Visible asks whether it is shown, meaning its own visible tick is on and nothing above it in the scene is hidden. Whether the camera can see it is a question for the On Screen check. Choices: Exists, In Group, Is Visible.	Exists
Ask The Opposite	Turns the answer round into "it has gone", "it is not one of those" or "it is hidden". It turns round the true-or-false answer the Check stores as well, which swapping the True Event and False Event over cannot do.	off
Wait for Actions	Holds the question back until every other action in the state has finished, so it asks about a node the state's own actions have just found or made. Left off, the default, it is asked as soon as the state runs.	off
Group	Only when Test is In Group. Is the node one of these? Press the box and pick it in the scene. You can pick one thing by its name, Every Coin for every copy of the coin scene (including ones made while playing), or Everything under a node for a set of markers. NodeWorthy does the rest.	—
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Property CHECK

Reads a setting off a node and compares it with a value, such as whether it is visible, its height is above a number, or a tick box is on. Anything the node has in its Inspector can be asked about.



Property as it appears in the Actions tab.

When to use it

For a value that lives on a node, not in the template's variables, such as a ProgressBar's value, whether a light is on, or a setting on a node you added from elsewhere. Where a check asks the question directly, such as Touching for contact, that check reads better on the box.

> Pick the name from the list, and do not type it. A typed name the node does not have gives no, and the warning points you back to the list.

See also

Get Property 80, **Set Property** 91, **Compare** 35, **Variable** 40

Try it

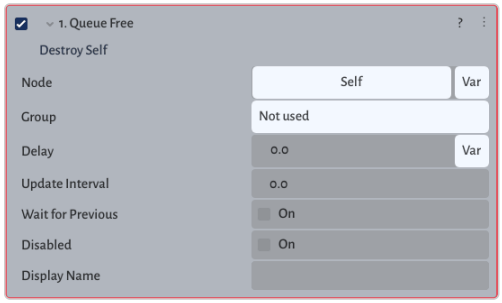
See the action work on its own with **Open Example** in the Add Action window, or open `examples/node/property/property.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node to read. Leave it empty for the node the machine works on. If it points at nothing, the answer is no and a warning asks you to pick the node. Var: a Node.	Self
Property	Which setting to look at, picked from the list of what that node has. A colon reaches inside one that has parts, so "position:y" is just the height. Only settings can be read. A name the node does not have gives no, with a warning that it has no property called that, and one of its methods gives no, with a warning that it is something the node does rather than has. Left empty, there is nothing to read at all.	empty
Test	How to compare. Is True, the default, says yes when the setting is on, and Is False is its opposite. Neither uses Value. Equals and Not Equals compare the setting with Value whatever kind of thing it is, while Greater and Less read both as numbers. Choices: Equals, Not Equals, Greater, Less, Is True, Is False.	Is True
Value	What the setting is compared with. It takes whatever you need, such as a number, some text, on or off, or a place. Is True and Is False ignore it, and it can sit at its default with those. Var: any kind, as the property picked.	on
Ask The Opposite	Turns the answer round, so "visible" becomes "hidden" without changing the test. It turns round the true-or-false answer the Check stores as well, which swapping the True Event and False Event over cannot do.	off
Wait for Actions	Holds the question back until every other action in the state has finished, so it reads a setting the state's own actions have just written. Left off, the default, it is asked as soon as the state runs.	off
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Queue Free

Removes a node from the game for good, either one node or every node in a group. It can wait a moment first.



Queue Free as it appears in the Actions tab.

When to use it

A coin collected, a defeated enemy, a puff of smoke that has played out. Name a Group to clear a set away in one go, such as every pickup and chest left on a floor before the next is built. Removal is permanent, so to get something back later, hide it with Set Visible.

> The action finishes as soon as it starts, and the state carries on without waiting out the Delay.

See also
Instantiate 84, **Set Visible** 216, **Group** 81, **Duplicate** 73

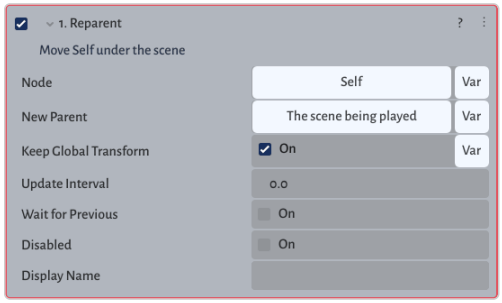
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/node/queue_free/queue_free.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node to remove. Leave it empty to remove Self, the node this Machine works on. An effect tidies itself away like this when it is done. The box hides once you fill in Group, and the group decides what goes. Var: a Node.	empty
Group	Pick something here to remove every one of them, wherever in the game they are, such as every coin or every bullet. Press the box and pick it in the scene. It can be one thing by its name, Every Coin for every copy of the coin scene (including ones made while playing), or Everything under a node for a set of markers. NodeWorthy does the rest.	empty
Delay	Wait this many seconds before it goes. 0, the default, removes it straight away. A delay lets an effect play out first, or a defeated guard lie on the ground a moment. The wait stands still while the game is paused, unless this machine is one that keeps running through a pause. Var: a float.	0.0

Reparent

Moves a node, and everything under it, to sit under another node, its new parent. From then on it moves with its new parent.



Reparent as it appears in the Actions tab.

When to use it

A passenger stepping onto a moving platform can be moved under it and ride along, and a card can go from the deck to the hand. Carry handles picking something up and putting it down again, while this only moves a node from one parent to another.

> A node cannot go under itself or under something inside it, and the top of the scene cannot be moved at all. Either gets a warning and nothing moves.

See also

Carry 92, **Get Parent** 79, **Instantiate** 84, **Move To** 97

Try it

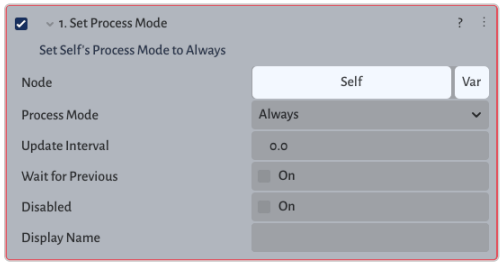
See the action work on its own with **Open Example** in the Add Action window, or open `examples/node/reparent/reparent.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node to move. Left empty, the default, it moves Self, the node this Machine works on. The Machine goes with it, as soon as this frame's work is done, and carries on in the state it is in rather than starting that state's actions again. Var: a Node.	Self
New Parent	The node it goes under. Left empty, the default, it goes under the scene being played. Something carried is dropped back into the level this way. Var: a Node.	empty
Keep Global Transform	On, the default, it stays where it is in the world, turned and sized as it was, as it changes parent. Off, it keeps its position, rotation and scale relative to its parent and jumps to the same spot on the new one. Var: a bool.	on

Set Process Mode

Chooses whether a node keeps running while the game is paused, using Godot's own five Process Modes. Everything under the node follows it, unless it has its own Process Mode set.



Set Process Mode as it appears in the Actions tab.

When to use it

A pause menu set to Always still answers its buttons while everything else is frozen, and When Paused suits the opposite case, such as a photo camera that only moves during a pause. It pauses nothing by itself. Pause Game does that.

> Setting the node this Machine works on to Disabled stops this Machine with it (unless the Machine has its own Process Mode set), and another Machine then has to change it back.

See also

Pause Game 222, **Enable Node** 75, **Set Time Scale** 66

Try it

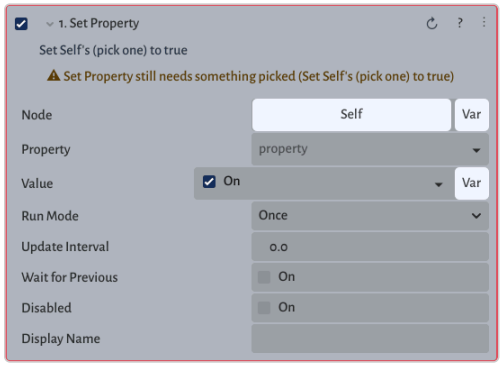
See the action work on its own with **Open Example** in the Add Action window, or open `examples/node/set_process_mode/set_process_mode.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node to set. Left empty, the default, it sets Self, the node this Machine works on, and with it this Machine. Var: a Node.	Self
Process Mode	Inherit does what the node above it does. Pausable runs while the game runs and stops while it is paused. When Paused runs only while the game is paused. Always, the default here, runs whether or not the game is paused. Disabled never runs, as if switched off, but stays on screen. Choices: Inherit, Pausable, When Paused, Always, Disabled.	Always

Set Property

Sets any one of a node's settings by name, whatever that node keeps, such as whether it is visible, its colour, its text or its speed. It sets it once, as the state is entered.



Set Property as it appears in the Actions tab.

When to use it

For a change no action in this book names, such as a label's font size or a light's brightness. Everyday jobs like Set Visible, Set Text and Set Volume have named actions, and those read better on the graph.

> Set Run Mode to Every Frame when the value should follow a variable that keeps changing, such as a light's Energy an Expression works out each frame, or has to be held against something else changing it. On Every Frame nothing else can change that setting while the state runs.

See also

Get Property 80, **Tween Property** 106, **Set Visible** 216, **Call Method** 71

Try it

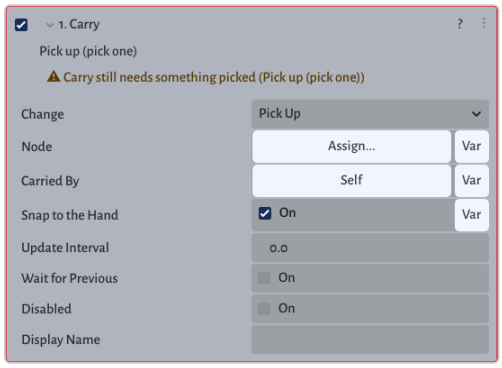
See the action work on its own with **Open Example** in the Add Action window, or open `examples/node/set_property/set_property.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node to change. Leave it empty for Self, the node this Machine works on. Press Var beside the box and pick a Node variable to change whatever that variable is holding, such as the enemy a raycast found or the coin just touched. While the variable holds no node nothing is set, a warning says Node points at nothing, and it never falls back to Self. Var: a Node.	Self
Property	Which of that node's settings to change. Press the little arrow beside the box and pick from the list. It reads the node named in Node and offers its settings, written as the Inspector writes them, such as Visible, Modulate, Text or Playing, and keeps the spelling right for you. You can also type one in, and typing is the only way to reach inside another value. A colon does that, as in "position:y" for the height on its own, or "material:albedo_color" for the colour of a material.	empty
Value	What to set it to. Press the button beside the box to say what kind of value this is, from a float, an int, a bool, a String, a Vector2 or Vector3, a Color, a node, a list, a table or a file, and the right sort of box appears beside it. Press Var instead to read the value from a variable while the game runs. It starts as true, which suits Visible and the other tick-box settings. If the kind of value does not match what that setting holds, nothing is changed and a warning says what it holds and what you gave it. Var: any kind, as the property picked.	on

Carry

Picks something up and carries it, then puts it down or throws it. It might be a crate, a key, a ball or a torch. What is carried moves with whatever carries it, as if held, and stops bumping into things and falling until it is let go. It works in 2D and in 3D.



Carry as it appears in the Actions tab.

When to use it

Anything the player holds and lets go of again, such as a crate carried onto a pressure plate. Put a Marker3D or Marker2D (an empty node that marks a spot) where the hand should be, and point Carried By at it.

> Put Down and Throw only work on something this action picked up. Used on anything else they do nothing.

See also

Move To 97, **Event On Collision** 109, **Drag** 140, **Give Item** 290

Try it

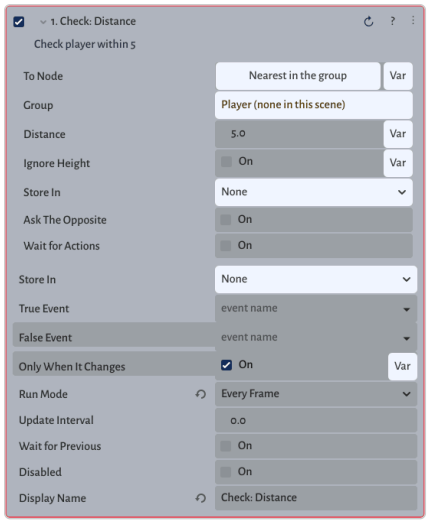
See the action work on its own with **Open Example** in the Add Action window, or open `examples/transform/carry/carry.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Change	What this one does. Pick Up starts carrying it, Put Down lets go of it where it stands, and Throw lets go and pushes it away. You normally use three of these actions in three different states, one for each, driven by the same button. Choices: Pick Up, Put Down, Throw.	Pick Up
Node	The thing being picked up, put down or thrown. It is usually a Node variable with Var pressed, such as the thing an Event On Collision has just noticed, so the character carries whatever they walked into. Unlike most Node boxes, leaving it empty carries nothing, and it does not fall back to Self. Var: a Node.	empty
Carried By	Only while Change is Pick Up. What holds the thing, normally a marker placed where the character's hand is. Leave it empty for Self, the node this Machine works on, which carries it at the character's middle. Var: a Node.	Self
Snap to the Hand	Only while Change is Pick Up. On, the thing jumps to the hand as it is picked up. That is right for a torch or a key. Off carries it from where it was, for a crate the character has walked right up to. Var: a bool.	on
Throw	Only when Change is Throw. How hard it is thrown, measured in the directions the carrier faces. In 3D, forward is minus Z and up is Y, in metres a second. In 2D, forward is X and down is Y, in pixels a second. It is two numbers in a 2D game and three in a 3D one, and starts at a gentle forward-and-up lob. The push only reaches a physics body, so anything else is let go where it stands.	—

Distance CHECK

Asks whether something is close enough. Is that node, or the nearest member of a group, within so many metres or pixels of the node the machine works on?



Distance as it appears in the Actions tab.

When to use it

Ten metres for a guard to notice the player, or two for a door to open as they walk up. It can search a group as well as a named node, and one template then works on every enemy in the game. For a character with a Perception node, measuring to whatever it is chasing, In Range fits better.

> The node the machine works on must have a place in the world, in two dimensions or three. On anything else the answer is always no, and a warning names the node and what type it is.

See also

In Range 95, **Can See** 250, **Keep Distance** 255, **Get Overlapping** 113

Try it

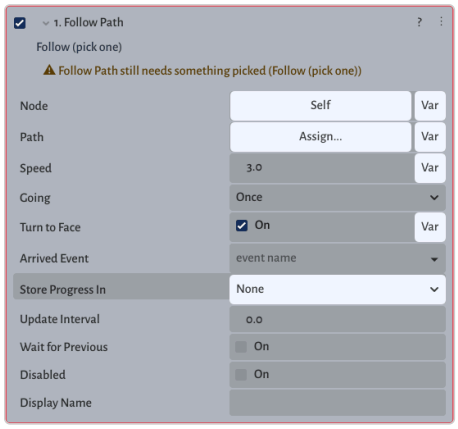
See the action work on its own with **Open Example** in the Add Action window, or open `examples/transform/distance/distance.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
To Node	The one node to measure to. Pick it and only that node counts. If it has gone from the scene the answer is no, and another member of the group standing nearby does not take its place. Leave it empty, the default, to search the group below instead. Var: a Node.	empty
Group	What to measure to when no node is named above, which is the nearest of what is picked here, such as the player. Press the box and pick it in the scene. You can pick one thing by its name, Every Coin for every copy of the coin scene (including ones made while playing), or Everything under a node for a set of markers. NodeWorthy does the rest. It starts on the player. If nothing in the scene is the player the answer is always no, and the problems button says so.	player
Distance	How close counts, in metres in a 3D game and in pixels in a 2D one. The default 5 is close range in 3D, and in a 2D game it starts at 160 pixels. Var: a float.	5.0 in 2D: 160 px
Ignore Height	Ticked, a 3D game measures along the ground, so somebody on the floor above is not treated as far away. A 2D game ignores this setting. Leave it off wherever up and down should matter, such as a pit you have fallen into. Var: a bool.	off
Store In	Name a variable and the node that was found goes into it, for the actions after it to chase or shoot at. Empty, the default, stores nothing.	empty
Ask The Opposite	Turns the answer round into "nothing is that close", and a guard can go back to patrolling on it. It turns round the true-or-false answer the Check stores as well, which swapping the True Event and False Event over cannot do.	off
Wait for Actions	Holds the question back until every other action in the state has finished, so the measurement is taken after the state has moved things. Left off, the default, it is asked as soon as the state runs.	off
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Follow Path

Slides something at a steady speed along a curve drawn on a Path2D or Path3D node in the scene, turning it to face the way it is going.



Follow Path as it appears in the Actions tab.

When to use it

When the route matters more than where it ends, such as a platform on a winding track, a car round a circuit, or a camera flying through a level for an opening shot. With Going on Once the state finishes at the end of the path. Loop and Ping Pong carry on for as long as the state runs.

> A path drawn with only one point has no length. Nothing moves, the action finishes straight away, and a warning says the path has no line drawn on it.

See also

Move To 97, **Translate** 105, **Patrol** 258, **Travel To** 170

Try it

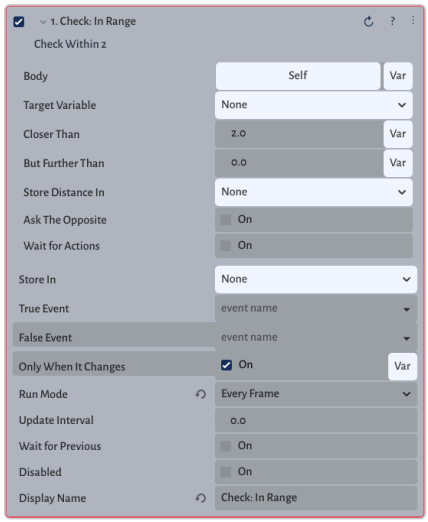
See the action work on its own with **Open Example** in the Add Action window, or open `examples/transform/follow_path/follow_path.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	What moves. Leave it empty, as it comes, for the thing this machine works on. A 2D thing must follow a 2D path and a 3D thing a 3D one. Mix them and it stays where it is while the progress and the Arrived event carry on as though it had travelled, and a warning names both nodes. Var: a Node.	Self
Path	The path node to follow, with its curve drawn in the scene. With no path, or a path with no curve on it, the action gives up straight away, the thing stays where it is, and a warning says whether the path is missing or has no line drawn. Var: a Node.	empty
Speed	How fast it travels along the curve, in pixels a second in a 2D game and metres a second in 3D. It is 3 to begin with, a walk in 3D. In a 2D game it starts at 96 pixels a second. Var: a float.	3.0 in 2D: 96 px
Going	Once, as it comes, runs from the start of the curve to the end and finishes the action there. Loop jumps back to the start and keeps going, for a closed loop. Ping Pong turns round at each end and never stops, as a lift or a patrolling platform does. Choices: Once, Loop, Ping Pong.	Once
Turn to Face	On to begin with, so the thing is turned to face the way it just moved and a car points along the track. Turn it off for something that keeps its facing, such as a lift or a floating platform. Var: a bool.	on
Arrived Event	An event sent when the end of the path is reached, and going back and forth, at each end. Empty, as it comes, sends nothing. Use it to make something happen at the far end without stopping the journey.	empty
Store Progress In	A variable kept up to date with how far along the path the thing is, from 0 at the start to 1 at the end. Empty, as it comes, keeps nothing. A progress bar or a check on "nearly there" can read it.	empty

In Range CHECK

Asks how near the character is to the thing its Perception node has found. It says yes while the gap is under a distance you set and, if you give one, over a second distance, which makes a band.



In Range as it appears in the Actions tab.

When to use it

An archer that should stand four to ten metres off asks Closer Than 10, But Further Than 4, and a guard starts swinging at Closer Than 2. It follows whatever the senses are on to, so one template works whoever the character is chasing.

> When the character's senses have not found anything, there is nothing to measure to and the answer is no.

See also

Distance 93, **Can See** 250, **Keep Distance** 255, **Chase** 251

Try it

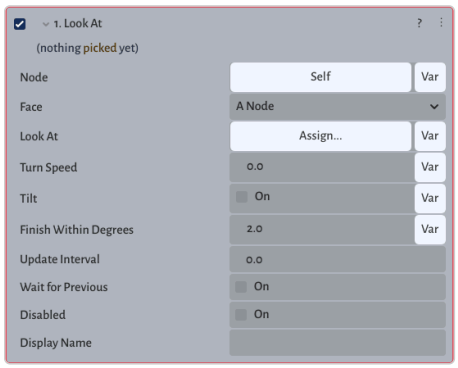
See the action work on its own with **Open Example** in the Add Action window, or open `examples/transform/in_range/in_range.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Body	The character being asked about, the one carrying the Perception node. Leave it empty for the node the machine works on. Var: a Node.	Self
Target Variable	Name a variable and the gap is measured to the node it holds, not to what the senses found. Empty, the default, follows the senses. Fill it in when another action has already chosen what to measure to, such as the nearest coin or the friend being escorted.	empty
Closer Than	Yes while the gap is this small or smaller, in metres in a 3D game and pixels in a 2D one. The default 2 is arm's reach in 3D. At 0 there is no near limit, and only the far limit below counts. Var: a float.	2.0 in 2D: 64 px
But Further Than	The gap must also be at least this big, turning the limit into a band such as "close, but not on top of me". The default 0 means no far limit. Var: a float.	0.0 in 2D: 0 px
Store Distance In	Name a variable and the gap goes into it as a number. A later action can use it, for instance to make a shot weaker at long range. Empty, the default, stores nothing.	empty
Ask The Opposite	Turns the answer round into "it is not in range", for a state that gives up and goes back to walking. It turns round the true-or-false answer the Check stores as well, which swapping the True Event and False Event over cannot do.	off
Wait for Actions	Holds the question back until every other action in the state has finished, so the distance is measured after the state has moved the character. Left off, the default, it is asked as soon as the state runs.	off
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Look At

Turns something to face another node, the mouse pointer, a place or a direction, snapping round in one step or turning at a set speed. It keeps facing while the state runs, so a turret set to the mouse follows the pointer wherever it goes.



Look At as it appears in the Actions tab.

When to use it

Anything that aims, such as an arrow over the player's head pointing at the next objective. With Face on A Direction it aims the way a stick is pushed and keeps its last aim when the stick is let go, as a twin-stick shooter aims.

> With Turn Speed at 0 it snaps round on its first frame and reports itself done on the next, and a state waiting for FINISHED moves on then.

See also

Rotate To 101, **Mouse Look** 191, **Get Input Vector** 143, **Chase** 251

Try it

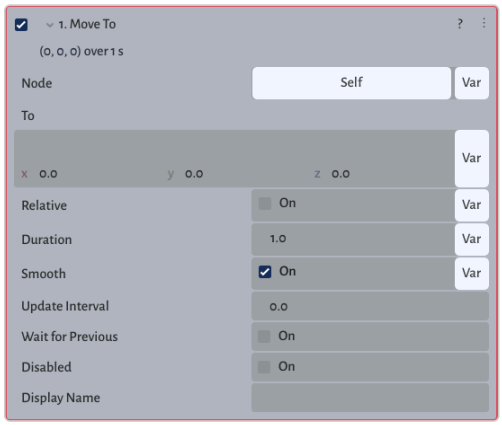
See the action work on its own with **Open Example** in the Add Action window, or open `examples/transform/look_at/look_at.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node that turns. Leave it empty, as it comes, for the thing this machine works on. Put the gun's node here to swing only the gun and leave the body alone. Var: a Node.	Self
Face	A Node, as it comes, faces another node in the scene. The Mouse faces wherever the pointer is. A Place faces a fixed spot in the world, and A Direction points the way a direction points, such as a stick's. Choices: A Node, The Mouse, A Place, A Direction.	A Node
Look At	The node to face, used when Face is A Node. Pick the player, an enemy or a marker. With nothing picked there is nothing to face. The action finishes straight away, the node stays as it was, and a warning asks for something to face. Var: a Node.	empty
Turn Speed	Degrees a second. 0, as it comes, snaps round to face it in a single step. At 180, a turret swings round slowly enough for a quick player to get behind it. Var: a float.	0.0
Tilt	Off to begin with, so a 3D thing stays level and turns only left and right. Tick it to tilt up and down as well, for a turret that can aim at something above it. It makes no difference in a 2D game. Var: a bool.	off
Finish Within Degrees	How close counts as facing it, 2 degrees to begin with. Within this, the action reports itself done and the state can send FINISHED and move on, while the turning carries on for as long as the state runs. A larger number lets a slow turret start shooting before it is lined up. Var: a float.	2.0
Place	Only when Face is A Place. The spot in the world to face, as two numbers in a 2D game or three in a 3D one.	—
Direction	Only when Face is A Direction. Which way to point, read as the world's own directions. That is two numbers in a 2D game, and in a 3D one two numbers meaning across and forward, as Get Input Vector hands them over. A direction of almost no length, such as a stick sitting at rest, is ignored and the last aim is kept.	—

Move To

Moves something to a place over a number of seconds and then finishes. It works on a 2D thing, a 3D thing and a panel on the screen alike.



Move To as it appears in the Actions tab.

When to use it

For a lift going up 4 metres in 2 seconds, tick Relative and set To at 0, 4, 0 and Duration 2. A transition on FINISHED carries on when it arrives. A character that must feel gravity and bump into walls walks with Move Character.

> It can only move something that has a place, such as a 2D node, a 3D node or a panel on the screen. On anything else it finishes straight away, nothing moves, and a warning names the node and its type.

See also

Tween Property 106, **Follow Path** 94, **Translate** 105, **Rotate To** 101

Try it

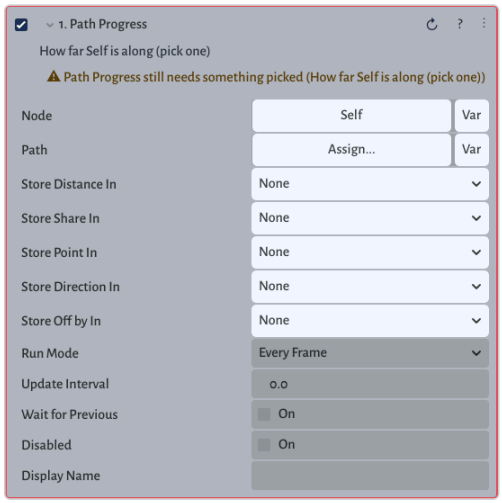
See the action work on its own with **Open Example** in the Add Action window, or open `examples/transform/move_to/move_to.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node to move. Leave it empty, as it comes, for the thing this machine works on. Put a door's node here to slide the door while the machine sits elsewhere. Var: a Node.	Self
To	Where to end up, as two numbers in a 2D game or on the screen, three in a 3D one. It is a spot in the world unless Relative is ticked, so typing 0, 0 sends the thing to the world's 0, 0 point (the top-left corner of the screen for a panel) rather than leaving it where it is. Var: a Vector2 or a Vector3, or a Node for where it stands.	0, 0, 0 in 2D: 0, 0
Relative	Off to begin with, so To is a place in the world. Ticked, To becomes an offset from wherever the thing starts. For example, 0, -100 lifts it a hundred pixels from where it is now, handy for a door that has been dragged into place in the scene. Var: a bool.	off
Duration	How many seconds the move takes, 1 to begin with. At 0 it goes there in a single step, useful for putting something back before a level starts. Var: a float.	1.0
Smooth	On to begin with, so the move eases gently into motion and gently to a stop, as a lift does. Off gives a constant speed, for a conveyor or a scrolling background. Var: a bool.	on
The Other Way	Only with Relative ticked. Go the other way by the same amount. A platform's Coming Back reads the same Travel as its Going and runs it backwards, so one setting changes both ways.	—

Path Progress

Finds out how far along a path something has got, however it got there, and where the nearest point of the path is. It works with a Path2D or a Path3D, the path nodes you draw a curve on.



Path Progress as it appears in the Actions tab.

When to use it

Something that moves freely along a route that matters, such as a rally car. The co-driver can call a corner 100 metres before the car reaches it, and a car off the road can be put back. The thing does not have to be on the path, because it is measured from the nearest point of it.

> A path that doubles back close to itself can make the nearest point jump from one part of it to the other, so draw routes that keep their distance from themselves.

See also

Follow Path 94, **Distance** 93, **Move To** 97, **Look At** 96

Try it

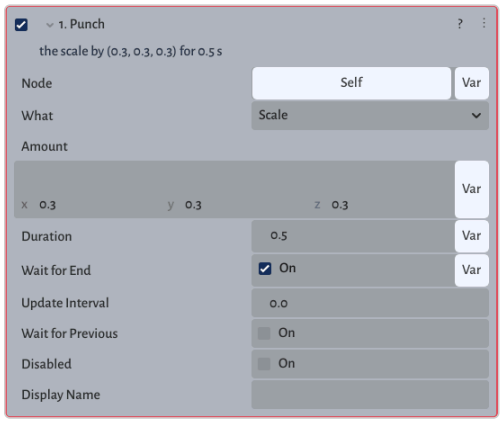
See the action work on its own with **Open Example** in the Add Action window, or open `examples/transform/path_progress/path_progress.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	What is measured. Leave it empty for the thing the machine works on, such as the car. Var: a Node.	Self
Path	The Path2D or Path3D to measure along, drawn from where the route starts to where it ends. With no path, or a path with no curve drawn on it, nothing is stored, and a warning says whether the path is missing or has no line drawn. Var: a Node.	empty
Store Distance In	Names a variable to keep how far along the path the thing is, in metres in 3D or pixels in 2D, counted from the start of the path. Empty keeps nothing. Compare it with the distance of the next corner to know when to call it.	empty
Store Share In	Names a variable to keep the share of the path covered, from 0 at the start to 1 at the end. Empty keeps nothing. A progress bar across the top of the screen can read it.	empty
Store Point In	Names a variable to keep the nearest point of the path, as a place in the world. Empty keeps nothing. Move a car there to put it back on the road.	empty
Store Direction In	Names a variable to keep the way the path runs at that point, as a direction pointing on toward the end. Empty keeps nothing. Use it to face a car back down the road after putting it there.	empty
Store Off by In	Names a variable to keep how far the thing is from the path, in metres in 3D or pixels in 2D. It is 0 on the path and more the further it has strayed. Empty keeps nothing. Use it to notice a car that has gone well off the road.	empty

Punch

Gives a node a sudden kick to its position, rotation or scale, and lets it spring back to where it was, wobbling as it settles. It works on a 2D node, a 3D node or a panel on the screen.



Punch as it appears in the Actions tab.

When to use it

The small touches of life that make a game feel good, such as a coin that squashes and stretches as it is picked up, or a door that rattles when it is locked. The kick is added on top of whatever else is moving the node, and a character keeps walking while it shudders. It always springs back. Move To and Scale To leave a node at its new place or size.

> Leaving the state part way through takes off what is left of the kick straight away, so the node is never left twisted or stretched. A panel whose pivot has never been moved grows and turns about its middle.

See also

Scale To 102, **Tween Property** 106, **Camera Shake** 186, **Move To** 97

Try it

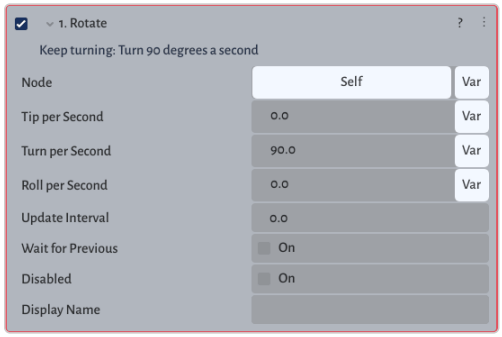
See the action work on its own with **Open Example** in the Add Action window, or open `examples/transform/punch/punch.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node to punch. Leave it empty, as it comes, for the thing this machine works on. Point it at a character's picture, not its body, when only the picture should squash. Var: a Node.	Self
What	Which part to kick. Position knocks it aside and back, Rotation twists it and back, and Scale, the default, squashes or stretches it and back. Choices: Position, Rotation, Scale.	Scale
Amount	How big the kick is, added to what the node has. For Scale, 0.5, -0.4 makes it half as wide again and a little shorter for a moment. For Position it is how far to knock it, in metres in a 3D game or pixels in a 2D one. For Rotation it is degrees, where a 2D node or a panel uses the first number and a 3D node tips, turns and rolls by all three. A single number stretches a node the same amount every way. Var: a Vector2 or a Vector3, or a Node for where it stands.	0.3, 0.3, 0.3 in 2D: 0.3, 0.3
Duration	How many seconds it takes to spring back and settle, half a second to begin with. Longer wobbles for longer. At 0 there is no time to spring back and no kick shows. A warning in the Output panel points this out. Var: a float.	0.5
Wait for End	On, as it comes, the action finishes only once the node has settled, and a transition on FINISHED waits for the wobble. Off, it finishes straight away and the wobble plays on by itself, even after the state has been left. Var: a bool.	on

Rotate

Keeps something turning at a steady speed, in degrees a second, for as long as the state runs. It does not hold the state up, because it starts the turning and reports itself done straight away.



Rotate as it appears in the Actions tab.

When to use it

Spinning coins, fans and radar dishes. The speed is the same on any computer, however fast it draws.

> It never stops by itself. Leaving the state is what stops it.

See also
Rotate To 101, **Translate** 105, **Look At** 96, **Tween Property** 106

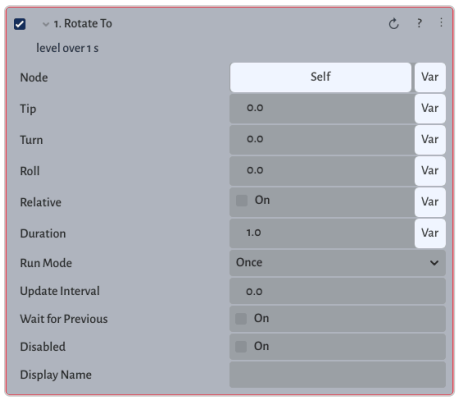
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/transform/rotate/rotate.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node to turn. Leave it empty, as it comes, for the thing this machine works on, or point it at the model inside a pickup to spin only the picture. Var: a Node.	Self
Degrees per Second	Only for a 2D thing or a panel on the screen. How fast it turns, 90 to begin with. Above zero turns clockwise and below zero the other way. 360 is a full turn every second. Var: a float.	90.0
Tip per Second	Only for a 3D thing. Degrees a second to tip it forward or back, around its X axis. 0, as it comes, leaves it level. Var: a float.	0.0
Turn per Second	Only for a 3D thing. Degrees a second to turn it left or right, around its up axis, 90 to begin with. Think of the spin of a pickup or a coin standing on its edge. Var: a float.	90.0
Roll per Second	Only for a 3D thing. Degrees a second to roll it around its forward axis, like a wheel seen from the side. Var: a float.	0.0

Rotate To

Turns something to an angle, in degrees. Once, it turns there over a number of seconds, the short way round, and then finishes. On Every Frame it is set to the angle each frame and follows a variable that keeps changing.



Rotate To as it appears in the Actions tab.

When to use it

A door swinging open with Turn at 90 over one second. On Every Frame, a dial faces whatever angle a variable holds while a Set Variable adds to it. A flat thing has one Angle, and a 3D thing has three numbers, Tip, Turn and Roll.

> Once reads its angles as the turn starts. To follow an angle that keeps changing, set Run Mode to Every Frame.

See also

Rotate 100, **Look At** 96, **Set Variable** 49, **Tween Property** 106

Try it

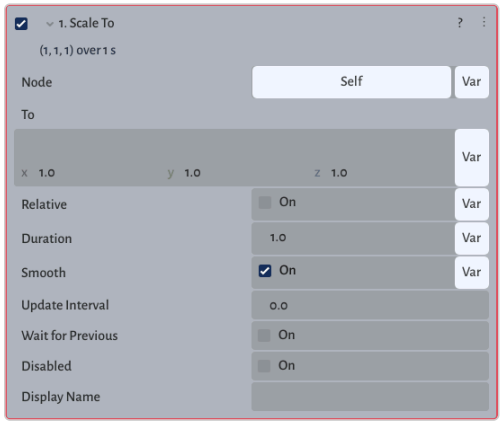
See the action work on its own with **Open Example** in the Add Action window, or open `examples/transform/rotate_to/rotate_to.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node to turn. Leave it empty, as it comes, for the thing this machine works on. Var: a Node.	Self
Angle	Only for a 2D thing or a panel on the screen. The angle to end up at, in degrees. 90 is a quarter turn clockwise on the screen and 0 is how it was drawn. Press Var beside it to read a float variable, such as an angle a Set Variable keeps adding to. Var: a float.	0.0
Tip	Only for a 3D thing. Degrees to tip it forward or back, around its sideways axis, the way a head nods or a drawbridge lowers. Above zero tips the front up, and a camera looks down with a Tip below zero, such as -30. It is measured from level, around the thing's own sideways axis however it has been turned. Var: a float.	0.0
Turn	Only for a 3D thing. Degrees to turn it left or right, around its up axis, the way a door swings or a turret pans. Above zero turns it to the left, seen from above. Most turns need only this one. Press Var to read a float variable. Var: a float.	0.0
Roll	Only for a 3D thing. Degrees to roll it to one side, around its forward axis, the way a plane banks or a head tilts. Above zero lifts its right side, rolling it to the left. Var: a float.	0.0
Relative	Only on Once. Off to begin with, so the angles are where it ends up. Ticked, they are how far to turn from wherever it starts, and a relative turn goes as far as you ask, so 360 is a full spin. Var: a bool.	off
Duration	Only on Once. How many seconds the turn takes, 1 to begin with. At 0 it turns in a single step. Var: a float.	1.0

Scale To

Grows or shrinks a node to a size over a number of seconds and then finishes. The size is the node's own scale, where 1 is the size it was made at, 2 is twice that and 0.5 is half.



Scale To as it appears in the Actions tab.

When to use it

A pickup swells as it appears, and an enemy shrinks away to nothing as it dies. It works like Move To and Rotate To, and one state can grow, move and turn something together. A quick squash that springs straight back is Punch's job.

> A RigidBody2D cannot change size, because Godot's physics keeps it at its own. Scale the picture under it. The warning in the Output panel says the same.

See also

Punch 99, **Move To** 97, **Rotate To** 101, **Tween Property** 106

Try it

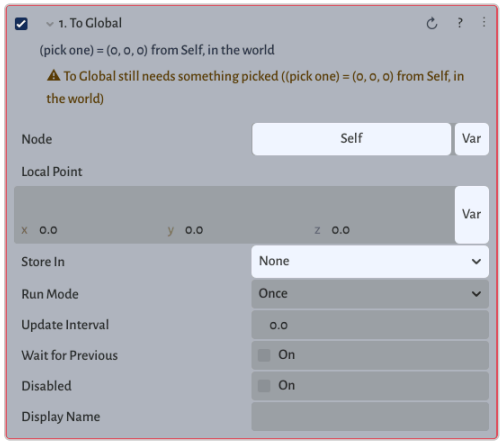
See the action work on its own with **Open Example** in the Add Action window, or open `examples/transform/scale_to/scale_to.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node to scale. Leave it empty, as it comes, for the thing this machine works on. Var: a Node.	Self
To	The scale to end at, with one number for each direction. That is two in a 2D game or on the screen, and three in a 3D one. A single number scales it the same way in every direction. It starts at 1, 1, 1, the node's own size. Var: a Vector2 or a Vector3, or a Node for where it stands.	1, 1, 1 in 2D: 1, 1
Relative	Off to begin with, so To is the size to end at. Tick it and To becomes a multiple of the size the node has when the state starts, so 2 doubles it and 0.5 halves it. Var: a bool.	off
Duration	How many seconds the change takes, one to begin with. Zero sets the size in a single step. Var: a float.	1.0
Smooth	On to begin with, and the change eases in and out. Off changes it at a steady rate all the way. Var: a bool.	on
The Other Way	Only with Relative ticked. Divides by To in place of multiplying, so a node doubled by one state comes back to its own size in another with the same number. A 0 in To cannot be divided by, and a warning in the Output panel reports it.	—

To Global

Turns a place in a node's own space, measured from the node along the directions it faces, into a place in the world. The answer follows the node as it moves, turns and grows. To Local turns it back.



To Global as it appears in the Actions tab.

When to use it

Finds the spot two metres in front of a gun where a bullet should start, or a place beside a door for a guard to walk to, wherever the door stands and whichever way it faces. On Every Frame it keeps the answer up to date as the node moves. In a 3D game a node's front is along minus Z, making two metres in front 0, 0, -2. In a 2D game its right is plus X.

See also

To Local 104, **Instantiate** 84, **Move To** 97, **Look At** 96

Try it

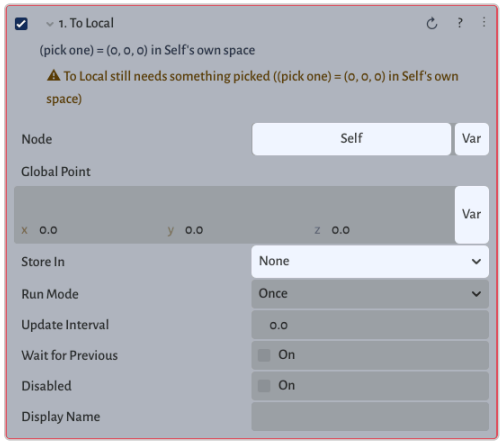
See the action work on its own with **Open Example** in the Add Action window, or open `examples/transform/to_global/to_global.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node whose own space the place is in. Leave it empty, as it comes, for the thing this machine works on. Var: a Node.	Self
Local Point	The place in the node's own space, as two numbers in a 2D game or three in a 3D one. 0, 0, -2 is two metres in front of a 3D node, and 200, 0 is two hundred pixels to a 2D node's right. 0, 0 is the node itself. Var: a Vector2 or a Vector3, or a Node for where it stands.	0, 0, 0 in 2D: 0, 0
Store In	The variable the place in the world goes into, ready for Move To, Look At or Instantiate to use. Left empty, the answer goes nowhere, and you are warned in the Output panel.	empty

To Local

Turns a place in the world into a place in a node's own space, measured from the node along the directions it faces. The answer follows the node as it moves, turns and grows. To Global turns such a place back into the world.



To Local as it appears in the Actions tab.

When to use it

Answers where something is compared with a node, not where it is in the level. It can tell whether an enemy is in front of the player or behind, or where on a moving platform something is standing. In a 3D game a node's front is along minus Z, and something in front of it comes out with a Z below 0. In a 2D game the node's right is plus X and its down is plus Y.

> A node scaled to 0 in some direction has squashed its space flat, and no place in the world can be found in it. A warning is printed and the variable is left alone.

See also
To Global 103, **Vector** 53, **Get Vector Components** 50, **Mouse Position** 150

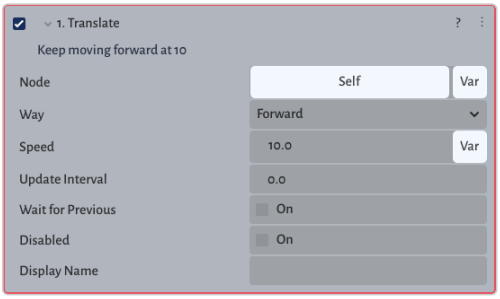
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/transform/to_local/to_local.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node whose own space to measure in. Leave it empty, as it comes, for the thing this machine works on. Var: a Node.	Self
Global Point	The place in the world, as two numbers in a 2D game or three in a 3D one. Press Var to read it from a variable, such as where the mouse is from Mouse Position. A node kept in a variable stands for where that node is. Var: a Vector2 or a Vector3, or a Node for where it stands.	0, 0, 0 in 2D: 0, 0
Store In	The variable the answer goes into, as two or three numbers the same way round as the place. Left empty, the answer goes nowhere and the Output panel warns about it.	empty

Translate

Keeps something moving at a steady speed for as long as the state runs, the way it faces or along a direction you give. It does not hold the state up. It starts the movement and reports itself done straight away.



Translate as it appears in the Actions tab.

When to use it

A bullet flying the way it was fired, or boxes riding a conveyor. Forward is the way the thing faces, so a bullet made at a turned gun flies the way the gun points with no sums at all.

> A physics body is handed the speed and does its own collisions, and walls still stop it. Anything else is only moved, and goes straight through everything.

See also

Move To 97, **Set Velocity** 122, **Follow Path** 94, **Move Character** 134

Try it

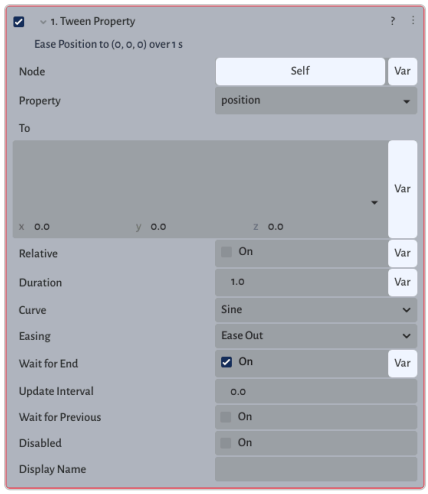
See the action work on its own with **Open Example** in the Add Action window, or open `examples/transform/translate/translate.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node to move. Leave it empty, as it comes, for the thing this machine works on, such as a bullet that moves itself. Var: a Node.	Self
Way	Forward, as it comes, means the way the thing faces, which is to its right in a 2D game and away from the screen in 3D. A Direction means a fixed direction in the world, the same whichever way the thing points. Choices: Forward, A Direction.	Forward
Speed	How fast, in pixels a second in a 2D game and metres a second in 3D. It is 10 to begin with, brisk in 3D. In a 2D game it starts at 320 pixels a second, and a bullet takes several hundred. Var: a float.	10.0 in 2D: 320 px
Direction	Only when Way is A Direction. The direction to travel, as two numbers in a 2D game or three in a 3D one, read as the world's own directions. Its length does not matter, only which way it points. Speed decides how fast.	—

Tween Property

Eases one of a node's settings from what it is now to a value you give, over a number of seconds, along a curve you choose. Moving, turning, growing and fading are all settings, and this one action covers them.



Tween Property as it appears in the Actions tab.

When to use it

A menu panel sliding onto the screen, a door dropping shut, or anything else that should change smoothly. Walking a node across the level is Move To's job, since it is built around going to a place.

> With Wait for End on, leaving the state stops the movement where it is, so a door caught half shut stays half shut.

See also

Set Property 91, **Move To** 97, **Rotate To** 101, **Ease Variable** 44

Try it

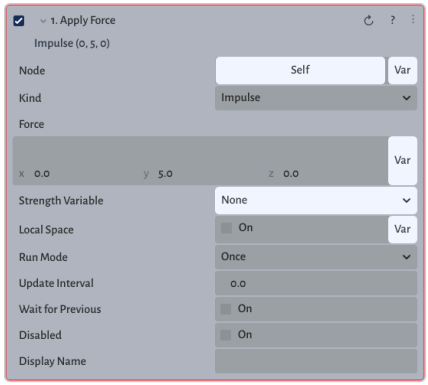
See the action work on its own with **Open Example** in the Add Action window, or open `examples/transform/tween_property/tween_property.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node to change. Leave it empty for Self, the node this Machine works on. Press Var beside the box to ease whatever node a variable is holding. Var: a Node.	Self
Property	Which setting is eased. Press the little arrow beside the box and pick from the list. It reads the node named in Node and offers only the settings that can sensibly be eased, such as Position, Rotation, Scale and Modulate, written as the Inspector writes them. Tick boxes and whole numbers are not offered, because there is nothing in between them to ease through. It starts on Position.	position
To	The value it should end at. Press the button beside the box to say what kind of value it is, or press Var to read it from a variable. A place typed with three numbers works for a 2D setting too, and the third number is left out. If the kind cannot be made to fit what the setting holds, nothing is eased and a warning says so. Var: any kind, as the property picked.	0, 0, 0
Relative	Off, the value above is where it ends up. On, it is a change from where the setting stands now, so a To of 100 on a position moves it 100 further along, not to 100. Right for a door that drops by its height wherever it stands. Var: a bool.	off
Duration	How many seconds the change takes. 1 second, the default, reads as a deliberate movement. 0.2 is a snap and 3 slow and grand. At 0 the value is set straight away, with no easing. Var: a float.	1.0
Curve	The shape of the change. Linear moves at one speed throughout. Sine, Quad, Cubic and Expo each gather and shed speed more sharply than the last. Elastic goes past the end and wobbles back, Back leans past the end and settles, and Bounce lands and bounces. It starts on Sine, which suits most things. Choices: Linear, Sine, Quad, Cubic, Expo, Elastic, Back, Bounce.	Sine
Easing	Where the movement is slow. Ease Out, the default, starts quickly and settles softly, and suits most movement on screen. Ease In sets off softly and speeds up, good for something setting off. Ease In Out is soft at both, and Ease Out In is quick at both. Choices: Ease Out, Ease In, Ease In Out, Ease Out In.	Ease Out
Wait for End	On, the action is not finished until the movement is, so a state with nothing else in it sends FINISHED once the door has finished closing. Off, it finishes straight away and the movement carries on by itself, even after the state has been left. That is good for a fade that should hold nothing up. Var: a bool.	on

Apply Force

Gives a physics body a push. It can be a steady shove, a single sharp one, or a spin.



Apply Force as it appears in the Actions tab.

When to use it

For things that get thrown about, such as a barrel kicked across a cellar, a ball struck, or a player knocked backwards by a blow. It is meant for a rigid body, the kind physics moves by itself, like a crate or a ball. On a character body, the kind a walking player or guard uses, the push goes into the body's speed. That only shows if a movement action such as Move Character or Chase is walking it in the same state or a parent state above it. A spin cannot turn a character body, and you are warned in the Output panel. On an area or on static scenery nothing is pushed, and that gets a warning too.

> A spin does nothing to a character body. On a 2D rigid body it comes from the single Spin number, on a 3D one from Force's three.

See also

Set Velocity 122, **Explosion** 110, **Recoil** 193, **Jump** 132

Try it

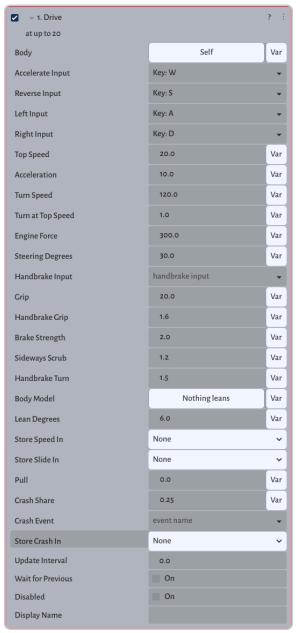
See the action work on its own with **Open Example** in the Add Action window, or open `examples/physics/apply_force/apply_force.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The body to push. Leave it empty to push the node this machine works on, for a knockback on the machine's own character. Anything that is not a rigid body or a character body is not pushed, and a warning in the Output panel gives the reason. Var: a Node.	Self
Kind	Impulse, the default, is one shove, the sort a blow or an explosion gives. Force is a gentle push spread out over time, for wind, a conveyor or a rocket burn. Give it Run Mode Every Frame, because on Once it pushes for a single physics step, barely a nudge, and a warning points this out. Torque turns it, and only a rigid body can be spun. On Once it is one sudden spin, and on Every Frame a steady twist that keeps spinning it faster. Choices: Force, Impulse, Torque.	Impulse
Force	Which way to push and how hard. That is two numbers in a 2D game, across and down, and three in a 3D one. For a spin in 3D the three numbers are the turn about each axis. The default, 0, 5, 0, pushes up in a 3D game and down in a 2D one, so type the direction and strength you want. Var: a Vector2 or a Vector3, or a Node for where it stands.	0, 5, 0 in 2D: 0, -160
Strength Variable	Name a variable and the push's strength is read from it each time, keeping only the direction you typed into Force. A charged shot can hit harder the longer it is held, or a gust of wind come and go. Empty, as it starts, uses the strength you typed. Once a variable is named only the direction of Force counts, not its length.	empty
Local Space	Off, the numbers are the world's directions, so up is always up, whichever way the body faces. On, they are the body's own directions, and a forward push always comes out of its front, right for a kick or a thruster. Var: a bool.	off
Torque	Only for Torque on a 2D body. It sets how hard to spin it, clockwise above zero and anticlockwise below. A flat thing turns about one axis only, hence one number. Force is put away while this shows.	—

Drive

Drives a vehicle with the keys or a gamepad. It speeds up, brakes into reverse, and steers, turning more the faster it goes and the right way round when backing up. Its speed is kept on the vehicle, and leaving the state and coming back later does not stop the car dead in the road.



When to use it

For a car, a kart, a boat or a tank the player is in charge of. It handles two kinds of vehicle. On a VehicleBody3D, Godot's car on wheels, it works the engine, the brake and the steering. On a character body it drives the arcade way, so speed and turning are worked out, not simulated, and the car's wheel models are turned to match. Any part under the car named wheel, tyre or tire counts. The front ones steer and all of them roll. Point Body Model at the car's body and it leans in the corners too. Give it a Handbrake and the car can be thrown sideways into a corner, as a rally car is. When an arcade car hits something it keeps only the speed that ran along the wall and loses the rest. Anything else that stops it, such as Set Velocity putting it back on the road, stops its driving speed too.

> Driving never holds a state up. Put it on the state that lasts for as long as the player is at the wheel.

See also

Move Character 134, **Apply Force** 107, **Get Input Vector** 143, **Set Velocity** 122

Try it

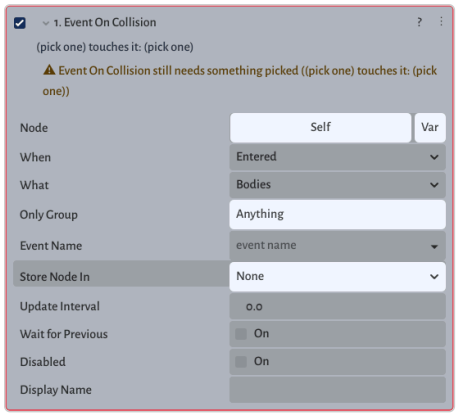
See the action work on its own with **Open Example** in the Add Action window, or open `examples/physics/drive/drive.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Body	The vehicle. Leave it empty for the node this machine works on, when the machine sits on the car. It must be a VehicleBody3D or a character body. Var: a Node.	Self
Accelerate Input	What speeds it up. That can be a key, a gamepad button or trigger, or an input action you named yourself. A trigger pulled half way gives half the throttle. W to begin with.	Key: W
Reverse Input	What slows it down, and then drives it backwards once it has stopped. Braking is twice as quick as speeding up. S to begin with.	Key: S
Left Input	What steers it left. A to begin with. A steering stick pushed half way steers half as far.	Key: A
Right Input	What steers it right. D to begin with, and it should be the same kind of control as Left Input so the two match.	Key: D
Top Speed	The fastest it goes forward, in metres per second in 3D or pixels per second in 2D. Reversing tops out at half of it. A VehicleBody3D is held to it too, as the engine stops pulling once it is reached. Var: a float.	20.0 in 2D: 640 px
Acceleration	How quickly it reaches top speed, and how quickly it coasts back down when nothing is pressed, in speed gained each second. About half the top speed feels heavy, several times it feels sharp. It is used for a character body. A VehicleBody3D gets its shove from Engine Force instead. Var: a float.	10.0 in 2D: 320 px
Turn Speed	How quickly a character body turns at full lock, in degrees each second. Standing still it does not turn at all, like a real car. A VehicleBody3D ignores this and steers by Steering Degrees. Var: a float.	120.0
Turn at Top Speed	How much of Turn Speed a character body still has at Top Speed, as a share. At 1, the default, it turns as sharply flat out as at a crawl. No real car can do that. At 0.5 it has half its turn left flat out, and a fast corner has to be braked for or taken by throwing the tail round with the handbrake. The cut eases in from a third of top speed, leaving slow driving unchanged. Var: a float.	1.0
Engine Force	Only in a 3D game, since a VehicleBody3D is a 3D body and a 2D game can have none. How hard the engine pulls, and how hard the brake bites. A heavier vehicle needs more, or it will barely move. It does nothing for a character body, which gets its shove from Acceleration instead. Var: a float.	300.0
Steering Degrees	Only in a 3D game, since a VehicleBody3D is a 3D body and a 2D game can have none. How far the front wheels turn at full lock, in degrees. Thirty is a road car. Much more and it darts about and tips over. A character body turns by Turn Speed instead, and only shows this on its wheel models. Var: a float.	30.0
Handbrake Input	What pulls the handbrake. Empty, the default, means no handbrake at all. On a car with wheels it cuts the drive and takes the grip out of the back wheels. On an arcade car it lets the car keep going the way it was while the nose comes round. It is not a brake pedal. Its job is to bring the tail round, and most of what a handbrake turn costs is scrubbed off by going sideways.	empty
Grip	Only in a 3D game. How quickly the car comes round to pointing where it is really going, in turns a second, for an arcade car. High, the default, keeps it on rails, like a kart or a road car. Lower it and the back runs wide in a fast corner, the feel of a loose surface. Var: a float.	20.0
Handbrake Grip	Only in a 3D game. The same while the handbrake is pulled. Small numbers let the back step right out, the way a rally car gets round a hairpin. At the default the car will sit sideways and still carry its speed through the turn. On a car with wheels this is how much grip the back wheels keep. Var: a float.	1.6
Brake Strength	How much harder the brake is than the accelerator. At 2, the default, it stops twice as quickly as it gets going. Raise it for something light and grippy, lower it for something heavy that has to be slowed early. Var: a float.	2.0
Sideways Scrub	Only in a 3D game, since a 2D car always goes the way it points. How quickly going sideways scrubs speed off, for an arcade car. This, rather than the handbrake, is what a slide costs you. Raise it and a drift kills your speed, lower it and the car carries its momentum through sideways. Var: a float.	1.2
Handbrake Turn	How much more the car turns while the handbrake is pulled, for an arcade car. At 0 it only breaks the grip, and higher throws the tail round harder. This and Handbrake Grip are most of how a handbrake turn feels. Var: a float.	1.5
Body Model	The car's body model, for an arcade car. Point it at the model and it leans over to the outside of a corner and dips its nose under braking, the way a body does on its springs, while the wheels stay flat on the ground. Empty leans nothing. A VehicleBody3D has real springs and leans by itself. Var: a Node.	empty
Lean Degrees	How far the body model leans in a hard corner, in degrees, for an arcade car. It dips half as far under hard braking and lifts its nose the same under power. Six, the default, is a road car on soft springs, and a couple of degrees is a stiff rally car. 0 keeps it rigid. Var: a float.	6.0
Store Speed In	Name a variable to keep the vehicle's speed in, positive going forward and negative backing up, for a speedometer or to pick an engine sound. Empty keeps nothing.	empty
Store Slide In	Only in a 3D game, since a 2D car always goes the way it points. Names a variable to keep how far sideways the car is going in, 0 pointing where it is going and 1 fully sideways. Empty keeps nothing. Use it to throw up smoke and leave marks only while the car is really sliding, and to score a drift.	empty
Pull	Steering the car does by itself, from -1 hard left to 1 hard right, added to whatever the driver is doing. It could be a puncture dragging the car to one side, a crosswind, or a bent track rod after a crash. 0, the default, runs straight. Press Var to read it from a variable, so damage can set it and a repair can put it back. Var: a float.	0.0
Crash Share	How much speed the vehicle has to lose at once for a hit to count as a crash, as a share of Top Speed. The default, 0.25, is a quarter of it. Scraping along a wall or brushing a post costs a little speed and is not a crash, but running into a tree is. Raise it for a car that shrugs off knocks. Var: a float.	0.25
Crash Event	An event sent when the vehicle crashes, that is, when one hit takes off at least Crash Share of its top speed. Empty, the default, sends nothing. Lead it to a state that dents the car, plays a crunch or spins it round.	empty
Store Crash In	Names a variable to keep how much speed the last crash took off, in metres a second in 3D or pixels a second in 2D, so damage can grow with how hard it hit. Empty keeps nothing. It is only written when there is a crash.	empty

Event On Collision

Sends an event as something touches this node, or stops touching it, and remembers what it was for the next action to use.



Event On Collision as it appears in the Actions tab.

When to use it

This is how a trigger works, whether the player steps into a doorway, a bullet meets a wall, or an enemy leaves its room. Point it at an Area2D or an Area3D, an invisible zone that notices what passes through without blocking anything, or at a rigid body, which it quietly switches over to reporting what it touches. A walking character body or static scenery cannot report a touch. Pointed at one of those it never sends an event, and a warning in the Output panel names the node. Hang a small area under that node and point at the area. Get Overlapping asks what is inside an area at a moment you choose, not as it happens.

> It only listens while its state is on, so put it on a state that stays on for as long as the trigger should work.

See also

Get Overlapping 113, **Touching** 125, **Explosion** 110, **Raycast** 116

Try it

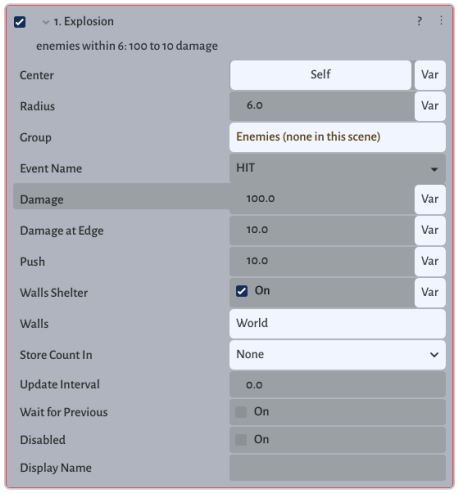
See the action work on its own with **Open Example** in the Add Action window, or open `examples/physics/event_on_collision/event_on_collision.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The area or body that does the watching. Leave it empty for the node this machine works on. It must be an area or a rigid body. Anything else can never report a touch, and a warning in the Output panel names it. Var: a Node.	Self
When	Entered fires as something arrives, Exited fires when something that was touching is no longer touching. Use two of these actions, one of each, for a pressure plate that stays down only while somebody stands on it. Choices: Entered, Exited.	Entered
What	Bodies watches physics bodies, like characters, crates and the ground. Areas watches other areas, such as another trigger or a sword's swing. Godot keeps the two apart, so pick the wrong one and no touch is ever heard. Choices: Bodies, Areas.	Bodies
Only Group	Only react to what is picked here, so a door can ignore bullets and crates and open only for the player. Press the box and pick it in the scene. You can pick one thing by its name, Every Coin for every copy of the coin scene (including ones made while playing), or Everything under a node for a set of markers. NodeWorthy does the rest. Anything, as it starts, reacts to anything that touches.	empty
Event Name	The event sent when it happens, which a transition on this state can catch. Leave it empty and no event is sent. With Store Node In filled in, it still remembers what touched.	empty
Store Node In	Name a variable and whatever touched is kept in it, ready for the next action to destroy it, count it or move it. It is written afresh each time something touches. Act on it straight away.	empty

Explosion

Works out an explosion's reach. Everything in a group within the radius is hurt, most at the centre and least at the edge. It is hurt through its Hurt Box when it has one, or else by an event carrying the damage. Rigid bodies nearby are thrown outward, and walls can shelter whatever is behind them.



Explosion as it appears in the Actions tab.

When to use it

This is the damage half of a rocket, a grenade or a barrel going up. Put it beside whatever draws the fireball. The centre can be any node with a place in the world, 2D or 3D, on any kind of body. The group decides what gets hurt, not the node kind. A group is a label you can hang on any number of nodes, and each labelled node needs a machine. One without is passed over. A node with a Hurt Box takes the damage through it, as from a Hit Box, and is not sent the event. Only rigid bodies are thrown. A walking character never is, so add an Apply Force if the player should be knocked back too.

> Each machine is told once however many of its nodes are in the group, and the centre's own node, and anything it hangs beneath, are never hurt by their own explosion.

See also
Apply Force 107, **Get Overlapping** 113, **Send Event** 31, **Instantiate** 84

Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/physics/explosion/explosion.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Center	The node the explosion is measured from. Leave it empty for the node this machine works on, when the machine sits on the rocket or the barrel itself. If that node has gone by the time the blast goes off, nothing happens and a warning is printed in the Output panel. Var: a Node.	Self
Radius	How far the blast reaches, in metres in 3D or pixels in 2D. Anything further away than this is untouched. At 0 or below it reaches nothing, and the Output panel warns about it. Var: a float.	6.0
Group	Who it hurts, so a blast can be aimed at enemies and leave the scenery alone. Press the box and pick it in the scene. You can pick one thing by its name, Every Coin for every copy of the coin scene (including ones made while playing), or Everything under a node for a set of markers. NodeWorthy does the rest. Every Machine In Reach catches every machine within the radius, whatever its group. Put two of these actions in one explosion if the enemies and the player should take different damage.	enemies
Event Name	The event each machine caught in the blast is sent, HIT to begin with. The damage travels with it as its value, so the transition that catches it can put the number straight into a variable. The thing being hurt needs a transition listening for this event, or the blast does nothing to it. Something with a Hurt Box is not sent it, because the Hurt Box takes the damage instead. Empty sends nothing, for a blast that only pushes.	HIT
Damage	The damage for something standing right at the centre. This is the biggest number the blast can send. Var: a float.	100.0
Damage at Edge	The damage for something at the very edge of the radius, with everything in between falling away evenly. Set it the same as Damage for a blast that hurts everything inside it alike. Var: a float.	10.0
Push	How hard rigid bodies inside the radius are thrown outward. It is hardest at the centre, fading to 0 at the edge. The throw is matched to each body's weight, and a heavy crate and a light one fly off at the same speed. At 0 nothing is thrown, for a blast that only does damage. Var: a float.	10.0
Walls Shelter	Ticked, as it starts, anything with something solid between it and the centre is spared, such as a wall, a floor or a closed door. The shelter holds back the throw as well as the damage, in 2D and in 3D. Untick it for a blast that reaches through everything. Var: a bool.	on
Walls	Which layers count as walls for sheltering. It begins with World, where the walls and ground stand. Tick any other layer your level's solid parts sit on. With no layer ticked there is no shelter and every blast reaches everywhere. It is only read when Walls Shelter is ticked.	World
Store Count In	Name a variable and it is filled with how many were caught, ready for a transition that rewards a multiple kill or plays a bigger sound. Empty stores nothing.	empty

Freeze

Holds a rigid body still where it is, lets it go again, or switches gravity off and back on for it.



Freeze as it appears in the Actions tab.

When to use it

Good for a crate that stays put until it is hit, a boulder waiting for the player to walk past, or a block caught by an ice spell. Only a rigid body can be frozen, the kind physics moves by itself, like a crate, a barrel or a ball. A walking character body, static scenery or an area is left alone, with a warning. To stop a character walking, take the Move Character out of its state. To make one lighter or heavier, use Set Gravity.

> Gravity On only does anything to a body whose gravity was switched off by this action earlier, so it cannot be used to make a body heavier than it was.

See also
Set Gravity 120, **Set Collision** 119, **Apply Force** 107, **Set Velocity** 122

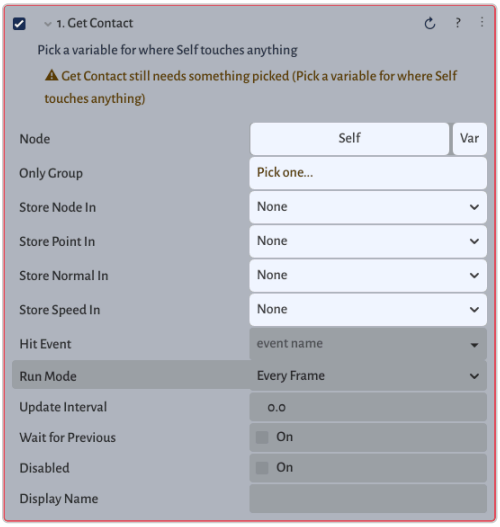
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/physics/freeze/freeze.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The rigid body. Leave it empty for the node this machine works on, such as a crate carrying the machine. Anything that is not a rigid body is refused with a warning. Var: a Node.	Self
Change	Freeze holds it where it is, though things still bump into it, and Unfreeze lets it move again. Gravity Off makes it stop falling and float wherever it is pushed, and Gravity On gives it back the weight it had before. Choices: Freeze, Unfreeze, Gravity Off, Gravity On.	Freeze
For Seconds	Change it back by itself after this many seconds, for an ice spell that wears off or a floating crate that settles. It counts game time and stops while the game is paused. At 0, the default, the change stays until something else undoes it. Var: a float.	0.0

Get Contact

Finds out where a rigid body is touching something, and how hard. It gives the node it touches, the point where they meet, the way that surface faces, and how fast they came together. It can send an event each time the body starts touching something new.



Get Contact as it appears in the Actions tab.

When to use it

Measures how hard things meet, so a crate's thud can grow louder the harder it lands, and a fall that was too high can do damage. It works on a rigid body, one moved by the physics, not by its own actions. A character body needs Get Last Collision.

> A rigid body only records what it touches when asked to. The first time this runs on one it switches that on itself, and the first answer comes a physics step or two later.

See also

Get Last Collision 129, **Event On Collision** 109, **Touching** 125, **Play Sound** 175

Try it

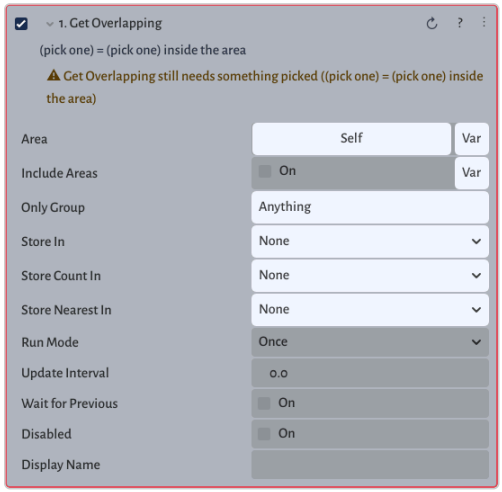
See the action work on its own with **Open Example** in the Add Action window, or open `examples/physics/get_contact/get_contact.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The rigid body, 2D or 3D. Leave it empty, as it comes, for the thing this machine works on. Anything else keeps no record of what it touches, and the Output panel warns about it. Var: a Node.	Self
Only Group	Only count what is picked here, so the floor a crate rests on does not hide the thing that struck it. Press the box and pick it in the scene. Left as Anything, it counts everything.	empty
Store Node In	The variable the node it touches goes into. It is emptied while the body touches nothing. Touching several things at once, it is the one it met fastest.	empty
Store Point In	The variable the point where they meet goes into, a place in the world. The last one is kept while the body touches nothing, so a spark can still be made there.	empty
Store Normal In	The variable the way the touched surface faces goes into, pointing out of it toward the body, such as up from a floor it rests on. Instantiate can use it to stand sparks or a dent on the surface.	empty
Store Speed In	The variable how fast they came together goes into, in metres a second in a 3D game or pixels a second in a 2D one. A crate that lands at seven metres a second gives 7, and resting on the floor it is close to 0. Use it for a thud's loudness, or for damage once it passes a limit.	empty
Hit Event	The event sent when the body starts touching something it was not touching a moment before, such as each bounce of a ball, a crate landing or a rock striking a wall. What it was already touching when it was first asked does not count, nor does the floor it is still resting on when its state starts again.	empty

Get Overlapping

Asks an area what is inside it at this moment and puts the answer in a list, with how many there are and which is nearest.



Get Overlapping as it appears in the Actions tab.

When to use it

For knowing what is there at a moment you choose, such as every enemy standing in a blast, everything on a pressure plate, or the nearest coin within reach. Event On Collision tells you as something arrives, but this asks when the state asks. It needs an Area2D or an Area3D, the invisible zone that notices what passes through it. On a walking character, a rigid body or static scenery the answer is always empty, and a warning in the Output panel names that node. Hang a small area under that node and point at the area.

> An area only notices things on the layers it is set to bump into. If the list is always empty, check those layers first.

See also

Event On Collision 109, **Explosion** 110, **For Each** 36, **Distance** 93

Try it

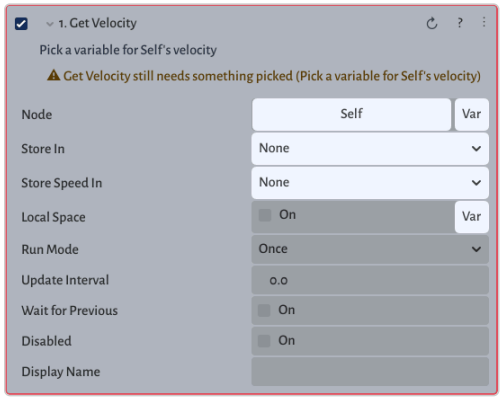
See the action work on its own with **Open Example** in the Add Action window, or open `examples/physics/get_overlapping/get_overlapping.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Area	The area to look inside. Leave it empty for the node this machine works on, when the machine sits on the trigger itself. It must be an area. Anything else can never say what is inside it, and a warning in the Output panel names it. Var: a Node.	Self
Include Areas	Off, the default, counts physics bodies only, like characters, crates and the ground. Ticked, other areas are counted as well, such as another trigger or a sword's swing. Var: a bool.	off
Only Group	Only count what is picked here, so a plate can count players and ignore falling crates. Press the box and pick it in the scene. You can pick one thing by its name, Every Coin for every copy of the coin scene (including ones made while playing), or Everything under a node for a set of markers. NodeWorthy does the rest. Anything counts everything the area can see.	empty
Store In	The variable the list of everything found goes into. Walk it with For Each to do the same thing to each one, or with the Array action to pick through it. Empty stores no list, but the count and the nearest are still filled in.	empty
Store Count In	The variable the number found goes into. 0 means the area is empty, and a transition can wait for 0 to let a door close again. Empty stores nothing.	empty
Store Nearest In	The variable the closest one to the area goes into, for the next action to act on that one alone. It is left empty when nothing was found, so check the count before trusting it.	empty

Get Velocity

Reads how fast a body is moving and which way, into variables. It works in 2D or 3D on the two kinds of body that take part in the physics. These are a character body, moved by its own actions, and a rigid body, pushed about by forces.



Get Velocity as it appears in the Actions tab.

When to use it

A character's running speed can choose between walking and running animations, and a rolling ball's tells you when it has come to a stop. Often Speed is all you need, as one number that is 0 when standing still.

> A character body's velocity is what its last move left it at, so it reads 0 until something such as Move Character has moved it.

See also

Set Velocity 122, **Move Character** 134, **Get Vector Components** 50, **Compare** 35

Try it

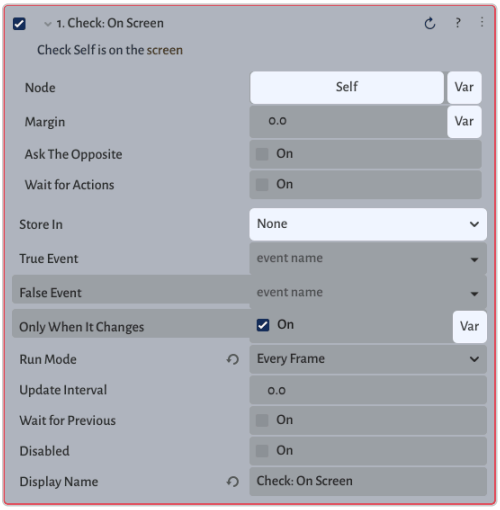
See the action work on its own with **Open Example** in the Add Action window, or open `examples/physics/get_velocity/get_velocity.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The body to read. Leave it empty, as it comes, for the thing this machine works on. Anything that is not a character body or a rigid body has no velocity to read, and the Output panel warns about it. Var: a Node.	Self
Store In	The variable the velocity goes into, giving which way and how fast, in metres a second in a 3D game or pixels a second in a 2D one, as two or three numbers. Empty stores nothing.	empty
Store Speed In	The variable its speed goes into, as one number for how fast, whichever way. 0 is standing still. Empty stores nothing. With both this and Store In empty, a warning says there is nowhere to put what it reads.	empty
Local Space	Off to begin with, so the velocity is given in the world's directions. Ticked, it is in the body's own directions, with forward speed along its front (minus Z in a 3D game, plus X in a 2D one). Var: a bool.	off

On Screen CHECK

Asks whether a node is on the screen right now, inside what the camera shows and in front of it. It works in 2D and in 3D, and nothing has to be added to the scene for it.



On Screen as it appears in the Actions tab.

When to use it

Lets an enemy hold its fire until the player can see it, or a spawner wait until its spot is out of view. It asks about the camera's view only, so something standing in the way does not count. A clear line of sight is Can See's question.

> In a 3D game it needs a camera showing the game. With none, the answer is always no and the Output panel warns about the missing camera.

See also

Can See 250, **Distance** 93, **Off Screen Indicator** 209, **Touching** 125

Try it

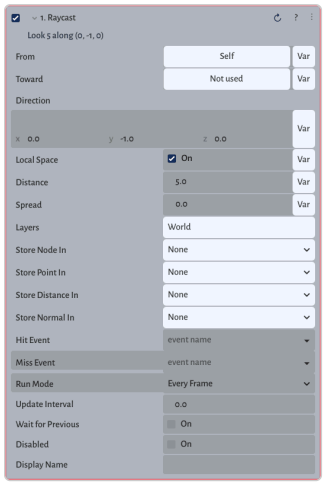
See the action work on its own with **Open Example** in the Add Action window, or open `examples/physics/on_screen/on_screen.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node to ask about. Leave it empty, as it comes, for the thing this machine works on. A node that is hidden is never on the screen. Var: a Node.	Self
Margin	How far off the edge still counts as on the screen, in pixels in a 2D game or metres in a 3D one. The node's middle is what is measured, so at 0, the default, it says no once the middle has gone. Give about half the thing's size and it counts while any of it shows. Below 0 it has to be that far inside the edge. Var: a float.	0.0
Ask The Opposite	Turns the question round into "is it off the screen", for clearing away a bullet once it has gone. The true-or-false answer the Check stores is turned round as well.	off
Wait for Actions	Holds the question back until every other action in the state has finished. Left off, the default, it is asked straight away.	off
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Raycast

Looks along a straight line and finds the first thing in the way. It stores what it found, where and how far, and can send an event on a hit or a miss.



When to use it

Handles a footstep test, the edge of a ledge, a gun that hits as it fires, or a check for a clear view of the player. The line can start from any node with a place in the world, of any kind, and it never finds the node it starts from, so a gun does not shoot itself. It looks at physics shapes, and the layers you tick decide what it can find, not whether the thing is a body, an area or scenery. A guard's sight cone, with a field of view and a memory of where the target went, is less work with Can See.

> The first look always reports what it found, and a state entered with the ground already underfoot still hears about it.

See also

Can See 250, **Get Overlapping** 113, **Event On Collision** 109, **Distance** 93

Try it

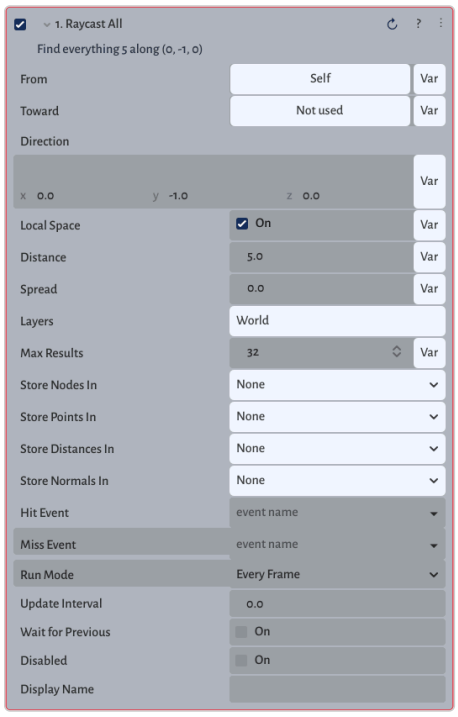
See the action work on its own with **Open Example** in the Add Action window, or open `examples/physics/raycast/raycast.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
From	Where the line starts. Leave it empty for the node this machine works on, or point at a small marker node placed at the muzzle of a gun or at a character's feet. Var: a Node.	Self
Toward	Name a node and the line runs straight to it, in place of a direction. A clear run to it counts as a miss, and anything in the way is a hit, stored as whatever blocks the view. Press Var to look toward the node held in a variable, such as the player. A look toward a node always goes straight to it, so with this filled in Direction, Local Space, Distance and Spread are not offered at all. Var: a Node.	empty
Direction	Only while Toward is empty. It sets which way to look, as two numbers in a 2D game and three in a 3D one. In 2D the second number counts down the screen, so 1 looks downward and -1 upward. In 3D, Y is up. Added to a 2D node it starts looking down the screen, and in 3D it starts looking down. Var: a Vector2 or a Vector3, or a Node for where it stands.	0, -1, 0 in 2D: 0, 1
Local Space	Only while Toward is empty. It is on to begin with, and the direction is read as the node's own and turns with it, forward always meaning the way it faces, as for a gun or a character. Off reads the direction as the world's, so down is always down, whatever the node is doing. Var: a bool.	on
Distance	Only while Toward is empty. How far to look, in metres in 3D or pixels in 2D. Keep it short for a ground test under a foot and long for a gun. Var: a float.	5.0 in 2D: 160 px
Spread	Only while Toward is empty. Aim up to this many degrees off at random, as a gun that is not perfectly accurate does, with a fresh direction inside that cone each time it looks. Zero, the default, looks straight along the direction. Press Var to read it from a variable so running or firing fast can widen it. Var: a float.	0.0
Layers	Tick the layers the line should notice, by name. World is for walls, ground and anything you add yourself, and Player, Enemies or Pickups are for the ready-made ones. Anything on none of the ticked layers is looked straight through. Areas count only on the Pickups layer and the Hit And Hurt Boxes layer, so a pickup is found when Pickups is ticked, while a trigger on the World layer never stops a look for the ground. Tick nothing and every look is a miss.	World
Store Node In	The variable the thing found goes into, ready for the next action to damage it, count it or pick it up. It is emptied when a look finds nothing, so a check can ask whether anything is there. Empty stores nothing.	empty
Store Point In	The variable the spot hit goes into, for Instantiate to put a bullet mark, sparks or a puff of dust right there. When a look finds nothing, the last spot hit is left as it was.	empty
Store Distance In	The variable the distance from the start to the hit goes into, such as how far below a foot the ground is, or how close the wall ahead has got. Empty stores nothing.	empty
Store Normal In	The variable the way the surface hit faces goes into, for example straight out of a wall or straight up from a floor. Instantiate uses it to stand a mark or sparks flat on that surface.	empty
Hit Event	Sent when the line finds something. It is sent only when the answer changes, not on every look. A line that keeps finding the ground sends it once, and again only after a look that found nothing. Empty sends nothing.	empty
Miss Event	Sent when the line finds nothing, again only when that is a change from the look before. A character notices the ground running out at the edge of a ledge this way. Empty sends nothing.	empty

Raycast All

Looks along a straight line and finds everything on it, not only the first thing in the way. It stores lists of what it found, where and how far, nearest first, and can send an event on a hit or a miss.



Raycast All as it appears in the Actions tab.

When to use it

A bullet that passes through every enemy in a row needs this, and so does counting the walls between two places. Raycast stops at the first thing hit and suits most guns and ground tests, but this one is for when what lies behind matters too. For Each can then work through the list.

> Like Raycast it starts on Every Frame, so set Run Mode to Once for a single shot.

See also

Raycast 116, **For Each** 36, **Get Overlapping** 113, **Shape Cast** 123

Try it

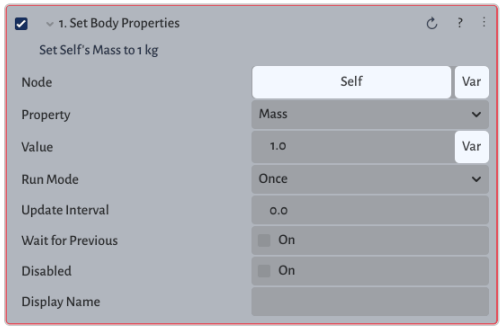
See the action work on its own with **Open Example** in the Add Action window, or open `examples/physics/raycast_all/raycast_all.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
From	Where the line starts. Leave it empty for the node this machine works on, or point at a marker at a gun's muzzle. A body looking from itself never finds itself. Var: a Node.	Self
Toward	Name a node and the line runs to it, not along a direction. Everything between here and it is found, but not the node itself. Press Var to look toward the node held in a variable. With this filled in, Direction, Local Space, Distance and Spread are not offered. Var: a Node.	empty
Direction	Only while Toward is empty. It is which way to look, as two numbers in a 2D game and three in a 3D one. In 2D the second number counts down the screen, and in 3D, Y is up. Var: a Vector2 or a Vector3, or a Node for where it stands.	0, -1, 0 in 2D: 0, 1
Local Space	Only while Toward is empty. On, the default, the direction turns with the node, so forward is the way it faces. Off reads it as the world's direction. Var: a bool.	on
Distance	Only while Toward is empty. How far to look, in metres in 3D or pixels in 2D. Var: a float.	5.0 in 2D: 160 px
Spread	Only while Toward is empty. Aim up to this many degrees off at random, as a gun that is not perfectly accurate does. At 0, the default, it looks straight along the direction. Press Var to read it from a variable. Var: a float.	0.0
Layers	Tick the layers the line should notice, by name, such as World, Player or Enemies. Anything on none of them is looked straight through. Areas count only on the Pickups layer and the Hit And Hurt Boxes layer, so a pickup is found when Pickups is ticked and a trigger on the World layer is looked past.	World
Max Results	The most things it finds on one look, nearest first. A bullet that goes through two enemies at most is 2. It starts at 32. Var: an int.	32
Store Nodes In	The Array variable the things found go into, nearest first, ready for For Each. Empty stores nothing.	empty
Store Points In	The Array variable the exact spots hit go into, in the same order as the things found.	empty
Store Distances In	The Array variable the distance to each hit goes into, in the same order.	empty
Store Normals In	The Array variable the way each surface hit faces goes into, in the same order, for standing marks or sparks flat on them.	empty
Hit Event	Sent when the line finds anything. As with Raycast, it is sent only when the answer changes. Empty sends nothing.	empty
Miss Event	Sent when the line finds nothing, again only when that is a change. Empty sends nothing.	empty

Set Body Properties

Changes how a physics body behaves while the game runs, one property at a time, picked by name. It can be its mass, how quickly it slows down or stops spinning, how bouncy it is, how much it grips, or whether it may turn at all.



Set Body Properties as it appears in the Actions tab.

When to use it

A crate filling with water grows heavier as its Mass is raised, say from 10 to 40 kilograms. A floor with Friction at 0 turns to ice, and a slowing field can raise Linear Damp on whatever enters it.

> The first time Bounce or Friction is set, the body is given a separate physics material. Changing one ball never changes the other balls made from the same scene.

See also

Set Gravity 120, **Freeze** 111, **Apply Force** 107, **Set Velocity** 122

Try it

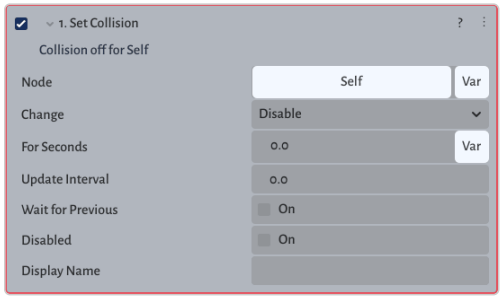
See the action work on its own with **Open Example** in the Add Action window, or open `examples/physics/set_body_properties/set_body_properties.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The body to change. Leave it empty for Self, the node this Machine works on. Mass, Linear Damp, Angular Damp and Lock Rotation need a RigidBody2D or RigidBody3D. Bounce and Friction work on a StaticBody as well, so a wall can be made bouncy. Anything else is left alone, and a warning names the node and the property it lacks. Var: a Node.	Self
Property	Which property to change. Mass is how heavy it is, in kilograms. Linear Damp is how quickly it slows down by itself, as though moving through water. Angular Damp is how quickly it stops spinning. Bounce is from 0, no bounce, to 1, bouncing back as high as it fell. Friction is from 0, like ice, to 1, like rubber. Lock Rotation keeps it from turning at all. Choices: Mass, Linear Damp, Angular Damp, Bounce, Friction, Lock Rotation.	Mass
Value	What the property becomes, such as kilograms for Mass or 0 to 1 for Bounce and Friction. Not offered for Lock Rotation. A mass of 0 or below is not used, so the body keeps its mass, and a warning asks for a Value above 0. Var: a float.	1.0
Lock Rotation	Only when Property is Lock Rotation. Ticked keeps it from turning, so it slides but never rolls or tips over. Unticked lets it turn again.	—

Set Collision

Switches a node's collision off or on, or changes which layers it sits on and which layers it bumps into. Give it a time and it puts everything back by itself.



Set Collision as it appears in the Actions tab.

When to use it

When something should stop being solid for a while, such as a player dropping through a one-way platform, or a coin that must not be collected twice. Unlike most physics actions it takes any kind of node, because a walking character, a rigid body, static scenery and an area all have collision shapes. A node with no collision shape at all is left as it is, with a warning in Godot's Output. To hold a rigid body still while it stays solid, use Freeze.

> Turning a walking character's collision off drops it straight through the floor. When only one sort of thing should be ignored, use Remove Collision Mask.

See also

Freeze 111, **Event On Collision** 109, **Touching** 125, **Set Visible** 216

Try it

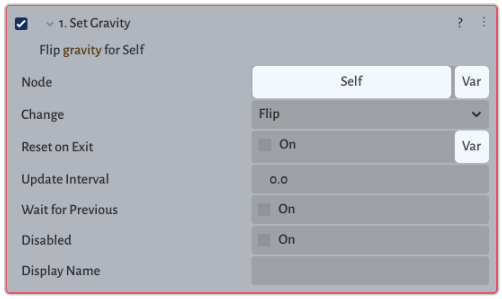
See the action work on its own with **Open Example** in the Add Action window, or open `examples/physics/set_collision/set_collision.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node to change. Leave it empty for the node this machine works on. Everything under it that has collision is changed as well, but another body nested inside it is left alone, so a crate carried by a character is not affected. Var: a Node.	Self
Change	Disable switches its collision shapes off, so nothing hits it and it hits nothing, and Enable switches them back. Add Collision Layer and Remove Collision Layer change which layers it can be found on, and taking a pickup off its layer stops it being collected. Add Collision Mask and Remove Collision Mask change what it notices. Taking the platform's layer out of the player's mask drops them through the platform. Choices: Disable, Enable, Add Collision Layer, Remove Collision Layer, Add Collision Mask, Remove Collision Mask.	Disable
For Seconds	Put the change back after this many seconds, for example 0.3 to drop through a platform, or five for a ghost that walks through walls for a while. Only the layers this action changed are put back, so any other change made meanwhile is kept. Zero, the default, leaves the change in place until something changes it back. Var: a float.	0.0
Layers	Only when Change is one of the four layer choices. Tick the layers to add or take away, by name, such as World, Player, Enemies, Pickups, or one of your own. Tick nothing and the action changes nothing.	—

Set Gravity

Changes gravity for one body. It can be lighter on the moon, heavier underwater, or turned over so it falls to the ceiling and walks about on it.



Set Gravity as it appears in the Actions tab.

When to use it

Gravity-flip puzzles, low-gravity levels and a swimmer who sinks slowly all start here. It works on a walking character body and on a rigid body, the kind physics moves by itself. An area or static scenery is left alone, with a warning. Jump and Move Character follow the change by themselves, so a jump set to two metres still reaches two metres under the new gravity. To stop a rigid body falling altogether, Freeze and its Gravity Off is simpler.

> Turning a character's gravity over also clears its up-and-down speed. It starts falling the new way from a standstill and does not shoot off.

See also
Freeze 111, **Jump** 132, **Move Character** 134, **Apply Force** 107

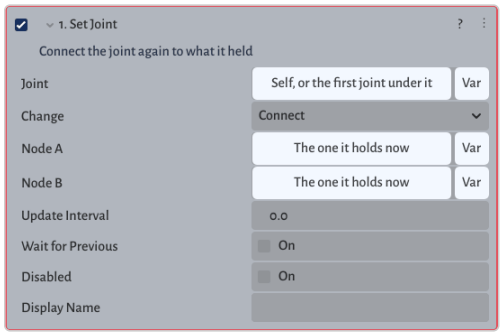
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/physics/set_gravity/set_gravity.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The character or physics body to change. Leave it empty for the node this machine works on. Nothing else can have its gravity changed. Var: a Node.	Self
Change	Gravity Scale sets how strong gravity is for this one body. Upside Down makes it fall upward and stand on ceilings, Right Way Up puts it back, and Flip, the default, swaps whichever it is now, so one action on a button flips gravity back and forth. Choices: Gravity Scale, Upside Down, Right Way Up, Flip.	Flip
Turn Texture	Used by the three flipping choices, and only for a 2D character. Ticked, the character's picture is turned over as well, so one standing on the ceiling is still the right way up on its feet. It does nothing in 3D, and nothing for a rigid body. Var: a bool.	on
Reset on Exit	Ticked, the change lasts only while this state runs. When the state ends, however it was left, the body's gravity is put back as it was. Handy for a glide, a hover, a magnet that holds a character up, or a room with less gravity, with no need to restore gravity in every state that can follow. Unticked, the default, the change stays until something else changes it. Var: a bool.	off
Strength	Only when Change is Gravity Scale. How strong gravity is for this body compared with usual. At 1 it is normal, 0.5 is the moon, 2 is heavy boots and 0 is no gravity at all. A number below 0 counts as the same number above it, as it always has on a rigid body. Upside Down makes a body fall up. It multiplies the gravity already in use and is not a speed, and you do not need to know the project's number.	—

Set Joint

Connects two bodies with a joint, lets them go again, or changes one of the joint's values while the game runs. A joint is a node that holds two physics bodies together at the place where it stands, like a pin, a hinge or a spring.



Set Joint as it appears in the Actions tab.

When to use it

A chain link that snaps is Disconnect on the joint holding it, and a rope the player grabs is Connect, with the player as Node B. The joint itself is a node placed in the scene, such as a PinJoint2D, DampedSpringJoint2D, HingeJoint3D or PinJoint3D.

> The joint holds the bodies at the place where the joint stands, so put it where the pin, hinge or rope's end should be before connecting.

See also

Freeze 111, **Apply Force** 107, **Set Body Properties** 118, **Reparent** 89

Try it

See the action work on its own with **Open Example** in the Add Action window, or open `examples/physics/set_joint/set_joint.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Joint	The joint to work on. Left empty it uses the node this Machine works on when that is a joint, or else the first joint anywhere under it. Var: a Node.	empty
Change	Connect joins the two bodies picked below. Disconnect lets go of both, so they move on their own. Set Value changes one of the joint's values. It starts on Connect. Choices: Connect, Disconnect, Set Value.	Connect
Node A	Only when Change is Connect. The first body the joint holds, such as the ceiling a lamp hangs from. It must be a physics body, such as a RigidBody or a StaticBody. Leave it empty to keep the one the joint holds now. With both left empty, Connect joins again the two it held before Disconnect let them go. Var: a Node.	empty
Node B	Only when Change is Connect. The second body the joint holds, such as the player grabbing a rope. Leave it empty to keep the one it holds now. A 2D joint holds nothing until it has both, and a warning asks for the other body. Var: a Node.	empty
Value	Only when Change is Set Value. Which value to change, by its name. The names are Bias, Softness, Length, Rest Length, Stiffness, Damping, Angular Limit Lower and Upper, the motor's Target Velocity, Angular Limit Enabled, Motor Enabled, or Disable Collision, which when on lets the two bodies pass through each other. Each kind of joint has some of them. Asking for one it lacks changes nothing, and the warning lists the kinds that have it.	—
Amount	Only when Change is Set Value and the value is a number. What it becomes, in degrees for the angular limits, degrees a second for the motor, or pixels for a 2D length. Setting a limit turns the limit on, and setting the motor's speed turns the motor on.	—
Enabled	Only when Change is Set Value and the value is on or off. Ticked turns it on, unticked turns it off. With Disable Collision, ticked means the two bodies pass through each other.	—

Set Velocity

Sets how fast a body is moving and which way, straight away, with no working up to it. An unticked axis keeps what the body is already doing along it, letting you change direction without cancelling a fall.



Set Velocity as it appears in the Actions tab.

When to use it

Sends a bullet or a ball off in a straight line, stops a body dead when every number is zero, or bounces something back the other way. It works on a rigid body and on a character body, in 2D and in 3D. An area, static scenery or a plain node is not moved, and a warning explains why. On a character body the speed is only a request, and a movement action such as Move Character or Chase must be walking it in the same state or a parent state above it. Without one, the body stands still with its speed set.

> With Local Space ticked the direction is turned to match the body, but the three ticks still pick the world's axes, so what an unticked axis keeps is the world's up and down, not the body's.

See also

Move Character 134, **Apply Force** 107, **Jump** 132, **Drive** 108

Try it

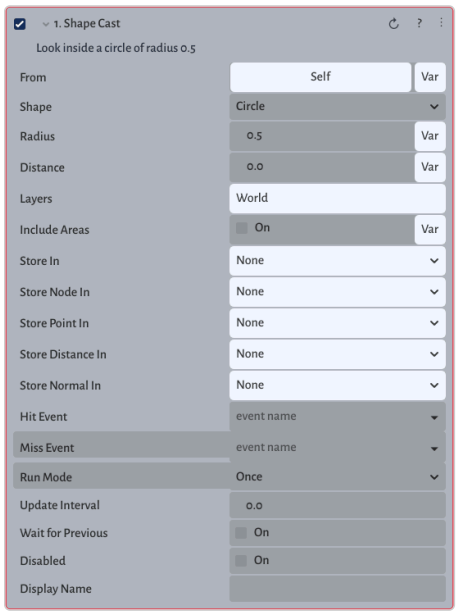
See the action work on its own with **Open Example** in the Add Action window, or open `examples/physics/set_velocity/set_velocity.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The body. Leave it empty for the node this machine works on, such as a bullet carrying the machine. Var: a Node.	Self
Velocity	The speed and direction, in metres per second in 3D or pixels per second in 2D, as two numbers in a 2D game and three in a 3D one. Remember the difference in units. A speed of 5 is a brisk run in 3D, but in 2D speeds run into the hundreds. Press Var to read it from a variable. Var: a Vector2 or a Vector3, or a Node for where it stands.	0, 0, 0 in 2D: 0, 0
Local Space	Off, the numbers are the world's directions. On, they are the body's, and forward is whichever way it faces, so a bullet leaves the front of a gun however the gun is pointing. Var: a bool.	off
Set X	Ticked, the sideways speed is set. Untick it to leave the body's sideways speed as it was and change only the other axes. Var: a bool.	on
Set Y	Ticked, the up-and-down speed is set. Untick it to turn a body round in mid-air without cancelling its fall, or to give a runner a new direction while it is still dropping. Var: a bool.	on
Set Z	Ticked, the forward-and-back speed is set. This is for 3D only, as a flat 2D body has no third axis and takes no notice of it. Var: a bool.	on

Shape Cast

Asks the physics world what is inside a circle or a box, with no area needed, or sweeps the shape along a line and finds what it runs into first, like a Raycast with thickness. It keeps what it found in variables and can send an event either way.



Shape Cast as it appears in the Actions tab.

When to use it

Finds every enemy caught by a sword swing, checks a spot is clear before spawning something there, or makes a bullet as wide as it looks. Nothing is added to the scene, because the shape is only looked with when the action runs. Where there is already an area, Get Overlapping asks it, and a thin line is cheaper with Raycast.

See also

Raycast 116, **Get Overlapping** 113, **Explosion** 110, **For Each** 36

Try it

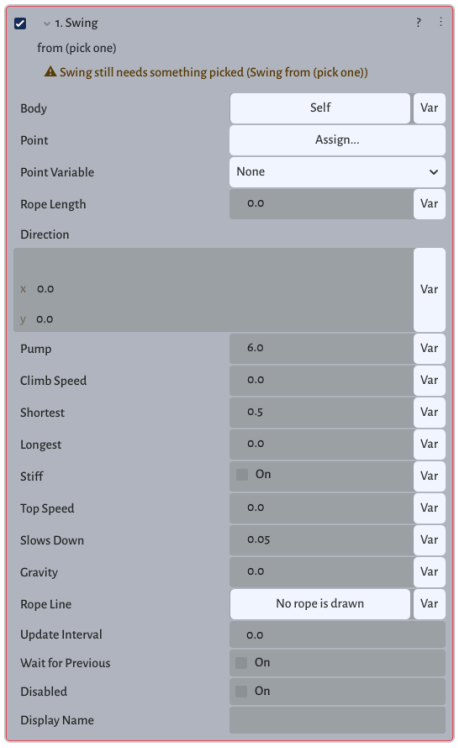
See the action work on its own with **Open Example** in the Add Action window, or open `examples/physics/shape_cast/shape_cast.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
From	Where the shape is, or where the sweep starts. Leave it empty, as it comes, for the thing this machine works on. It must be a 2D or 3D node. The node itself, when it is a body, is never found. Var: a Node.	Self
Shape	The shape to look with, either Circle, the default (a ball in a 3D game), or Box. Choices: Circle, Box.	Circle
Radius	Only for Circle. How big the circle is, from its middle to its edge, in metres in a 3D game or pixels in a 2D one. Var: a float.	0.5 in 2D: 16 px
Distance	How far to sweep the shape. 0, the default, does not sweep it and looks only where it is. Anything above 0 moves it along Direction until it touches something. Var: a float.	0.0 in 2D: 0 px
Layers	What it can find, by physics layer. World is walls, ground and anything you add yourself, and the ready-made player, enemies and pickups each have a separate layer. Nothing ticked finds nothing.	World
Include Areas	Off to begin with, so only bodies are found. Tick it to find areas too, such as pickups and trigger zones. Var: a bool.	off
Store In	The variable the list of everything found goes into. That is everything inside the shape, or everything it is touching where a sweep stopped. For Each can then do something to each one.	empty
Store Node In	The variable the one thing found goes into. That is what a sweep ran into, or the nearest of everything inside. It is emptied when nothing is found.	empty
Store Point In	The variable the point where the shape touches it goes into. The last one is kept when nothing is found.	empty
Store Distance In	The variable how far the shape went before it touched something goes into. Without a sweep, it is how far away the nearest thing inside is. A sweep that starts already touching something has gone nowhere and gives 0.	empty
Store Normal In	The variable the way the touched surface faces goes into, such as straight out of a wall or up from a floor.	empty
Hit Event	The event sent when something is found. On Every Frame it is sent when the answer changes, not every frame, and the first look always says what it found.	empty
Miss Event	The event sent when nothing is found, in the same way as Hit Event.	empty
Size	Only for Box. How big the box is, edge to edge, as two numbers in a 2D game or three in a 3D one. With Local Space ticked, the box is turned with the node.	—
Direction	Only when Distance is above 0. Which way to sweep, in the world's directions unless Local Space is ticked. In a 2D game Y counts down the screen, so down is 0, 1 in 2D and 0, -1, 0 in 3D.	—
Local Space	On to begin with, so Direction is the node's own, forward means its forward, and a box is turned the way the node faces. Off uses the world's directions and keeps the box square to the world.	—

Swing

Swings a character from a point as if on a rope. It falls round the point like a pendulum, fastest at the bottom and slowing at the ends.



Swing as it appears in the Actions tab.

When to use it

For a grappling hook caught on a ring, a vine or a chain. It moves the body itself, so give it the swinging state to itself, and leave that state on an event, such as the grab button let go or a jump, to let go. The character flies on with the speed it had, so let go at the front of a swing and it is thrown forward and up. Holding a way pumps the swing higher, and in a 2D game holding up or down climbs the rope. It needs a character body (a CharacterBody2D or CharacterBody3D) and a point with a place in the world of the same kind.

> The rope holds the character out, never in. Swung up past the point, it goes slack and the character falls until the rope snaps taut again, losing some speed as a real rope would. Stiff makes it a pole that swings all the way round. While it stays taut the swing keeps its height, and left alone it comes back up to where it began. If something else carries the character far away, such as a teleport or a respawn, the rope lets out to where the character is and does not yank it back.

See also
Move Character 134, **Jump** 132, **Dash** 128, **Machine Variable** 27, **Get Input Vector** 143

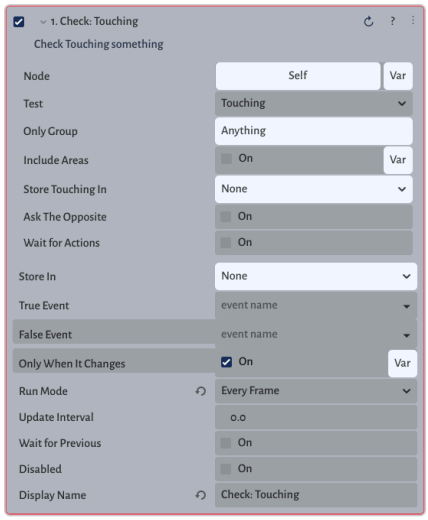
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/physics/swing/swing.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Body	The character body that swings. Leave it empty for the node this machine works on. Var: a Node.	Self
Point	The point it swings from. It can be any node with a place in the world, such as a ring or a hook picked in the scene.	empty
Point Variable	Swing from the node held in this variable, letting one template swing from whichever point it reached. Rings that set this variable when the character comes near make a level full of rings work with one template.	empty
Rope Length	How long the rope is, in metres in 3D or pixels in 2D. Zero, as it starts, takes however far the character is from the point when it catches hold. Press Var to read it from a variable. Var: a float.	0.0
Direction	The way held, which pumps the swing and, in 2D, climbs the rope. Press Var and pick the variable a Get Input Vector fills in, and the player's stick or keys drive it. Var: a Vector2 or a Vector3, or a Node for where it stands.	0, 0
Pump	How hard holding a way pushes the swing along, in metres a second each second in 3D or pixels in 2D. It only ever pushes along the swing, never along the rope, and only while the rope is taut. Zero swings only as gravity takes it. Var: a float.	6.0 in 2D: 300
Climb Speed	How fast holding up or down climbs the rope, shortening or lengthening it. A shorter rope swings faster. Zero, as it starts, keeps the rope as it is. Climbing works in a 2D game only. In 3D the way held has no up or down, and the rope keeps its length. Var: a float.	0.0
Shortest	The shortest the rope can be climbed to. Only used when Climb Speed is above zero, so only in a 2D game. Var: a float.	0.5 in 2D: 16
Longest	The longest it can be let out to by climbing down. Zero lets it out no further than it was when the character caught hold. Only used when Climb Speed is above zero, so only in a 2D game. Var: a float.	0.0
Stiff	A pole in place of a rope. It holds the character at its length both ways and never goes slack, and a swing pumped hard enough goes up over the point and all the way round. Off, as it starts, a rope holds the character out but lets it fall inside when swung up past the point. Var: a bool.	off
Top Speed	The fastest it can swing, in metres a second in 3D or pixels in 2D, however hard it is pumped. Without it a pole pumped round and round speeds up for ever. Zero, as it starts, sets no limit. Var: a float.	0.0
Slows Down	How much of its speed it loses each second, from 0 for a swing that never dies down to 1 for one that barely moves. A little keeps a swing from growing wild. Var: a float.	0.05
Gravity	How strongly it is pulled down, in metres a second each second in 3D or pixels in 2D. Zero, as it starts, uses the project's gravity. Var: a float.	0.0
Rope Line	A Line2D drawn as the rope from the character to the point while it swings, and rubbed out when it lets go. Empty draws nothing. Var: a Node.	empty

Touching CHECK

Asks whether something is in contact right now, such as a crate resting on a pressure plate or a character with its shoulder against a wall. It works the same in 2D and in 3D.



Touching as it appears in the Actions tab.

When to use it

A burning state lasts as long as the character stands in the fire and ends when the contact does. Event On Collision fires once as something arrives, but this keeps answering while it is there. On Floor allows a jump only when the character is standing on something.

> Asking a physics-driven body what it is touching switches on its contact reporting. There is no need to set that up in the Inspector.

See also

Event On Collision 109, **Get Overlapping** 113, **Distance** 93, **Tile** 245

Try it

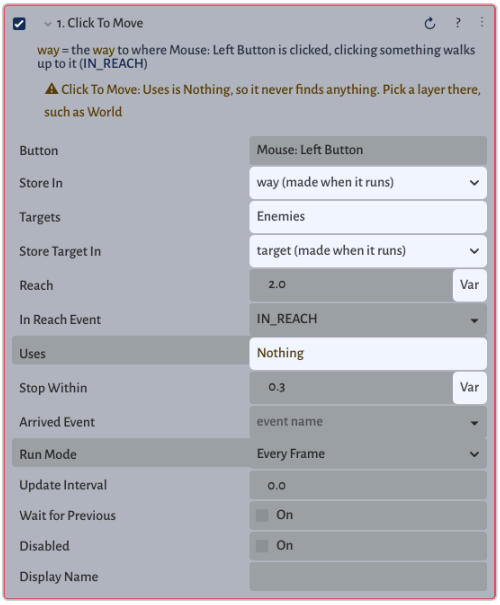
See the action work on its own with **Open Example** in the Add Action window, or open `examples/physics/touching/touching.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The area or body being asked about. An area is a marked-off space that notices what is inside it, and a body is a solid thing the physics pushes around or stops. Leave it empty for the node the machine works on. Var: a Node.	Self
Test	Touching, the default, asks whether anything is in contact. On Floor, On Wall and On Ceiling ask about a character body that moves itself, and answer from the last move it made. Is it standing, scraping along a wall, or has it bumped its head? At A Ledge asks whether a character walking on the ground has a drop just ahead of it, the way it is going, or the way it faces when it stands still. An enemy can turn round there and stay on its platform. Choices: Touching, On Floor, On Wall, On Ceiling, At A Ledge.	Touching
Only Group	Count only what is picked here, so a door can ignore everything but the player. Press the box and pick it in the scene. You can pick one thing by its name, Every Coin for every copy of the coin scene (including ones made while playing), or Everything under a node for a set of markers. NodeWorthy does the rest. Anything, the default, counts anything at all, and a plate can then trigger on a stray bullet.	empty
Include Areas	Ticked, other areas count as well as solid bodies, so one trigger area can notice another. It is off by default, and it only does anything when the node being asked about is itself an area, because a body cannot be asked what areas it is inside. Only the Touching test uses it. Var: a bool.	off
Store Touching In	Name a variable and whatever is being touched goes into it, for the next action to push, hurt or pick up. On Floor, On Wall and On Ceiling put in what the character last slid against, and every test empties the variable again when nothing is being touched. Empty, the default, stores nothing.	empty
Ask The Opposite	Turns the answer round into "nothing is touching", so a character can notice it has walked off a ledge. It turns round the true-or-false answer the Check stores as well, which swapping the True Event and False Event over cannot do.	off
Wait for Actions	Holds the question back until every other action in the state has finished, so contact is read after the character has moved this frame. Left off, the default, it is asked as soon as the state runs.	off
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Click To Move

Lets the player walk by clicking. Click the ground and the character walks there, hold the button and it follows the pointer, or click an enemy and it walks up to it and says when it is close enough to strike. It does not move anything itself. It fills in the way to go, and Move Character does the walking.



Click To Move as it appears in the Actions tab.

When to use it

Mouse-driven games seen from above or at an angle, such as a dungeon crawler or a farm, walk their hero this way. Put it after Get Input Vector in the same state, with Move Character below them both. Keys and clicks then work side by side, and pressing a key forgets the click. Where the level has walkable ground (NodeWorthy, Bake Navigation Mesh), it finds the way round walls. Without it, it walks in a straight line.

> It only works out where to go. Without Move Character reading the same variable, nothing walks.

See also

Get Input Vector 143, **Move Character** 134, **Mouse Pick** 149, **Navigate To** 239

Try it

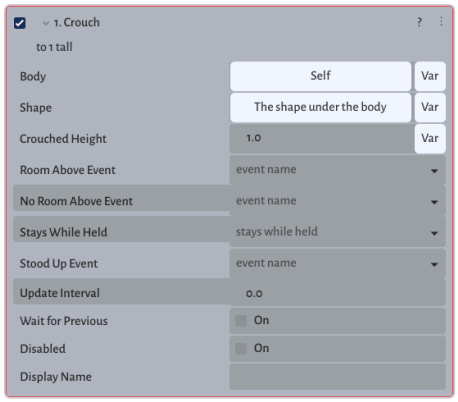
See the action work on its own with **Open Example** in the Add Action window, or open `examples/character/click_to_move/click_to_move.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Button	The button that is clicked, or held, to walk. The left mouse button is the usual one.	Mouse: Left Button
Store In	The direction variable Move Character reads, the same one Get Input Vector fills in from the keys. Pick that one, or the character will never move.	way
Targets	What can be clicked on to walk up to it rather than to the ground under it, picked by layer, such as Enemies. A layer is a named group of things, such as everything on the Enemies layer. Empty makes every click a walk to the ground.	Enemies
Store Target In	What was clicked on is kept in this variable, so the state that strikes knows what to aim at, for instance through Attack's Aim at Variable. Empty keeps nothing.	target
Reach	How close to what was clicked on it walks before it stops and sends In Reach Event, such as the reach of a sword, in pixels in 2D and metres in 3D. Too small and the character bumps into the enemy without ever striking. Var: a float.	2.0 in 2D: 60
In Reach Event	The event sent once, when what was clicked on is within reach. Give a transition this event to go to the state that swings. Empty sends nothing.	IN_REACH
Uses	What can be clicked on to walk up to it and use it, not strike it, such as a chest, a lever or a door. It is picked by layer, such as Pickups. Empty uses nothing.	Nothing
Stop Within	How close to a spot on the ground counts as there, in pixels in 2D and metres in 3D. Too small and the character can dither on the spot. Var: a float.	0.3 in 2D: 10
Arrived Event	The event sent when the character gets to a spot on the ground it was sent to. Empty sends nothing.	empty
Used Event	The event sent to the thing that was clicked on to be used, once the character is within reach of it. A Chest opens when it is sent USED.	—

Crouch

Makes a character shorter while the state runs, and gives it back its height when the state is left.



Crouch as it appears in the Actions tab.

When to use it

For crouching, ducking under a swing, crawling through a gap or sliding along the floor. It shortens the collision shape itself, and a shorter character really does fit under things a tall one cannot. The feet stay where they are, so the character ducks and does not sink through the floor. It needs a CollisionShape under it with a height to change. That means a capsule, a cylinder, a box or a rectangle. A circle or a sphere has no height and is left alone, with a warning.

> The shape's resource is copied before it is changed, and other characters built from the same shape keep their full height. Leaving the crouch while something is still overhead gives the character its height back anyway, with a warning in the Output panel. Otherwise it would be stuck down for good. Leave the crouch on the Room Above event and this never happens.

See also

Move Character 134, **Step Up** 136, **Touching** 125, **Set Property** 91

Try it

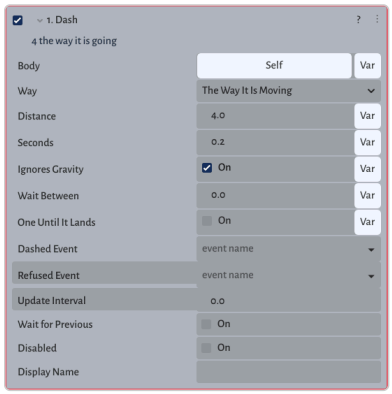
See the action work on its own with **Open Example** in the Add Action window, or open `examples/character/crouch/crouch.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Body	The character body to shorten. Leave it empty for the node this machine works on. Var: a Node.	Self
Shape	The CollisionShape to shorten. Leave it empty for the first one under the body. Var: a Node.	empty
Crouched Height	How tall the character is while crouched, in metres in 3D or pixels in 2D. About half its standing height reads as a crouch and about a third of it reads as a crawl. A height no shorter than the current one does nothing, and a warning says so. Var: a float.	1.0 in 2D: 32 px
Room Above Event	Event sent when there is room overhead to stand up. It is sent straight away if there is room as it crouches, and again each time room comes back after something was in the way. Leave the crouch on this event to stand as soon as there is room. Empty sends nothing.	empty
No Room Above Event	Event sent when something overhead is in the way of standing up. It comes straight away if it crouches under something, and again each time something gets in the way after there was room. Use it to keep the character down, or to show the player why they cannot get up. Empty sends nothing.	empty
Stays While Held	Name the control the player holds to crouch and the action does the rest. It stays down while that control is held and stands up by itself once it is let go AND there is room overhead. Hold-to-crouch is easy to build wrongly, because letting go and having room must agree before the character may get up, and this box handles both. Empty leaves standing up to your own template, using the Room Above and No Room Above events. On a stick it stays down while the stick is pushed at least half way down, and rolling the thumb back toward level to run stands it up.	empty
Stood Up Event	Only with Stays While Held. Event sent when it has stood up by itself, having been let go with room overhead. Leave the crouching state on this event. Empty sends nothing.	empty

Dash

Sends a walking character shooting off in one direction for a moment, then finishes.



Dash as it appears in the Actions tab.

When to use it

When a swing needs dodging or a ledge is one jump too far. You give how far the dash goes and how long it takes. The speed is worked out, and every dash covers the same ground. It holds its state up while it runs, making that state the dash itself, so leave it on FINISHED and the character carries on. It needs a character body (a CharacterBody2D or CharacterBody3D), the kind a walking player or guard uses. On a rigid body or static scenery it does not dash, and the Output panel warns about it.

> A dash with nowhere to go, such as The Way It Is Moving while the character stands still, is refused with a warning in the Output panel. It never fails quietly.

See also

Move Character 134, **Jump** 132, **Wall Cling** 138, **Set Velocity** 122

Try it

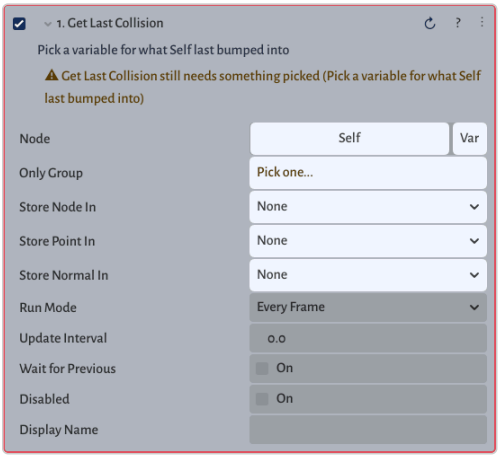
See the action work on its own with **Open Example** in the Add Action window, or open `examples/character/dash/dash.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Body	The character body to send. Leave it empty for the node this machine works on, as on the player's machine. Var: a Node.	Self
Way	Which way the dash goes. The Way It Is Moving uses the direction it is already travelling, laid flat, so a character falling past a gap dashes across it, not down into the floor. Standing still there is no direction, and the dash is refused. The Way It Is Facing goes straight ahead, away from the screen in 3D and to the right in 2D. A Direction uses the direction below. Drop in the variable Get Input Vector fills and the player dashes the way they are holding. Toward A Node aims at another node wherever it has got to. Choices: The Way It Is Moving, The Way It Is Facing, A Direction, Toward A Node.	The Way It Is Moving
Distance	How far the dash carries the character, in metres in 3D or pixels in 2D. Press Var to read it from a variable, so a stronger dash can go further. Var: a float.	4.0 in 2D: 128 px
Seconds	How long the dash takes. Shorter is sharper. About a fifth of a second reads as a snap, and half a second reads as a run-up. The speed is this and the distance together, so changing one changes how fast it looks. Var: a float.	0.2
Ignores Gravity	Ticked, as it starts, nothing pulls the character down while it dashes, and an air dash goes where it is pointed without sagging. Untick it for a dash that falls as it goes, such as a heavy roll along the ground. Var: a bool.	on
Wait Between	Refuse another dash until this many seconds have passed, so holding the button down cannot chain dash after dash. Zero, as it starts, allows one every time the state is entered. Var: a float.	0.0
One Until It Lands	Ticked, only one dash is allowed until the character touches the ground again, however long it stays in the air. Without it a player could cross a whole level without landing. It counts from the dash itself, and the character must really stand on something before the next one. Var: a bool.	off
Dashed Event	Event sent as it dashes, for a sound, a trail of dust or a change of animation. Empty sends nothing.	empty
Refused Event	Event sent when it cannot dash, because it is too soon after the last one, none are left until it lands, or there is nowhere to dash at all. The action finishes either way and the state does not hang. Empty sends nothing.	empty
Direction	Only when Way is A Direction. The direction to dash along. A flat direction, such as the one Get Input Vector stores, fits straight in. Press Var to read it from a variable. In 3D a flat direction is read as the ground's two directions, as Move Character reads it, and a stick pushed up dashes forward. On a body seen from the side (Move Character's Space Side On) it is read as along and up, and up dashes up.	—
Toward	Only when Way is Toward A Node. The node to dash at, wherever it is when the dash starts.	—
Toward Variable	Only when Way is Toward A Node. Dash at the node held in this variable in place of the one picked in Toward. One template can then dash at whatever it is dealing with, such as the target Look For Target stored.	—

Get Last Collision

Finds out what a character body last bumped into as it moved. It gives the node, the point where they met, and the way that surface faces. A character body is a node moved by its own actions, such as the player or an enemy that walks.



Get Last Collision as it appears in the Actions tab.

When to use it

Tells you what a player ran into, what a car hit, or what a character should bounce off. It reads what the last move left behind and works alongside Move Character, Jump and the other movement actions. A character on the ground touches the floor at every step, and the floor is often the last thing it met, so pick a group in Only to find the last thing of that kind. A rigid body needs Get Contact.

> It only knows about the character's last move. A character that nothing is moving keeps the answer from that move.

See also

Get Contact 112, **Touching** 125, **Move Character** 134, **Event On Collision** 109

Try it

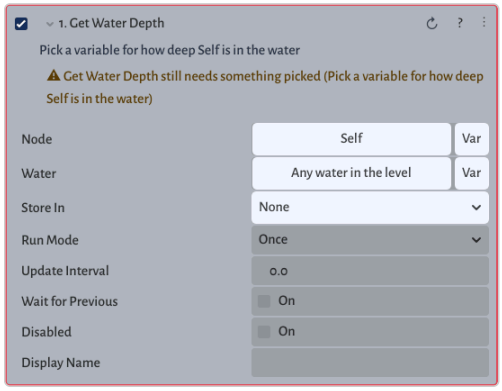
See the action work on its own with **Open Example** in the Add Action window, or open `examples/character/get_last_collision/get_last_collision.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The character body, 2D or 3D. Leave it empty, as it comes, for the thing this machine works on. Anything else never slides into anything, and a warning in the Output panel tells you so. Var: a Node.	Self
Only Group	Only count what is picked here, so the floor under a character's feet does not hide the enemy it ran into. Press the box and pick it in the scene. Left as Anything, it counts whatever it met last.	empty
Store Node In	The variable the node it bumped into goes into. It is emptied while the character bumps into nothing.	empty
Store Point In	The variable the point where they met goes into, a place in the world. The last one is kept while it bumps into nothing.	empty
Store Normal In	The variable the way the surface faces goes into, such as straight out of a wall or up from a floor. Instantiate can use it to stand sparks on that surface.	empty

Get Water Depth

Reads how far below the surface someone's feet are into a variable, or 0 out of the water.



Get Water Depth as it appears in the Actions tab.

When to use it

To make water matter more the deeper it gets. Set a player's speed from the depth so wading slows them, show how far down a diver is, or fade the music as they sink. Water is found the way In Water finds it, a Terrain's lake or sea, or any Area in the group water. Set Run Mode to Every Frame to keep the number up to date while they move.

> The depth is from the top of the water to the bottom of who is measured, their feet. When they are in two waters at once, the deeper is read.

See also

In Water 131, **Swim** 137, **Set Variable** 49, **Move Character** 134

Try it

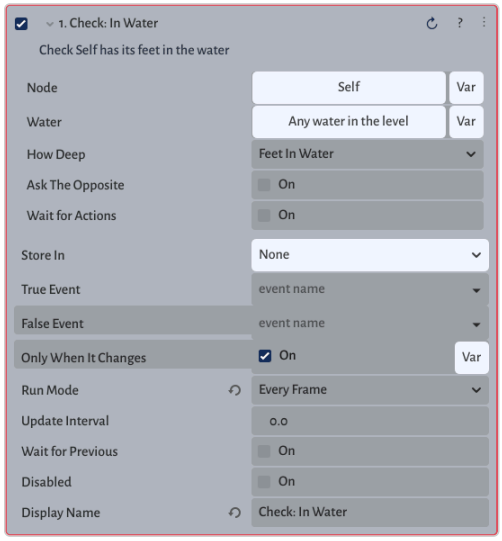
See the action work on its own with **Open Example** in the Add Action window, or open `examples/character/get_water_depth/get_water_depth.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	Who to measure. Leave it empty for the node this machine works on. Var: a Node.	Self
Water	The water to measure in, an Area2D or an Area3D, or a Terrain with a lake or sea. Leave it empty for any water in the level. Var: a Node.	empty
Store In	The float variable the depth goes into, in metres in 3D or pixels in 2D, or 0 out of the water.	empty

In Water CHECK

True while someone is in the water, with their feet in it, waist deep, or with their head under.



In Water as it appears in the Actions tab.

When to use it

For anything that changes in water. A player who wades slower once the water is waist deep, a splash as their feet go in, an air bar that runs down while their head is under, or an enemy that will not follow into a lake. Water is an Area, an invisible shape you draw in the scene. A Terrain's lake or sea is already water, and so is any Area you put in the group water. What happens in the water is yours to build with states, such as a Holding Breath state that a Timer and Set Variable count down.

> How deep is measured from the top of the water's shape to the collision shape of who is asked about, so a tall character and a small one each have their own waist. A node with no collision shape is measured at its place. In 2D, where the screen counts down, the top of the water is still the surface.

See also

Swim 137, **Get Water Depth** 130, **Touching** 125, **Timer** 68

Try it

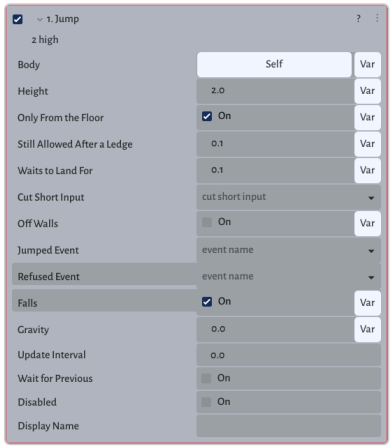
See the action work on its own with **Open Example** in the Add Action window, or open `examples/character/in_water/in_water.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	Who to ask about. Leave it empty for the node this machine works on. Var: a Node.	Self
Water	The water to ask about, an Area2D or an Area3D, or a Terrain with a lake or sea. Leave it empty for any water in the level, which is every Area in the group water. Var: a Node.	empty
How Deep	How deep counts as in. Feet In Water is as soon as the feet are in it, Waist Deep is when it reaches the middle, and Head Under is when it is over the head. Choices: Feet In Water, Waist Deep, Head Under.	Feet In Water
Ask The Opposite	Ask the question the other way round, so yes becomes no and no becomes yes. Use it for "is not touching the ground" and the like. It turns round the answer kept in Store In as well, which swapping the two events cannot do.	off
Wait for Actions	Only ask once every action in the state has finished.	off
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Jump

Makes a walking character jump. You set how high it should reach and the action works out the push from the gravity it is under. The jump is set as a height you can see, not found by trial and error.



Jump as it appears in the Actions tab.

When to use it

Covers the player's jump, a double jump, a wall jump, or an enemy hopping a gap. It needs a character body (a CharacterBody2D or CharacterBody3D), the kind a walking player or guard uses, and does nothing at all on a rigid body, an area or static scenery. A crate is launched with Apply Force. A jumping character normally has a Move Character running too, on the same state or the parent state above it, to keep steering in the air.

> The push is made in the physics step, so the action finishes on the physics step after the jump, once the body has really left the ground.

See also

Move Character 134, **Apply Force** 107, **Set Gravity** 120, **Set Velocity** 122

Try it

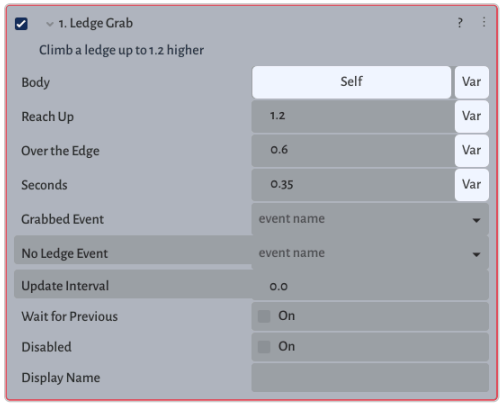
See the action work on its own with **Open Example** in the Add Action window, or open `examples/character/jump/jump.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Body	The character body to push. Leave it empty for the node this machine works on, as on the player's machine. Point it at anything else and no jump happens. Var: a Node.	Self
Height	How high the jump should reach, in metres in 3D or pixels in 2D. Two metres is a comfortable human hop. Put in a 2D game it starts at 64 pixels instead, about two character heights for a small sprite. Press Var to read it from a variable so a charged jump can go higher. Var: a float.	2.0 in 2D: 64 px
Only From the Floor	Ticked, as it starts, it refuses to jump unless the body is standing on something. Untick it for a double jump and count the jumps yourself with a variable, because without a count the player can climb the sky. Var: a bool.	on
Still Allowed After a Ledge	Only while Only From The Floor is ticked. How long after the character walks off a ledge the jump is still allowed, in seconds. Games that feel good to jump in have this, and the starting tenth of a second is about right. A player who pressed a moment too late gets the jump they meant, and nobody notices the grace time. Set it to zero to allow a jump only while the character is really standing on something. One jump uses the allowance up, and it can never turn into a second jump in mid air. Var: a float.	0.1
Waits to Land For	Only while Only From The Floor is ticked. A jump pressed just before the character lands waits this long for the ground, in seconds, and happens as it touches down. Without it, a press a few hundredths of a second early is swallowed and the player feels the game missed it. A tenth of a second, as it starts, is about right. Set it to zero to refuse straight away. The refused event is only sent once the wait is over. Var: a float.	0.1
Cut Short Input	Name the jump button, and letting go of it while the character is still rising cuts the jump short. A tap is a little hop, and a held press goes the full height. Empty, the default, always jumps the full height. The button is remembered on the body, and the jump is still cut short if the machine has moved on to another state by then.	empty
Off Walls	Ticked, it can also jump off a wall it is sliding down, not only off the floor (a wall jump). Off to begin with. Var: a bool.	off
Jumped Event	Event sent as it jumps, for a sound, a puff of dust or a change of animation. Empty sends nothing.	empty
Refused Event	Only while Only From The Floor is on. With that off no jump is ever refused. The event sent when it would not jump because it was not standing on anything, letting the template grunt, play an empty-handed animation or do nothing gracefully. Empty sends nothing.	empty
Falls	Leave it ticked, because this is what pulls the character down again once it has jumped. Untick it only in a game with no gravity at all, or the character goes up and never comes back. A 2D body whose Motion Mode is Floating is seen from above and never falls, even with this ticked. Var: a bool.	on
Gravity	The gravity the jump is worked out against, which decides how hard the push has to be. Zero, the default, uses the Gravity of the Move Character walking the character, or the project's gravity when that is zero too. Any change made by Set Gravity is counted on top, and the jump still reaches the height you asked for. Var: a float.	0.0
Away From Wall	Only when Off Walls is ticked. How hard a wall jump pushes the character away from the wall, in metres per second in 3D or pixels per second in 2D. The push fades by itself, and steering does not drag the character straight back. Too small a number and it slides down the same wall again straight away.	—

Ledge Grab

Catches the top of a wall the character is against and pulls it up onto it, then finishes.



Ledge Grab as it appears in the Actions tab.

When to use it

A jump that falls a little short becomes a climb, not a drop. Think of it as a tall step taken slowly. It looks for a top within reach of whatever the character is pressed against and carries it up and over in the time you give it. While it climbs it holds its state up and places the character itself. Give it a separate state and leave that state on FINISHED. The character has to be against something for there to be a ledge at all. Normally the state before this one had a Move Character pushing it into the wall.

> The character is placed along its climb, not pushed. A pull-up is a fixed move onto a known spot, and physics would scrape it up the face of the wall. It goes up first and over second, as a climbing person does. At the end it is set down on the ledge with a real move, and the next state knows straight away that it stands on the floor and is no longer against the wall.

See also

Step Up 136, **Jump** 132, **Wall Cling** 138, **Move Character** 134

Try it

See the action work on its own with **Open Example** in the Add Action window, or open `examples/character/ledge_grab/ledge_grab.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Body	The character body to pull up. Leave it empty for the node this machine works on. Var: a Node.	Self
Reach Up	The highest top it can pull itself onto, measured up from where the character is, in metres in 3D or pixels in 2D. Anything above this is out of reach and stays a wall. Press Var to read it from a variable, and a stronger character can then climb higher. Var: a float.	1.2 in 2D: 38.4 px
Over the Edge	How far past the edge it ends up, in metres in 3D or pixels in 2D. Enough that it stands on the ledge, not teetering on its lip. About half the character's width works well. Var: a float.	0.6 in 2D: 19.2 px
Seconds	How long the pull-up takes. Longer reads as an effort, shorter as a vault. Var: a float.	0.35
Grabbed Event	Event sent as it catches the ledge, for a grunt or a change of animation. Empty sends nothing.	empty
No Ledge Event	Event sent when there is no top within reach, letting the template go back to falling. The action finishes either way and the state never hangs. Empty sends nothing.	empty

Move Character

Walks a character about. Every physics step it pushes the body along, keeps it on the ground, slides it round walls and, if you ask, turns it to face where it is going.



When to use it

The action that moves a walking character, whether a player driven by the keys, a guard closing in or a shopkeeper on his rounds. It needs a character body (a CharacterBody2D or CharacterBody3D), which Godot only moves when something walks it. A rigid body, an area or static scenery cannot be walked, and the Output panel warns about it. Apply Force or Set Velocity is the way to move those. Jump, Set Velocity and anything else that sets a speed need this running in the same state or a parent state above it, or the body will not budge. It heads straight at things, while Navigate To walks round walls and furniture.

> Walking never holds a state up, and a state with only this in it counts as done straight away. Put the action on the state that should keep walking, or on a parent state so everything nested inside keeps walking too.

See also

Jump 132, **Chase** 251, **Navigate To** 239, **Set Velocity** 122

Try it

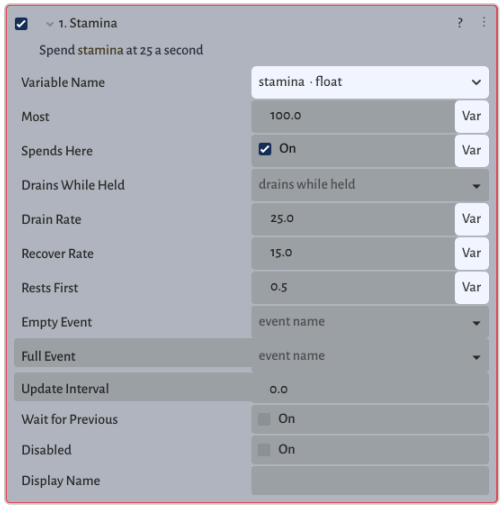
See the action work on its own with **Open Example** in the Add Action window, or open `examples/character/move_character/move_character.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Body	The character body to move. Leave it empty for the node this machine works on, as on the player's machine. Var: a Node.	Self
Heading	A Direction follows a direction, usually the one Get Input Vector fills in from the keys, at Speed. Toward A Node heads for a node as it moves, and Toward A Place walks to a spot and stops. A Velocity takes direction and speed in one, as a Vector or Expression works it out. Choices: A Direction, Toward A Node, Toward A Place, A Velocity.	A Direction
Direction	For A Direction or A Velocity, it says which way to go as a pair of numbers. Press Var to read the keys' variable. For A Direction anything longer than one is trimmed, so a diagonal is not faster. For A Velocity its length is the speed. In 3D a direction typed as a Vector3 keeps its up and down in Global and Local space. With Falls off, that is how a climb goes up a wall. Var: a Vector2 or a Vector3, or a Node for where it stands.	0, 0
Space	Used when Heading is A Direction. Global reads the direction as the world's, Local reads it as the body's so forward means the way it faces, and Camera reads it as the camera's. In a 3D game players expect the keys to work that way. In a 2D game Camera means the same as Global. Side On is for a 3D game played from the side, as a 2D platformer is. Right is along X, it never moves in depth, and with Falls off a stick's up and down climb up and down. It marks the body as seen from the side, which lets Dash read a stick the same way and Slips Round Corners work in 3D too. Choices: Global, Local, Camera, Side On.	Global
Camera	Used when Space is Camera. Name the camera the directions should be read from. Empty, the default, uses whichever camera is current and keeps working when the game switches camera. Var: a Node.	empty
Speed	How fast at full tilt, in metres a second in 3D or pixels a second in 2D. Five is a brisk run in 3D, and put in a 2D game it starts at 160 pixels a second instead. Not shown for A Velocity. Var: a float.	5.0 in 2D: 160 px
Feel	Fills in the five numbers below for you, without your needing to know what any of them mean. Tight is quick to get going and quick to stop, heavy on the way down with a sharp moment at the top. That makes it precise and fair, for a game about landing on things. Floaty is slow to build up with a long hang at the top. Heavy gives little steering once it is off the ground, and a jump becomes a decision. Arcade puts everything back to instant full speed and a plain arc, with no weight to it at all. Three of the five are worked out from the Speed above, which makes one choice work in a 2D game and a 3D one alike, so set Speed first. Once the numbers are filled in it goes back to Custom and they are yours to change. Picking another choice writes over them again. Choices: Custom, Tight, Floaty, Heavy, Arcade.	Custom
Acceleration	How quickly it works up to speed and back down, in speed gained each second. Zero, the default, starts and stops on the spot, for an arcade feel. A few times the speed gives a short slide, but too small a number and the character feels like it is on ice. Var: a float.	0.0
Turn to Face	Off to begin with. Ticked, the body is turned to face the way it is walking. Seen from the side in a 2D game where things fall, it flips the pictures left and right and the character stays upright, never rolling over. Var: a bool.	off
Turn Speed	Only matters with Turn to Face ticked. It sets how quickly it swings round, as a share of the way each second. Ten, the default, turns smartly, and three or four gives a visible, heavier turn. Var: a float.	10.0
Falls	Ticked, gravity pulls it down whenever it is off the ground. Untick it for a game seen from above where nothing falls, and then the up-and-down part of the direction is walked as well, so the character can move freely about the floor. A 2D body whose Motion Mode is Floating is seen from above and never falls, even with this ticked. Var: a bool.	on
Gravity	How strongly it is pulled down while Falls is ticked. Zero, the default, uses the project's gravity setting. A bigger number gives a heavier, faster fall. Var: a float.	0.0
Falls Faster Than It Rises	How much harder gravity pulls on the way down than on the way up. At 1, how it starts, a jump is a plain arc. Above 1 it rises floatily and drops smartly, as most platform games are tuned. A value of 1.5 to 2 gives weight without feeling slow. Var: a float.	1.0
Floats at the Top	How strongly gravity pulls around the top of a jump, as a share of its usual strength. At 1, how it starts, nothing changes. Below 1 the character hangs there a moment, long enough for a player to see where they will land. Var: a float.	1.0
Fastest Fall	The fastest the character will ever fall, in metres per second in 3D or pixels per second in 2D. Zero, how it starts, lets it keep speeding up. Cap it for long drops, because a fall that keeps getting faster looks wrong, and can pass through a thin floor in a single step. Var: a float.	0.0
Slips Round Corners	This works seen from the side, with Falls ticked, in a 2D game or in a 3D one with Space set to Side On (then in metres). A jump that clips the corner of a ceiling by a few pixels slips this far sideways round it and carries on up, instead of bumping its head and dropping. 4 to 6 pixels makes jumps past a ledge forgiving without looking like cheating. Zero, the default, bumps as a plain body does. Var: a float.	0.0
Acceleration in the Air	How quickly it gets up to speed while off the ground, in speed gained per second. Zero, how it starts, uses the same figure as Acceleration. A smaller one makes a character commit to a jump, since it cannot turn on the spot in mid air. Var: a float.	0.0
Toward	Only when Heading is Toward A Node. The node to head for, and it keeps heading for it as it moves. It goes straight at it, and a wall in between will stop it. Navigate To finds a way round.	—
Toward Variable	Only when Heading is Toward A Node. Name a variable holding a node and it heads for whatever that variable holds, in place of the node named above. A guard can then change who it chases while the game runs, such as the one Can See or Event On Collision has just filled in. Empty uses Toward. Named but empty at the time, it heads nowhere and stands still.	—
Toward Place	Only when Heading is Toward A Place. The spot to walk to, as two numbers in a 2D game and three in a 3D one. Press Var to read it from a variable, such as the one Mouse Position fills in, for a point-and-click game.	—
Stop Within	Only when Heading is Toward A Node or Toward A Place. Stop this far short of what it is heading for, in metres in 3D or pixels in 2D, so it does not shove into it. Too small a number and the character presses against what it is chasing and never settles.	—
Arrived Event	Only when Heading is Toward A Node or Toward A Place. Event sent the first time it gets that close, once for each visit to the state, so a guard can look about or a shopkeeper can start talking. Empty sends nothing.	—

Stamina

Keeps a meter that empties while something is being done and fills up again while it is not.



Stamina as it appears in the Actions tab.

When to use it

For a sprint, put it in the Sprint state and it drains there. Put it in the Walk state with Spends Here unticked and it fills there. It keeps one variable between 0 and Most, and the rest of the template reads that variable like any other. Show it on a bar with Set Value, ask about it with a Check, and leave the state when it runs out. It never holds a state up.

> Both events are sent on the way past a limit, never over and over while the meter sits at one, so nothing listening for them is flooded.

See also

Timer 68, **Expression** 45, **Clamp** 42, **Set Value** 215

Try it

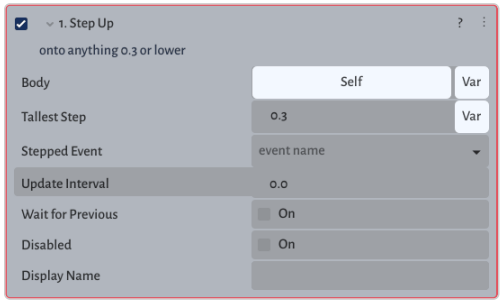
See the action work on its own with **Open Example** in the Add Action window, or open `examples/character/stamina/stamina.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Variable Name	The variable the meter is kept in. It is made if the template has not got one, and it starts full, ready for a player's first sprint.	stamina
Most	How full a full meter is. It never goes above this or below 0, so nothing else has to keep it in range. Var: a float.	100.0
Spends Here	Ticked, as it starts, this is a state that spends the meter. Untick it for a state that only lets the meter come back, such as resting, walking or standing about. A template usually has the spending in one state and this unticked in the state it returns to. That pair is a complete stamina system with no sums in it. Var: a bool.	on
Drains While Held	Drain only while this control is held, so one state can both spend and recover. Empty, as it starts, it drains for as long as the state runs, the state itself being the spending.	empty
Drain Rate	How much the meter empties by each second while it is being spent. Var: a float.	25.0
Recover Rate	How much it fills by each second while it is not being spent. At 0 it never recovers in this state, for a tank refilled only by a pickup. Var: a float.	15.0
Rests First	Wait this long after the last of the spending before it starts filling again. A short rest here stops a player tapping the button to keep the meter full. Only shown while Recover Rate is above zero. Var: a float.	0.5
Empty Event	Event sent as it reaches 0. Leave the state on this to stop the sprint, the swim or the block. It is sent once on the way past, not every frame it sits at 0. Empty sends nothing.	empty
Full Event	Event sent once, when it is full again. Empty sends nothing.	empty

Step Up

Lets a walking character walk up a step that would otherwise stop it dead.



Step Up as it appears in the Actions tab.

When to use it

Without it a character body treats a kerb, a stair, the lip of a doorway or a low crate as a wall, and a player made to jump over a kerb thinks the game is broken. Put it in the same state as the Move Character walking the character, above it in the list. It works in 2D and in 3D and costs nothing in the open, because it only acts while the character stands on something and is pressed against something else.

> Which way is forward comes from the wall the character is pressed against, not from its speed. Walking into something takes that speed away, and a blocked character looks still to anything that asks.

See also

Move Character 134, **Jump** 132, **Touching** 125, **Raycast** 116

Try it

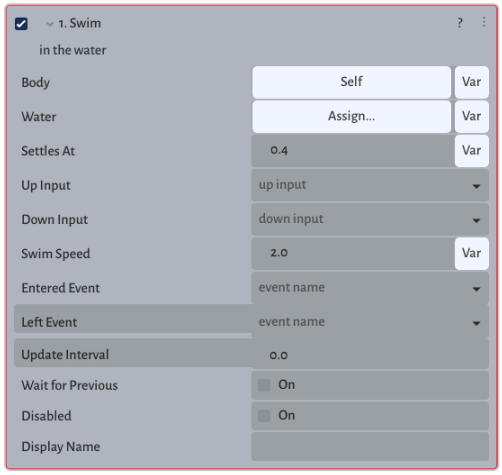
See the action work on its own with **Open Example** in the Add Action window, or open `examples/character/step_up/step_up.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Body	The character body to help up. Leave it empty for the node this machine works on. Var: a Node.	Self
Tallest Step	The tallest thing the character can step up onto, in metres in 3D or pixels in 2D. Anything taller is still a wall, and the character cannot stroll up the scenery. Keep it well below what the character can jump, or there is nothing left for jumping to do. Press Var to read it from a variable, letting a bigger character take a bigger step. Var: a float.	0.3 in 2D: 9.6 px
Stepped Event	Event sent each time it steps up onto something, for a footstep or a grunt. Empty sends nothing.	empty

Swim

Lets a character move through water without falling through it, with controls to swim up and down.



Swim as it appears in the Actions tab.

When to use it

For a pool, a river, a lake or a flooded room. Inside the water gravity stops pulling, and the character settles, rises or sinks at the speed you choose. The water is an Area with a collision shape, an invisible shape you draw in the scene. A pool can be any shape, and the surface is wherever you drew it. Put it beside the Move Character already walking the character, which goes on steering. A lower Speed on that Move Character in the swimming state makes the water feel thick.

> The character's weight is given back as it leaves the water, and again when the state ends, so it can never be left hanging weightless on dry land. Gravity is switched off the way Set Gravity does it, and the walking actions go on working without knowing about the water.

See also

In Water 131, **Get Water Depth** 130, **Move Character** 134, **Set Gravity** 120, **Touching** 125, **Event On Collision** 109

Try it

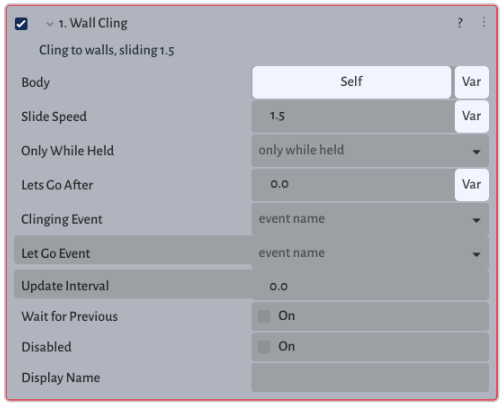
See the action work on its own with **Open Example** in the Add Action window, or open `examples/character/swim/swim.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Body	The character body that swims. Leave it empty for the node this machine works on. Var: a Node.	Self
Water	The Area the water fills, an Area2D in a 2D game or an Area3D in a 3D one, or a Terrain with a lake or sea, whose water it finds by itself. Without one the action has no water to work with, and it warns about it. Var: a Node.	empty
Settles At	How fast it drifts while no swimming control is held, in metres per second in 3D or pixels per second in 2D. Above zero it sinks slowly, below zero it bobs up to the surface, and zero hangs it where it is. Var: a float.	0.4 in 2D: 12.8 px
Up Input	The control that swims it upward. Empty leaves it with no way up but the drift above.	empty
Down Input	The control that swims it downward. Empty leaves it with no way down but the drift above.	empty
Swim Speed	How fast it swims up or down while one of those controls is held, in metres per second in 3D or pixels per second in 2D. Var: a float.	2.0 in 2D: 64 px
Entered Event	Event sent as it gets into the water, for a splash. Empty sends nothing.	empty
Left Event	Event sent when it is out of the water again. Leave the swimming state on this event. Empty sends nothing.	empty

Wall Cling

Catches a falling character on a wall so it slides down slowly, or stops on the wall altogether.



Wall Cling as it appears in the Actions tab.

When to use it

Paired with a Jump set to go Off Walls, it builds wall jumping. On its own it turns a shaft into something a player climbs, not falls down. Put it in the state beside the Move Character already walking the character. It never holds the state up. It works for as long as the state runs, and does nothing while the character stands on the ground or is nowhere near a wall. It needs a character body (a CharacterBody2D or CharacterBody3D).

> Clinging only ever slows a fall. It does not hold the character against the wall sideways. A Move Character pressing into the wall keeps it there.

See also

Jump 132, **Dash** 128, **Move Character** 134, **Touching** 125

Try it

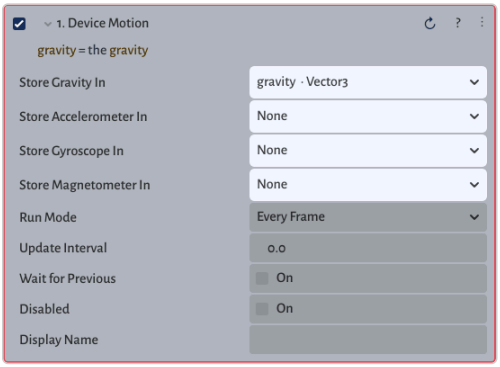
See the action work on its own with **Open Example** in the Add Action window, or open `examples/character/wall_cling/wall_cling.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Body	The character body to catch. Leave it empty for the node this machine works on. Var: a Node.	Self
Slide Speed	How fast it slides down the wall, in metres per second in 3D or pixels per second in 2D. At 0 it sticks to the wall without sliding. The speed is a cap, not a push, and a character still rising from a jump is left alone, not shoved downward. Press Var to read it from a variable, and a better grip can be earned. Var: a float.	1.5 in 2D: 48 px
Only While Held	Name a control and the character only clings while it is held. The player holds it to stay on the wall and releases it to drop. Empty, as it starts, clings whenever it touches a wall.	empty
Lets Go After	Let go by itself after this many seconds on the wall, however long the state runs, so no wall can be rested on for ever. Zero, as it starts, holds on for as long as the state does. Var: a float.	0.0
Clinging Event	Event sent as it catches the wall, for a scrape of sound or a change of animation. Empty sends nothing.	empty
Let Go Event	Event sent when it comes off, however that happened. It may have landed, left the wall, had the control let go, or run out of time. Leave the state on this event to go back to falling. Empty sends nothing.	empty

Device Motion

Reads a phone's or tablet's motion sensors into variables. They tell which way is down, how hard it is being shaken, how fast it is turning, and which way a compass would point. Each reading is three numbers, X, Y and Z.



Device Motion as it appears in the Actions tab.

When to use it

Tilt a phone and a marble can roll round a maze, because Store Gravity In gives which way is down and that direction pushes the ball. A car can be steered by turning the phone like a wheel, and shake-to-reroll dice watch the accelerometer jump. On a computer there is nothing to read.

> On a phone each sensor you read must also be ticked in Project Settings, under Input Devices > Sensors, or it reads (0, 0, 0). A warning names each one that is not ticked.

See also

Get Input Vector 143, **Virtual Joystick** 158, **Vibrate** 157, **Touch Gesture** 156

Try it

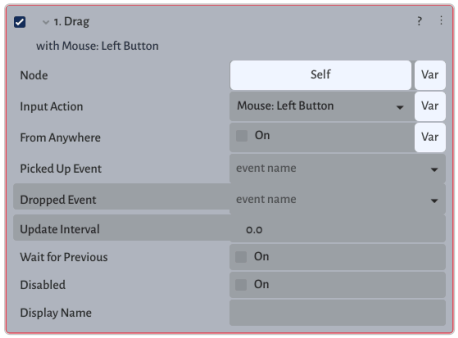
See the action work on its own with **Open Example** in the Add Action window, or open `examples/input/device_motion/device_motion.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Store Gravity In	The Vector3 variable that gets gravity (which way is down, as the phone is held right now). Tilting the phone moves it, and tilt steering reads this one. It starts as gravity, and empty stores nothing.	gravity
Store Accelerometer In	The Vector3 variable that gets the accelerometer. That is how hard the phone is being pushed about, with gravity included, in metres per second per second. A shake makes it jump well above the pull of gravity. Empty, the default, stores nothing.	empty
Store Gyroscope In	The Vector3 variable that gets the gyroscope. It says how fast the phone is turning round each of its three lines, in radians a second. It is 0 while the phone is held still. Empty, the default, stores nothing.	empty
Store Magnetometer In	The Vector3 variable that gets the magnetometer. It shows which way the Earth's magnetic field runs through the phone, as a compass feels it. Empty, the default, stores nothing.	empty

Drag

Lets the player pick something up with the mouse and move it. They press on it, move the mouse, and let go to drop it. It works in 2D and in 3D, and in 3D the node slides across flat ground at the height it was picked up. A physics body that would normally fall is held still while it is carried and falls again as soon as it is dropped.



Drag as it appears in the Actions tab.

When to use it

A game of patience needs only one state with this in it. The player can pick up and drop cards as often as they like, because the action keeps working for as long as the state lasts and never holds the state up. Puzzle pieces, inventory icons and furniture in a room work the same way.

> The node needs a collision shape to be pressed on, unless it is a sprite or a control, which are pressed on wherever they are drawn.

See also

Mouse Pick 149, **Mouse Position** 150, **Carry** 92, **Touch Gesture** 156

Try it

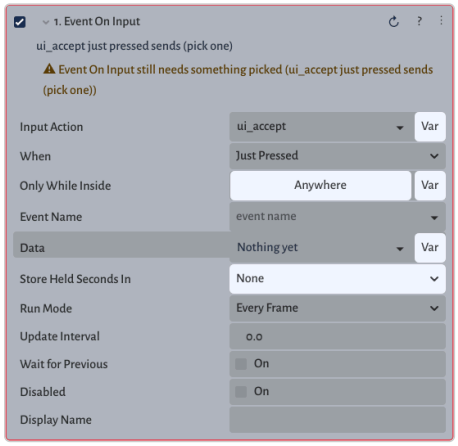
See the action work on its own with **Open Example** in the Add Action window, or open `examples/input/drag/drag.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	What gets dragged. Left empty it drags the node the machine works on, or point it at another node to drag that one. A node is one item in the scene tree, such as a sprite, a body or a label. Var: a Node.	Self
Input Action	The input that holds the node while it is carried. The list names keys, mouse buttons and gamepad inputs, such as Mouse: Left Button or Key: Space. It also names your own input actions, the named controls set up under Project Settings > Input Map, where one name can stand for several keys. Any Key or Button takes whatever the player presses. The node follows the pointer while this is held down and drops when it is let go. The default, the left mouse button, gives a plain press-move-release drag. A key the player has just chosen on a Controls page does not count as a press here until they let it go. Var: a String.	Mouse: Left Button
From Anywhere	Off, the default, means the node is picked up only when the pointer is over it, and the player can choose between several things on the screen. On means the press picks it up wherever the pointer happens to be. That suits a game where only one thing is ever draggable. Var: a bool.	off
Picked Up Event	An event sent as the node is picked up, for a sound, a small lift, or dimming everything else while it is in the player's hand. Left empty, no event is sent.	empty
Dropped Event	An event sent when the player lets go, to snap the node to a slot, check whether it landed somewhere legal, or play a thud. Left empty, none is sent. It is not sent if the state ends while the node is still held.	empty

Event On Input

Watches one key, mouse button or gamepad input and sends an event when the player presses it, lets it go, holds it, taps it twice quickly, or holds it down for a time. It can also keep how long it has been held in a variable, so a shot charges up while the button is down and fires with that charge when it is let go.



Event On Input as it appears in the Actions tab.

When to use it

Most player controls start here. Space going down sends JumpPressed, and a transition from the standing state leaves on it to the jumping state. Put it in the state that should be listening and lead a transition off the event it sends. A transition that only needs to know whether a key is down right now can ask the Input check in its condition.

> Pressed sends its event on every frame the input is down. Keep it for when you want a stream of events, and pick Just Pressed for a one-off.

See also

Input 144, **Input Combo** 145, **Get Input Vector** 143, **Send Event** 31

Try it

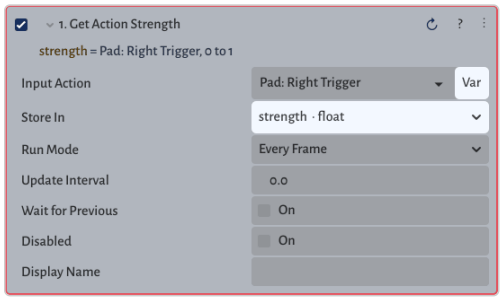
See the action work on its own with **Open Example** in the Add Action window, or open `examples/input/event_on_input/event_on_input.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Input Action	What to watch. Pick a key, mouse button or gamepad input from the list, such as Key: Space or Pad: Bottom Button (A / Cross), or one of your own input actions from Project Settings > Input Map, where one name can cover several keys. Any Key or Button suits a "press any key to start" screen. Var: a String.	ui_accept
When	Which moment sends the event. Just Pressed fires once as the input goes down and Just Released once as it comes up, however long it is held in between. Pressed sends the event again on every frame the input is down. Double Tapped needs a second press within Seconds of the first. Held For sends the event once after the input has been down for Seconds without being let go. On a stick, a direction counts as pressed only once the stick is half way over, and for Double Tapped the stick must come back near the middle between the two taps. A thumb wandering as the player walks never dashes. Choices: Just Pressed, Just Released, Pressed, Double Tapped, Held For.	Just Pressed
Only While Inside	Only while the player stands inside this area, such as a door opened with E only when you are beside it. Pick the area, such as the door's Near. Empty counts the press anywhere. When a transition's When list is used to pick "E is pressed beside it", this is set for you, with the Near area made round the thing. Var: a Node.	empty
Event Name	The event sent when the chosen moment arrives. Name it for what happened, such as JumpPressed or FireReleased, and lead a transition off that name. Left empty, no event goes out.	empty
Data	A value carried with the event, such as which key was pressed or how hard to jump. A transition waiting for an event that carries a value can put it in a variable, and Get Event Data can read it too. It starts empty, and the event then carries no value. Var: any kind.	empty
Store Held Seconds In	Names a variable that counts up in seconds while the input is down and keeps its last value after it is let go. The state the player lands in can then read how long the button was held. Leave it empty and nothing is kept. The count restarts at 0 each time the state begins, and a button already down as the state starts does not arrive part-charged.	empty
Seconds	Only when When is Double Tapped or Held For. For Double Tapped it is the most time allowed between the two presses, and for Held For it is how long the input must stay down before the event is sent. The default of 0.3 makes a brisk double tap. Raise it towards 0.5 if players find the tap too fussy, and use something like 1 for a charged attack.	—

Get Action Strength

Puts how far an input is pressed into a variable, as a number from 0 to 1. A trigger being pulled, or a stick pushed one way, gives every value in between. A key gives 1 while it is held and 0 when it is not.



Get Action Strength as it appears in the Actions tab.

When to use it

Good for when "pressed or not" is not enough. A car can pull away gently on a half-squeezed trigger. The variable reads about 0.5, ready to scale the speed, and the engine note can rise with it. It only stores a number, so to send an event on a press, use Event On Input.

> A key or a button only ever reads 0 or 1. The in-between numbers come from a trigger or a stick.

See also

Get Input Vector 143, **Input** 144, **Set Velocity** 122, **Ease Variable** 44

Try it

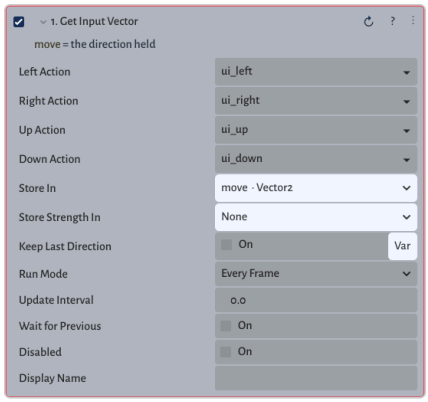
See the action work on its own with **Open Example** in the Add Action window, or open `examples/input/get_action_strength/get_action_strength.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Input Action	What to read. Pick a key, mouse button or gamepad input from the list, such as Pad: Right Trigger or Key: W, or one of your own input actions from Project Settings > Input Map, which can stand for several keys. The default is a pad's right trigger, the usual accelerator. Var: a String.	Pad: Right Trigger
Store In	Names the variable the number goes into, always between 0 and 1. The default, strength, is fine for one of these. When a node reads two triggers, give each a different name, such as throttle and brake.	strength

Get Input Vector

Puts the direction the player is holding into a variable. The arrow keys, WASD and a stick are all read the same way. Another action, such as Move Character, then uses that variable to move something, and a machine drives a character with no code at all.



Get Input Vector as it appears in the Actions tab.

When to use it

A player character usually starts with two actions, this one reading the keys into move, and Move Character or Set Velocity using move. A stick pushed gently gives a short direction and pushed all the way a full one, and the character walks or runs to match. For a second stick that only aims, tick Keep Last Direction. On a phone, Virtual Joystick fills the same kind of variable from a thumb on the screen.

> Fill in both of a pair or neither, because a direction needs both halves. With no whole pair filled in, nothing moves and the action prints a warning.

See also

Move Character 134, **Virtual Joystick** 158, **Get Action Strength** 142, **Set Velocity** 122

Try it

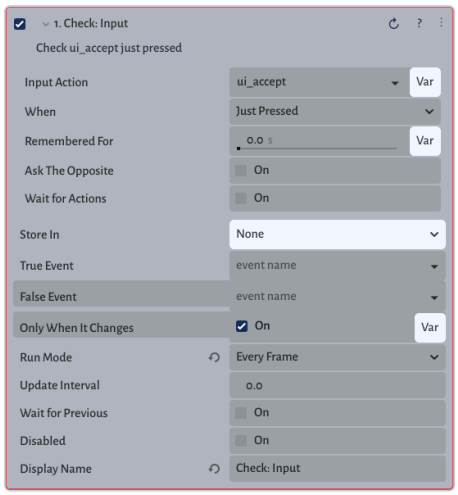
See the action work on its own with **Open Example** in the Add Action window, or open `examples/input/get_input_vector/get_input_vector.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Left Action	What counts as going left. Pick a key, mouse button or gamepad input from the list, such as Key: A or Pad: Left Stick Left, or one of your own input actions (a name under Project Settings > Input Map that can cover several keys). The default, ui_left, is the engine's built-in left, answered by the arrow keys and a pad's d-pad.	ui_left
Right Action	What counts as going right, chosen the same way. Take all four directions from one family, such as four keys, one stick, or four of your own input actions. A mixture only half works.	ui_right
Up Action	What counts as going forward in a 3D game, or up in a 2D one. With WASD this is Key: W. Clear this and Down for a side-scroller. The character then only goes left and right, and the other pair carries on working.	ui_up
Down Action	What counts as going back in a 3D game, or down in a 2D one. With WASD this is Key: S. Cleared along with Up, nothing ever pushes forward or back.	ui_down
Store In	Names the variable the direction goes into, as a flat direction running from -1 to 1 each way. The default is move. Nothing reads it by itself, so point Move Character's Direction at it with Var. Give it another name when one node has two of these, such as one to walk and one to aim.	move
Store Strength In	Optionally names a second variable holding how far the direction reaches, from 0 when nothing is held up to 1 for a full push. Use it to choose between a walk and a run animation, or to leave a state as soon as the player stops pushing. Leave it empty and nothing is kept.	empty
Keep Last Direction	Off, the default, means the direction drops back to 0 as soon as the player lets go, as walking needs. On means the last direction held stays in the variable after the player lets go, and an aim on a stick keeps pointing where it was. The stored strength still falls to 0 while nothing is held, and that tells you the player has stopped pushing. Var: a bool.	off

Input CHECK

Asks whether a key, a mouse button or a gamepad control is being pressed. It can answer "held down right now", or only on the single tick it goes down or comes back up.



Input as it appears in the Actions tab.

When to use it

The gate on a state. Keep climbing while the up key is held, or let go of a ledge when the grab button comes up. Event On Input sends an event by itself, but this gives a yes-or-no answer to hang a transition on, and it combines with other checks.

> One press moves a machine one step. A state entered because of a press does not take that same press again in the same frame, and a single press cannot run you through a chain of states.

See also
Event On Input 141, **Get Action Strength** 142, **Input Combo** 145, **Get Input Vector** 143

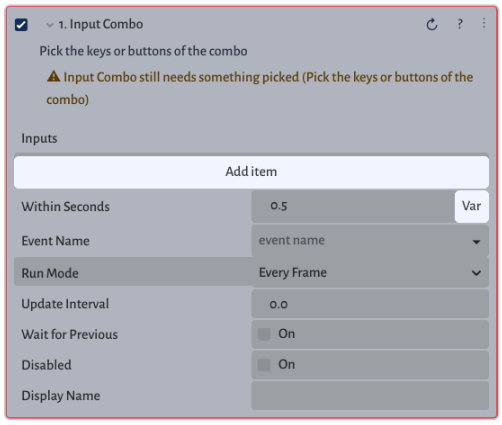
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/input/input/input.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Input Action	What to watch. Type the name of an input action set up under Project Settings > Input Map, such as "ui_accept", or name a key, mouse button or gamepad control directly. The default "ui_accept" is Enter and Space. A name that matches nothing never says yes. Var: a String.	ui_accept
When	Pressed says yes for as long as the control is down. Just Pressed, the default, says yes only on the frame it goes down, and Just Released only on the frame it comes back up. Use Pressed for something that goes on, like a crouch, and Just Pressed for something that happens once, like a jump. A stick direction counts as pressed only once the stick is half way that way, so a thumb a little off level while running never presses the way beside it. An input action you added to the Input Map, used on its own, keeps the Deadzone set there. Set 0.5 on one for a stick. Choices: Pressed, Just Pressed, Just Released.	Just Pressed
Remembered For	Just Pressed only. A press this many seconds ago still counts, once, if no check has taken it yet. Set 0.15 and a jump pressed a moment before landing still happens on landing, and fast controls feel fair. At 0, how it starts, only the press this frame counts. On a stick, one push is one press, however much the thumb wobbles while holding it. Var: a float.	0.0
Ask The Opposite	Turns the answer round into "it is not being pressed", so a run state drops back to a walk when the key is let go. It turns round the true-or-false answer the Check stores as well, which swapping the True Event and False Event over cannot do.	off
Wait for Actions	Holds the question back until every other action in the state has finished, so the press is read after the state's own work. Left off, the default, it is asked as soon as the state runs.	off
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Input Combo

Sends an event when a run of keys or buttons is pressed one after another, in the right order, each soon enough after the one before. Pressing anything else in between, or waiting too long, starts the run again from the beginning.



Input Combo as it appears in the Actions tab.

When to use it

Fighting-game special moves are built with it, such as Down then Attack for a slide kick, and a secret door can open on Left, Right, Left. It watches for as long as the state lasts. Sequence sounds similar but runs a fixed series of things the game does, while this one listens for presses the player makes.

> With Inputs left empty the event is never sent.

See also

Event On Input 141, **Input** 144, **Sequence** 38, **Send Event** 31

Try it

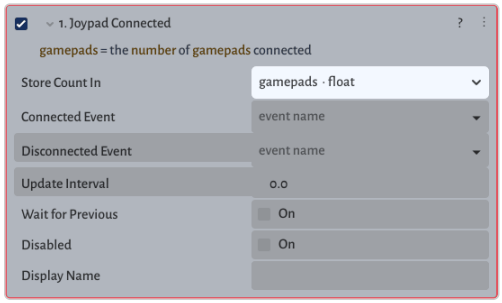
See the action work on its own with **Open Example** in the Add Action window, or open `examples/input/input_combo/input_combo.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Inputs	The keys or buttons of the combo, in the order they must be pressed. Add a row for each one and pick it from the list, which offers keys, mouse buttons and gamepad inputs by name as well as your own input actions, the named controls you can set up under Project Settings > Input Map. Two or three are plenty, because a long run is hard for a player to land.	empty
Within Seconds	The most time allowed between one press and the next. The default of 0.5 is brisk but fair. Below about 0.2 only a practised player will manage it, and much above 1 the combo starts catching presses the player never meant as a combo. Var: a float.	0.5
Event Name	The event sent when the last press of the run lands. Lead a transition off it to the state that does the special move. The run then starts again from the beginning, so the player can do it twice in a row.	empty

Joypad Connected

Keeps the number of gamepads plugged in up to date in a variable, and sends an event when one is plugged in or taken out while this state runs.



Joypad Connected as it appears in the Actions tab.

When to use it

When a pad's battery dies mid-fight, the game can pause and ask for the controller back. When a second pad goes in, "Press Start" can appear for player two. At the title screen the count decides whether two-player is offered at all. It listens without holding the state up.

> It only hears about pads while this state is running, so put it in a state the game spends its time in, such as the one that is active during play.

See also

Last Device 147, **Pause Game** 222, **Variable** 40, **Rebind Input** 152

Try it

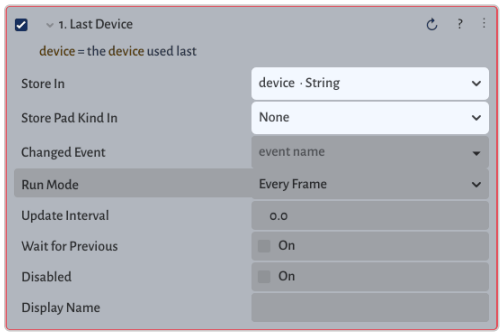
See the action work on its own with **Open Example** in the Add Action window, or open `examples/input/joypad_connected/joypad_connected.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Store Count In	Names the variable the number of gamepads goes into. It is set as the state begins and kept up to date while the state runs. The default, gamepads, is fine. A check on it can then ask whether it is 0, or 2 or more, and lead somewhere accordingly.	gamepads
Connected Event	An event sent when a gamepad is plugged in. Fill it in to show a prompt or bring a second player into the game. Left empty, none is sent.	empty
Disconnected Event	An event sent when a gamepad is unplugged or its battery runs down. Fill it in to pause the game and put a message on the screen. Left empty, none is sent.	empty

Last Device

Keeps which device the player used last in a variable, as the words keyboard, mouse, gamepad or touch, and sends an event when it changes. The screen can then show the right prompt, such as "Press E" at the keyboard and "Press A" on a pad.



Last Device as it appears in the Actions tab.

When to use it

A prompt over a door, the hints along the bottom of a menu and the icons on a Controls page all depend on it. Store Pad Kind In also names the make of pad, and an Xbox pad can then get Xbox button pictures and a PlayStation pad PlayStation ones.

> A knock of the desk or a stick resting slightly off centre does not count as using that device, so the prompts do not flicker.

See also

Joypad Connected 146, **Rebind Input** 152, **Set Text** 213, **Variable** 40

Try it

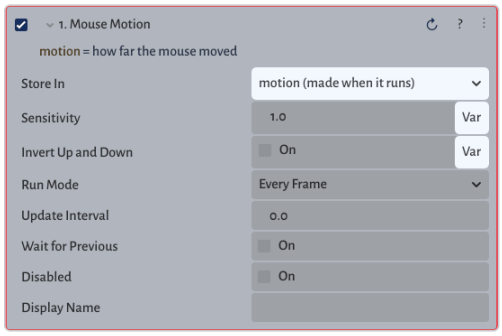
See the action work on its own with **Open Example** in the Add Action window, or open `examples/input/last_device/last_device.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Store In	Names the variable the device goes into, always one of the words keyboard, mouse, gamepad or touch. The default, device, is fine. A check on that variable then chooses which prompt to show. It says keyboard until the player has touched anything.	device
Store Pad Kind In	Optionally names a variable holding the make of the pad, as the words xbox, playstation, nintendo or other, for choosing the right button pictures. It is the pad used last, or the first one plugged in when none has been used yet. With no pad plugged in it is set to empty text. Leave the box empty and nothing is kept.	empty
Changed Event	An event sent as soon as the player swaps from one device to another. Lead it to a state that redraws the prompts. Left empty, no event is sent and the variable changes quietly under whatever is reading it.	empty

Mouse Motion

Puts how far the mouse moved since the last frame into a variable, as two numbers for across and down, in pixels on the screen. It is 0, 0 while the mouse is still.



Mouse Motion as it appears in the Actions tab.

When to use it

To turn a turret with the mouse, add the across number to its angle each frame. A Sensitivity of 0.2 slows the turn. It keeps working when Set Mouse Mode hides the mouse and holds it in the window, where the pointer never moves. A first-person view is simpler with Mouse Look.

See also

Mouse Look 191, **Mouse Position** 150, **Set Mouse Mode** 154, **Drag** 140

Try it

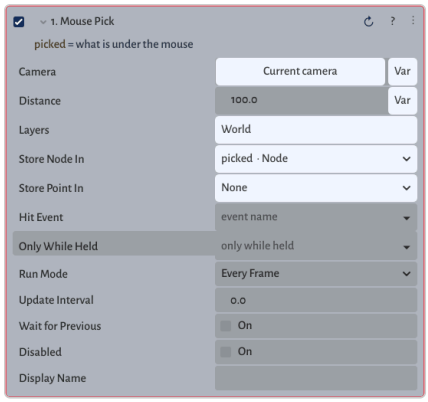
See the action work on its own with **Open Example** in the Add Action window, or open `examples/input/mouse_motion/mouse_motion.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Store In	The variable the movement goes into, called motion to start with. Right and down are positive. A variable that holds anything other than two numbers is left alone, and a warning names it and what it holds.	motion
Sensitivity	How much each pixel the mouse moves counts for. 1, the default, gives pixels and 0.1 gives a tenth, for a slower turn. A negative number turns it the other way. Var: a float.	1.0
Invert Up and Down	On, moving the mouse up gives a positive number, for players who like to aim that way round. Off to start with. Var: a bool.	off

Mouse Pick

Works out what is under the mouse pointer and puts it in a variable. In a 3D game it casts a line from the camera through the cursor and takes the first thing it meets. In a 2D game it takes whatever collision shape the cursor is inside.



Mouse Pick as it appears in the Actions tab.

When to use it

Point-and-click games start here. In a strategy game it puts the unit under the cursor into picked, and other actions give that unit its orders. Hit Event lets a transition fire as the cursor comes onto something new. It finds the thing under the pointer, and the place under it comes from Mouse Position.

> The thing being picked needs a collision shape on one of the chosen layers. A picture with no shape is never found.

See also

Mouse Position 150, **Drag** 140, **Raycast** 116, **Set Mouse Mode** 154

Try it

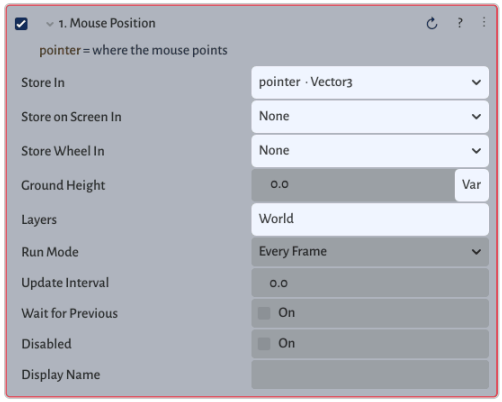
See the action work on its own with **Open Example** in the Add Action window, or open `examples/input/mouse_pick/mouse_pick.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Camera	Only in a 3D game. In 2D nothing is cast from a camera and the box is not offered. The camera to look through. Left empty, it uses the camera showing the game. Name one only when the scene has several cameras and you must be sure which does the looking. Var: a Node.	empty
Distance	Only in a 3D game, for the same reason. How far along the line to look, in metres. The default of 100 reaches most of a room-sized scene. Raise it for a wide outdoor level, and lower it when only things close to the camera should ever be picked. Var: a float.	100.0
Layers	Which layers count as something the cursor can pick. The default, World, holds walls, ground and anything you add yourself. Tick Player, Enemies or Pickups to pick the ready-made ones too. When the cursor should see through walls, glass or scenery, put the clickable things on a separate layer (Name a new layer...) and tick only that one.	World
Store Node In	Names the variable the thing under the cursor goes into. The default is picked. Over empty space the variable is emptied, and a check on it then tells you whether the cursor is over something right now.	picked
Store Point In	Optionally names a variable holding the place the cursor landed on, in the game's world. Use it to put a marker where the player clicked, or to send something to that spot. Leave it empty and nothing is kept.	empty
Hit Event	An event sent when something new comes under the cursor, not on every frame it rests there. The cursor sitting still on the same thing does not send it again, and moving off it and back does. Fill it in to highlight the thing or show its name. Left empty, none is sent.	empty
Only While Held	Left empty, it looks on every update, for hovering. Name a key, mouse button, gamepad input or one of your own input actions from Project Settings > Input Map and it only looks while that is held down, which turns it into a click. Hold the left mouse button and whatever is under the cursor is picked. With Any Key or Button, any press counts. A key the player has just chosen on a Controls page does not count until they let it go.	empty

Mouse Position

Puts where the mouse pointer is into a variable, as a place in the game's world that other actions can use. In a 2D game that is the spot in the level under the pointer. In a 3D game it is the spot on whatever the pointer is over, or on flat ground where it is over nothing.



Mouse Position as it appears in the Actions tab.

When to use it

Aiming usually starts here. With the pointer stored, Look At can turn a turret to face it, Move To can send a character to where the player clicked, and Instantiate can drop something on the spot. For a cursor drawn on a layer over the game, Store on Screen In gives the screen's own pixels.

> In a 3D game where nothing is under the pointer and the camera is aimed away from that flat ground, no new place can be worked out and the variable keeps whatever it had.

See also
Mouse Pick 149, **Look At** 96, **Move To** 97, **Instantiate** 84

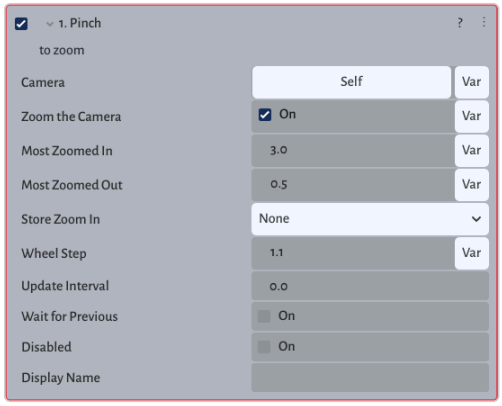
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/input/mouse_position/mouse_position.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Store In	Names the variable the place in the world goes into, as two numbers in a 2D game and three in a 3D one. The default is pointer. Left empty, no world place is worked out, and that is fine when you only want the screen place or the wheel.	pointer
Store on Screen In	Optionally names a variable holding where the pointer is on the screen, in pixels from the top left corner. Use it to move a control, such as a drawn crosshair or a tooltip, to the pointer. Leave it empty and nothing is kept.	empty
Store Wheel In	Optionally names a variable holding how far the wheel turned this frame, counted in notches. Pushing it away from you is positive, pulling it towards you negative, and it sits at 0 while the wheel is still. A trackpad gives fractions of a notch. Leave it empty and nothing is kept.	empty
Ground Height	In a 3D game, the height in metres of the invisible flat ground the pointer lands on where there is nothing under it. The default of 0 is ground level. Raise it to the height of a tabletop or a deck if that is the surface the player is pointing at. Var: a float.	0.0
Layers	In a 3D game, which layers count as something under the pointer. The default is World. Untick a layer and the pointer passes through it and lands on the flat ground.	World

Pinch

Zooms with two fingers on a phone or tablet. Spread them apart to zoom in and bring them together to zoom out. It zooms a camera by itself, or puts how far the fingers have spread into a variable for something else to use. On a computer the mouse wheel does the same, handy for testing without a phone.



Pinch as it appears in the Actions tab.

When to use it

Strategy maps and puzzle boards are the usual home for it, where the player spreads two fingers for a closer look. It works with a Camera2D or a Camera3D and keeps working for as long as the state lasts without holding the state up. Here the player does the zooming. A zoom the game makes by itself, to a set amount over a set time, is Camera Zoom.

> The camera it is pointed at has to be a Camera2D or a Camera3D as the state begins, or there is nothing for it to zoom and only the variable changes.

See also

Camera Zoom 187, **Touch Gesture** 156, **Virtual Joystick** 158, **Mouse Position** 150

Try it

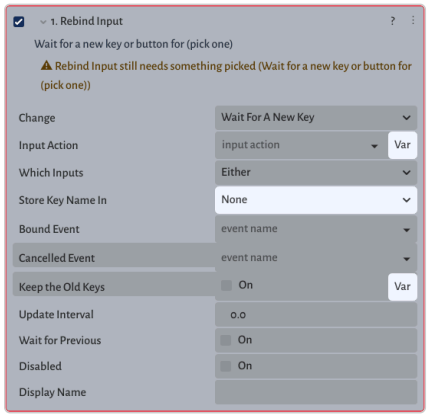
See the action work on its own with **Open Example** in the Add Action window, or open `examples/input/pinch/pinch.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Camera	The camera to zoom. Leave it empty for the node the machine works on, when the machine sits on the camera itself. Otherwise point it at the Camera2D or Camera3D you mean. The view the camera has as the state begins is what everything below is measured against. Var: a Node.	Self
Zoom the Camera	On, the default, the camera zooms as the player pinches. Off, the camera is left alone and only the number in Store Zoom In changes, for when something else, such as a hand-built map view, should do the zooming. Var: a bool.	on
Most Zoomed In	How far in the player may zoom, counted as times the view the camera started with. The default of 3 lets them get three times as close. Lower it when the art starts to look coarse close up. Var: a float.	3.0
Most Zoomed Out	How far out the player may zoom, counted the same way. The default of 0.5 lets them pull back to half as close. Set it no lower than the point where the player can see past the edge of the level. Var: a float.	0.5
Store Zoom In	Optionally names a variable holding the zoom as times the view it started with, where 1 is where it began. Use it to show a zoom level on the screen, or to drive something else off the same pinch. Leave it empty and nothing is kept.	empty
Wheel Step	How much one notch of the mouse wheel zooms, as times the view before it. The default of 1.1 is a tenth closer per notch. Raise it for a faster wheel. Set it to 1 to leave the wheel alone when it already does something else in your game. Var: a float.	1.1

Rebind Input

Lets the player choose their own keys, the way a Controls page in an Options menu does. The state waits for the next key, mouse button or gamepad input they press and binds it to a control. The choice is kept and comes back the next time the game runs. It can also just report which key a control is on, for the label beside it, or put every control back as the game first had it.



Rebind Input as it appears in the Actions tab.

When to use it

Build a Controls page with it. Give it one row per control, each row showing its key with Only Show The Key, and a state that waits for a new key when the player clicks the row. A Reset button runs it once with Put Back Every Default. A control here is an input action, one of the named controls under Project Settings > Input Map, such as jump or fire, and it can answer to several keys. The player changes those controls, and the raw keys stay as they are.

> With Wait For A New Key the state does not finish until the player presses something or gives up. Give that state a way out other than FINISHED if it must not sit there for ever.

See also

Last Device 147, **Event On Input** 141, **Set Text** 213, **Grab Focus** 208

Try it

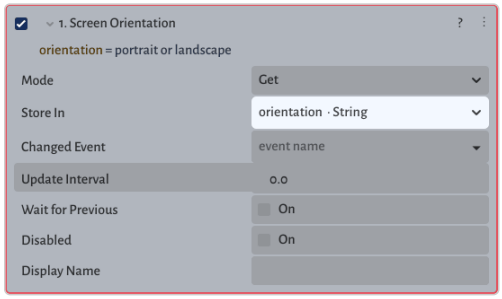
See the action work on its own with **Open Example** in the Add Action window, or open `examples/input/rebind_input/rebind_input.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Change	What this one does. Wait For A New Key stops and listens until the player presses something, then binds it. Only Show The Key changes nothing and reports the key the control is on, for the label on each row of the page. Put Back Every Default forgets everything the player has changed and restores the keys the game shipped with. Choices: Wait For A New Key, Only Show The Key, Put Back Every Default.	Wait For A New Key
Input Action	The control being changed or shown, such as jump or fire. It is hidden when Change is Put Back Every Default, which touches every control. Left empty there is nothing to bind and the action ends. Var: a String.	empty
Which Inputs	Which kind of input to wait for and to report. Either takes whatever the player presses, Keyboard And Mouse only keys and mouse buttons, and Gamepad only pad buttons, sticks and triggers. Use the last two for a page with a keyboard column and a gamepad column. A new input only replaces inputs of the same kind, so changing a control's key leaves its gamepad button alone. A control on a gamepad button never takes a stick push, and a thumb resting on the stick while the page waits cannot put Dash on walking left. Walking, on a stick or on keys alone, takes one. While it waits for one kind, a press of the other kind does nothing at all, and a gamepad's right button cannot close the menu behind a keyboard box. Only Escape and a gamepad's Select / Back give up the wait. Choices: Either, Keyboard And Mouse, Gamepad.	Either
Store Key Name In	Optionally names a variable holding what a player would call the key, such as W, Space, Mouse Right Button or Pad A, ready for Set Text to put in the label beside the row. It is filled in before the wait and again as soon as a new key is bound, so the label keeps itself up to date. Leave it empty and nothing is kept.	empty
Bound Event	Only when Change is Wait For A New Key. An event sent once the new key has been bound, to take the menu back to its resting state. Left empty nothing is sent, and the state still finishes on its own once the key is caught.	empty
Cancelled Event	Only when Change is Wait For A New Key. An event sent when the player gives up without choosing a key, leaving the control as it was. They give up with Escape, or a gamepad's Select / Back when Which Inputs is Keyboard And Mouse or Gamepad. Fill it in so a player who changes their mind gets back to the menu.	empty
Keep the Old Keys	Only when Change is Wait For A New Key. Off, the default, means the new key replaces the keys the control was on, as on most Controls pages. On means the new key is added to them, and a control can be built up to answer to several keys. Var: a bool.	off

Screen Orientation

Finds out whether the screen is upright (portrait) or on its side (landscape), and sends an event when it turns. It can also turn a phone's screen to the way you choose.



Screen Orientation as it appears in the Actions tab.

When to use it

With Get, a menu or a HUD can lay itself out again when the player turns the phone, or the game can pause until it is turned back. Set suits a game that plays upright but shows its map on its side, or one that lets the phone turn freely only on some screens. The way round a game starts is set in Project Settings, under Display > Window > Handheld, and this action changes it while the game runs.

> Set on a computer changes nothing, and warns about it the first time only.

See also

Screen Settings 225, **Last Device** 147, **Touch Gesture** 156, **Event On Input** 141

Try it

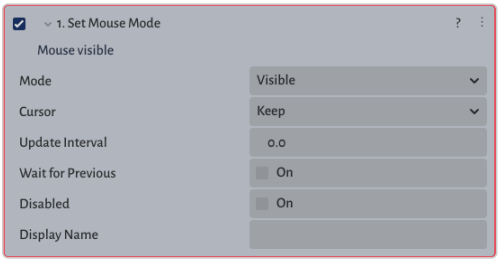
See the action work on its own with **Open Example** in the Add Action window, or open `examples/input/screen_orientation/screen_orientation.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Mode	Get reads which way the screen is, from the shape of the game's window, and keeps listening while the state runs. It works on a computer too, where the window can be dragged taller than it is wide. Set turns the screen, which only phones and tablets can do. Choices: Get, Set.	Get
Store In	Only when Mode is Get. The variable, holding words, that gets portrait when the screen is taller than it is wide and landscape otherwise. It starts as orientation, and empty stores nothing.	orientation
Changed Event	Only when Mode is Get. The event sent each time the screen turns from portrait to landscape or back while the state runs. A window that changes size without turning does not send it. Empty, the default, sends no event.	empty
Orientation	Only when Mode is Set. The way to turn the screen. Landscape and Portrait are the usual ways round, the Reverse ones are the other way up, and the Sensor ones let the phone follow how it is held (sideways only, upright only, or either).	—

Set Mouse Mode

Shows, hides, captures or confines the mouse pointer, and can swap the pointer for your own picture. A menu needs the pointer visible, while a game where the view turns with the mouse captures it, keeping it inside the window.



Set Mouse Mode as it appears in the Actions tab.

When to use it

As a level starts, capture the pointer for a first-person or over-the-shoulder view, and give it back when a menu opens. It also swaps the arrow for crosshairs. A pause menu needs none of this, because Pause Game frees the pointer and puts it back as it was on resuming.

See also
Mouse Look 191, **Mouse Position** 150, **Pause Game** 222, **Screen Settings** 225

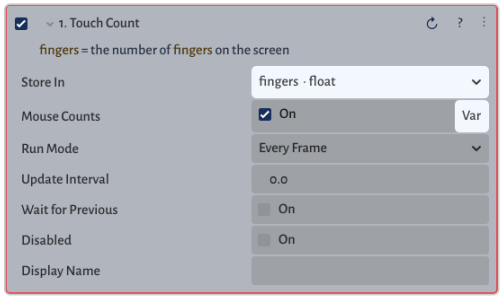
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/input/set_mouse_mode/set_mouse_mode.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Mode	Visible, where it starts, shows the ordinary pointer. Hidden hides it but lets it move about. Captured hides it and holds it in the window, for a game that turns the view with the mouse. It also stops the pointer straying onto a second screen. Confined keeps a visible pointer inside the window. Choices: Visible, Hidden, Captured, Confined.	Visible
Cursor	Keep, where it starts, leaves the pointer's picture alone and changes only whether it is shown. Custom puts your own texture in its place, such as crosshairs. Default brings the ordinary arrow back, for a menu after a level has set crosshairs. Choices: Keep, Custom, Default.	Keep
Texture	Only when Cursor is Custom. The picture shown in place of the arrow, such as crosshairs, a hand or a sword. Keep it small, because it is drawn at its own size. 32 by 32 pixels is usual. With no picture chosen the old pointer stays.	—
Hotspot	Only when Cursor is Custom. The spot in the picture that does the pointing, counted in pixels from its top left corner, such as the middle of crosshairs or the tip of a pointing finger. Left at 0, 0 the top left corner does the clicking, and crosshairs feel off by half their width.	—

Touch Count

Puts the number of fingers on the screen right now into a variable, 0 when none are. On a computer the left mouse button counts as a finger.



Touch Count as it appears in the Actions tab.

When to use it

Here the number of fingers is the control. Two fingers down might open the map, and a boost can last for as long as fingers stays above 0, while a thumb rests on the glass.

See also

Touch Gesture 156, **Pinch** 151, **Virtual Joystick** 158, **Drag** 140

Try it

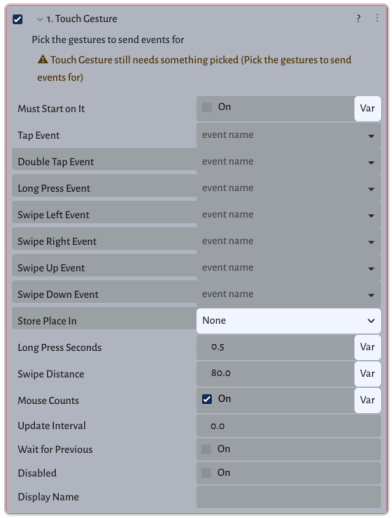
See the action work on its own with **Open Example** in the Add Action window, or open `examples/input/touch_count/touch_count.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Store In	The variable, a number, that gets how many fingers are down. It starts as fingers. Left empty, the action prints a warning.	fingers
Mouse Counts	On, the default, means the left mouse button held down counts as one finger, for trying a phone game on a computer. Off counts only real fingers. Var: a bool.	on

Touch Gesture

Sends an event when a finger taps, taps twice, presses and holds, or swipes left, right, up or down on a phone or tablet. Leave a gesture's event empty to ignore that gesture. On a computer the left mouse button counts as a finger.



Touch Gesture as it appears in the Actions tab.

When to use it

Most of a touch game's controls can come from this one action. A runner might jump on a swipe up, shoot on a tap and open its menu on a long press. With Must Start On It ticked, any sprite becomes an on-screen button, with no menu of controls to build. It keeps watching for as long as the state lasts and never holds the state up.

> A finger that has already sent Long Press Event sends nothing when it lifts, so a long press never also counts as a tap.

See also

Virtual Joystick 158, **Pinch** 151, **Drag** 140, **Event On Input** 141

Try it

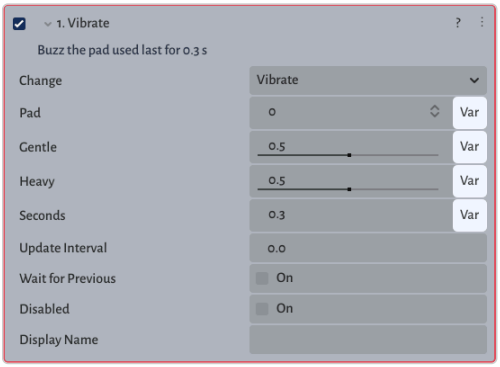
See the action work on its own with **Open Example** in the Add Action window, or open `examples/input/touch_gesture/touch_gesture.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Must Start on It	Off, the default, means a touch anywhere on the screen counts, for tap-to-shoot and swipes across the whole game. On means only touches that begin on the node count, whether over its collision shape, over its picture if it is a sprite, or inside its box if it is a control. This is how a sprite becomes a button. Var: a bool.	off
Tap Event	An event sent when a finger touches and lifts again quickly without moving. Fill it in to fire, to select, or to advance a line of dialogue. Left empty, taps are ignored.	empty
Double Tap Event	An event sent when a second tap follows the first quickly and close to it. Both taps still send Tap Event as well, so do not use both for two things that clash, such as fire on a tap and reload on a double tap.	empty
Long Press Event	An event sent once when a finger has been held still for Long Press Seconds. Good for a context menu, a charged action, or picking something up. A finger that has sent this sends nothing more when it lifts.	empty
Swipe Left Event	An event sent when a finger moves quickly to the left and lifts. Left empty, swipes to the left are ignored.	empty
Swipe Right Event	An event sent when a finger moves quickly to the right and lifts. Fill in the swipes in pairs, left and right or up and down, so the player is not left with a gesture that works only one way.	empty
Swipe Up Event	An event sent when a finger moves quickly up the screen and lifts. Left empty, swipes up are ignored.	empty
Swipe Down Event	An event sent when a finger moves quickly down the screen and lifts. Left empty, swipes down are ignored.	empty
Store Place In	Optionally names a variable holding where the gesture happened in the game's world, such as the spot of the tap or where the swipe began. Use it to spawn something where the player tapped. It is filled in for every gesture, even one whose event is left empty, and can be used by itself.	empty
Long Press Seconds	How long a finger has to stay still before it counts as a long press. The default of 0.5 feels deliberate without feeling slow. Below about 0.3 ordinary taps start being taken as long presses. Var: a float.	0.5
Swipe Distance	How far a finger has to travel, in pixels on the screen, before it counts as a swipe rather than a tap. The default of 80 works on a phone. Raise it on a large tablet where a small drag is easy to make by accident. Var: a float.	80.0
Mouse Counts	On, the default, the left mouse button counts as a finger, so a touch game can be played in the editor before it ever reaches a phone. Untick it on a game that is meant to read the mouse and the touch screen differently, so that clicking does not also set the gestures off. Var: a bool.	on
Node	Only when Must Start On It is ticked. The node a touch has to begin on for it to count. Leave it empty for the node the machine works on, or point it at another node to watch that one.	—

Vibrate

Makes a gamepad buzz for a hit taken, an explosion close by, or an engine running. On a phone or tablet with no gamepad, the device itself buzzes instead. It can also stop a buzz that is still going.



Vibrate as it appears in the Actions tab.

When to use it

Put it in the state that already does the thing being felt, beside the sound and the screen shake. That is the state where the player is hit, where the shot fires, or where the car crashes. Keep the buzzes short and let the big ones be rare, or the player stops noticing them.

> With no gamepad plugged in nothing is felt at all, unless the game is running on a phone or tablet, where the device buzzes instead.

See also

Camera Shake 186, **Joypad Connected** 146, **Play Sound** 175, **Screen Effect** 202

Try it

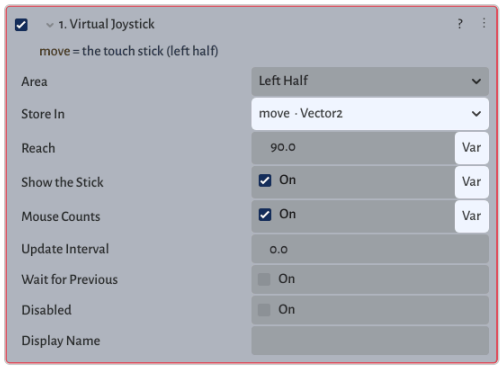
See the action work on its own with **Open Example** in the Add Action window, or open `examples/input/vibrate/vibrate.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Change	Vibrate, the default, starts a buzz. Stop ends one straight away. Use it when the engine cuts out, or when the player pauses the game part way through a long rumble. Choices: Vibrate, Stop.	Vibrate
Pad	Which gamepad buzzes. The default of 0 means the one the player used last, for a one-player game. 1 is the first pad plugged in, 2 the second, and so on, for a game where each player should feel their own hits. Var: an int.	0
Gentle	Only while Change is Vibrate. How hard the light, fast buzz is, from 0 to 1. Use it for small things like a footstep, a tick or a pick-up. The default of 0.5 is a clear but polite buzz. Set it to 0 when only the heavy rumble should be felt. Var: a float.	0.5
Heavy	Only while Change is Vibrate. How hard the deep, slow rumble is, from 0 to 1. It carries the big things, such as an explosion, a crash or a heavy landing. Set it to 0 for a small, sharp buzz with no weight behind it. Var: a float.	0.5
Seconds	Only while Change is Vibrate. How long the buzz lasts. The default of 0.3 is about right for a hit. Use 0.1 for a tick and a second or more for an explosion, and remember that a long buzz over a nervous moment quickly becomes annoying. Var: a float.	0.3

Virtual Joystick

An on-screen joystick for a phone or tablet. The player puts a thumb down in its part of the screen and pushes, and the direction goes into a variable in the same form Get Input Vector gives from the keys. Move Character then walks a character with it. The stick draws itself where the thumb comes down and disappears when it lifts. On a computer the left mouse button counts as a thumb, for trying it without a phone.



Virtual Joystick as it appears in the Actions tab.

When to use it

A twin-stick phone game puts one on the left half for walking into move and a second on the right half for aiming into aim. Two sticks never take the same thumb, and both work together. It keeps working for as long as the state lasts and never holds the state up.

> The direction goes back to 0 when the thumb lifts and when the state ends, and whatever it was moving stops.

See also

Get Input Vector 143, **Move Character** 134, **Touch Gesture** 156, **Pinch** 151

Try it

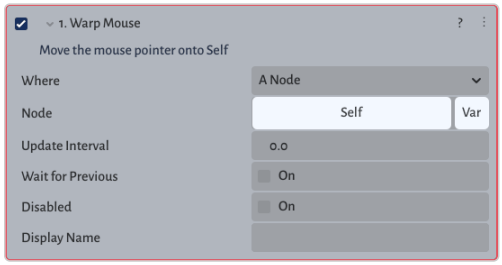
See the action work on its own with **Open Example** in the Add Action window, or open `examples/input/virtual_joystick/virtual_joystick.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Area	Which part of the screen a thumb has to come down in for this stick to claim it. Left Half is the usual place for walking, Right Half for aiming, and Whole Screen suits a game with only one stick. Two sticks on the same half would fight over the thumb, so give each a different half. Choices: Left Half, Right Half, Whole Screen.	Left Half
Store In	Names the variable the direction goes into, a flat direction running from -1 to 1 each way. The default is move. Point Move Character's Direction at it with Var, and give a second stick a different name, such as aim.	move
Reach	How far the thumb has to push, in pixels on the screen, to reach full speed. The default of 90 suits a phone. Lower it so a small thumb movement runs, and raise it on a tablet to give finer control between a walk and a run. Var: a float.	90.0
Show the Stick	On, the default, the stick is drawn on the screen where the thumb lands, over everything else, and disappears when the thumb lifts. Turn it off when you have drawn your own stick, or want none shown at all. The direction is still read either way. Var: a bool.	on
Mouse Counts	On, the default, means the left mouse button counts as a thumb, for testing on a computer. Untick it where clicking should do something else entirely and must not also drive the stick. Var: a bool.	on

Warp Mouse

Moves the mouse pointer in one jump, as if the player had moved the mouse there. It can go to the middle of the screen, to a place on it, or onto a node such as a button or a character.



Warp Mouse as it appears in the Actions tab.

When to use it

When a shop opens, putting the pointer on its first button saves the player hunting for it. After a cutscene it can go back where it was, and a tutorial can move it onto the enemy it is talking about.

> Only Windows, macOS and Linux let a game move the pointer. On a phone, a tablet or a web page nothing moves, and a single warning says the platform cannot do it.

See also

Mouse Position 150, **Set Mouse Mode** 154, **Grab Focus** 208, **Mouse Pick** 149

Try it

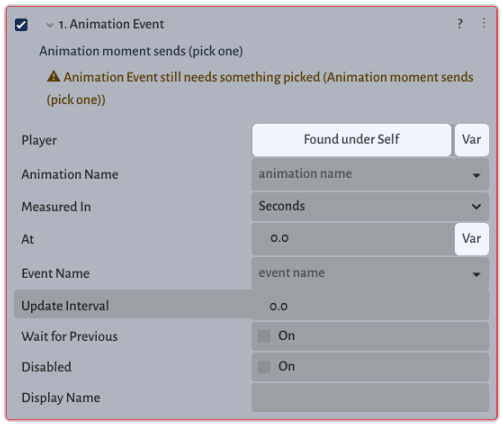
See the action work on its own with **Open Example** in the Add Action window, or open `examples/input/warp_mouse/warp_mouse.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Where	Where the pointer goes. A Node, the default, is where a node shows on the screen, such as the middle of a button, or where a 2D or 3D node is seen. A Place is a spot on the screen you give. The Middle is the middle of the game's view. Choices: A Node, A Place, The Middle.	A Node
Node	Only when Where is A Node. The node to move the pointer onto. Left empty it is the node this Machine works on, so a Machine on a button moves the pointer onto that button. A 3D node needs a camera showing it, and one behind the camera leaves the pointer where it was. Var: a Node.	Self
Position	Only when Where is A Place. The spot on the screen, in pixels from the top left of the game's view, the same way Mouse Position gives a spot on the screen. It starts at the top left corner.	—

Animation Event

Sends an event at a chosen moment inside an animation while it plays. Every time the animation passes that moment, loop after loop, the event goes again.



Animation Event as it appears in the Actions tab.

When to use it

A footstep sound as each foot lands, at 0.3 seconds into the walk and again at 0.8, is two of these in the walking state. It never holds the state up, so the state's other actions carry on alongside it. To know when an animation has finished, let Play Animation wait for the end, or use Play Sprite Animation's Finished Event.

> It only watches while its own state is running, so keep it in the state where the animation is playing.

See also

Play Animation 162, **Play Sprite Animation** 163, **Send Event** 31, **Travel To** 170

Try it

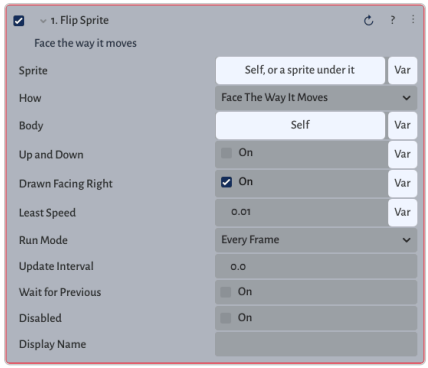
See the action work on its own with **Open Example** in the Add Action window, or open `examples/animation/animation_event/animation_event.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Player	The AnimationPlayer (Godot's node that plays animations along a timeline) or the animated sprite to watch. Left empty it uses the node this Machine works on, or the first player or animated sprite anywhere under it, whatever it has been named. Fill it in only when there is more than one and you mean a particular one. Var: a Node.	empty
Animation Name	The name of the animation to watch. Left empty it watches whichever animation happens to be playing, so the moment fires in every one of them. Type a name and the moment fires only while that animation plays.	empty
Measured In	How the moment below is counted, either in Seconds, the way an AnimationPlayer keeps time, or in Frames, the way an animated sprite counts. It starts at Seconds. Set it to match the node you are watching. Pick the other and the node's counting is used anyway, with a warning saying the number was read as seconds or as a frame. Choices: Seconds, Frames.	Seconds
At	The moment itself. It is seconds counted from the start of the animation when Measured In is Seconds, or which frame when it is Frames, counting the first frame as 0. It starts at 0, the very beginning. Var: a float.	0.0
Event Name	The event sent each time the moment goes past. Point a transition out of this state at it, or let another state hear it. Left empty, nothing is sent and a warning asks for one under Event Name.	empty

Flip Sprite

Turns a sprite round so it faces the way it is going. Most 2D characters are drawn facing one way and flipped for the other.



Flip Sprite as it appears in the Actions tab.

When to use it

Any 2D character that walks both ways needs one. Left on Face The Way It Moves it watches the body every frame and turns the picture with it, with nothing else to set up. Face A Direction follows the stick, not the body's real speed. Flipped, Not Flipped and Toggle turn it once and leave it. Something that should rotate, not mirror, turns with Look At or Rotate To.

> On Face The Way It Moves and Face A Direction the state keeps watching and never completes by itself, so give it another way out. The other three complete straight away.

See also

Play Sprite Animation 163, **Look At** 96, **Rotate To** 101, **Move Character** 134

Try it

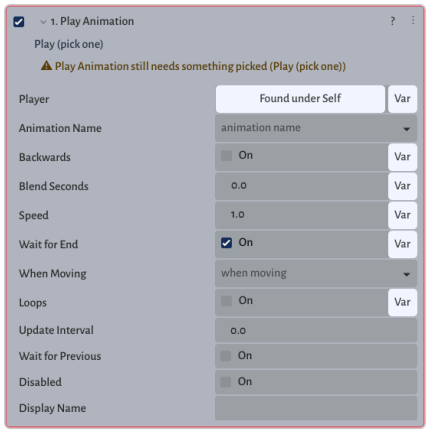
See the action work on its own with **Open Example** in the Add Action window, or open `examples/animation/flip_sprite/flip_sprite.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Sprite	The sprite to turn. It can be a Sprite2D, an AnimatedSprite2D, or their 3D versions for flat art in a 3D world. Empty, the default, uses the node this Machine works on if that is a sprite, or the first sprite directly under it. Var: a Node.	empty
How	Face The Way It Moves, the default, watches the body's own speed every frame and flips to match. Face A Direction follows a direction held in a variable. Flipped turns it round and leaves it, Not Flipped puts it back the way it was drawn, and Toggle swaps whichever way it was. Those three happen once, as the state starts, so the state can go on to complete. Choices: Face The Way It Moves, Face A Direction, Flipped, Not Flipped, Toggle.	Face The Way It Moves
Body	The body whose movement to follow, for Face The Way It Moves. It is taken off the panel for the other ways. Empty, the default, uses the node this Machine works on. A body with no speed to read (anything but a character body or a physics body) never flips. Var: a Node.	Self
Up and Down	On, it flips top to bottom, for something that turns over. Off, the default, flips left to right, as for a walking character. Var: a bool.	off
Drawn Facing Right	On, the default, says your artwork is drawn facing right, so moving left is the flipped one. Untick it if your art is drawn facing left and everything turns the other way. It only matters for the two ways that follow something, and is taken off the panel for the rest. Var: a bool.	on
Least Speed	How fast the body has to be moving before it counts as having changed direction, so the picture does not flicker when the character is all but still. It starts at 0.01. Raise it if a character being nudged on a slope keeps twitching round. It only matters for the two ways that follow something. Var: a float.	0.01
Direction Variable	Only when How is Face A Direction. The variable holding the direction the sprite should face, as an across-and-up pair such as the one Get Input Vector fills in from the stick or the keys. A character pushed backwards still faces the way the player is aiming. Left empty nothing is followed and the sprite never turns.	—

Play Animation

Plays a named animation on an AnimationPlayer, Godot's node that animates anything along a timeline. It can hold the state open until the animation ends, so the next thing follows on straight away.



Play Animation as it appears in the Actions tab.

When to use it

Doors swinging open, chest lids, levers and cutscene camera moves are all animated on a timeline. It can animate a character by itself too. Blend Seconds smooths one clip into the next, and Speed keeps a run in step with how fast the character moves. A 2D character drawn as frames is played with Play Sprite Animation, and a rig that already carries its animations and the blends between them is driven with Travel To.

> Pointed at a node that is not an AnimationPlayer, it plays nothing and finishes straight away. The warning names the node and suggests leaving Player empty, so one is looked for under the machine's node.

See also

Play Sprite Animation 163, **Animation Event** 160, **Travel To** 170, **Tween Property** 106

Try it

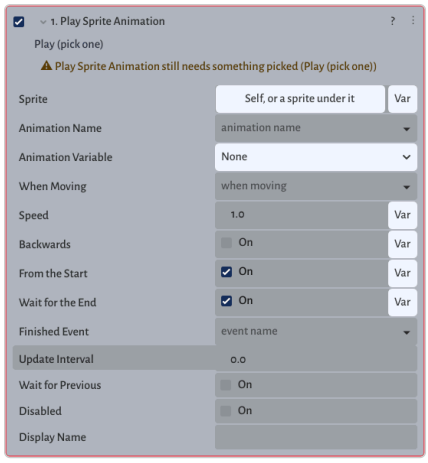
See the action work on its own with **Open Example** in the Add Action window, or open `examples/animation/play_animation/play_animation.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Player	The AnimationPlayer to play on. Left empty it uses the node this Machine works on if that is one, or the first AnimationPlayer anywhere under it, whatever it has been named. Most scenes need nothing filled in here. Var: a Node.	empty
Animation Name	The animation's name, spelled as it is in the player. With no animation by that name nothing plays, the action finishes straight away, and a warning names the player and the animation asked for, so check the spelling first. Read from a variable with Wait For End off, the clip follows the variable while the state runs. One action can play a jump clip while rising and a fall clip coming down.	empty
Backwards	On, the animation runs from its end to its start. Off is the default and plays it the normal way round. A door's opening animation played backwards closes it, with no second animation to make. Var: a bool.	off
Blend Seconds	How long this animation takes to take over from whatever was playing before it, in seconds. At zero, the default, it takes over in a single frame. That is right for a door or a lever, but on a person or an animal it reads as a jolt. A tenth or two of a second carries a walk into a run without the jolt, and much beyond half a second starts to feel soft and late. Var: a float.	0.0
Speed	How fast to play it. 1, the default, is the animation's own speed, 2 is twice as fast and 0.5 is half. Read from a variable, it keeps in step with that variable for as long as it plays. Match a run to the character's real speed and the feet stop sliding along the ground. It has to be above 0. At 0 or below the clip plays at its own speed and a warning says so. To play it the other way round, tick Backwards. Var: a float.	1.0
Wait for End	On, the default, the state finishes only once the animation has ended, and a transition on FINISHED carries on into the next thing. Off, the state finishes straight away and the animation plays on by itself. A looping animation never ends, so with one of those the state finishes straight away whatever this says. Var: a bool.	on
When Moving	Another clip to play while the character moves, such as walk, and the one in Animation Name again once it stands still. It is picked from the same list of the player's clips as Animation Name. It keeps switching for as long as the state lasts. Empty always plays the one above.	empty
Loops	Plays the clip round and round, even when the file says it plays once, as models from elsewhere often do. Idle, walk and run need it on, but a jump or an attack does not. Var: a bool.	off
Pace	Only when When Moving is filled in. The speed, in pixels or metres a second, the moving clip was made for. It plays faster or slower as the character goes faster or slower, so the feet keep up with the ground. 0 plays it at Speed.	—

Play Sprite Animation

Plays one of an animated sprite's named animations, such as walk or attack. An AnimatedSprite2D holds a set of animations made of frames, and this plays one by name.



Play Sprite Animation as it appears in the Actions tab.

When to use it

Every 2D character and most 2D props animate this way. A Running state plays run, and a Hurt state plays hurt once and leaves on FINISHED. It works on an AnimatedSprite3D too, for flat art in a 3D world. An AnimationPlayer, which animates along a timeline, is played with Play Animation.

> An animation set to loop in the sprite's frames can never be waited for. When a state runs straight on and you expected it to stop, look there first. A character with no animations yet gets them from a sprite sheet. Right-click it in the scene and pick NodeWorthy > Looks > Import Sprite Sheet....

See also

Flip Sprite 161, **Play Animation** 162, **Animation Event** 160, **Set Texture** 214

Try it

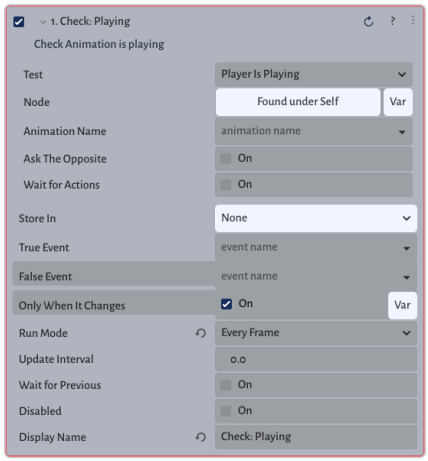
See the action work on its own with **Open Example** in the Add Action window, or open `examples/animation/play_sprite_animation/play_sprite_animation.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Sprite	The AnimatedSprite2D or AnimatedSprite3D to play on. Empty, the default, uses the node this Machine works on if that is an animated sprite, or the first one directly under it. Var: a Node.	empty
Animation Name	The animation to play, spelled as it is named in the sprite's frames. A name the sprite does not have plays nothing and finishes the state straight away, and a warning names the sprite and the missing animation. A character drawn from several sides, with walk down-left, walk up-right and so on, is asked for plain walk. The side facing the way it goes is the one played, and it keeps turning as the character does.	empty
Animation Variable	Read the animation's name from this variable, so one state can play whichever animation is called for, such as a weapon that names its own swing or a character whose walk changes with what it is carrying. Empty, the default, uses the name typed above. Whatever it holds is still looked up in the sprite's frames, so a name that is not there plays nothing, and a warning names it. Not waiting for the end, it follows the variable while the state runs. One In The Air state plays jump while rising and fall while falling, from a variable a formula keeps.	empty
When Moving	Played while the character is moving, such as run, and the Animation again once it stands still, switching by itself for as long as the state runs. One state that both stands and runs, as a player on the ground does, then looks right either way. Empty always plays the Animation. With it set, the state carries on straight away, with no end to wait for.	empty
Speed	How fast to play it, as a multiple of the animation's own speed. It starts at 1, its normal speed. Twice as fast is 2 and half is 0.5. Use it to speed a walk cycle up as a character runs. It has to be above 0. At 0 or below it plays at its own speed and a warning says so. Var: a float.	1.0
Backwards	On, it plays from the last frame to the first. Off is the default. It is how a door closes with its opening animation, or a character stands up with the sitting-down one, without drawing anything twice. Var: a bool.	off
From the Start	On, the default, it starts again from the first frame even if that animation is already playing. Turn it off to let an animation already running carry on, which keeps a walk cycle smooth when the state is entered again and again. Var: a bool.	on
Wait for the End	On, the default, the state finishes when the animation does, so a transition on FINISHED carries straight on into the next thing. Off, it finishes straight away and the animation plays on while the state gets on with something else. Added to a state that already does something and leaves on FINISHED, such as a jump, it starts off, so a clip longer than the jump never holds it up. An animation set to loop never ends, so with one of those the state finishes straight away whatever this says. Var: a bool.	on
Finished Event	Sent when the animation ends, whether or not the state waited for it. Empty sends nothing. A looping animation never ends, so it never sends this, and leaving the state before the animation is over means it is not sent either.	empty

Playing CHECK

Asks what an animation or a sound is doing. It can ask which state an AnimationTree is in, whether an AnimationPlayer, an animated sprite or a sound player is still going, whether music is on, or how far through a clip it has got.



Playing as it appears in the Actions tab.

When to use it

An attack state can end when the swing ends, with no guessing at a number of seconds. Player Past Fraction lands the hit nine tenths of the way through.

> Player Past Fraction also says yes once the named clip has stopped, because a clip that ran to its end is past any fraction of it. A clip that was never played is not.

See also

Play Animation 162, **Animation Event** 160, **Play Sprite Animation** 163, **Timeout** 67

Try it

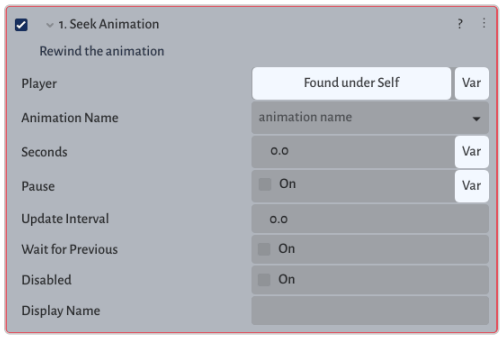
See the action work on its own with **Open Example** in the Add Action window, or open `examples/animation/playing/playing.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Test	What is being asked. Tree Is In State asks which state an AnimationTree is in, that being the node that blends animations together. Player Is Playing, the default, asks whether an AnimationPlayer is still running, and Player Past Fraction how far through its clip it is. Sprite Is Playing asks about an animated sprite, Sound Is Playing about a sound player, and Music Is Playing about the background music. Choices: Tree Is In State, Player Is Playing, Player Past Fraction, Sprite Is Playing, Sound Is Playing, Music Is Playing.	Player Is Playing
Node	Which node to ask. Leave it empty and the node the machine works on is used when it is the right kind of node, otherwise the first node of that kind under it is found, whatever it is called. Music Is Playing does not use this box at all. Var: a Node.	empty
Animation Name	The state name or clip name being asked about. Empty asks about whatever is playing, for "is anything still running", and for Player Past Fraction it means the clip played last. Tree Is In State needs a name, and left empty the answer is no. The two sound tests ignore it.	empty
Ask The Opposite	Turns the answer round into "it has stopped" or "the music is off". It turns round the true-or-false answer the Check stores as well, which swapping the True Event and False Event over cannot do.	off
Wait for Actions	Holds the question back until every other action in the state has finished, so it does not answer before the action that starts the animation has run. Left off, the default, it is asked as soon as the state runs.	off
State Machine	Only when Test is Tree Is In State. It names the state machine inside the AnimationTree, for a tree that holds more than one. When the tree's root is itself the state machine, as in most trees, it is found whatever this says.	—
Fraction	Only when Test is Player Past Fraction. How far through counts as past, from 0 at the start to 1 at the end. The default 0.9 is nine tenths through, a good place to land the blow of a swing.	—
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Seek Animation

Jumps a clip on an AnimationPlayer, the node that plays keyframed clips, to a point in its timeline, and the pose changes to match straight away. Seconds of 0 rewinds it to the start.



Seek Animation as it appears in the Actions tab.

When to use it

A saved game loads with a chest already open. Jump the chest's opening clip to its end with Pause on, and it shows open without playing. Animated sprites work by frame, through Stop Sprite Animation's Show Frame.

> An AnimationPlayer played through an AnimationTree shows what the tree decides. Jumping it then changes nothing you can see, and a warning points this out.

See also

Play Animation 162, **Stop Animation** 168, **Set Animation Speed** 166, **Stop Sprite Animation** 169

Try it

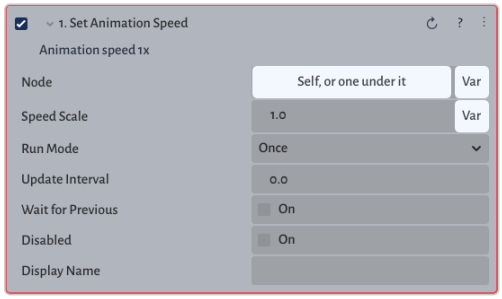
See the action work on its own with **Open Example** in the Add Action window, or open `examples/animation/seek_animation/seek_animation.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Player	The AnimationPlayer. Left empty it uses the node this Machine works on if that is one, or the first AnimationPlayer anywhere under it, whatever it has been named. Var: a Node.	empty
Animation Name	The clip to jump in. Left empty it keeps the clip the player has now, playing or paused. A clip the player does not have brings a warning.	empty
Seconds	The time to jump to, in seconds from the clip's start. 0, the default, rewinds it. A time past the clip's end brings a warning, and the clip jumps to its end. Var: a float.	0.0
Pause	On, the clip is held still at that moment, as Stop Animation's Pause does, and Stop Animation's Resume plays it on from there. Off, a playing clip plays on from the new moment at the speed it was playing, and a clip that was not playing shows the pose at that moment and stays still. Var: a bool.	off

Set Animation Speed

Changes how fast an animation plays while it plays. It sets the Speed Scale of an AnimationPlayer, an AnimatedSprite2D or an AnimatedSprite3D, and everything that node plays runs at that multiple until it is set again.



Set Animation Speed as it appears in the Actions tab.

When to use it

For a fan winding up as its power comes on, read Speed Scale from a variable that climbs from 0 to 3, with Run Mode on Every Frame. Play Animation's Speed only sets the speed a clip starts at, but this changes it later. Slowing everything in the game is a job for Set Time Scale.

> The speed stays with the player or sprite for every clip after this one, so set it back to 1 when the hurry is over.

See also

Play Animation 162, **Stop Animation** 168, **Play Sprite Animation** 163, **Set Time Scale** 66

Try it

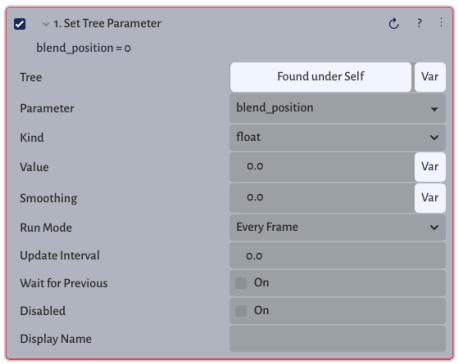
See the action work on its own with **Open Example** in the Add Action window, or open `examples/animation/set_animation_speed/set_animation_speed.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The AnimationPlayer, AnimatedSprite2D or AnimatedSprite3D. Left empty it uses the node this Machine works on if it is one, else the first AnimationPlayer under it, else the first animated sprite under it. Var: a Node.	empty
Speed Scale	How fast it plays. 1, the default, is as made, 2 twice as fast, 0.5 half as fast, and 0 holds it still. A negative number plays it backwards. Read it from a variable with Run Mode on Every Frame and the speed follows the variable. Var: a float.	1.0

Set Tree Parameter

Sets one of an AnimationTree's parameters. That can be a blend position, a blend amount, a condition or a OneShot. The Machine tells the character what is happening, such as how fast it is going, and the tree works out which animations to show.



Set Tree Parameter as it appears in the Actions tab.

When to use it

An AnimationTree is Godot's node that mixes animations together. Feed its blend position the character's speed every frame and the character eases from standing to walking to running. Fire a OneShot and a punch plays over the walk without stopping it. Turning a condition on lets the tree take the transitions that wait for it. Travel To is the other half of the pair. It says which state the tree's state machine is in, and this sets everything else.

> Picking a condition or a OneShot also sets Run Mode to Once. A condition is set as the state starts, and a OneShot asked again every frame would start over every frame and never be seen. With Smoothing above 0 nothing is written as the state starts, only from the first frame after, because writing straight away would leave nothing to ease.

See also

Travel To 170, **Play Animation** 162, **Stop Animation** 168, **Ease Variable** 44, **Set Property** 91

Try it

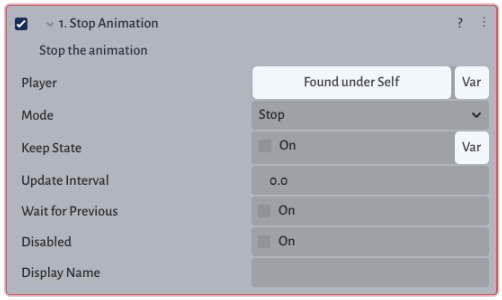
See the action work on its own with **Open Example** in the Add Action window, or open `examples/animation/set_tree_parameter/set_tree_parameter.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Tree	The AnimationTree to set. Left empty it uses the node this Machine works on if that is one, or the first AnimationTree anywhere under it, whatever it has been named. Var: a Node.	empty
Parameter	The parameter, picked from the list beside it, which shows what the tree really has. Each blend, each condition and each OneShot is there, with a line saying what it takes. It is written without the "parameters/" at the front, such as "Locomotion/blend_position", "conditions/is_jumping" or "Attack/request". A name the tree does not have changes nothing, and a warning names the tree and the parameter.	blend_position
Kind	What the parameter holds, set by itself when the parameter is picked from the list. It is float for a blend amount or a one-way blend, Vector2 for a two-way blend space, bool for a condition, and OneShot for a OneShot's request. Only the settings the kind uses are shown. Choices: float, Vector2, bool, OneShot.	float
Value	The number written when Kind is float. Usually it is read from one of this Machine's variables with Var. Point it at the variable holding the character's speed and the character keeps up by itself. Var: a float.	0.0
Smoothing	For a float or a Vector2 with Run Mode on Every Frame. It sets how fast the value eases toward the new one, in units a second. At 0, where it starts, the value is set straight away and can snap a character between walking and running, while a few units a second gives a soft change. On Once it is always set straight away. Var: a float.	0.0
Direction	The across-and-up pair written when Kind is Vector2, for a two-way blend space. It starts at (0, 0), the middle of the space. Press Var to read it from a variable, such as the one Get Input Vector fills in.	—
Condition	For a condition. On lets the tree's transitions that wait for it be taken, and off holds them back. Press Var to follow a bool variable, such as one saying whether the character is on the ground.	—
Request	For a OneShot. Fire plays it over whatever else is playing, Abort stops it straight away, and Fade Out lets it fade away over the OneShot's own fade-out time.	—
Wait for End	For a OneShot fired. On, the state waits until it has played to its end, so a transition on FINISHED carries on after the attack. Off, the default, finishes straight away and lets it play.	—

Stop Animation

Stops, pauses or resumes a clip on an AnimationPlayer, such as one Play Animation started.



Stop Animation as it appears in the Actions tab.

When to use it

A fan switched off, or an alarm light that stops flashing, has to stop before its clip does. With Toggle on one key, a pause menu can hold a clip still and let it carry on again.

> Resume carries on only after a Pause. After Stop, or with no clip ever played, it does nothing and a warning says there is nothing paused to resume.

See also

Play Animation 162, **Stop Sprite Animation** 169, **Set Tree Parameter** 167, **Playing** 164

Try it

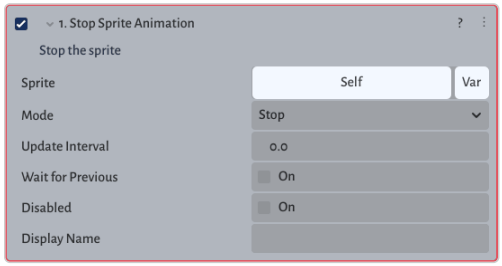
See the action work on its own with **Open Example** in the Add Action window, or open `examples/animation/stop_animation/stop_animation.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Player	The AnimationPlayer to stop. Left empty it uses the node this Machine works on if that is one, or the first AnimationPlayer anywhere under it, whatever it has been named. Var: a Node.	empty
Mode	Stop ends the clip and goes back to its start, as Godot's stop() does. Pause holds it where it is. Resume plays it on from where Pause left it. Toggle pauses a playing clip and resumes a paused one, for one key that pauses and unpauses. Choices: Stop, Pause, Resume, Toggle.	Stop
Keep State	Only for Stop. On, the pose the clip had reached stays. Off, the default, goes back to how the clip began. Var: a bool.	off

Stop Sprite Animation

Stops, pauses or resumes an AnimatedSprite2D or AnimatedSprite3D, or holds it on one of its frames.



Stop Sprite Animation as it appears in the Actions tab.

When to use it

A switch drawn as frame 0 off and frame 1 on is held on the right one by Show Frame, reading the frame from the switch's variable with Var. Pause suits a coin that stops spinning once it is taken, or a character frozen mid-step while a pause menu is up.

> A Play Sprite Animation still running with When Moving keeps switching the sprite between its standing and moving animations, and so starts it again. Stop that state too, or put this in the state that replaces it.

See also
Play Sprite Animation 163, **Stop Animation** 168, **Flip Sprite** 161

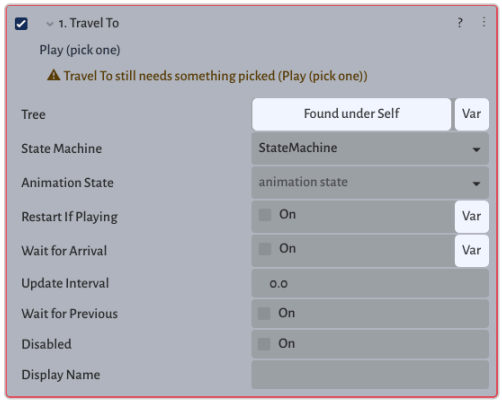
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/animation/stop_sprite_animation/stop_sprite_animation.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Sprite	The AnimatedSprite2D or AnimatedSprite3D to stop. Empty, the default, uses the node this Machine works on if that is an animated sprite, or the first one directly under it. Var: a Node.	empty
Mode	Stop ends the animation on its first frame, as Godot's stop() does. Pause holds it on the frame it is showing. Resume plays it on from there. Toggle pauses a playing sprite and resumes a paused one. Show Frame holds the frame below. Choices: Stop, Pause, Resume, Toggle, Show Frame.	Stop
Animation Name	For Show Frame, this is the animation to show a frame of, picked from the list beside it. Empty keeps the one the sprite already has.	—
Frame	Used by Show Frame. The frame to hold, counting from 0, so the first is 0 and the fourth is 3. Press Var to read it from a variable, such as a switch's on or off as 1 or 0. Asking for a frame the animation does not have brings a warning giving the last one it has.	—

Travel To

Tells an AnimationTree's state machine to go to one of its states (start running, draw a weapon) and lets the tree work out the blend between the two animations itself.



Travel To as it appears in the Actions tab.

When to use it

With a character built on an AnimationTree, Godot's node that mixes animations together, the Machine never names an animation. It says "be running now" and the character eases into it. A door or a chest with a single timeline is simpler with Play Animation, and how fast the character runs is set with Set Tree Parameter.

> With Wait for Arrival on it waits until the tree really is showing that state, so if the tree has no line drawn from where it is to where you are sending it, the wait never ends.

See also

Set Tree Parameter 167, **Play Animation** 162, **Play Sprite Animation** 163, **Go To State** 25

Try it

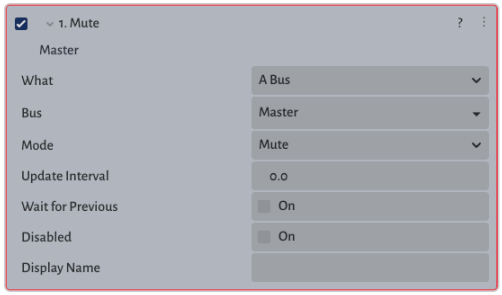
See the action work on its own with **Open Example** in the Add Action window, or open `examples/animation/travel_to/travel_to.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Tree	The AnimationTree to drive. Left empty it uses the node this Machine works on if that is one, or the first AnimationTree anywhere under it, whatever it has been named. Var: a Node.	empty
State Machine	The name of the state machine inside the tree, for a tree that holds more than one, such as a Locomotion one for the legs and an UpperBody one for the arms. In the usual setup the tree itself is the state machine, and then it is found whatever this says, so it can be left as it is or emptied.	StateMachine
Animation State	The state to travel to, spelled as it is in the tree. Left empty, nothing happens, the action finishes straight away and a warning asks for a state. A name the tree has no state for is caught before anything plays, so the action finishes without waiting, and the warning names the state asked for.	empty
Restart If Playing	On, the state is started again from its beginning even when the tree is already on it, as for an attack that can be swung twice in a row. Off, the default, leaves a state that is already playing alone, so a walk carries on smoothly. Var: a bool.	off
Wait for Arrival	Off, the default, the state finishes straight away and the blend goes on in the background. On, the state finishes only once the tree has actually arrived, so a transition on FINISHED waits until the weapon really is drawn. Var: a bool.	off

Mute

Silences a mixer bus, one sound player or the music, and lets it be heard again. A mixer bus is a named channel, set up in Godot's Audio tab, that sound travels through on its way to the speakers. Everything sent down it is turned up or down together. Muting leaves every volume where it was, and unmuting brings it all back at the level it stood at.



Mute as it appears in the Actions tab.

When to use it

The Mute box in an Options menu is this action with Mode on Toggle. For any on-or-off choice it saves remembering a level and putting it back afterwards. A slider from silent to loud is Set Volume's job.

> Choosing The Music pauses the music player itself, not the channel it plays through. The rest of the game keeps its sound, and letting the music be heard again carries on from where it stopped.

See also

Set Volume 180, **Pause Sound** 172, **Stop Music** 181, **Stop Sound** 182

Try it

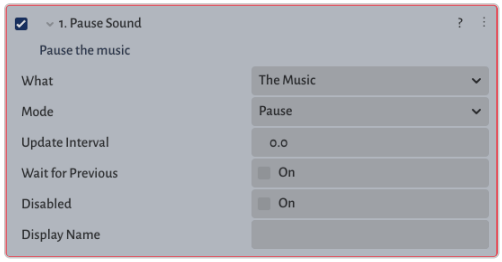
See the action work on its own with **Open Example** in the Add Action window, or open `examples/audio/mute/mute.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
What	Which of three things to silence. A Bus is a named channel every sound sent to it travels through, A Player is one sound-playing node in the scene, and The Music silences the music. It starts at A Bus, the choice for an Options menu. Choices: A Bus, A Player, The Music.	A Bus
Bus	Only when What is A Bus. The name of the channel, letter for letter as Godot's Audio tab has it. Master carries everything, so muting it silences the game. Music, Effects and Voice are made for you while the game runs if the project has none of that name, and a Mute box for the effects works in any project. A name no bus has mutes nothing and prints a warning.	Master
Mode	Mute silences it, Unmute lets it be heard, and Toggle swaps between the two so one tick box or one key can do both. It starts at Mute. Choices: Mute, Unmute, Toggle.	Mute
Player	Only when What is A Player. Point at the sound-playing node you want silenced. Left empty, or pointed at something that is not a sound player, nothing is silenced and a warning is printed. A player is silenced by holding its sound still, not by turning it down, and letting it be heard again carries on from the point it stopped.	—

Pause Sound

Holds a sound still where it is, and later lets it go on from that point, not from the beginning.



Pause Sound as it appears in the Actions tab.

When to use it

An engine noise can wait while the car is stopped and pick up again as it pulls away. The music keeps playing while the game itself is paused, so a pause menu that should go silent needs this action as well.

> A held sound keeps its place, and letting it go on picks up mid-note. Nothing here starts it again from the top, so play it again for that.

See also

Stop Sound 182, **Stop Music** 181, **Mute** 171, **Pause Game** 222

Try it

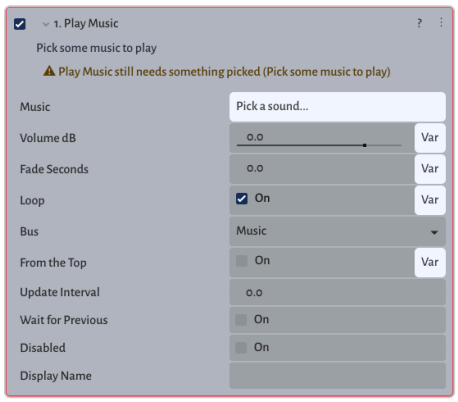
See the action work on its own with **Open Example** in the Add Action window, or open `examples/audio/pause_sound/pause_sound.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
What	The Music, A Player (one sound-playing node you point at), or Every Sound At Self, which reaches every sound player on the node this Machine works on, however deeply it is tucked inside. It starts at The Music. Choices: The Music, A Player, Every Sound At Self.	The Music
Mode	Pause stops it where it stands, Resume lets it carry on from there, and Toggle swaps between the two so one button can do both. It starts at Pause. Choices: Pause, Resume, Toggle.	Pause
Player	Only when What is A Player. Point at the sound-playing node to hold. Left empty, or pointed at a node that is not a sound player, nothing is held and a warning is printed.	—

Play Music

Plays a music track. One track plays at a time. It carries on when the scene changes and keeps playing under a pause menu while the game is paused.



Play Music as it appears in the Actions tab.

When to use it

Put the level's track in the level's first state. Asking for the track already playing leaves it playing, and a state entered over and over does not restart the music. Asking for a different one, such as the boss theme as the boss wakes, fades the old track out as the new one comes in, over Fade Seconds.

> Bus starts at Music. An Options menu's music slider or a Mute box then turns the music up and down without touching the rest of the game. Change it to Master only when nothing needs to set the music apart.

See also

Stop Music 181, **Set Volume** 180, **Play Sound** 175, **Pause Sound** 172

Try it

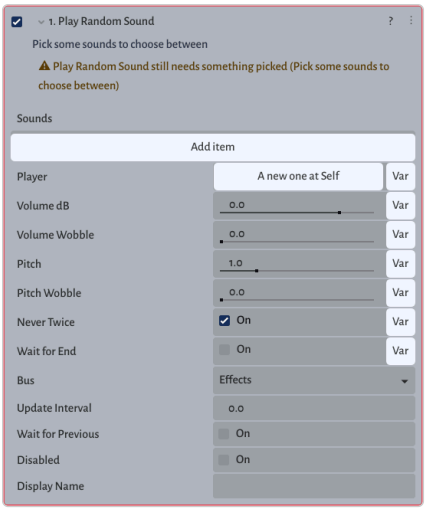
See the action work on its own with **Open Example** in the Add Action window, or open `examples/audio/play_music/play_music.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Music	The track to play, dragged in from the FileSystem dock. With nothing here nothing plays and a warning is printed.	empty
Volume dB	Volume dB in decibels, the measure Godot's Audio tab uses. 0 is the track's own level, -6 is noticeably quieter, and -40 is nearly silent. It is applied even when the track asked for is already playing, and the same action can quieten music that is already going. Var: a float.	0.0
Fade Seconds	Seconds the new track takes to come up from silence, and the same time the track before it takes to go down. It starts at 0, so the music begins at full loudness straight away. One or two seconds makes a gentle change between levels. Var: a float.	0.0
Loop	On, the default, the track begins again at its end and plays for ever, as level music should. Off lets it play through once and stop. Only the music's own copy is told to loop, never the file. The same sound played by Play Sound elsewhere does not start looping. Var: a bool.	on
Bus	The named channel the music travels through. It starts at Music, the channel an Options menu's music slider or a Mute box turns up and down, and the music answers to them with nothing set. Music, Effects and Voice are made for you while the game runs when the project has none of that name. Master is everything. Any other name has to exist in Godot's Audio tab, or the music plays through Master with a warning.	Music
From the Top	Off, the default, the track already playing is left where it is. On starts it again from its first note every time this runs, for a short sting that must be heard in full. Var: a bool.	off

Play Random Sound

Plays one of several sounds, picked at random, and can let the loudness and the pitch wander a little each time. The same sound played over and over makes a game sound cheap.



Play Random Sound as it appears in the Actions tab.

When to use it

Footsteps are the classic case. Take four recorded steps with Never Twice on and a Pitch Wobble of 0.05, and no two steps in a row sound alike. For most repeated sounds two or three recordings with a touch of wobble are enough. A sound that must finish after the thing that made it has been destroyed needs Play Sound At Position.

> Never turn Wait for End on for a sound set to loop. A looping sound has no end, so the state would wait for ever.

See also

Play Sound 175, **Play Sound At Position** 176, **Set Pitch** 178, **Stop Sound** 182

Try it

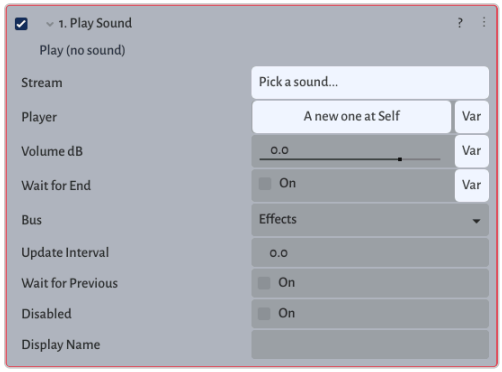
See the action work on its own with **Open Example** in the Add Action window, or open `examples/audio/play_random_sound/play_random_sound.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Sounds	The sounds to choose between, added one at a time from the FileSystem dock. Four or five suits footsteps, and three is plenty for a grunt. With none here nothing plays and a warning is printed.	empty
Player	Point at a sound-playing node in the scene to play through it. Left empty, a player is made at the node this Machine works on, of the right kind for the game. In a flat game the sound comes from the right place on the screen, and in a 3D game from the right place in the world. That player is taken away again when the sound ends. Var: a Node.	empty
Volume dB	Volume dB in decibels. 0 is the sound's own level, -6 is noticeably quieter, and it goes down to -40. Var: a float.	0.0
Volume Wobble	How far the loudness may wander either way each time, in decibels. It starts at 0, every one at the same level, and players notice the sameness. One or two decibels is enough. Var: a float.	0.0
Pitch	How fast the sound plays, and with it how high it sounds, as a multiple of its normal speed. 1 is its own pitch, 2 an octave up, 0.5 an octave down. Var: a float.	1.0
Pitch Wobble	How far the pitch may wander either way each time. It starts at 0, so every play is identical. A wobble of 0.1, a tenth either way, is plenty. More than that and a footstep starts to sound like a different creature each step. Var: a float.	0.0
Never Twice	On to start with, so the same sound is never played twice running while there is more than one to choose from. Off lets the same sound come up twice in a row, and players hear that as a fault. Var: a bool.	on
Wait for End	Off, the default, the action finishes straight away and the sound plays on by itself. On, it does not finish until the sound has ended, and a transition on FINISHED waits for it. Var: a bool.	off
Bus	Shown while Player is empty, because a player you point at keeps its own. The named channel that a player made for this action plays through. It starts at Effects, the channel an Options menu's effects slider turns up and down, and the sound follows that slider with nothing set. Music, Effects and Voice are made for you while the game runs if the project has none of that name. Master, the main one, is everything. Any other name has to exist in Godot's Audio tab or it is ignored.	Effects

Play Sound

Plays a sound at the node the Machine works on, or through a sound player you point at.



Play Sound as it appears in the Actions tab.

When to use it

The everyday sound action, for a door creak or a button click. A sound played here belongs to the node and goes with it. When the node is about to be destroyed or hidden, Play Sound At Position lets the sound finish.

> A sound set to loop, played through a player made for the action, is stopped when the state is left, but a one-shot sound plays out to its end even after its state has gone.

See also

Play Random Sound 174, **Play Sound At Position** 176, **Stop Sound** 182, **Playing** 164

Try it

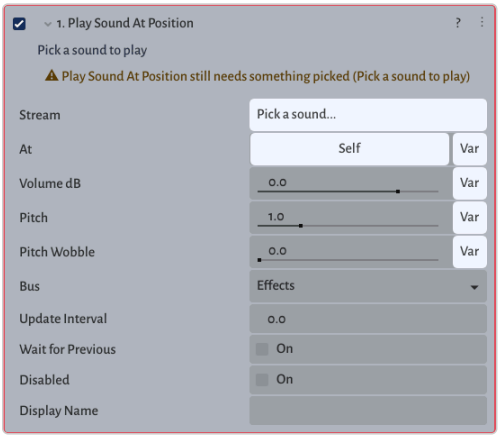
See the action work on its own with **Open Example** in the Add Action window, or open `examples/audio/play_sound/play_sound.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Stream	The sound to play, as a sound file you drag in from the FileSystem. Leave it empty and point Player at a sound player that already holds its sound, and that player plays the sound it has, at its set volume. Set a sound up once on a player, loop and all, and start it from any state. With no sound here and none on the player, no sound plays and a warning gives the reason.	empty
Player	Point at a sound-playing node in the scene to play through it, when the sound needs settings set on that node, such as how far away it can be heard. Left empty, a player is made at the node this Machine works on, of the right kind for the game (a flat player for a flat game, a 3D one for a 3D game), and taken away again when the sound ends or is stopped. Var: a Node.	empty
Volume dB	Volume dB in decibels. 0 is the sound's own level, -6 is noticeably quieter, and it goes down to -40. Set the level on the sound file itself where you can, and use this where one place needs it louder or softer. Var: a float.	0.0
Wait for End	Off to start with, so the action finishes straight away and the sound plays on by itself. On, it does not finish until the sound has ended. A state can wait out a line of speech this way, with no guessed number of seconds. Var: a bool.	off
Bus	Shown only while Player is empty, since a player you point at has its own channel. A player made for this action plays through the channel named here, Effects to start with (the one an Options menu's effects slider turns up and down). Music, Effects and Voice are made for you while the game runs if the project lacks them. Master is everything, and any other name has to exist in Godot's Audio tab or it is ignored.	Effects

Play Sound At Position

Plays a sound where something stands, on a separate player left standing in the world. The sound plays out to its end even after the thing that made it has gone.



Play Sound At Position as it appears in the Actions tab.

When to use it

An exploding barrel is freed as its bang starts. Play Sound's player belongs to the barrel and would cut the bang off halfway. Played from here, it rings out where the barrel stood. For a node that stays, Play Sound is simpler and the sound follows it.

> The sound plays on even while the game is paused. The player stays where the thing was when the sound started and does not follow anything still moving.

See also

Play Sound 175, **Play Random Sound** 174, **Queue Free** 88, **Instantiate** 84

Try it

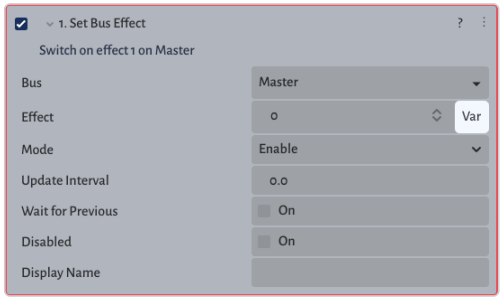
See the action work on its own with **Open Example** in the Add Action window, or open `examples/audio/play_sound_at_position/play_sound_at_position.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Stream	The sound file, dragged in from the FileSystem dock. With nothing here nothing plays and a warning is printed.	empty
At	The node whose place the sound comes from. A node is one of the pieces listed in the Scene dock. Leave it empty for the node this Machine works on. Press Var beside it and pick a Node variable to use whatever node that variable holds, such as the enemy a raycast found. Var: a Node.	Self
Volume dB	Volume dB in decibels. 0 is the sound's own level, -6 is noticeably quieter, and it goes down to -40. Var: a float.	0.0
Pitch	How fast it plays, and therefore how high it sounds, as a multiple of the sound's normal speed. 1 is its own pitch, 2 an octave up, 0.5 an octave down. Var: a float.	1.0
Pitch Wobble	How far the pitch may wander either way each time, so twenty barrels do not all explode on the same note. It starts at 0, which plays every one identically, and 0.1 is a tenth either way. Var: a float.	0.0
Bus	The named channel it plays through, Effects to start with, which an Options menu's effects slider turns up and down with nothing else set. If the project has no bus called Music, Effects or Voice, one is made for you while the game runs. Master, the main one, is everything. Any other name has to exist in Godot's Audio tab or it is ignored.	Effects

Set Bus Effect

Switches an effect on a mixer bus on or off, and everything playing through that bus is heard with it or without it.



Set Bus Effect as it appears in the Actions tab.

When to use it

When the player dives, switching on a Low Pass Filter on the Master bus muffles everything, and a second one switched off as they surface brings the sound back. A mixer bus is one of the channels in Godot's Audio tab, at the bottom of the editor. The effect itself is added and set up there first, and this action only switches it.

> A bus with no effects, or fewer than the one picked, has nothing to switch, and the warning gives how many effects the bus has.

See also

Mute 171, **Set Volume** 180, **Play Music** 173, **Play Sound** 175

Try it

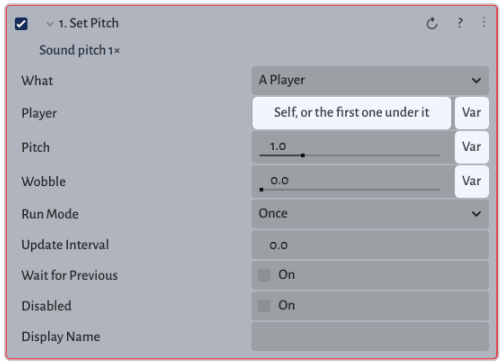
See the action work on its own with **Open Example** in the Add Action window, or open `examples/audio/set_bus_effect/set_bus_effect.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Bus	The mixer bus the effect is on, picked from the project's own buses in the Audio tab. It starts as Master, the bus everything goes through in the end.	Master
Effect	The effect to switch, picked by name from the ones that bus has, top first, such as Reverb or Low Pass Filter. It starts on the top one. The list is empty until the bus has an effect in the Audio tab. Warnings count the effects from 1, top first, so effect 2 in a warning is the second one down. Var: an int.	the top one
Mode	Enable switches the effect on, Disable switches it off, and Toggle switches it the other way from how it is. One key can then turn the echo on and off. Choices: Enable, Disable, Toggle.	Enable

Set Pitch

Sets how fast a sound plays, and so how high it sounds.



Set Pitch as it appears in the Actions tab.

When to use it

Take an engine note that rises with the car's speed. Press Var beside Pitch, pick a variable that goes from 1 at a standstill to about 2 flat out, and set Run Mode to Every Frame. The pitch follows the variable as it changes. A sound that only needs to come out slightly different each time is less work with Play Random Sound, which wobbles the pitch already.

> It needs a sound player (an AudioStreamPlayer, AudioStreamPlayer2D or AudioStreamPlayer3D) named or somewhere under the node the Machine works on. With none, it warns and sets nothing.

See also

Set Volume 180, **Play Sound** 175, **Play Random Sound** 174, **Play Music** 173

Try it

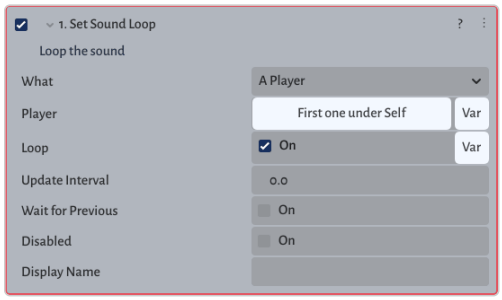
See the action work on its own with **Open Example** in the Add Action window, or open `examples/audio/set_pitch/set_pitch.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
What	A Player, one sound-playing node, or The Music. It starts at A Player. Choices: A Player, The Music.	A Player
Player	Shown when What is A Player. Point at the sound-playing node to set. Leave it empty for the node this Machine works on if it is itself a player, or else the first sound player found inside it, whatever it is called. When there is none, nothing is set and a warning is printed. Var: a Node.	empty
Pitch	How fast it plays, as a multiple of its normal speed. 1 is its own pitch, 2 an octave up, 0.5 an octave down, and it runs from 0.1 to 4. Small steps are the useful ones, and 1.05 is a noticeable step up without sounding comic. Press Var to read it from a variable. Var: a float.	1.0
Wobble	How far the pitch may wander either way each time it is set, for a sound that should not be identical twice. It starts at 0, which sets the pitch above unchanged. On Every Frame a new wobble is taken every frame. That is good for a crackle and wrong for a steady note. Var: a float.	0.0

Set Sound Loop

Makes a sound or the music loop, starting again each time it reaches its end, or stop looping so it plays to its end and stops. It works while the sound is playing, and a playing sound carries on from where it was.



Set Sound Loop as it appears in the Actions tab.

When to use it

An alarm can ring over and over, then finish its last ring once it is switched off. Turn Loop off and the ring plays to its end. A shop tune can loop the same way and play out once when the player leaves. The sound file itself is never changed. The sound player gets a separate copy, and every other player using that file carries on as before.

> Only WAV, Ogg Vorbis and MP3 sounds can be told to loop. A warning names any other kind, or a player with no sound.

See also

Play Sound 175, **Play Music** 173, **Stop Sound** 182, **Pause Sound** 172

Try it

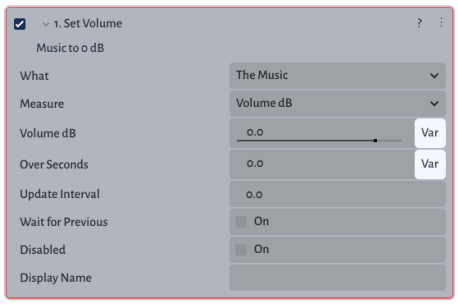
See the action work on its own with **Open Example** in the Add Action window, or open `examples/audio/set_sound_loop/set_sound_loop.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
What	A Player sets looping on one sound player, and The Music sets it on the music Play Music started. Choices: A Player, The Music.	A Player
Player	Only when What is A Player. The sound player to set. Left empty it is the first sound player under the node this Machine works on. Var: a Node.	empty
Loop	On, the default, the sound starts again from the beginning each time it reaches its end. Off, it plays to its end and stops. Var: a bool.	on

Set Volume

Sets how loud one sound player, the music or a mixer bus is, straight away or eased over a number of seconds. A mixer bus is a named channel set up in Godot's Audio tab. Everything sent down it is turned up or down together.



Set Volume as it appears in the Actions tab.

When to use it

The sliders in an Options menu set the Music and Effects buses, and everything playing through them follows. It can also duck the music under a line of dialogue. Ease it down over half a second, then back up when the line ends. For plain on or off, Mute is better, since it leaves every level where it was.

> A bus keeps whatever it is set to for the rest of the session and across scene changes, so a saved Options setting has to be applied again when the game next starts.

See also

Mute 171, **Play Music** 173, **Play Sound** 175, **Set Pitch** 178

Try it

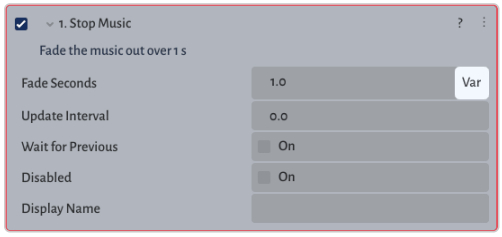
See the action work on its own with **Open Example** in the Add Action window, or open `examples/audio/set_volume/set_volume.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
What	A Player (one sound-playing node), The Music, or A Bus (a whole channel). It starts at The Music. Choices: A Player, The Music, A Bus.	The Music
Measure	How you give the loudness. Volume dB is the measure Godot's Audio tab uses, where 0 is full and -6 noticeably quieter. Volume Linear is the way a slider gives it, 0 silent and 1 full, and suits an Options menu. Choices: Volume dB, Volume Linear.	Volume dB
Volume dB	Shown while Measure is Volume dB. The loudness, from -60 to 12. 0 is full, -6 is noticeably quieter, -40 is nearly silent. Var: a float.	0.0
Over Seconds	Seconds to ease over. It starts at 0, which sets the loudness straight away, for a slider. Half a second to a second ducks the music under a line of speech and brings it back afterwards. The ease keeps running while the game is paused and still works from a pause menu. Var: a float.	0.0
Player	Only when What is A Player. Point at the sound-playing node to set. Leave it empty for the node this Machine works on if it is itself a player, or else the first sound player found inside it. When there is none, nothing is set and a warning is printed.	—
Bus	Only when What is A Bus. The name of the channel, spelt as in Godot's Audio tab. Master is everything. Music, Effects and Voice are made for you while the game runs if the project has none of that name, and an Options menu's sliders work in any project. A name no bus has sets nothing, with a warning.	—
Volume Linear	Only when Measure is Volume Linear. A share of full loudness, from 0, silent, to 1, full. Press Var beside it and pick the variable your volume slider writes its value into, and the slider becomes the volume. 0 is as quiet as the mixer goes, not truly off, but Mute gives real silence.	—

Stop Music

Stops the music, straight away or faded out over a number of seconds, whatever started it and wherever.



Stop Music as it appears in the Actions tab.

When to use it

Silence before a cutscene or on a game over screen starts here. A change to a different track needs no stop first, because Play Music fades the old one out as the new one comes in.

See also

Play Music 173, **Pause Sound** 172, **Set Volume** 180, **Mute** 171

Try it

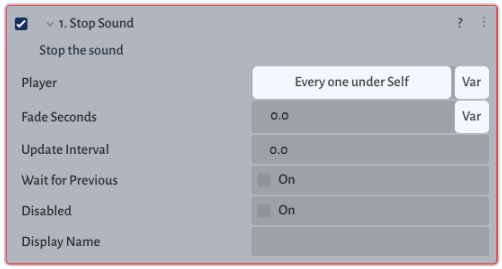
See the action work on its own with **Open Example** in the Add Action window, or open `examples/audio/stop_music/stop_music.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Fade Seconds	Seconds the music takes to fade down to silence before it stops. It starts at 1, a gentle ending. 0 cuts it dead, for a jump scare or the player's death. The fade keeps running while the game is paused and still finishes from a pause menu. Var: a float.	1.0

Stop Sound

Stops a sound, straight away or faded out over a number of seconds.



Stop Sound as it appears in the Actions tab.

When to use it

A sound set to loop never ends by itself. A car's engine hum plays until this stops it, perhaps faded over half a second as the engine dies. The music has a separate action, Stop Music.

> Left empty it silences every sound player under the node. Name the player when only one of several should stop.

See also
Play Sound 175, **Pause Sound** 172, **Stop Music** 181, **Play Random Sound** 174

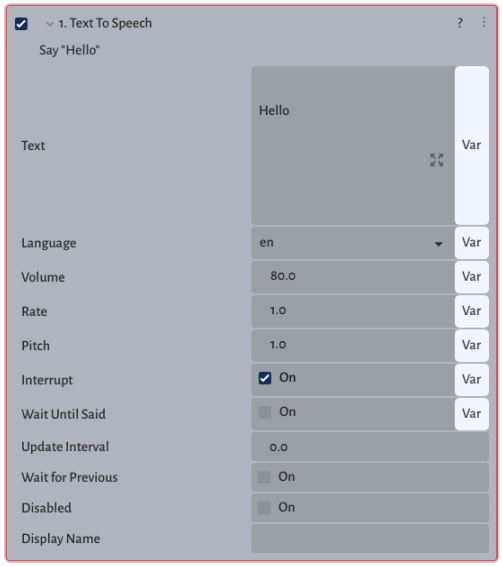
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/audio/stop_sound/stop_sound.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Player	The sound player to stop. Leave it empty to stop every sound player anywhere under the node this Machine works on, however deeply it is tucked away, and the node itself when it is a player. Point it at a node that is not a sound player and there is nothing to stop, and a warning is printed. Var: a Node.	empty
Fade Seconds	Seconds to fade down before stopping. It starts at 0, which stops it straight away. The volume is put back to what it was once the fade has finished, and the next sound through that player is heard at its usual level. The fade keeps running while the game is paused. Var: a float.	0.0

Text To Speech

Says words out loud in the computer's voice, with no recording needed. The words can hold variables, such as a score.



Text To Speech as it appears in the Actions tab.

When to use it

A rally co-driver can call each corner as the car nears it, at a Rate of 1.5. It speaks and lets the state carry on straight away, unless Wait Until Said is on.

> It uses the voice the computer speaks with, and sounds different on a Mac and on Windows. A computer with no voice stays silent, and a warning gives the reason.

See also

Play Sound 175, **Set Text** 213, **Format String** 54, **Show Choices** 217

Try it

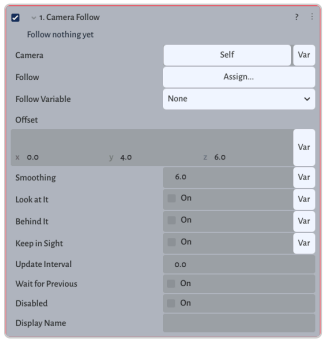
See the action work on its own with **Open Example** in the Add Action window, or open `examples/audio/text_to_speech/text_to_speech.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Text	The words to say. Write a variable's name in curly brackets, such as {score}, anywhere to have its value said. Press Var to read all the words from a text variable. A list of calls can be read out one after another that way. Var: a String.	Hello
Language	The language of the voice, as two letters, such as en for English, fr for French and de for German. The first voice the computer has for that language is used, or its own voice when it has none. Var: a String.	en
Volume	How loud, from 0 for silent to 100 for as loud as the voice goes. The voice does not go through the game's sound buses, and a music or effects slider does not change it, so read this from a variable to give it a separate slider. Var: a float.	80.0
Rate	How fast the words come. 1, the default, is normal speech. 1.5 is brisk, as a co-driver talks, and 0.7 is slow and clear. Var: a float.	1.0
Pitch	How high the voice is. 1, the default, is its own. Less is deeper and more is higher. Var: a float.	1.0
Interrupt	On, the default, cuts off whatever is still being said and the new words are heard straight away. That is right for a co-driver. Off, the new words wait their turn after the old ones. Var: a bool.	on
Wait Until Said	Off, the default, lets the state carry on while the words are spoken. On, the state waits until they have all been said, as a narrator does before the story goes on. Var: a bool.	off

Camera Follow

Keeps a camera on something that moves, such as the player, a car or a thrown ball. It eases after its target with no sudden snap, and the offset decides where the target sits on the screen. It works in 2D and 3D alike. Point it at the camera and it works out which kind it is moving.



When to use it

A side-scroller where the camera trails the player is the plain case. For a chase camera behind a car, tick Behind It as well and the camera stays on the car's tail through the corners. It never holds the state up, so put it beside whatever else that state is doing.

> With Follow and Follow Variable both empty the camera stays where it is and warns about it. A Follow Variable that is only empty for now gets no warning, because a template often fills it a moment later.

See also

Switch Camera 196, **Orbit Camera** 192, **Camera Limits** 185, **Camera Zoom** 187

Try it

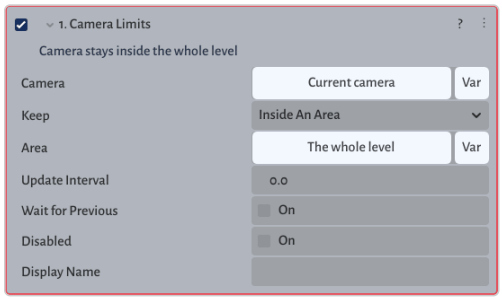
See the action work on its own with **Open Example** in the Add Action window, or open `examples/camera/camera_follow/camera_follow.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Camera	The camera that moves. Leave it empty when the machine sits on the camera itself. Otherwise pick your Camera2D or Camera3D. It moves whatever node you pick. Pick the sprite beside the camera and the sprite slides along, with a warning naming it as a node that is not a camera. Var: a Node.	Self
Follow	The node the camera keeps up with, usually the player, picked here in the scene. With this and Follow Variable both empty the camera stays where it is, and a warning says it has nothing to follow.	empty
Follow Variable	Empty by default, and the camera follows whatever you picked in Follow. Name a variable here and the camera follows whichever node that variable holds at the time. The game can then move the camera on to whichever player is in the lead, or to the enemy that just took the ball. It wins over Follow, so if the variable is empty or is not holding a node, the camera follows nothing and stays put.	empty
Offset	Where the camera sits relative to the thing it follows. In a 2D game type two numbers, across and down the screen, so -80 down puts the view 80 pixels above the target's head. In a 3D game type three. The default of 0, 4, 6 stands the camera four metres up and six metres back. Var: a Vector2 or a Vector3, or a Node for where it stands.	0, 4, 6 in 2D: 0, 0
Smoothing	How much of the gap the camera closes each second. The default of 6 is a soft, unhurried follow. 12 or more keeps the camera tight on a fast player. 0 pins it on the offset with no easing at all, and that is fine for a puzzle game but stiff in an action one. Var: a float.	6.0
Look at It	Off by default, so the camera keeps whatever way it is facing. Tick it in a 3D game and the camera also turns to face what it follows. A chase camera needs this to swing round as the car corners. A 2D camera ignores it. Var: a bool.	off
Behind It	Off by default. The offset is then fixed in the world, and six metres back means six metres south whichever way the target faces. Top-down and side views use this. Tick it in a 3D game and the offset turns with the way the target faces. A camera set behind a car stays on its tail through every corner, and with Smoothing on it hangs back a little as the car turns, as a chase camera does. Only the turn on the ground counts, so a car that leans or dips does not tip the view. A 2D camera has no use for it. Var: a bool.	off
Keep in Sight	Off by default. Tick it and a hill, a wall or a roof never comes between the camera and what it follows. When one does, the camera rises over it, and where rising is not enough, as under a roof, it comes in front of it instead. A side-on hero walking past hills is the case it was made for, and the Side-On Platform Hero has it ticked. 3D only. Var: a bool.	off

Camera Limits

Keeps a 2D camera inside an area, so the view never slides past the edge of the level and shows the empty grey beyond it. It is for 2D games only. It works on a Camera2D and has nothing to do in a 3D game.



Camera Limits as it appears in the Actions tab.

When to use it

Run once at the start of a level with nothing picked, it pens the camera into the whole level. Run it again each time the player walks into a new room and the view stops at that room's walls. No Limits takes the limits off for a cutscene that pans away, or for an endless runner where the level has no edge. It sets the limits and finishes straight away, costing the state no time.

> The limits are set once as the state is entered, so to change rooms the state has to be entered again with the new room's Area.

See also

Camera Follow 184, **Camera Zoom** 187, **Switch Camera** 196

Try it

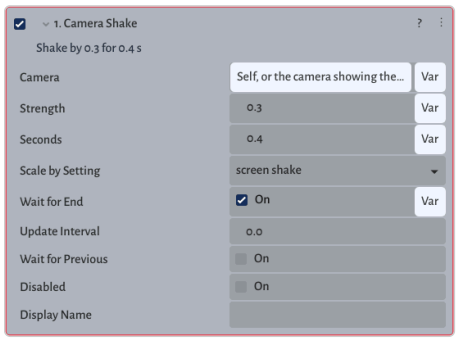
See the action work on its own with **Open Example** in the Add Action window, or open `examples/camera/camera_limits/camera_limits.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Camera	Leave it empty and it limits whichever Camera2D is showing the game at that moment. Fill it in only when you need to set the limits on a camera that is not being looked through yet. Var: a Node.	empty
Keep	Inside An Area, the default, keeps the view inside the whole level, or inside the area the node picked under Area covers. No Limits opens the view right up again, for an endless level or a camera that has to fly out past the edge of the map. Choices: Inside An Area, No Limits.	Inside An Area
Area	Shown when Keep is Inside An Area. Left empty, as it starts, it is the whole level. That means every StaticBody2D in the scene, measured by its collision shapes, such as the ready-made Ground and Walls, and every TileMapLayer with tiles on it, put together. Pick a node for less. Then it measures the StaticBody2Ds and tile layers under it, the tiles actually drawn on a TileMapLayer, or the rectangle of any Control, such as a ReferenceRect dragged out to mark a room. Other kinds of body are not measured. With nothing there to measure the camera is left with no limits. A warning appears while the game runs, and the problems button points it out before then. Var: a Node.	empty

Camera Shake

Rattles the camera for a moment after an explosion, a heavy landing or a hit taken. The shake fades by itself and the view goes back where it was at the end, never left crooked. It works with a Camera2D and a Camera3D alike.



Camera Shake as it appears in the Actions tab.

When to use it

Put it in the state that fires the gun, sets off the bomb or takes the damage. It shifts the camera's offset, not its position, and a camera that is following the player keeps following while it shakes.

> Two shakes on one camera together, say an explosion during a heavy landing, wobble around the same resting place and settle back to it when the last one ends.

See also

Vibrate 157, **Screen Effect** 202, **Camera Follow** 184, **Recoil** 193

Try it

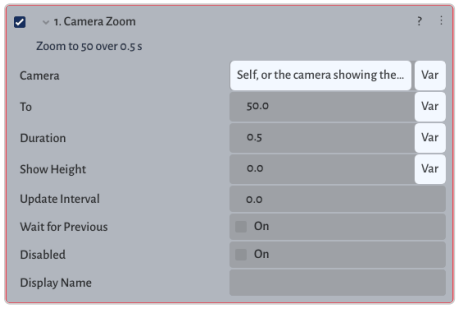
See the action work on its own with **Open Example** in the Add Action window, or open `examples/camera/camera_shake/camera_shake.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Camera	The camera to rattle. Left empty, it shakes the camera showing the game, for a machine that sits on the player or the bomb. A machine that sits on a camera shakes that one. Point it at something that is not a Camera2D or a Camera3D and the camera showing the game is shaken in its place, with a warning naming the node. With no camera in the game at all nothing shakes, and a warning says there is no camera. Var: a Node.	empty
Strength	How far the view moves at the worst of it, in pixels in a 2D game or metres in a 3D one. It starts at 0.3, a small knock in 3D, or 9.6 pixels when put in a 2D game. 8 to 20 pixels is a solid hit in 2D, and 0.1 is plenty in 3D. The shake fades from this down to 0 over its length. Var: a float.	0.3 in 2D: 9.6 px
Seconds	How long the shake lasts, fading out over that time. The default of 0.4 suits a hit or a landing, and 1 or more suits a collapsing building. Zero means no shake at all. The action finishes and the view never moves. Var: a float.	0.4
Scale by Setting	The name of a global number from 0 to 1 that every shake is multiplied by. It is screen shake unless you change it. Put a Screen Shake slider on your Options page that stores into a global variable of that name and a player who dislikes shaking can turn it down or off across the whole game. A game that has no such variable shakes at full strength, and clearing the box ignores any setting.	screen shake
Wait for End	On by default, so the state waits for the shake to finish before moving on. That is handy when the shake is the point of the state. Turn it off when the shake is one of several things the state is doing. The action then finishes straight away and the shake carries on fading in the background. Var: a bool.	on

Camera Zoom

Zooms a camera in or out over a few seconds, closer for a dialogue or wider for a fight. A 2D camera's zoom is a multiplier, where bigger is closer. A 3D camera's is its field of view in degrees, where smaller is closer. It works with both, and the Show Height setting is for 2D pixel art only.



Camera Zoom as it appears in the Actions tab.

When to use it

As a boss walks into the room, a 2D camera can pull back to a zoom of 0.6 over a second and fit the arena on screen. In a 2D pixel-art game, set Show Height in place of To. The camera then zooms until a fixed number of the game's pixels fill the window's height, and the art is the same size on every screen. Zooming is not the same as moving closer. To bring the view nearer to something, give Camera Follow a smaller offset.

> With Show Height set the action keeps fitting the view to the window for as long as the state is active, so leave that state when you want it to stop.

See also

Camera Follow 184, **Camera Limits** 185, **Switch Camera** 196, **Screen Settings** 225

Try it

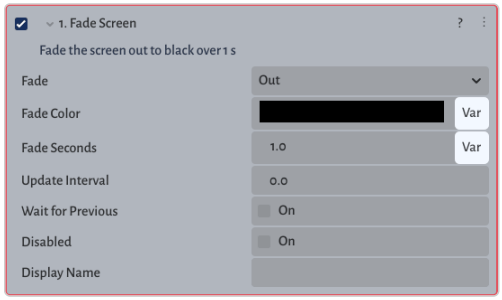
See the action work on its own with **Open Example** in the Add Action window, or open `examples/camera/camera_zoom/camera_zoom.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Camera	The camera to zoom. Left empty, it zooms the camera showing the game, for a machine that sits on the player. A machine that sits on a camera zooms that one. Pointed at something that is not a Camera2D or a Camera3D, it zooms the camera showing the game and warns about the node it was given. With no camera in the game at all it finishes straight away, warning that there is no camera to zoom. Var: a Node.	empty
To	What to end at. That is a zoom multiplier for a 2D camera, where 2 is twice as close, or a field of view in degrees for a 3D one, where 50 is normal and 30 is a long lens. It is ignored, and hidden, while Show Height is above zero. A field of view is kept between 1 and 179 degrees however large a number you type. Var: a float.	50.0 in 2D: 1.5
Duration	How many seconds the zoom takes, easing gently in and out. The default of 0.5 is quick but visible, and 2 suits a slow reveal. Zero snaps to the new zoom in a single frame. It is ignored, and hidden, while Show Height is above zero. Var: a float.	0.5
Show Height	For a 2D camera only. The camera zooms so this many pixels of the game fill the height of the window, whatever size that window is, and keeps doing so while the state runs as the window is resized. Zero, the default, turns it off and the action zooms to To. 180 or 240 are usual choices for pixel art. Var: a float.	0.0
Whole Steps	Only when Show Height is above zero. On by default, so the camera zooms only by whole steps (2, 3, 4 and so on) and every pixel of the art is drawn at the same size, with a little more of the game on screen than you asked for. Turn it off to fill the window exactly, at the cost of some pixels being drawn slightly larger than others.	—

Fade Screen

Fades the screen to a colour, or back from it, over a number of seconds, and finishes once the fade is done.



Fade Screen as it appears in the Actions tab.

When to use it

When the player dies, fade to black over 1.5 seconds. The state waits for the fade, so the next state can bring up the game-over menu on a dark screen. A fade that only goes with a change of scene needs no separate action, because Change Scene has its own Fade Seconds.

> The colour stays over the screen, even through a change of scene, until a Fade Screen set to In takes it away. A level that fades out needs a Fade Screen In somewhere after it.

See also

Change Scene 220, **Screen Effect** 202, **Pause Game** 222, **Wait** 69

Try it

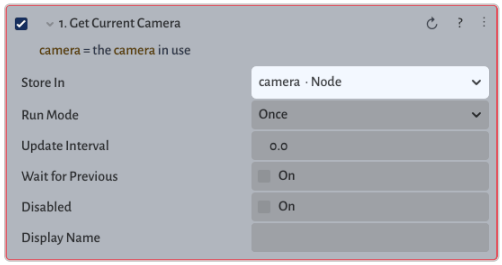
See the action work on its own with **Open Example** in the Add Action window, or open `examples/camera/fade_screen/fade_screen.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Fade	Out covers the screen with the colour. In takes the colour away so the game shows again. On a screen that is not covered already it starts fully covered, and a level opens out of black that way. Choices: Out, In.	Out
Fade Color	The colour the screen fades to, black to start with. With In it is the colour the screen fades back from when it is not covered already. A screen that is already covered keeps the colour it has. Var: a Color.	0, 0, 0, 1
Fade Seconds	How many seconds the fade takes, 1 to start with. 0 covers or uncovers the screen in one step and finishes straight away. Var: a float.	1.0

Get Current Camera

Puts the camera the game is being seen through right now into a Node variable, so other actions can work on whichever camera is in use.



Get Current Camera as it appears in the Actions tab.

When to use it

Before a cutscene switches cameras, store the one in use. Afterwards Switch Camera can go back to it. It also lets Camera Shake or Camera Zoom work on whatever camera the player is looking through. It works with 2D and 3D cameras.

See also
Switch Camera 196, **Camera Shake** 186, **Camera Zoom** 187, **Camera Follow** 184

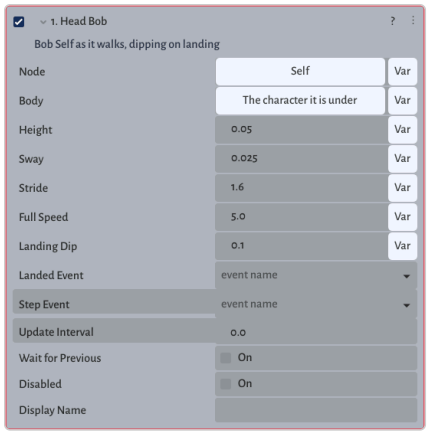
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/camera/get_current_camera/get_current_camera.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Store In	The Node variable that gets the camera in use. It starts as camera. With no camera showing the game it is left empty and a warning says no camera is showing the game.	camera

Head Bob

Bobs the view up and down as a first-person character walks, and dips it when they land, so moving feels like walking, not gliding. The faster the character goes the harder it bobs. Standing still or in mid-air it settles. It is for 3D games only.



Head Bob as it appears in the Actions tab.

When to use it

Put it in the state your player walks in, and a second one with Height at 0.1 in the sprinting state for a heavier sprint. Step Event, sent as each foot comes down, gives footstep sounds in time with the walk, and Landed Event can bring a thud or a puff of dust.

> Bob the node between the body and the camera rather than the camera itself, or the bob and the player's looking up and down will fight each other.

See also

Mouse Look 191, **Weapon Sway** 197, **Move Character** 134, **Play Random Sound** 174

Try it

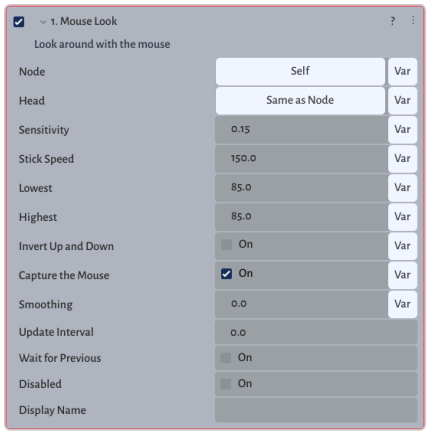
See the action work on its own with **Open Example** in the Add Action window, or open `examples/camera/head_bob/head_bob.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node that bobs. Make it a plain Node3D sitting between the character's body and its camera (body, then head, then camera) and point this at the head. Leave it empty for the thing the machine works on. Var: a Node.	Self
Body	The character whose walking drives the bob. It is a CharacterBody3D, the kind of body that knows its own speed and whether it is standing on the floor. Leave it empty and the action looks up through the bobbing node's parents for the first character body above it. The usual body, head, camera arrangement needs nothing more. Var: a Node.	empty
Height	How far the view dips with each step, in metres. The default of 0.05 is a gentle walk, and 0.1 feels heavy and tiring. Above about 0.15 people start to feel sick. Var: a float.	0.05
Sway	How far the view rocks from side to side over a pair of steps, in metres. The default of 0.025, about half the up-and-down, reads as a natural walk. Set it to 0 for a bob straight up and down. Var: a float.	0.025
Stride	How far one step carries the character at Full Speed, in metres. Longer strides mean the view bobs more slowly. The default of 1.6 is a person's walk. Shorten it for a small character. Going faster than Full Speed lengthens the stride as well as quickening it, and a sprint does not patter. Var: a float.	1.6
Full Speed	The speed that gives a full bob, in metres a second. Set it to whatever your walk speed is, 5 by default. Below it the bob fades away smoothly as the character slows. Above it the bob grows, up to half as much again and no further. Var: a float.	5.0
Landing Dip	How far the view is pushed down on a hard landing, in metres, springing back afterwards. The default of 0.1 is a solid landing. A short drop, no faster than a step off a kerb, does not dip at all, and the dip grows with the speed of the fall. Set it to 0 to leave landings alone. Var: a float.	0.1
Landed Event	Empty by default, and no event is sent. Type an event name here and it is sent as the character lands from a real fall. Lead it to a state that plays a thud or puffs out some dust.	empty
Step Event	Empty by default. Type an event name and it is sent twice per bob, as each foot comes down, and more often the faster the character walks. Lead it to a state with Play Random Sound for footsteps that keep time by themselves.	empty

Mouse Look

Lets the player look around with the mouse, the way a first-person game does. Moving the mouse left and right turns the body, and moving it up and down tilts the head. The right stick on a gamepad turns the view too. It is for 3D games only.



Mouse Look as it appears in the Actions tab.

When to use it

This is the looking half of a first-person player. The usual arrangement is a CharacterBody3D with a Camera3D under it. Put this action on the body and set Head to the camera. Put Move Character beside it with Space set to Local and the player walks wherever they are looking.

> While the mouse is captured the player cannot click anything on screen, so leave this state before showing a menu.

See also

Orbit Camera 192, **Move Character** 134, **Head Bob** 190, **Set Mouse Mode** 154

Try it

See the action work on its own with **Open Example** in the Add Action window, or open `examples/camera/mouse_look/mouse_look.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node that turns left and right, usually the character's body, and the character then faces where the player looks. Leave it empty for the thing the machine works on, when the machine sits on the body. Var: a Node.	Self
Head	The node that tilts up and down, usually the camera under the body. Leave it empty and the body itself tilts. That suits a camera flying about on its own, but a walking character's body would tip over. Var: a Node.	empty
Sensitivity	How far the view turns for each pixel the mouse moves, in degrees. The default of 0.15 suits most people. 0.05 is slow and steady and 0.4 is very fast. Read it from a variable and an options page can set it. Var: a float.	0.15
Stick Speed	How fast the right stick on a gamepad turns the view when pushed all the way, in degrees a second. The default of 150 is a comfortable speed. Set it to 0 to leave the stick alone, for a game meant for a mouse only. Var: a float.	150.0
Lowest	How far the view can tilt down, in degrees below straight ahead. The default of 85 lets the player look almost at their own feet without the view turning over. It is capped at 90 however large a number you type. Var: a float.	85.0
Highest	How far the view can tilt up, in degrees above straight ahead. The default of 85 lets the player look almost straight up. It is capped at 90 however large a number you type. Var: a float.	85.0
Invert Up and Down	Off by default, so pushing the mouse forward looks up. Tick it and pushing the mouse forward looks down. Some players much prefer that, so offer it on the options page. Var: a bool.	off
Capture the Mouse	On by default, so the pointer is hidden and held inside the window while the state runs, so it never wanders onto another screen mid-fight. Esc gives it back and a click in the game takes it again, and it is given back when the state is left. Turn it off for a view you look around with the pointer still free. Var: a bool.	on
Smoothing	How many seconds the view takes to catch up with the mouse. The default of 0 turns with no delay, as in most shooters. Between 0.03 and 0.08 takes the jitter out of a cheap mouse or a shaky stick. Much above that the view starts to feel like it is on ice. Var: a float.	0.0

Orbit Camera

A third-person camera. It circles something at a distance, turned by the mouse or the right stick, zoomed in and out with the wheel, and pulled in front of any wall that would block the view. It is for 3D games only.



When to use it

Adventures and 3D platformers watch the character from behind with it. Put it on the Camera3D, set Around to the player, and give the player Move Character with Space set to Camera. Pushing forward then walks away from the camera whichever way the camera is looking.

> The camera begins from wherever it already sits in the scene, so place it behind and above the player for the view to start the right way round.

See also

Mouse Look 191, **Camera Follow** 184, **Move Character** 134, **Switch Camera** 196

Try it

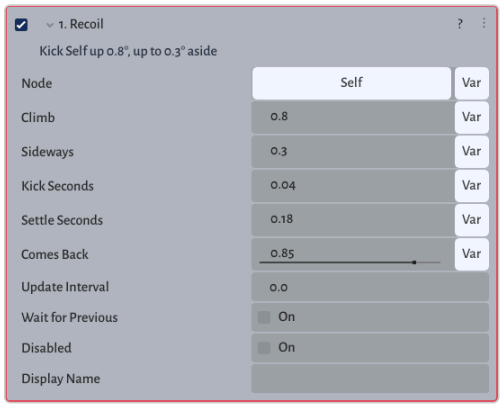
See the action work on its own with **Open Example** in the Add Action window, or open `examples/camera/orbit_camera/orbit_camera.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Camera	The camera that does the circling. Leave it empty only when the machine sits on the camera. On the player, pick the camera, or nothing moves and a warning says so. Var: a Node.	Self
Around	What the camera circles, such as the player, a car or a building. Leave it empty and it circles the thing the machine works on, for a machine on the player with the camera as another node. If that turns out to be the camera itself there is nothing to circle and the action does nothing. Var: a Node.	Self
Height	How far above the circled node's own origin the camera aims, in metres. The default of 1.5 looks at a standing person's head rather than at their feet. Raise it for a tall character, lower it for a car. Var: a float.	1.5
Over the Shoulder	How far to one side the whole view slides, in metres. At 0 the camera sits straight behind and the character stands in the middle with its back to you, hiding what is ahead. Around half a metre puts it to one side and opens the view up. Positive is over the right shoulder, negative the left, and it stays on that shoulder as the view turns. Var: a float.	0.0
Distance	How far away the camera starts, in metres. The default of 5 is a comfortable third-person view. 2 is over the shoulder and 10 shows a lot of the level. It is kept between Closest and Farthest. Var: a float.	5.0
Closest	How close the wheel can bring the camera, in metres. Var: a float.	2.0
Farthest	How far out the wheel can take it, in metres. Var: a float.	12.0
Zoom Step	How far one notch of the mouse wheel moves the camera, in metres. The default of 0.75 takes a few notches to cross the whole range. Set it to 0 to leave the wheel alone, and the distance then never changes. Var: a float.	0.75
Sensitivity	How far the view turns for each pixel the mouse moves, in degrees. The default of 0.25 is a comfortable swing. Lower it for a slower, more cinematic camera. Var: a float.	0.25
Stick Speed	How fast the right stick on a gamepad turns the view when pushed all the way, in degrees a second, 150 by default. The stick turns the view whenever it is pushed, even when Only While Held is set and nothing is being held. Set this to 0 if the stick should do nothing at all. Var: a float.	150.0
Lowest	How far below the thing it circles the camera may swing, in degrees. The default of 20 lets the player look down on their character from slightly under it. 0 keeps the camera level or above. It stops just short of straight down whatever you type. Var: a float.	20.0
Highest	How far above the thing it circles the camera may swing, in degrees. The default of 70 is close to a top-down view. It stops just short of straight up whatever you type. Var: a float.	70.0
Invert Up and Down	Off by default, so pushing the mouse forward, or the right stick up, swings the camera down behind the character and the view tilts up, as in most games. Tick it and pushing forward swings the camera up and over the character. Some players prefer that. Var: a bool.	off
Only While Held	Empty by default, so the mouse turns the view whenever it moves. Set it to something like Mouse: Right Button and the view turns only while that is held, leaving the pointer free for things to click. The right stick turns the view either way, so set Stick Speed to 0 to stop that too.	empty
Capture the Mouse	On by default, so the pointer is hidden and held in the window while the view is being turned. Esc gives it back and a click in the game takes it again, with nothing to build. It also comes back when the state is left. Ignored while Only While Held is filled in. Var: a bool.	on
Keep Out of Walls	On by default. If a wall, a floor or a pillar comes between the camera and what it circles, the camera is pulled in front of it. What it circles is ignored, and the character's own body never pushes the camera in. Var: a bool.	on
Smoothing	How much of the gap the camera closes each second as the thing it circles moves. The default of 0 sticks to it, crisp and with no lag. Try 8 to 15 for a softer camera that drifts a little behind a fast player. Var: a float.	0.0

Recoil

Kicks the view up, and a little to one side, the way a gun climbs in the hands as it fires, then lets most of it come back down. It is for 3D games only, and it never holds the state up.



Recoil as it appears in the Actions tab.

When to use it

Put it in the state that fires a single shot, beside whatever spawns the bullet. Held fire then climbs by itself, because each new kick arrives before the last has settled, and letting go lets the view sink back. It tilts the same node Mouse Look tilts, usually the camera, on top of wherever the player is looking, and aiming while firing keeps working.

> Whatever does not come back becomes where the player is now aiming, so a low Comes Back on a fast weapon walks the aim off the target in a couple of seconds.

See also

Camera Shake 186, **Weapon Sway** 197, **Mouse Look** 191, **Instantiate** 84

Try it

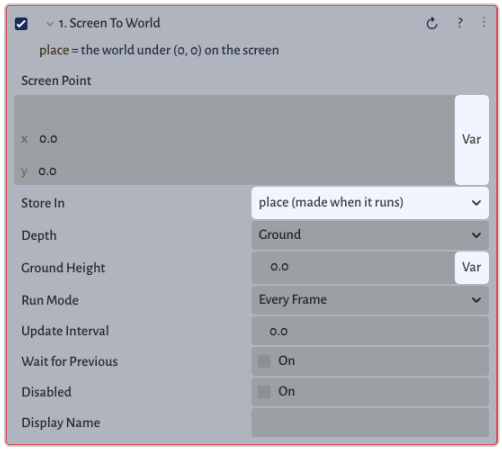
See the action work on its own with **Open Example** in the Add Action window, or open `examples/camera/recoil/recoil.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node that tilts, which should be the same one Mouse Look tilts (usually the camera). Leave it empty for the thing the machine works on. Var: a Node.	Self
Climb	How far each shot kicks the view up, in degrees. The default of 0.8 is a light weapon you can hold on target, and 2 or more is a heavy one that walks off the target quickly. Var: a float.	0.8
Sideways	How far each shot may kick the view to one side, in degrees, picked at random to the left or the right each time. The default of 0.3 makes a burst wander a little off a straight climb. Set it to 0 for a perfectly vertical climb. Var: a float.	0.3
Kick Seconds	How long the kick itself takes. The default of 0.04 is a sharp snap. Much above 0.1 it starts to feel like a push, not a kick. Var: a float.	0.04
Settle Seconds	How long the view takes to sink back after the kick. The default of 0.18 settles between shots of a fairly slow weapon. Lengthen it and fast fire will climb higher, because each shot lands before the last has come down. Var: a float.	0.18
Comes Back	How much of each kick sinks back again, from 0 to 1. At the default of 0.85 most of it returns and a little is left behind, and long bursts creep upwards. At 1 the view always returns to where it was aimed, and at 0 none of it comes back and the aim climbs shot by shot. Var: a float.	0.85

Screen To World

Turns any point on the screen into a place in the game's world and puts it into a variable, ready for Move To, Look At or Instantiate. For the mouse pointer alone, Mouse Position is simpler.



Screen To World as it appears in the Actions tab.

When to use it

A crosshair sits in the middle of the screen. This finds the place in the level it points at, where a shot can land or a marker can go. It turns the spot a finger touched into a place in the same way. In a 2D game the place follows the camera wherever it has moved or zoomed to.

> In a 3D game, a camera looking away from the ground has no place on it to give. The variable is left as it was, and a warning is printed.

See also

Mouse Position 150, **World To Screen** 198, **Mouse Pick** 149, **Move To** 97

Try it

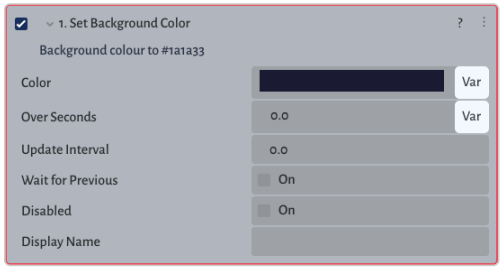
See the action work on its own with **Open Example** in the Add Action window, or open `examples/camera/screen_to_world/screen_to_world.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Screen Point	The point on the screen, in pixels from the top left of the game's view. 0, 0 to start with is the top left corner. Read it from a variable to use a finger's position, or the place Mouse Position stores in its Store On Screen In. Var: a Vector2 or a Vector3, or a Node for where it stands.	0, 0
Store In	The variable the place in the world goes into, called place to start with. It holds two numbers in a 2D game and three in a 3D one. A variable of another kind, such as a number, is left alone and a warning is printed.	place
Depth	Only in a 3D game, where a point on the screen stands for a line going away from the camera. Z Depth takes the place a set distance in front of the camera. Ground, the default, takes where the line meets flat ground. Intersect Ray takes the first thing on Layers the line meets, or the flat ground where it meets nothing. Choices: Z Depth, Ground, Intersect Ray.	Ground
Ground Height	Only when Depth is Ground or Intersect Ray. How high the flat ground is, in metres. 0 to start with suits a level whose floor is at 0. Var: a float.	0.0
Z Depth	Only when Depth is Z Depth. How far in front of the camera the place is, in metres (10 to start with).	—
Layers	Only when Depth is Intersect Ray. Which collision layers count as something the line meets. World, the default, is walls, ground and anything you add yourself.	—

Set Background Color

Changes the colour behind everything, where nothing else is drawn, at once or eased over a number of seconds.



Set Background Color as it appears in the Actions tab.

When to use it

A sky can turn orange over 20 seconds at sunset, or each level can open on a different colour. In a 2D game it changes the colour Godot clears the screen to, and that stays changed into the next scene. In a 3D game it changes the Custom Color of the WorldEnvironment, a node that sets how the 3D world looks, when its Background Mode is Custom Color.

> A 3D world with a sky behind it has no colour there to change, and the warning names the Background Mode to pick.

See also
Set Property 91, **Switch Camera** 196, **Camera Shake** 186, **Screen Settings** 225

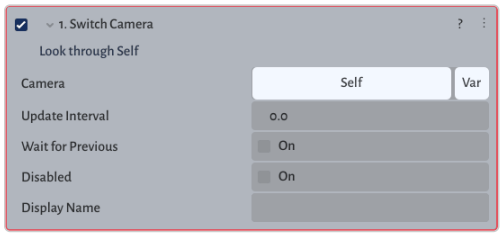
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/camera/set_background_color/set_background_color.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Color	The colour to show behind everything. It starts as a dark blue. Var: a Color.	0.1, 0.1, 0.2, 1
Over Seconds	How many seconds to ease from the colour there now to this one, and the state waits until it is done. Zero, the default, changes it straight away. Var: a float.	0.0

Switch Camera

Looks through a different camera, such as a cutscene angle, a security camera or the view over a player's shoulder. It works with a Camera2D and a Camera3D alike.



Switch Camera as it appears in the Actions tab.

When to use it

Cut to a fixed camera as the player walks into the boss room, and back again with another Switch Camera pointed at the player's camera when the fight is won.

> Pointed at anything that is not a Camera2D or a Camera3D, the view stays as it was and a warning says the node is not a camera.

See also

Camera Follow 184, **Camera Zoom** 187, **Camera Limits** 185, **Orbit Camera** 192

Try it

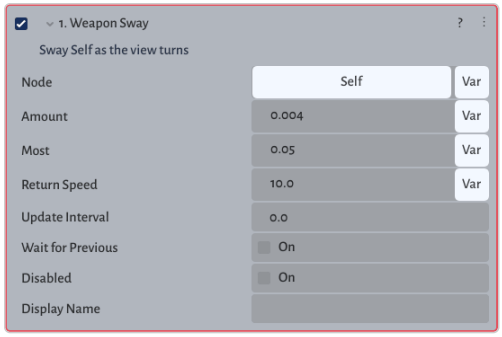
See the action work on its own with **Open Example** in the Add Action window, or open `examples/camera/switch_camera/switch_camera.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Camera	The camera to look through. Leave it empty for the thing the machine works on, when the machine sits on the camera itself. A Camera2D with its Enabled box unticked is ticked for you, and a camera you left switched off in the editor still works. Var: a Node.	Self

Weapon Sway

Lets something held in front of the camera lag behind as the view turns, and drift back when it stops, so it feels held in the hands, not glued to the screen. It follows the turning itself, and the mouse, a gamepad stick and a recoil kick all sway it the same way. It is for 3D games only.



Weapon Sway as it appears in the Actions tab.

When to use it

Put it in the state the player looks around in, once for each thing held in front of the camera, such as a gun, a torch or a pair of hands. Without it a first-person gun sits dead still in the corner of the picture. It moves the thing on top of wherever it already is, and an aiming or reloading movement easing the same node still works while it sways.

> The swaying thing must be a child of the node that turns, usually the camera. A gun parented straight to the level has no turning to follow and will never sway.

See also

Recoil 193, **Head Bob** 190, **Mouse Look** 191, **Tween Property** 106

Try it

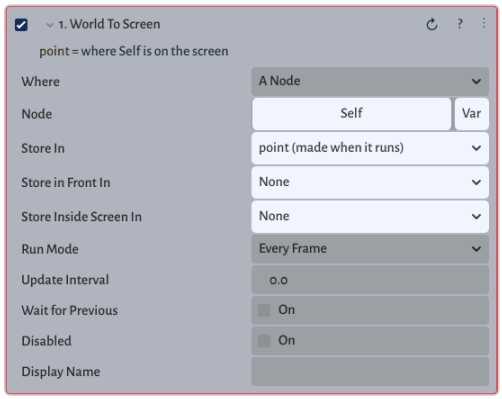
See the action work on its own with **Open Example** in the Add Action window, or open `examples/camera/weapon_sway/weapon_sway.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The thing that sways, such as a gun sitting under the camera. Leave it empty for the thing the machine works on. It must sit under a 3D node that turns (the camera), because the sway is worked out from that parent's turning. Var: a Node.	Self
Amount	How far it swings for a quick turn, in metres for each degree the view turns in one frame. The default of 0.004 is a subtle lag and 0.01 is a heavy weapon that swings noticeably. 0 turns the sway off. Var: a float.	0.004
Most	The furthest it is ever allowed to get from where it belongs, in metres. The default of 0.05 keeps a fast spin from flinging the gun out of the picture. Lower it if the weapon ever leaves the edge of the screen. Var: a float.	0.05
Return Speed	How quickly it drifts back to where it belongs once the view stops turning, from about 1 (a slow, floaty drift) to 30 (back almost at once). The default of 10 settles in a fraction of a second. Var: a float.	10.0

World To Screen

Finds where on the screen something in the world shows, and puts that point into a variable, in pixels from the top left. It can also say whether the thing is in front of the camera and whether it is on the screen at all.



World To Screen as it appears in the Actions tab.

When to use it

A "!" over a character's head needs to know where the head is on the screen. Store the point, then move a Control in your screen's display there with Set Property. A marker that stays over something and points the way when it goes off the screen is already built as Off Screen Indicator.

> In a 3D game it needs a Camera3D showing the game. With none, it warns and leaves the variables alone.

See also

Off Screen Indicator 209, **Screen To World** 194, **Set Property** 91, **Mouse Position** 150

Try it

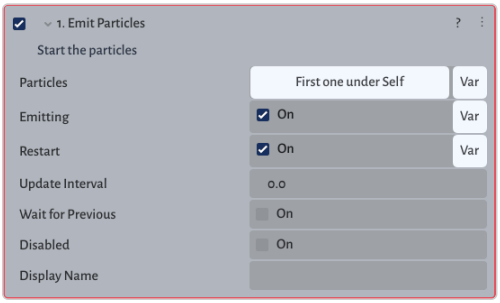
See the action work on its own with **Open Example** in the Add Action window, or open `examples/camera/world_to_screen/world_to_screen.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Where	What to find on the screen. A Node, the default, finds a node, and A Place finds a place typed in or read from a variable. Choices: A Node, A Place.	A Node
Node	Only when Where is A Node. The node to find. Left empty it is the node this Machine works on. A 2D node or a Control gives the point it is drawn at, even on a CanvasLayer, a layer drawn over the game. Var: a Node.	Self
Store In	The variable the point on the screen goes into, called point to start with. It holds two numbers, across and down in pixels from the top left.	point
Store in Front In	The variable told whether the thing is in front of the camera, true or false. Something behind the camera still gets a point, a mirror image that means nothing, so check this before showing a marker there. In a 2D game it is always true. Empty stores nothing.	empty
Store Inside Screen In	The variable told whether the thing is on the screen now (in front of the camera and inside the edges of the view). Empty stores nothing.	empty
Place	Only when Where is A Place. The place in the world, as two numbers in a 2D game and three in a 3D one.	—

Emit Particles

Starts or stops a particle effect, such as dust, sparks, smoke or a burst on a hit. It works with any of the four kinds of particle node, in 2D and in 3D.



Emit Particles as it appears in the Actions tab.

When to use it

It suits any effect that should begin or end with a state, such as dust while a character runs and none while they stand. For a one-shot burst like an explosion, leave Restart ticked and the burst plays from the beginning every time the state is entered. It finishes straight away and holds nothing up.

> It only flicks the effect on or off and finishes straight away. If the next state must not begin until the smoke has cleared, put a Wait beside it.

See also

Screen Effect 202, **Set Shader Parameter** 204, **Play Sound** 175, **Set Visible** 216

Try it

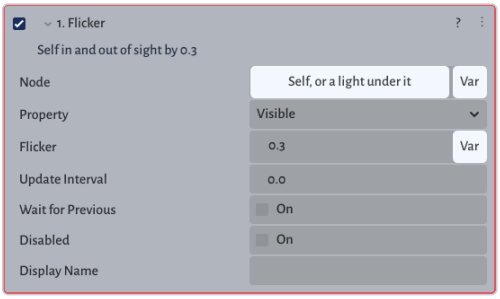
See the action work on its own with **Open Example** in the Add Action window, or open `examples/effects/emit_particles/emit_particles.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Particles	The particle node to start or stop. Leave it empty and the action uses the thing the machine works on when that is itself a particle node, or else the first particle node found anywhere beneath it, such as an effect hung under a character. Var: a Node.	empty
Emitting	On, the default, starts the particles, and off stops them. To switch an effect back off, put a second copy of this action, with it turned off, in the state you leave to. Var: a bool.	on
Restart	On by default, so the effect begins again from its first particle and does not carry on from where it was. It only matters when Emitting is on. Leave it on for a burst you want to see in full each time. Turn it off for a steady stream like smoke, which would otherwise jerk back to the start. Var: a bool.	on

Flicker

Makes something flicker for as long as its state is active. It can drop out of sight and back at random moments, or a light's brightness can waver like a flame. When the state is left it is put back as it was.



Flicker as it appears in the Actions tab.

When to use it

Give a broken lamp Visible at 0.3 and it cuts out now and then. Give a campfire Energy and its light wavers. Everything is put back when the state ends, so a light flickering in a Power Failing state is steady again once the machine moves on.

> It never finishes, so its state does not send FINISHED. Leave that state on another event.

See also

Set Light 274, **Set Visible** 216, **Set Modulate** 273, **Screen Effect** 202

Try it

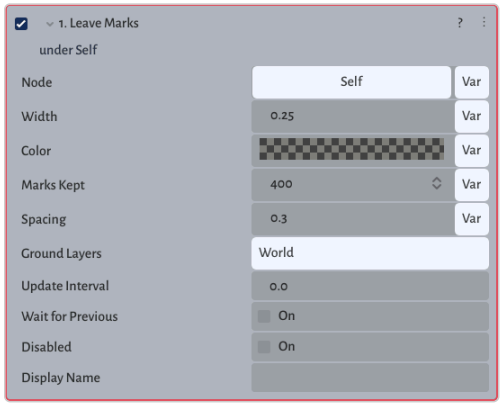
See the action work on its own with **Open Example** in the Add Action window, or open `examples/effects/flicker/flicker.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The thing to flicker, such as a sprite, a mesh, a sign or a light. Left empty, it is the thing the machine works on. With Energy, it is the first light under it when it is not a light itself. Var: a Node.	empty
Property	Visible switches it out of sight and back at random moments, and works on anything drawn and on a light. Energy wavers a light's brightness the way a flame does, never quite going out. Choices: Visible, Energy.	Visible
Flicker	How much it flickers, where 0 is steady. With Visible, 0.3 cuts out now and then like a failing bulb and 1 is out nearly as much as it is on. With Energy, 0.3 is a torch and 1 a fire in a draught. Var: a float.	0.3

Leave Marks

Lays marks on the ground under something for as long as its state is on. Think of tyre marks behind a sliding car, grooves in snow or a scorched line behind a rocket sled. The marks laid stay on the ground after the state is left.



Leave Marks as it appears in the Actions tab.

When to use it

Put it in the state that means the marks should be made, such as a car's Sliding state, and point it at the part that touches the ground (a back wheel, a foot or a runner). A car's two back wheels need one each and leave two lines. Each mark is laid on whatever is under the part, and the marks follow the ground over bumps and hills.

> A new line starts each time the state does. A line breaks, leaving a gap, when the part leaves the ground or is moved somewhere else in one go.

See also

Emit Particles 199, **Instantiate** 84, **Raycast** 116, **Drive** 108

Try it

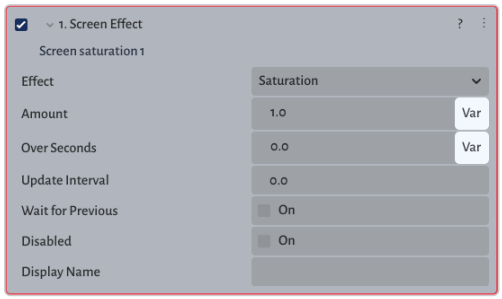
See the action work on its own with **Open Example** in the Add Action window, or open `examples/effects/leave_marks/leave_marks.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	Where the marks come from. This is the wheel, the foot or the runner that touches the ground. Leave it empty for the thing the machine works on. It must be a 3D node. On a 2D one nothing is laid and a warning says the node is not 3D. Var: a Node.	Self
Width	How wide each mark is, in metres. A car tyre is about 0.25, the default, and a tank track might be half a metre. Var: a float.	0.25
Color	The colour of the marks, and how see-through they are. Try dark and half clear for rubber on tarmac, or a pale grey for grooves in snow. Press Var to read it from a variable, and each level can set a different colour. Var: a Color.	0.08, 0.07, 0.06, 0.55
Marks Kept	How many marks stay on the ground at once for each part leaving them. Once there are this many, the oldest are taken up to make room for new ones, and a long race never fills the ground. 400, the default, is about 120 metres of line. Var: an int.	400
Spacing	How far the part moves between one mark and the next, in metres. Shorter follows a tight slide more closely but uses up Marks Kept sooner. The default, 0.3, suits a car. Var: a float.	0.3
Ground Layers	Which layers count as ground to lay marks on. World, where the ground and anything you add yourself stand, is ticked to begin with. If your road is on a separate layer, tick that one too, or the marks are laid on the ground under the road and hidden by it.	World

Screen Effect

Changes how the whole screen looks. It can make it darker and greyer when the player is low on health, washed out in a flashback, or red all over. Brightness, Contrast and Saturation change the Environment the game is shown with (the camera's own when it has one, else the scene's WorldEnvironment). Tint changes the CanvasModulate a 2D game uses to colour everything at once. Set it in a single frame or ease it over a few seconds.



Screen Effect as it appears in the Actions tab.

When to use it

It changes the mood of the whole picture. While the player is poisoned, a sickly green Tint can wash over a 2D scene. As health drops, Saturation can drain towards 0. Fog, glow, ambient light and the sky's colour belong to Set Environment, and one sprite or one model on its own is a job for Set Modulate or Set Shader Parameter.

> It needs a WorldEnvironment with an Environment in the scene or a camera with its own Environment, or a CanvasModulate for Tint. With neither, nothing changes and a warning is printed.

See also

Set Environment 203, **Set Modulate** 273, **Set Shader Parameter** 204, **Camera Shake** 186

Try it

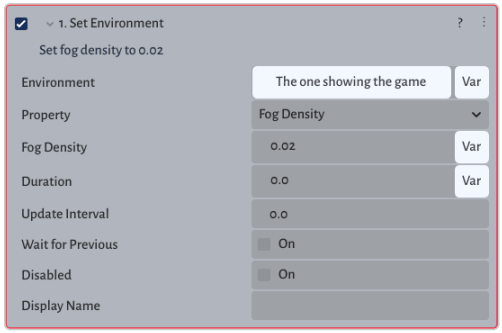
See the action work on its own with **Open Example** in the Add Action window, or open `examples/effects/screen_effect/screen_effect.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Effect	Which part of the look to change. Brightness, Contrast and Saturation are the plain picture controls, and Saturation at 0 gives black and white. Tint is the single colour a 2D scene is washed with. Choices: Brightness, Contrast, Saturation, Tint.	Saturation
Amount	How much, for every effect but Tint, where it is hidden. Brightness, Contrast and Saturation are all 1 as the scene was made, so 0.5 is half and 1.5 is half as much again. Var: a float.	1.0
Over Seconds	Eases the change in over this many seconds, and the state waits for the ease to finish. Zero, the default, changes it in a single frame, for a hit. Two or three seconds suits a mood settling in, such as colour draining away as health runs out. Var: a float.	0.0
Tint	Only when Effect is Tint. The colour the whole 2D scene is washed with. White, the default, leaves it as made, a deep blue makes night and a dark red makes a warning. Darker colours also darken the scene, since the colour is laid over everything.	—

Set Environment

Changes how the world looks while the game runs, such as the fog's thickness and colour, the ambient light, the glow, or the background. It changes the Environment (the settings for how the scene is lit and drawn) of the scene's WorldEnvironment, straight away or eased over a time.



Set Environment as it appears in the Actions tab.

When to use it

Fog rolling in on the moor is Fog Density eased from 0 to 0.05 over four seconds. Each action changes one thing, so a state that thickens the fog and turns it blue holds two. Brightness, contrast and saturation are not here, because they belong to Screen Effect.

> A 2D game has no background colour here to change. The colour behind a 2D game is Default Clear Color, under Rendering > Environment in the Project Settings, and the warning printed points there.

See also
Screen Effect 202, **Set Light** 274, **Tween Property** 106, **Set Property** 91

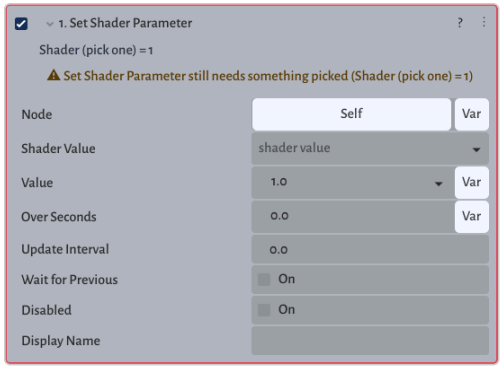
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/effects/set_environment/set_environment.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Environment	The WorldEnvironment, or a Camera3D that has an Environment set. Left empty, it is the one the game is shown with. That is the camera's own when the camera showing the game has one, otherwise the first WorldEnvironment in the scene. Var: a Node.	empty
Property	What to change. Fog Density is how thick the fog is and switches the fog on, and Fog Light Color is its colour. Ambient Light Color and Ambient Light Energy are the colour and brightness of the light that reaches everywhere, even into the shade. Glow Intensity is how strongly bright things glow and switches the glow on. Background Color is the colour behind everything, and Background Energy Multiplier how bright the sky or that colour is. Choices: Fog Density, Fog Light Color, Ambient Light Color, Ambient Light Energy, Glow Intensity, Background Color, Background Energy Multiplier.	Fog Density
Fog Density	Only with Fog Density. How thick the fog is. 0 is none, 0.01 a light haze, 0.05 thick and 0.1 hard to see through. Fog that was off is eased in from none. Var: a float.	0.02
Duration	How many seconds the change takes. At 0 it changes straight away. Var: a float.	0.0
Color	Only with Fog Light Color, Ambient Light Color or Background Color. The colour it becomes. An ambient colour set here replaces the sky as the source of that light, and a background colour is shown in place of the sky.	—
Ambient Light Energy	Only with Ambient Light Energy. How bright the light that reaches everywhere is. 1 is as made, 0 leaves the shade black and 2 is twice as bright.	—
Glow Intensity	Only with Glow Intensity. How strongly bright things glow, where 0 is none, 0.8 is Godot's usual and 2 glows a lot. Glow that was off is eased in from none.	—
Background Energy Multiplier	Only with Background Energy Multiplier. How bright the sky or the colour behind everything is. 1 is as made, 0 black and 2 twice as bright.	—
Curve	Only when Duration is above zero. The shape of the change, Godot's Tween transition (a tween is a change eased over time). Linear keeps one speed all the way, and the others speed up, slow down or go past and back.	—
Easing	Only when Duration is above zero. Where the change is slow. Ease Out slows at the end, Ease In at the start, Ease In Out at both ends and Ease Out In in the middle.	—
Wait for End	Only when Duration is above zero. On, the action finishes only when the change is done, so FINISHED comes after it. Off, it finishes straight away and the change plays on.	—

Set Shader Parameter

Changes one of the values a shader reads, picked from a list of the values that shader has. Set it in a single frame or ease it there over a few seconds. It works on anything in 2D or 3D whose material is a ShaderMaterial (a material driven by a shader), and only this one thing changes, never everything else drawn with the same material.



Set Shader Parameter as it appears in the Actions tab.

When to use it

Effects that live in the artwork are driven from here, such as a sprite flashing white when hit or an enemy dissolving over a second as it dies. The change belongs to one node. For an ordinary property on the node, such as its colour or its position, use Tween Property or Set Property.

> With no Shader Value picked, no shader material on the node, a name the shader does not have, or a value of the wrong kind, nothing is set and the warning says which of these went wrong.

See also

Screen Effect 202, **Tween Property** 106, **Set Property** 91, **Emit Particles** 199

Try it

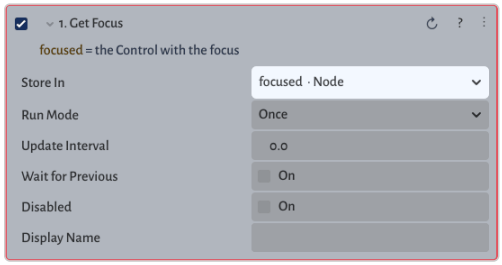
See the action work on its own with **Open Example** in the Add Action window, or open `examples/effects/set_shader_parameter/set_shader_parameter.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node whose shader you are changing. Leave it empty for the thing the machine works on. Its material must be a ShaderMaterial. On a 3D model, either the material override on the whole model or the material on its first surface will do. Var: a Node.	Self
Shader Value	Which value to change, picked from a list of the ones that shader has, so there is no name to remember. Left empty, nothing is set and a warning says no value was picked.	empty
Value	What to set it to. It should be the sort of thing the shader expects, such as a number for a number, a colour for a colour and a Vector for a vector. A whole number and a number are turned into one another, a colour fills a three- or four-number value, and a Vector fills one of another size. Anything else is not set, and a warning names the value and the kind it wanted. Var: any kind.	1.0
Over Seconds	Eases the value across over this many seconds, with the state waiting for the ease to finish. Zero, the default, sets it in a single frame, for a hit flash. Half a second to two seconds suits a dissolve. The ease starts from whatever the shader holds now, or, if it has never been set, from the shader's own starting value. Var: a float.	0.0

Get Focus

Puts the button, slider or box that has the focus now, the one the keyboard or a pad's D-pad is on, into a Node variable. With nothing focused the variable is left empty.



Get Focus as it appears in the Actions tab.

When to use it

In a shop played with a pad, it tells a description box which item is highlighted. Before a pop-up opens, store the focus here, and Grab Focus can put it back when the pop-up closes.

See also

Grab Focus 208, **Show Choices** 217, **Set Text** 213, **Set Visible** 216

Try it

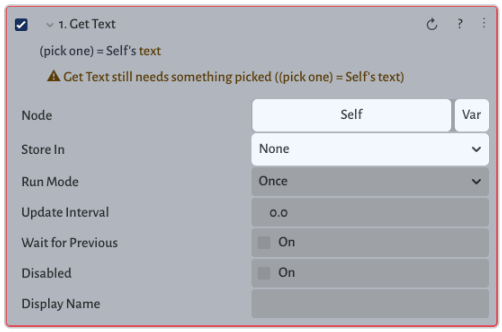
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ui/get_focus/get_focus.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Store In	The Node variable that gets the focused control. It starts as focused. Set Run Mode to Every Frame to keep it up to date as the player moves round the menu.	focused

Get Text

Reads the words in a label, a text box, a button or a rich text box into a text variable. Set Text writes words onto the screen, and this reads them back.



Get Text as it appears in the Actions tab.

When to use it

On a new-game page it keeps the name the player typed. On Once it reads the words a single time, as the OK button is pressed. With Run Mode on Every Frame the variable keeps up while the player is still typing, and a line elsewhere can repeat the name letter by letter. A number from a slider or a spin box comes from Get Value.

> A rich text box gives its words as the player sees them, without the tags that make them bold or coloured.

See also

Set Text 213, **Get Value** 207, **String Compare** 57, **Event On Signal** 76

Try it

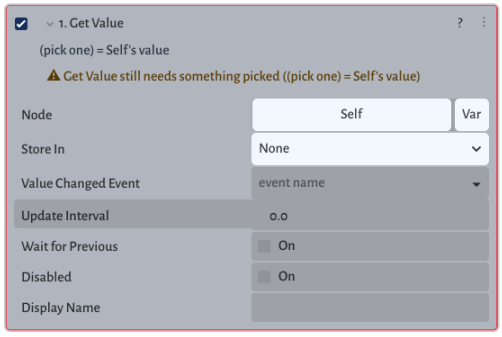
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ui/get_text/get_text.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The label, text box (a LineEdit for one line, a TextEdit for several), button or rich text box to read. Left empty it reads the node this Machine works on, which must then be one of those. Given anything without words, it writes nothing and warns about the node. Var: a Node.	Self
Store In	The variable the words go into, which should hold text (a String). A variable of another kind, such as a number, is left as it was, and a warning explains why. Left empty, the words go nowhere, and that is warned about too.	empty

Get Value

Reads the value of a slider, a spin box, a scroll bar or a progress bar into a variable, and keeps it up to date while its state runs. Each time the value changes it can also send an event carrying the new value.



Get Value as it appears in the Actions tab.

When to use it

Point it at the volume slider on an Options page and the variable follows the slider as the player drags it. Value Changed Event lets another state act as soon as it moves, for example setting the volume with Set Volume. It also reads how many players were picked on a spin box, or how full a bar is.

> The action finishes as soon as its state starts and listens in the background, so it never holds up FINISHED. The listening stops when the state is left.

See also

Set Value 215, **Set Volume** 180, **Event On Signal** 76, **Get Text** 206

Try it

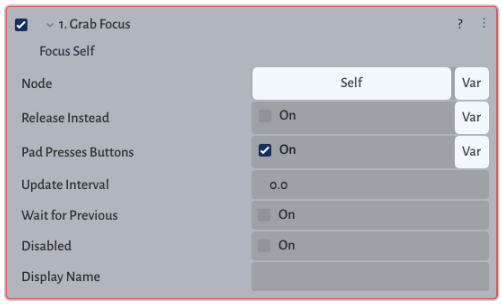
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ui/get_value/get_value.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The slider, spin box, scroll bar or progress bar to read. Left empty it reads the node this Machine works on, which must then be one of those. Given anything else, it writes nothing and a warning says the node has no value to read. Var: a Node.	Self
Store In	The variable the value goes into. It is a float, or an int for a slider that moves in whole steps, which then stays a whole number. It is read as the state starts and written again every time the value changes while the state runs. Leave it empty when the value is only wanted with the event.	empty
Value Changed Event	The event sent each time the value changes while this state runs, whether the player moved it or another action did. It carries the new value, and Store In is written before it is sent, so the state it leads to finds the new number already there. Left empty, no event is sent.	empty

Grab Focus

Puts the keyboard on a button, so it is highlighted and Enter presses it. A menu played with a gamepad or the arrow keys needs something focused as soon as it opens.



Grab Focus as it appears in the Actions tab.

When to use it

In the state that shows a menu, point it at the first button, and the player can move through the menu with the arrow keys or the D-pad without touching the mouse. On a text box, the player can start typing straight away. Buttons made by Show Choices need no Grab Focus, because that action focuses the first answer already.

> A button inside a scrolling list is scrolled into sight as well, even on a page the same state has only just shown.

See also

Show Choices 217, **Set Visible** 216, **Event On Signal** 76, **Rebind Input** 152

Try it

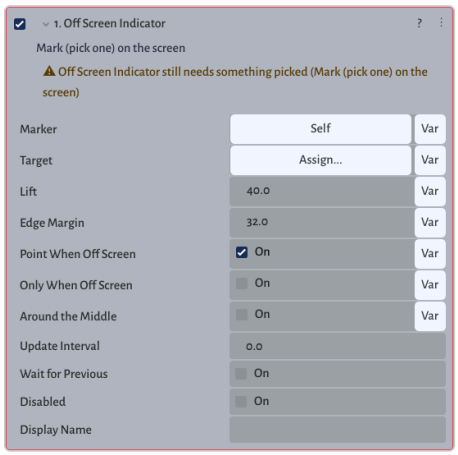
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ui/grab_focus/grab_focus.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The button, slider or text box to focus. Leave it empty, as it comes, for the thing this machine works on. Only a Control, something drawn on the screen, can take the focus. Anything else is left alone and a warning names it. A Control whose Focus Mode takes no focus gets a warning too, saying to set it to All. Var: a Node.	Self
Release Instead	Off to begin with, putting the focus on. Ticked, it takes the focus away and nothing is highlighted, for when a menu closes and the game takes the controls back. Var: a bool.	off
Pad Presses Buttons	On to begin with, which lets the pad's bottom button (A on an Xbox pad, Cross on a PlayStation one) press the highlighted button, as Enter does. Godot's own settings leave that button out, so without this a pad can move round the menu but not choose anything. It only ever adds it while the game runs, and only when the project has given no pad button to accepting already. Var: a bool.	on

Off Screen Indicator

Keeps a marker from your HUD, any picture or panel you have made, on the screen over something in the world. When that thing goes off the screen the marker waits at the nearest edge, turned to point at it, and the player knows which way to look.



Off Screen Indicator as it appears in the Actions tab.

When to use it

An arrow over the next objective, a name riding over a character's head, or a "!" over someone worth talking to. With Around the Middle ticked it becomes a damage arrow. Read the thing it marks from a variable, and one marker can follow whatever is important right now.

See also

Set Visible 216, **Set Texture** 214, **Get Event Data** 20, **Camera Follow** 184

Try it

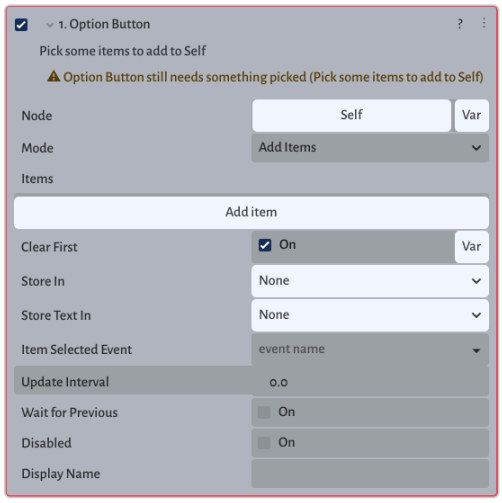
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ui/off_screen_indicator/off_screen_indicator.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Marker	The marker itself, something drawn on the screen such as a picture of an arrow in your HUD. Leave it empty, as it comes, for the thing this machine works on. Point it at a sprite or a body out in the world and no marker appears, and a warning names the node and says the marker must be a Control. Var: a Node.	Self
Target	What the marker points at or sits over. If it is gone, or not in the game, the marker hides itself, so a marker over a defeated enemy leaves the screen with it. With nothing picked it stays hidden, and a warning asks for a Target. Var: a Node.	empty
Lift	How far above the thing the marker sits, in pixels on the screen, 40 to begin with. That is enough to put a name over a character's head and not across their chest. Ignored when Around the Middle is ticked. Var: a float.	40.0
Edge Margin	How far in from the edge of the screen the marker waits while the thing is off screen, in pixels, 32 to begin with. Too small and the arrow is half cut off. Ignored when Around the Middle is ticked. Var: a float.	32.0
Point When Off Screen	On to begin with, so the marker turns to point at the thing while it waits at the edge, and straightens again once the thing is back in view. Draw the marker's picture pointing right, because it is turned from there. Ignored when Around the Middle is ticked. Var: a bool.	on
Only When Off Screen	Off to begin with, and the marker is always there. Ticked, it hides whenever the thing can be seen, for an arrow that only shows the way to something out of sight. Ignored when Around the Middle is ticked. Var: a bool.	off
Around the Middle	Off to begin with. Ticked, the marker leaves the thing and sits on a ring round the middle of the screen, turned the way the thing lies, so straight ahead is up and behind is down. This makes the damage arrow that tells a player where they were shot from. Var: a bool.	off
Radius	Only when Around the Middle is ticked. How far out from the middle of the screen the ring sits, in pixels, 180 to begin with. Keep it well clear of the crosshair in the centre.	—

Option Button

Fills, clears, reads or sets a drop-down list, such as the window sizes on an Options page, the languages a game speaks, or Easy, Normal and Hard. While its state runs it can also send an event each time the player picks an item.



Option Button as it appears in the Actions tab.

When to use it

As the Options page opens, Add Items fills the list and Select shows the choice the player saved last time. Get Selected, or Item Selected Event, tells you what the player picked. Keep the item's number with Save Game, and use its words on a label or with String Compare. Show Choices is a different thing. It puts a row of buttons on the screen and waits for one to be pressed.

> The action finishes as soon as its state starts and listens in the background while the state runs, and the listening stops when the state is left.

See also
Show Choices 217, **Save Game** 235, **Set Language** 226, **Screen Settings** 225

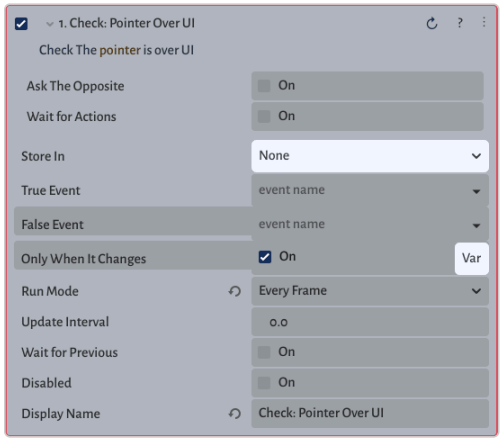
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ui/option_button/option_button.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The drop-down list, an OptionButton. Left empty it uses the node this Machine works on, which must then be one. Anything else gets a warning, and nothing changes. Var: a Node.	Self
Mode	Add Items puts the items below into the list, and Clear empties it. Get Selected only stores which item is picked, and Select picks the item below. Whichever it is, the variables below are written afterwards. Choices: Add Items, Clear, Get Selected, Select.	Add Items
Items	Only when Mode is Add Items. The words of each item, top to bottom, such as Easy, Normal and Hard. Left empty, nothing is added and a warning says there were no items.	empty
Clear First	Only when Mode is Add Items. Ticked, as it starts, the list is emptied before the items are added, so a page opened a second time does not list everything twice. Untick it to add items under the ones already there. Var: a bool.	on
Store In	The variable the number of the picked item goes into, counting from 0 at the top, or -1 when nothing is picked. It is written after the change, and again each time the player picks an item while the state runs. Left empty, it stores nothing.	empty
Store Text In	The variable the words of the picked item go into, such as "1920 x 1080", written at the same times as Store In. Left empty, it stores nothing.	empty
Item Selected Event	The event sent each time the player picks an item while this state runs. It carries the number of the item, and the two variables above are written before it is sent. Left empty, no event is sent.	empty
Selected	Only when Mode is Select. The item to pick, counting from 0 at the top, and -1 picks nothing. Press Var to pick the one a variable holds, such as a choice kept with Save Game. A number the list does not have gets a warning giving the last number there is, and the pick is left as it was. Select does not send Item Selected Event, and showing a saved choice is not taken for the player choosing.	—

Pointer Over UI CHECK

True while the mouse pointer, or a finger, is over something on the screen that takes clicks, such as a button, a slider, a text box, a panel or an open inventory. With it, a click on a button does only what the button does.



Pointer Over UI as it appears in the Actions tab.

When to use it

Ask it before a click walks the hero. A yes means the click landed on a button or a panel, and the hero should stay put. A Check with it on Every Frame can also light up a hint while the pointer rests on a panel. Words that let clicks through, and an empty container stretched over the whole screen to hold the HUD, do not count, so clicks there really do reach the game.

> It answers from where the pointer last moved to. A finger counts because Godot moves the pointer to each touch, unless Emulate Mouse From Touch has been turned off in the Project Settings.

See also
Click To Move 126, **Mouse Position** 150, **Mouse Pick** 149, **Input** 144

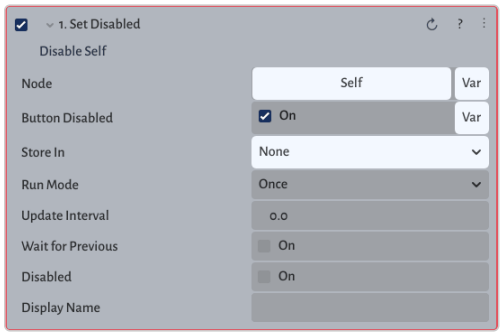
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ui/pointer_over_ui/pointer_over_ui.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Ask The Opposite	Turns the answer round into "the pointer is over the game, not the screen", for letting a click through to the world. It also turns round the true-or-false answer the Check stores. Swapping the True Event and False Event over would not do that.	off
Wait for Actions	Holds the question back until every other action in the state has finished. Left off, the default, it is asked as soon as the state runs.	off
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Set Disabled

Greys a button out so it cannot be pressed, or makes it pressable again. It works on any button, whether a plain button, a tick box, a switch or a drop-down list.



Set Disabled as it appears in the Actions tab.

When to use it

A Continue button with no saved game to carry on from should be greyed out, and so should a Buy button while the player has too little gold. Press Var beside Button Disabled to follow a true-or-false variable and set Run Mode to Every Frame. The button then greys out and comes back by itself as the variable changes.

> On Every Frame, following a variable, it only looks again when that variable changes, and does no work while the variable stays the same.

See also

Set Visible 216, **Toggle Button** 218, **Grab Focus** 208, **Save Game** 235

Try it

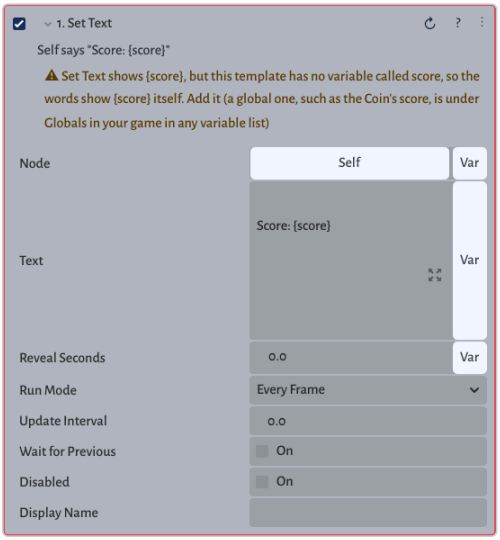
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ui/set_disabled/set_disabled.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The button to grey out or bring back. Left empty it uses the node this Machine works on, which must then be a button. Anything that is not a button gets a warning, and nothing changes. Var: a Node.	Self
Button Disabled	Ticked, as it starts, greys the button out so clicks and keys do nothing to it, and unticked makes it pressable again. Press Var to follow a true-or-false variable, such as one that is true while there is no saved game. Var: a bool.	on
Store In	The variable true or false goes into, for whether the button is greyed out now, after the change. Left empty, nothing is stored.	empty

Set Text

Puts words on a label, a button, a rich text box or a text field. Any variable's name in curly brackets is swapped for its value, and a score board of several numbers is one action. With Reveal Seconds the words appear a letter at a time, as a character speaking.



Set Text as it appears in the Actions tab.

When to use it

On Every Frame, "Ammo: {ammo}" stays up to date, and the words are rewritten only when a variable named in the box changes. A score board costs nothing while the score stands still.

See also

Show Choices 217, **Set Value** 215, **Print** 248, **Set Texture** 214

Try it

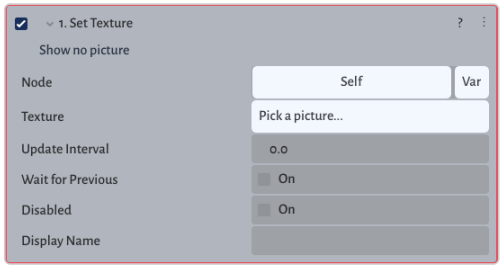
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ui/set_text/set_text.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node whose words change. Leave it empty, as it comes, for the thing this machine works on. Point it at your score label, your dialogue box or a button whose wording changes. Var: a Node.	Self
Text	The words to show. Write a variable's name in curly brackets anywhere to drop its value in, as in Score: {score}, and {target} for the name of the node. After a colon you can say how a number should look, such as {time:mm:ss} as minutes and seconds, {score:0000} padded with zeros, {speed:0.0} to one decimal, or {volume:%} as a percentage. A name belonging to no variable is left on the screen as typed, and a spelling mistake shows up that way. {x:0} and {x:0.0} round to that many places, each part of a Vector too, so {place:0} gives (100, 20). A list comes out as its items with commas between, and a table as each name with its value, such as potion: 2, key: 1. The {x:mm:ss} form counts whole seconds, as a clock does. Var: a String.	Score: {score}
Reveal Seconds	0 to begin with, so the whole line goes up in one go. Give it a number and the letters appear one by one over that many seconds, the way a character speaks in a story game, and the state waits for the last letter before it moves on. Only a Label or a RichTextLabel can reveal words this way. On anything else a warning says Reveal Seconds does nothing. Var: a float.	0.0
Skip Input	Only when Reveal Seconds is above 0. The name of a control from the project's Input Map, such as ui_accept, that shows the rest of the words when pressed. Empty, as it comes, the line cannot be skipped and an impatient player sits through every one.	—

Set Texture

Changes the picture something shows. It works on a sprite in a 2D or 3D scene and on a picture or a picture button on the screen.



Set Texture as it appears in the Actions tab.

When to use it

A chest drawn open once it has been looted, or a portrait whose face changes when the character is hurt. It swaps one still picture for another.

See also

Set Text 213, **Set Visible** 216, **Flip Sprite** 161, **Play Sprite Animation** 163

Try it

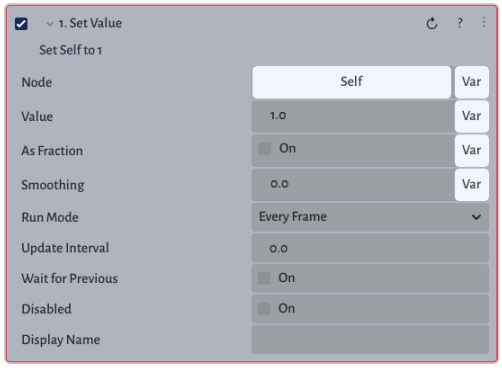
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ui/set_texture/set_texture.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node showing the picture. Leave it empty, as it comes, for the thing this machine works on. If that node cannot show a picture, a warning lists the ones that can, which are a Sprite2D, a Sprite3D, a TextureRect and a TextureButton. Var: a Node.	Self
Texture	The picture to show, dragged in from the FileSystem dock. Empty, as it comes, shows no picture, which is a tidy way to clear an inventory slot.	empty

Set Value

Sets the value of a slider, a progress bar or a spin box, or anything with a smallest and a largest value, such as a health bar.



Set Value as it appears in the Actions tab.

When to use it

A volume slider set to its saved value as the options menu opens, or a bar showing how charged an attack is. Read the value from a variable with the Var button and the bar follows that number.

See also

Set Text 213, **Ease Variable** 44, **Modify Stat** 268, **Set Visible** 216

Try it

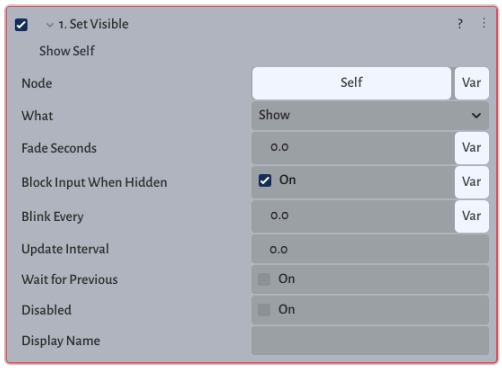
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ui/set_value/set_value.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The slider, progress bar or spin box. Leave it empty, as it comes, for the thing this machine works on. Anything else has no value to set, so nothing moves and a warning names the node. Look for that warning first when a bar sits still. Var: a Node.	Self
Value	What to set it to, 1 to begin with. Unless As Fraction is ticked it is a raw number and must lie between the bar's smallest and largest values, set on the bar itself in the Inspector. Read from a variable with a Min or Max (the Var button), the bar's two ends are set from them as it goes. For example, health never above 3 fills the bar at 3, and the bar's own Max Value can stay as it is. Var: a float.	1.0
As Fraction	Off to begin with, and Value is a plain number. Ticked, Value is a share of the bar's range, 0 for empty and 1 for full. No percentages to work out when the bar runs from 0 to 100 but your variable does not. Var: a bool.	off
Smoothing	0 to begin with, so the bar jumps straight to the new value. Give it a number and the bar slides there at that many units a second, and a health bar drains away. It slides only on Every Frame. On Once the value is always set in one step. Var: a float.	0.0

Set Visible

Shows or hides anything, such as a menu, a panel, a health bar or a sprite. With a fade it eases in or out, where a plain show blinks on.



Set Visible as it appears in the Actions tab.

When to use it

Every menu and HUD leans on it to show the pause panel, or hide the health bar during a cut scene. It makes things out in the world appear and vanish too. A fade keeps running while the game is paused, and a pause menu can fade in over the frozen game.

> Starting a fade stops one still running on the same node. Showing something already shown does nothing, and a key held down will not restart the fade every frame.

See also

Set Texture 214, **Grab Focus** 208, **Pause Game** 222, **Tween Property** 106

Try it

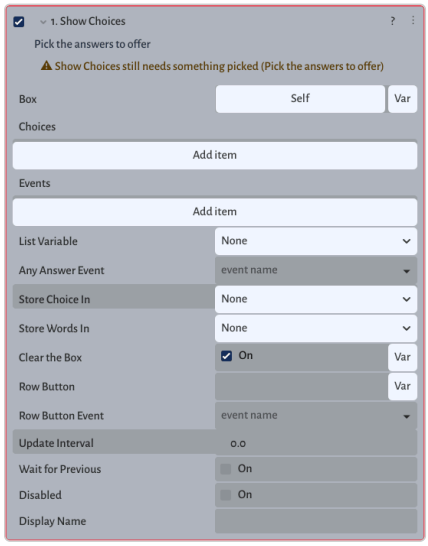
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ui/set_visible/set_visible.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node to show or hide. Leave it empty, as it comes, for the thing this machine works on. Point it at the whole panel, not at one button inside it, so everything goes together. Var: a Node.	Self
What	Show, as it comes, makes it visible, Hide takes it away, and Toggle flips whichever it is now, so one key can open and close an inventory. Choices: Show, Hide, Toggle.	Show
Fade Seconds	0 to begin with, so it appears or disappears in one step. A quarter of a second feels smooth and holds nobody up. The fade ends at however see-through the panel was designed to be, so a panel made half see-through comes back half see-through. A 3D thing, and a whole screen layer drawn over the game, cannot fade and are shown or hidden in one step. Var: a float.	0.0
Block Input When Hidden	On to begin with, so a hidden panel also stops answering the mouse, and an invisible menu cannot be clicked by accident. Shown again, it answers the mouse as it was set to before. Turn it off only if something else is managing that. Var: a bool.	on
Blink Every	Keeps flipping it between shown and hidden, this many seconds apart, for as long as the state runs, like an alarm light or a warning that flashes. It starts the way Show, Hide or Toggle says, and there is no fade while it blinks. 0, as it comes, shows or hides it once. Leaving the state puts it back as it was before the blinking began. Var: a float.	0.0

Show Choices

Puts a button on the screen for each answer the player can choose, and sends the event that goes with whichever one is picked. The answers are typed in, or taken from an Array variable when they are only known while the game runs. The buttons are made inside a box in your own panel and taken away again once an answer is chosen.



Show Choices as it appears in the Actions tab.

When to use it

A shop's Buy, Sell and Leave, each answer sending its own event. The state waits until an answer is picked, so give it a transition for each event. The first button has the keyboard and gamepad from the start, and a player with no mouse can still choose.

> Only an answer moves the state on. For a question that times out, put a Timer or a Wait in a state that sits around this one.

See also

Set Text 213, **Grab Focus** 208, **Set Visible** 216, **Get Event Data** 20

Try it

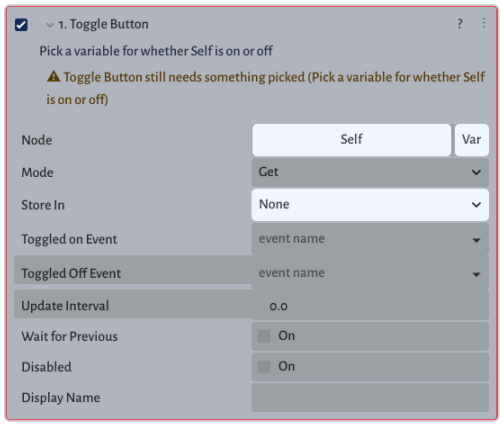
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ui/show_choices/show_choices.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Box	The box the buttons are put into. It is a container in your dialogue panel that stacks its children in a row or a column. Leave it empty, as it comes, for the thing this machine works on. With no box there, no buttons appear and a warning asks for a VBoxContainer or an HBoxContainer. Var: a Node.	Self
Choices	The answers, one to a row, in the order the buttons appear. Write a variable's name in curly brackets to drop its value into an answer, such as Buy it for {price} coins. With no answers typed in, the action does nothing, finishes straight away, and warns that there are no answers to offer.	empty
Events	The event each answer sends, in the same order as the answers. Each one needs a transition on this state waiting for it, or picking that answer takes the buttons away and leaves the machine standing still. An answer with no event beside it sends nothing.	empty
List Variable	An Array variable to take the answers from, in place of the rows above. It makes one button for each item in it, in its order, with the item's words as they are. The saves that exist, the games opened lately or what is in an inventory can be offered this way. With an empty list no buttons appear and the state completes straight away, so give it a way out for that. A variable holding something other than an Array shows no buttons, and a warning says what it holds.	empty
Any Answer Event	An event sent whichever answer is picked, after that answer's own. With answers from a list there is no telling how many there will be. Wait for this event, and Store Words In says which was picked.	empty
Store Choice In	A variable given the number of the answer picked, counting from 1 for the first. Empty, as it comes, keeps nothing. Handy when several answers lead to the same state and the next state still has to tell which was chosen.	empty
Store Words In	A String variable that is given the words on the button that was picked. With answers from a list this is how the machine knows what was chosen. Empty keeps nothing.	empty
Clear the Box	On to begin with, so any buttons already sitting in the box, such as the last question's answers, are taken away before these are added. Anything in the box that is not a button, such as the portrait or the words, is left alone. Turn it off only if you mean to add these answers below buttons that are already there. Var: a bool.	on
Row Button	Words for a small button at the end of each answer's row, such as Delete, Sell or Drop. Empty, as it comes, puts no button there and the answers are plain buttons as before. Pressing a row's button keeps the answer in Store Choice In and Store Words In, just as picking it would, but sends Row Button Event in place of the answer's own events. A list of saves with a Delete button on each is made this way. Var: a String.	empty
Row Button Event	Sent when a row's own button is pressed, and the answer's events are not. Give the state a transition waiting for it, such as one to a page that asks whether to delete the save in Store Words In.	empty

Toggle Button

Reads or sets a button that stays on or off, such as a tick box or a switch on an Options page, or any button with Toggle Mode on. While its state runs it sends one event when the player turns it on and another when they turn it off.



Toggle Button as it appears in the Actions tab.

When to use it

A Fullscreen tick box on an Options page is the usual case. Its Toggled On Event leads to a state where Screen Settings goes fullscreen, and Toggled Off Event to one that goes back to a window. Set, as the page opens, shows a saved setting, and Get reads one. A button that is only pressed, never left on, is heard with Event On Signal.

> The action finishes as soon as its state starts and listens in the background while the state runs, and the listening stops when the state is left.

See also

Set Disabled 212, **Screen Settings** 225, **Event On Signal** 76, **Save Game** 235

Try it

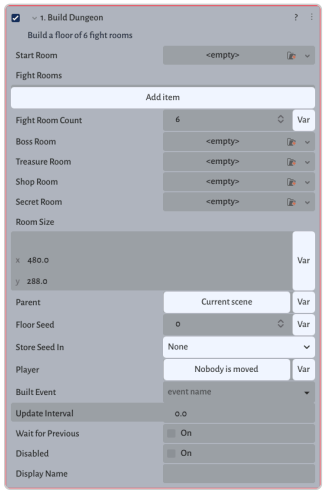
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ui/toggle_button/toggle_button.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The tick box (a CheckBox), switch (a CheckButton) or other button with Toggle Mode on. Left empty it uses the node this Machine works on. A button that springs back when it is let go is never on or off, and gets a warning. Var: a Node.	Self
Mode	Get only stores whether it is on. Set turns it on or off, as the setting below says, and Toggle turns it off when it is on and on when it is off. Set and Toggle change only what the button shows and send neither event, because showing a saved setting is not taken for the player changing it. Choices: Get, Set, Toggle.	Get
Store In	The variable true or false goes into, for whether the button is on after the change, and again each time the player turns it on or off while the state runs. Left empty, nothing is stored.	empty
Toggled on Event	The event sent when the player turns the button on while this state runs. Store In is written first. Left empty, turning it on sends nothing.	empty
Toggled Off Event	The event sent when the player turns the button off while this state runs. Store In is written first. Left empty, turning it off sends nothing.	empty
Button Pressed	Only when Mode is Set. Ticked, as it starts, turns the button on, and unticked turns it off. Press Var to show a true-or-false variable, such as a setting kept with Save Game.	—

Build Dungeon

Lays out a floor of rooms from room scenes you have drawn by hand, a new layout every run. The rooms sit side by side on a grid around the start room, every one of them reachable. Each room and each door is told what it is, and can open, close or lock itself.



When to use it

A dungeon crawl, or a run-based game where the player clears room after room and should find a different floor each time. Every room scene must be the same size, with a door node in the middle of each side named DoorNorth, DoorSouth, DoorEast or DoorWest.

> Building a floor takes away the floor built before it, so asking for a new one never leaves the old rooms behind.

See also

Instantiate 84, **Random Seed** 48, **Change Scene** 220, **New Run** 293

Try it

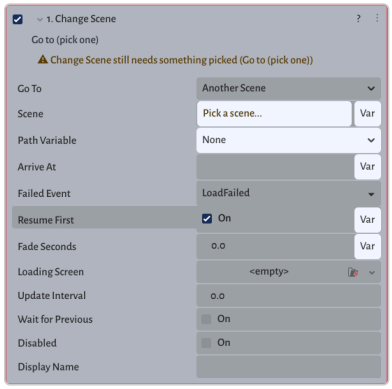
See the action work on its own with **Open Example** in the Add Action window, or open `examples/scene/build_dungeon/build_dungeon.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Start Room	The room scene the player begins in. It is placed in the middle of the grid and the rest of the floor grows out from it. Nothing is built at all while this is empty.	empty
Fight Rooms	The room scenes used for fighting. One is picked at random for each fight room on the floor, so a list of five or six different rooms keeps a run from looking the same twice. Nothing is built while this list is empty.	empty
Fight Room Count	How many fight rooms the floor has, six to begin with. Raise it for a longer floor and lower it for a quick one. This is the main dial for how big a run feels. Var: an int.	6
Boss Room	The room the boss waits in. It is put as far from the start as the layout allows, so the player has to work their way to it. Leave it empty and the floor has no boss room.	empty
Treasure Room	A reward room, placed as a dead end off the floor so the player has to choose to walk to it. Leave it empty to leave it out.	empty
Shop Room	A shop, also a dead end off the floor. Leave it empty to leave it out.	empty
Secret Room	A hidden room, tucked into a space between rooms and joined to each of them. The doors into it are told they are secret, so you can keep them looking like wall until the player finds them. Leave it empty to leave it out.	empty
Room Size	How big one room is, in pixels, and so how far apart the rooms are set on the grid. It must match the size you drew your rooms at, 480 by 288 to begin with. Get it wrong and the floor comes out with gaps between rooms, or rooms on top of each other. Var: a Vector2 or a Vector3, or a Node for where it stands.	480, 288
Parent	The node the rooms are added under. Leave it empty and they go under the running scene. Point it at one of your nodes to keep the floor together in one place. Var: a Node.	empty
Floor Seed	The number the layout is worked out from. 0, as it starts, gives a new floor every time. Put a number in and that floor comes back every run, so you can test one layout over and over, or let two players play the same floor. Var: an int.	0
Store Seed In	The name of a variable the number used is written into. Show it on the screen so a player can write it down, and keep it with Save Game to build that floor again later. Empty stores nothing.	empty
Player	The player, moved into the start room as the floor is built. It goes onto a node called PlayerStart if the start room has one, otherwise to the middle of the room. Leave it empty and the player stays wherever they were standing. Var: a Node.	empty
Built Event	An event sent once the whole floor is standing, for fading the screen in, starting the music or showing the floor number. Empty sends nothing.	empty

Change Scene

Leaves the scene that is running and starts another one in its place, such as a menu starting the game, a door to the next level, or this level again after losing. The screen can fade out and back in around the change, and a loading screen can show while a big level loads.



Change Scene as it appears in the Actions tab.

When to use it

After the change the old scene is gone, with every node and every ordinary variable in it. Three things survive. They are global variables, what Save Game wrote down, and the player, if their template has Carries Over Between Scenes ticked in the Variables tab. A carried player takes their health, gold, level, pack and sword into the next scene, and the scene they left is remembered for when they come back. Adding one thing to the running scene, with the rest left standing, is Instantiate's job.

> A second change asked for while one is already under way is ignored, so a button pressed twice still changes scene only once. Going to a scene with nobody to carry the player on, such as a title screen, ends the journey, and the next game starts fresh, with no scene remembered.

See also

Save Game 235, **Pause Game** 222, **Quit Game** 224, **Screen Effect** 202

Try it

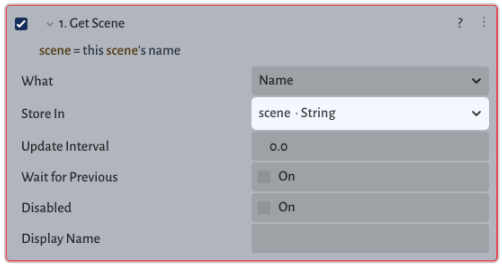
See the action work on its own with **Open Example** in the Add Action window, or open `examples/scene/change_scene/change_scene.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Go To	Another Scene, as it starts, goes to the scene you pick below. Reload Current Scene starts the running scene over from the beginning, for a Try Again button after losing, with no scene to pick. A player carried in from another scene starts it again as they walked in, with the health and gear they had then, at the same spot. Choices: Another Scene, Reload Current Scene.	Another Scene
Scene	Normally empty, and hidden while it is. It holds the scene itself, as Godot's Inspector sets it. A scene held this way is carried inside the template, so two scenes that hold each other cannot load. Pick the scene in Scene below.	empty
Scene	The scene to go to, picked from your own scenes by name, or any scene with Pick a file.... It keeps the scene's path, so a title screen and its level can go to each other. It shows in warning colour until one is picked. Var: a String.	empty
Path Variable	The name of a variable holding the place of the scene to go to, read as the change happens, so one Change Scene can go to any level. A level-select menu writes the chosen level into that variable and this reads it back. Filled in, it overrides the Scene above. Left empty, the Scene above is used.	empty
Arrive At	Only for Another Scene. The name of a spot in the new scene, such as a Marker2D called CaveDoor, where the player arrives. A player whose template has Carries Over Between Scenes ticked is stood there, and when nobody is carried over the new scene's own player is. In 3D they also face the way the spot faces. Coming out of a cave then puts the player at the cave door, not back at the start of the village. Leave it empty and they start where the new scene puts them. Var: a String.	empty
Failed Event	The event sent when there is nothing to load, because there is no scene at that path or the scene running now cannot be started again. It comes set to LoadFailed. Nothing else happens in that case, and with no event the menu sits there as if the button were broken.	LoadFailed
Resume First	Lets the game run again before the change, which it does to begin with. A pause menu that froze the game must leave this ticked, or the new scene comes up frozen and nothing in it moves. Var: a bool.	on
Fade Seconds	How long the screen takes to fade out, and again to fade back in, around the change. 0, as it starts, cuts straight across with no fade. A third of a second is a gentle change that hides the join. Var: a float.	0.0
Loading Screen	A scene of your own shown while the next one loads quietly in the background, such as a picture, a line of advice or a bar. Leave it empty and the old scene stays on screen until the new one is ready, fine for a level that loads quickly.	empty
Put Globals Back	Shown for Reload Current Scene. On, as it starts, the game's global variables go back to what they were when this scene began. A restart after losing then has the score the level began with, and its coins can be taken again without counting twice. Off keeps every global as it is now, for a game whose settings, such as volume, are globals the player may have changed in the pause menu.	—
Keep as They Are	Shown while Put Globals Back is on. The globals to leave alone, picked from the game's globals, such as lives, so losing one is not undone by starting again, or a best time.	—
Fade Color	Only when Fade Seconds is above 0. The colour the screen fades to and comes back from. Black to begin with. White suits a bright game, and a dark red a death.	—
Store Progress In	Only when a Loading Screen is set. How far the loading has got, counted from 0 to 1, goes into a global variable of this name, for a bar on the loading screen. Give the loading screen's own machine a variable with this name and Is Global ticked, or the bar has nothing to follow and never moves.	—

Get Scene

Puts the name of the scene running now, or the file it was loaded from, into a text variable.



Get Scene as it appears in the Actions tab.

When to use it

Store the level's Scene File Path, and a Game Over screen's Retry can hand it to Change Scene to play that level again. It asks about the game's current scene, and works the same from a machine in a HUD that stays from scene to scene.

> A scene made while the game runs and never saved has no file. Scene File Path gives nothing then, and a warning says the scene was never saved.

See also

Change Scene 220, **Set Text** 213, **Save Game** 235, **Format String** 54

Try it

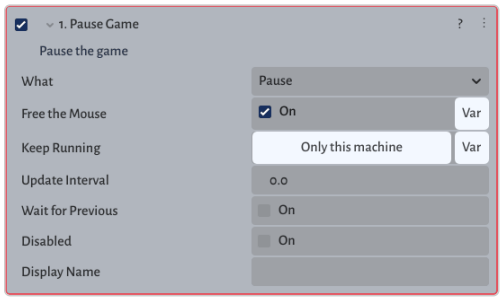
See the action work on its own with **Open Example** in the Add Action window, or open `examples/scene/get_scene/get_scene.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
What	Name gives the name of the scene's top node, such as Forest. Scene File Path gives the scene's file inside the project, the path Change Scene loads to play it again. Choices: Name, Scene File Path.	Name
Store In	The text variable (a String) it goes into. It starts as scene. A number variable, or any other kind that is not text, is left alone and a warning names it.	scene

Pause Game

Freezes the game, or lets it run again. Everything holds still except what you name to carry on, so a pause menu keeps working while the world behind it is frozen.



Pause Game as it appears in the Actions tab.

When to use it

Escape opens a pause menu and its Resume button closes it. A level won or lost can stop the world while the score shows. The machine that pauses always keeps running, so the state that unpauses can still be reached, and it goes back to normal once the game runs again.

> A pause menu that is not named under Keep Running freezes with everything else, so its buttons will not respond.

See also

Set Time Scale 66, **Change Scene** 220, **Set Mouse Mode** 154, **Quit Game** 224

Try it

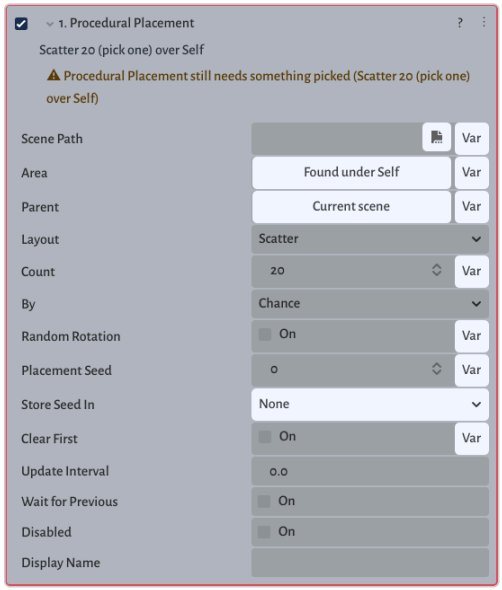
See the action work on its own with **Open Example** in the Add Action window, or open `examples/scene/pause_game/pause_game.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
What	Pause freezes the game, Resume lets it run again, and Toggle does the opposite of now, for a single Escape key. It starts on Pause. Choices: Pause, Resume, Toggle.	Pause
Free the Mouse	Shows the mouse pointer while the game is paused, and puts it back as the game had it once the game runs again. A pause menu in a game that hides or holds the pointer needs this. Ticked to begin with. Untick it only if you handle the pointer yourself. Var: a bool.	on
Keep Running	The pause menu, or anything else that must keep working while the game is frozen. Its buttons can still be pressed and everything inside it still moves. Leave it empty and only the machine doing the pausing keeps running. Resume puts everything back by itself and ignores this box. Var: a Node.	empty

Procedural Placement

Puts many copies of a scene over an area in one go, scattered or one to each cell of a grid, spread evenly at random or gathered in clumps by noise. The same seed places them the same way every time.



Procedural Placement as it appears in the Actions tab.

When to use it

A forest with clearings is 300 trees over a box shape, placed By Noise so they gather in woods with open ground between. The area is a collision shape drawn over the ground, the same kind Instantiate's Somewhere Inside uses.

> The scene has to match the area, so a 3D scene goes over a 3D shape and a 2D scene over a 2D one. With the wrong kind a warning is printed and no copies are placed.

See also
Instantiate 84, **Build Dungeon** 219, **Random Seed** 48, **Duplicate** 73

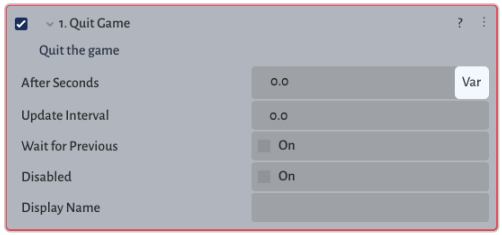
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/scene/procedural_placement/procedural_placement.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Scene Path	The scene to put down, picked from your scenes by name, such as a tree, a rock or an enemy. Var: a String.	empty
Area	The area to fill. It is a collision shape, or a node with one under it, such as an Area2D. In 3D it can be a box, sphere, cylinder or capsule, and everything stands at the height of the shape's middle. In 2D it can be a rectangle, circle or capsule. Left empty it uses the node this Machine works on. Var: a Node.	Self
Parent	The node to add them under. Left empty they go under the scene being played. Var: a Node.	empty
Layout	Scatter puts Count of them anywhere in the area. Grid puts one in each cell of a grid across it, in neat rows unless Jitter nudges them. Choices: Scatter, Grid.	Scatter
Count	Only when Layout is Scatter. How many to put down. By Noise they all go in the clumps, and fewer are placed if the clumps have no more room. Var: an int.	20
By	Chance places them evenly at random. Noise gathers them in clumps with gaps between, the way woods and clearings look. Choices: Chance, Noise.	Chance
Random Rotation	On turns each one a random way round, about its up in 3D or flat on the screen in 2D, so trees and rocks look less alike. Var: a bool.	off
Placement Seed	The same number places everything in the same places every time, so a world comes out the same after loading a saved game. 0, the default, follows the game's own random choices, which Random Seed sets. Var: an int.	0
Store Seed In	A variable to put the seed used into, to place the same again after loading a saved game. Leave it empty to store nothing.	empty
Clear First	On takes away the copies of this scene placed under the same parent before. Running it again then replaces the old lot and does not add to it. Var: a bool.	off
Cell Size	Only when Layout is Grid. How far apart the cells are, in metres in 3D or pixels in 2D. Cells so small that the grid would have more than five thousand are refused with a warning.	—
Jitter	Only when Layout is Grid. How far each one may be nudged from the middle of its cell, as a share of a cell. 0 keeps neat rows and 1 lets it go anywhere in its cell.	—
Chance	Only when Layout is Grid and By is Chance. How likely each cell is to get one, from 0 for never to 1, the default, for always.	—
Frequency	Only when By is Noise. How big the clumps are. A smaller number makes bigger woods and wider clearings, a bigger one many small clumps. In a 2D game, where distances are in pixels, it needs to be much smaller, around 0.004.	—
Threshold	Only when By is Noise. How much of the area the clumps cover, from -1 for nearly all of it to 1 for nearly none, and 0 covers about half.	—

Quit Game

Closes the game, as a Quit button on a title screen does. Run from the editor, it stops the test and leaves the editor open.



Quit Game as it appears in the Actions tab.

When to use it

The Quit button of a title screen or pause menu, or the Yes button of an "Are you sure?" page. Anything the player should keep must be written down with Save Game before this runs, because nothing else survives the game closing.

> Once the wait has started it cannot be called off. The game closes even if the machine has moved on to another state, and even if the game is paused.

See also

Change Scene 220, **Save Game** 235, **Pause Game** 222

Try it

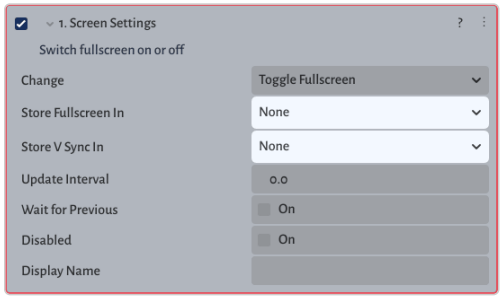
See the action work on its own with **Open Example** in the Add Action window, or open `examples/scene/quit_game/quit_game.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
After Seconds	How long to wait before the game closes, so a button click, a short tune or a fade has time to finish. 0, as it starts, closes straight away. A quarter of a second covers a click, and for a tune or a fade, match its length. Var: a float.	0.0

Screen Settings

Does what an Options menu does to the screen. It sets fullscreen or a window, the size of the window, and V-Sync. It can also read how the screen is set now into variables, so a menu's tick boxes start out right.



Screen Settings as it appears in the Actions tab.

When to use it

An Options page. Run it once with Change set to Nothing as the page opens, and the boxes show the real settings before the player touches anything. Keep the player's choices in variables, write them down with Save Game and set them again as the game starts. Otherwise the player chooses them all over again every time.

> While the game is being tested inside the editor the window is left alone, so run the game in its own window to see fullscreen and size changes.

See also

Save Game 235, **Set Language** 226, **Set Mouse Mode** 154, **Set Volume** 180

Try it

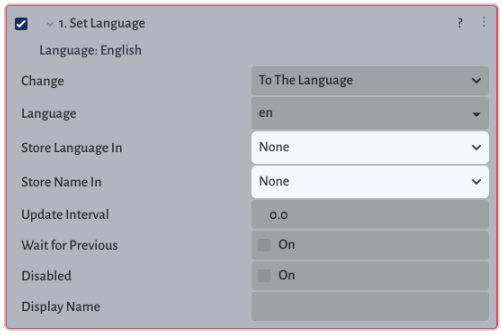
See the action work on its own with **Open Example** in the Add Action window, or open `examples/scene/screen_settings/screen_settings.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Change	What to change. Nothing changes nothing and only reads how the screen is set now. Fullscreen fills the screen and Windowed puts the game back in a window. Toggle Fullscreen, where it starts, swaps between those two, for one tick box or one key. Window Size puts the game in a window of the size below, and V Sync turns V-Sync on or off. Choices: Nothing, Fullscreen, Windowed, Toggle Fullscreen, Window Size, V Sync.	Toggle Fullscreen
Store Fullscreen In	The name of a variable set to true or false for whether the game is fullscreen, read after the change has happened. Point a tick box at it and the box is always right. Empty stores nothing.	empty
Store V Sync In	The name of a variable set to true or false for whether V-Sync is on, read after the change. Empty stores nothing.	empty
Window Size	Only when Change is Window Size. How big the window is made, in pixels across and down, 1280 by 720 to begin with. Offer two or three sizes in a menu, not a box to type in. Asking for a size also brings the game out of fullscreen.	—
V Sync On	Only when Change is V Sync. Ticked keeps the picture in step with the screen and stops it tearing across the middle, and most games leave it ticked. Unticked lets the game draw as fast as it can, and some players prefer the quicker response.	—

Set Language

Changes the language the game is shown in, or steps on to the next one the project has. Every label, button and line of text that has a translation changes straight away.



Set Language as it appears in the Actions tab.

When to use it

A Language button in an Options menu, with one button for each language or a single button that steps through them all. It can also read back which language is in use, for the button's label. The translations themselves are added in the project's settings, under Localization. This action only chooses between them.

See also

Set Text 213, **Screen Settings** 225, **Save Game** 235, **Show Choices** 217

Try it

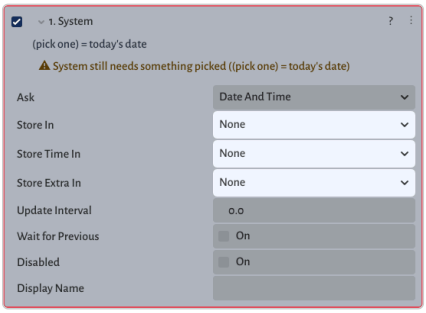
See the action work on its own with **Open Example** in the Add Action window, or open `examples/scene/set_language/set_language.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Change	To The Language, where it starts, switches to the one named below. To The Next One steps on to the next language the project has and comes back round to the first after the last, enough for a single Language button. Nothing changes nothing and only reads the language in use now. Choices: To The Language, To The Next One, Nothing.	To The Language
Language	The language to switch to, written as its short name, such as en for English, fr for French or de for German. It is used only by To The Language. A short name the project has no translation for leaves the game showing its untranslated text.	en
Store Language In	The name of a variable the short name of the language is written into after the change, such as fr. Keep it with Save Game so the game starts next time in the language the player chose. Empty stores nothing.	empty
Store Name In	The name of a variable the language's full name goes into after the change, such as French. Show this one on the button, because French reads better than fr. Empty stores nothing.	empty

System

Asks the computer or phone the game is running on for the few things a game needs from it. It can open a web page, open a file with the computer's own program, copy text out or paste text in, and find today's date and time, what kind of machine the game is on, and where the player's folders are.



System as it appears in the Actions tab.

When to use it

A daily reward that has to know what day it is, or touch controls shown on a phone and hidden on a computer. One setting, Ask, decides which question it puts, and the boxes below change with it.

> The date and time come from the player's own machine, so a player who puts their clock forward can take tomorrow's daily reward today.

See also

Run Program 234, **Choose Folder** 229, **Format String** 54, **Save Game** 235

Try it

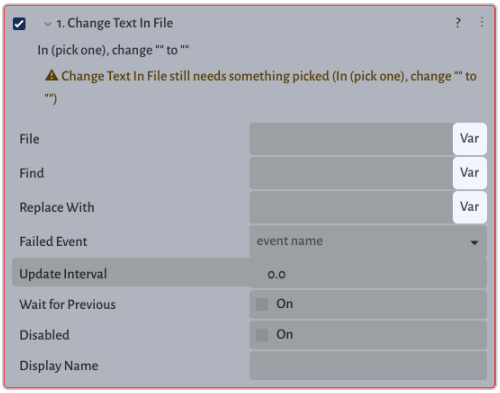
See the action work on its own with **Open Example** in the Add Action window, or open `examples/scene/system/system.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Ask	What to ask for. Open Web Page opens the address below in the player's browser. Copy Text puts the text below where the player can paste it. Paste Text reads back whatever the player last copied. Date And Time, where it starts, gives you today's date, the time and the day of the week. OS Name gives you one plain word for the machine, and whether it has a touch screen. Open A File opens a file or a folder the way double-clicking it would. Where A Folder Is stores the path of one of the computer's folders. Choices: Open Web Page, Copy Text, Paste Text, Date And Time, Os Name, Open A File, Where A Folder Is.	Date And Time
Store In	The name of the variable the answer goes into. That is the pasted text, the date as text such as 2026-09-13, one of windows, mac, linux, android, ios or web, or a folder's path with no slash on the end. Empty stores nothing, and for the asks that only give an answer, the action then does nothing you can see.	empty
Store Time In	For Date And Time, the name of the variable the time goes into, as text such as 14:05. It is the hour and minute only, on a twenty-four hour clock. Empty stores nothing.	empty
Store Extra In	For Date And Time, the name of the variable the day of the week goes into, as text such as Sunday. For OS Name, the same box takes true or false for whether the machine has a touch screen, so check it before showing touch buttons. Empty stores nothing.	empty
Text	Only when Ask is Open Web Page, Copy Text or Open A File. It is the web address for the first, the words to copy for the second, and the path of the file or folder for the third, such as a manual to read or the folder a screenshot was saved into. A variable's name in curly brackets drops that variable's value in. An address with no https:// in front of it is opened as though it had one. With nothing at the path, Open A File opens nothing and warns that there is nothing to open at that path.	—
Folder	Only when Ask is Where A Folder Is. Which folder's path to store. It can be the player's Documents, where it starts, their Desktop or their Downloads. Where The Game Is gives the folder the game's own program sits in (run from the editor, that is your project's folder), and The Game's Save Folder is where this game keeps its saves and settings.	—

Change Text In File

Replaces every place some words appear in a text file on the computer. The new words can come from a variable.



Change Text In File as it appears in the Actions tab.

When to use it

Straight after Copy Folder, this makes the copy the player's own, for example by putting the name they typed into the new project's settings. It is meant for small, known changes to plain text files. A game's own data belongs in Save Game.

See also

Copy Folder 230, **Format String** 54, **Save Game** 235, **File Exists** 231

Try it

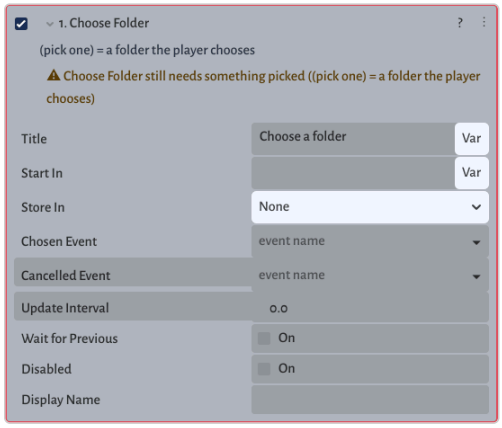
See the action work on its own with **Open Example** in the Add Action window, or open `examples/save_and_files/change_text_in_file/change_text_in_file.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
File	The text file to change. A plain name such as notes.txt is the one in the game's save folder, where Write File puts it. A variable's name in curly brackets drops its value in. Var: a String.	empty
Find	The words to look for, exactly as they are written in the file, capitals and spaces included. Every match is replaced, so choose words that appear only where you mean to change them. Var: a String.	empty
Replace With	What to put in their place. A variable's name in curly brackets drops its value in, such as the project name the player typed. Var: a String.	empty
Failed Event	Sent when there is no such file, it could not be written, or the words were not in it. Left empty, no event is sent, but a warning in the Output panel still gives the reason.	empty

Choose Folder

Opens the computer's usual folder window so the player can pick a folder, and puts its path into a variable. The state waits until they choose one or close the window.



Choose Folder as it appears in the Actions tab.

When to use it

For the moments when the player, not you, decides where something goes, such as the folder a new project is kept in, or where screenshots are saved. It belongs more in a launcher or a level editor made with NodeWorthy than in a game. Follow it with Copy Folder, or keep the path with Save Game so the player is asked only once.

> It needs a real computer, because on a phone or in a web browser there is no folder window to open.

See also

Copy Folder 230, **System** 227, **File Exists** 231, **Save Game** 235

Try it

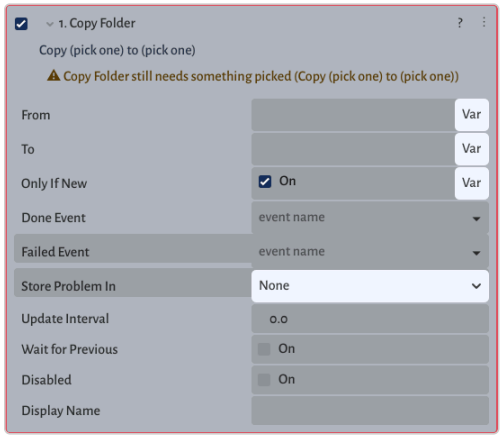
See the action work on its own with **Open Example** in the Add Action window, or open `examples/save_and_files/choose_folder/choose_folder.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Title	The words along the top of the folder window. Say what the folder is for, such as Where shall your games be kept? The player has nothing else to go on. Var: a String.	Choose a folder
Start In	The folder the window opens on. A variable's name in curly brackets drops its value in, for example the Documents folder the System action found. Empty, as it comes, opens wherever the computer last was. A folder that does not exist is ignored. Var: a String.	empty
Store In	The String variable the chosen folder's path goes into. It is left as it was when the player closes the window without choosing.	empty
Chosen Event	Sent when a folder is chosen. Give the state a transition waiting for it. Empty sends nothing, and the state still completes.	empty
Cancelled Event	Sent when the window is closed with nothing chosen. Send the player back to where they were, and do not carry on with an empty path.	empty

Copy Folder

Copies a folder and everything in it to another place on the computer. When it cannot, it gives the reason in a plain sentence. Anything already at the far end is left untouched unless you allow it to be written over.



Copy Folder as it appears in the Actions tab.

When to use it

Hands the player a copy to make their own. That could be a starter project into the folder they chose, a set of example levels, or a backup of their saves before a risky change. Because it will not write over what is already there, you can safely point it at a folder the player picked.

> A very large folder is copied in one go, and the game stands still until it is done. Show a Please wait message in the state before.

See also

Choose Folder 229, **Change Text In File** 228, **File Exists** 231, **Run Program** 234

Try it

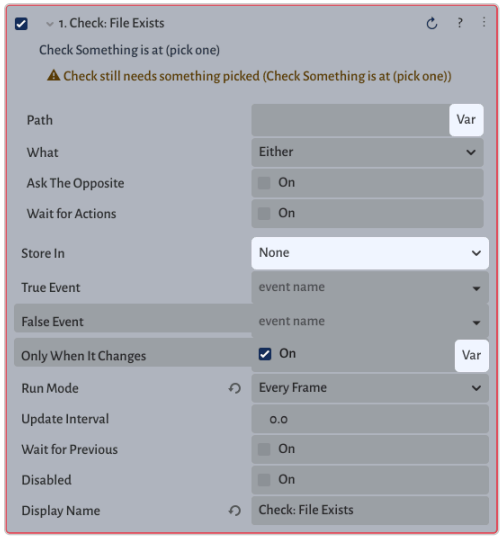
See the action work on its own with **Open Example** in the Add Action window, or open `examples/save_and_files/copy_folder/copy_folder.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
From	The folder to copy. A variable's name in curly brackets drops its value in, so a path can be built from where the game is and a name. Var: a String.	empty
To	Where the copy goes, written as the new folder's own path, not the folder it goes inside. To make First Game inside Documents, write the Documents path, a slash, and First Game. Curly brackets work here too. Var: a String.	empty
Only If New	On to begin with, so the copy is refused when anything is already at To, and a player's work is never written over by accident. Turned off, files with the same names are replaced and everything else there is left alone. Var: a bool.	on
Done Event	Sent once everything has been copied.	empty
Failed Event	Sent when it could not be done. The folder to copy may not be there, something may already be at To, or the computer may not allow the writing. Give it a transition to a state that tells the player.	empty
Store Problem In	A String variable that is given the reason it failed, as a plain sentence such as there is already something at that path, ready to put on the screen with Set Text.	empty

File Exists CHECK

Asks whether there is a file or a folder at a place on the computer.



File Exists as it appears in the Actions tab.

When to use it

Check before relying on something being there. Before offering Continue, ask whether a save exists. Before Run Program starts a program, ask whether it is where it should be. Before Copy Folder makes a new project, ask whether the name the player typed is already taken.

See also

Copy Folder 230, **Run Program** 234, **Save Game** 235, **Choose Folder** 229

Try it

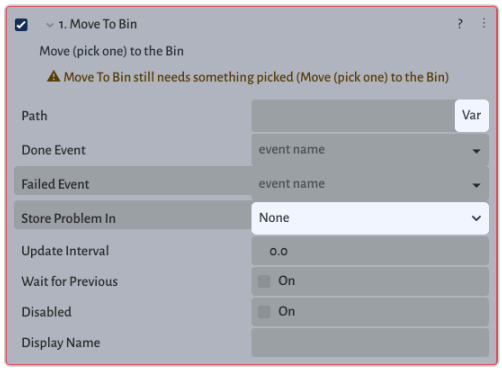
See the action work on its own with **Open Example** in the Add Action window, or open `examples/save_and_files/file_exists/file_exists.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Path	The place to look. A plain name such as notes.txt is looked for in the game's save folder, where Write File puts it. A variable's name in curly brackets drops its value in. An empty path is never there. Var: a String.	empty
What	What counts. Either, where it starts, is true for a file or a folder. A File and A Folder are true only for that kind, so a folder with the name of the file you wanted does not fool it. Choices: A File, A Folder, Either.	Either
Ask The Opposite	Turns the answer round. The check is then true when there is no file or folder at the path, handy for asking "is this name still free?". The true-or-false answer the Check stores is turned round as well.	off
Wait for Actions	Holds the question back until every other action in the state has finished, so it sees a file the state's own actions have just made. Left off, where it starts, it is asked as soon as the state runs.	off
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Move To Bin

Moves a file or a folder, and everything in it, to the computer's Bin, which is the Recycle Bin on Windows. When it cannot, it gives the reason in a plain sentence.



Move To Bin as it appears in the Actions tab.

When to use it

Lets the player throw something away, such as an old save, a level they made, or a whole game in a launcher. The game cannot bring it back, but the player can still take it out of the Bin by hand, so a slip is never the end of their work. Ask first, on a page with a Keep It button, before a state with this action in it.

> Nothing is ever deleted outright. Everything goes to the Bin, where the player can find it again until they empty it.

See also

Copy Folder 230, **File Exists** 231, **Show Choices** 217, **Choose Folder** 229

Try it

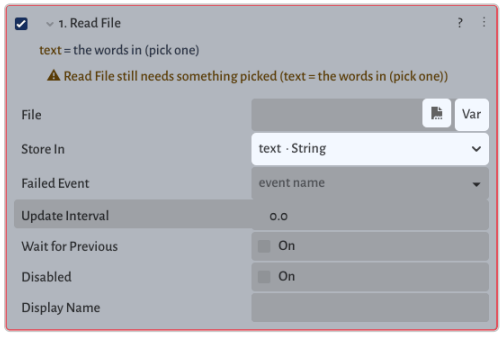
See the action work on its own with **Open Example** in the Add Action window, or open `examples/save_and_files/move_to_bin/move_to_bin.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Path	The file or folder to move. A plain path inside the game's save folder starts with user://, and a whole path on the computer works too. A variable's name in curly brackets drops its value in, such as the folder of the game the player picked. The top of a drive, the player's home folder, their Documents folder and the game's own folder are never moved, whatever the path says. Var: a String.	empty
Done Event	Sent once it is in the Bin.	empty
Failed Event	Sent when it could not be moved. There may be nothing at the path, it may be one of the places that are never moved, or the computer may not allow it. Give it a transition to a state that tells the player.	empty
Store Problem In	A String variable that is given the reason it was not moved, as a plain sentence such as there is nothing at that path, ready to put on the screen with Set Text.	empty

Read File

Reads a text file, line breaks and all, into a String variable. The file can be one that ships with the game or one the game wrote into its save folder.



Read File as it appears in the Actions tab.

When to use it

The words of a letter the player finds can live in letter.txt and be read in as the letter opens. Notes the player wrote earlier with Write File come back the same way. The game's variables are better kept with Save Game, which keeps numbers as numbers.

> A game's own .txt file is only packed into the finished game when the export is told to take it. In the export window's Resources tab, add *.txt to the filters for files that are not resources.

See also

Write File 236, **File Exists** 231, **Save Game** 235, **Change Text In File** 228

Try it

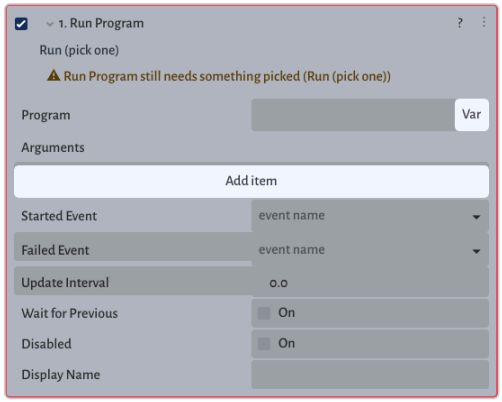
See the action work on its own with **Open Example** in the Add Action window, or open `examples/save_and_files/read_file/read_file.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
File	The file to read. Pick one of the game's own files here, or type a plain name such as notes.txt for a file in the game's save folder, where Write File puts it. Put a variable's name in curly brackets, such as {slot}, to drop its value into the name. Var: a String.	empty
Store In	The String variable the file's words go into. A variable that holds something else, such as a number, is left as it was, with a warning.	text
Failed Event	An event sent when there is no such file or it could not be read, such as notes not written yet. With none set, a warning says what went wrong.	empty

Run Program

Starts another program on the computer and carries on without waiting for it. Both keep running side by side.



Run Program as it appears in the Actions tab.

When to use it

Made for launchers that start the game they launch, and for tools that open an editor on the file they have just made. Arguments tells the program what to open. To open a document or a folder in whatever the computer normally uses, the System action's Open A File is simpler and needs no path to a program.

> It works on Windows, Mac and Linux. On a phone or in a web browser it starts nothing and sends the Failed Event.

See also

System 227, **File Exists** 231, **Copy Folder** 230, **Quit Game** 224

Try it

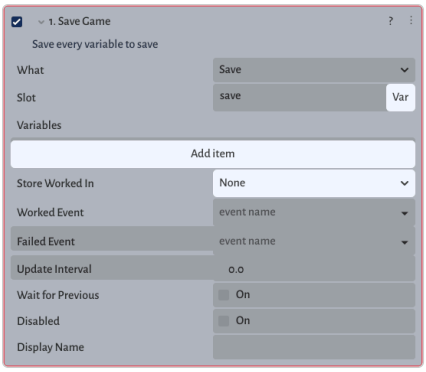
See the action work on its own with **Open Example** in the Add Action window, or open `examples/save_and_files/run_program/run_program.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Program	The program to start, as its path on the computer. A variable's name in curly brackets drops its value in, letting you build the path from where the game itself is. On a Mac an app, whose name ends .app, is started the way double-clicking it would. Var: a String.	empty
Arguments	What to hand the program as it starts, one to a row, such as a file for it to open. Each row can use curly brackets. What a program understands is up to that program.	empty
Started Event	Sent once the program has started.	empty
Failed Event	Sent when it could not be started, because there is nothing at that path or the computer refused. Give it a transition to a state that tells the player what to check.	empty

Save Game

Keeps a machine's variables from one run of the game to the next, such as a high score or the level reached. It can write a save, read one back, throw one away, or only ask whether one exists.



Save Game as it appears in the Actions tab.

When to use it

An Options menu keeps the player's choices for next time with it, and a title screen's Continue button can light up only when Exists finds a save. Each value comes back as the kind it went in, so a whole number is still whole and a colour still a colour. It keeps the variables of the machine the action sits in. Put it on the machine that holds them, or mark those variables global so every machine carrying them sees one value.

> A variable holding a node is never written into a save and is never overwritten by loading one. With Load The Scene, each machine starts again in the state it was in, so what that state does as it starts happens once more, and then its variables are put back. The machine doing the loading carries on where it is.

See also

Machine Variable 27, **Set Variable** 49, **Change Scene** 220, **Screen Settings** 225

Try it

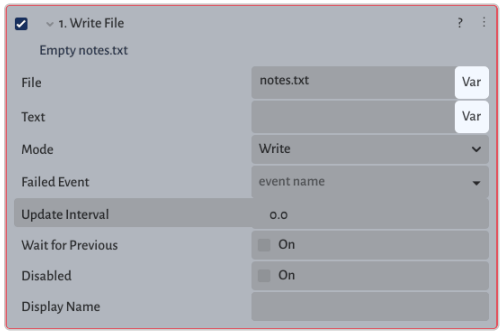
See the action work on its own with **Open Example** in the Add Action window, or open `examples/save_and_files/save_game/save_game.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
What	Save writes the variables down, Load reads them back, Delete throws the save away, and Exists only looks. Save The Scene keeps the whole scene as it is. That means every machine's variables and the state it is in, where each machine's node stands and which way it faces, whether it is shown and how fast it moves, what has been destroyed and what was spawned. Load The Scene puts all of that back, in the scene it was saved in. What was destroyed before the save stays gone, and what has been destroyed since comes back as the scene first had it. It starts on Save. Choices: Save, Load, Delete, Exists, Save The Scene, Load The Scene.	Save
Slot	The name of the save. A game with more than one save gives each a name. Leave it as save for a game with a single save, or try slot1, slot2 and slot3 for a game with three. Press Var beside the box and a menu can write the slot the player picked into a variable, so one Save Game serves all of them. Var: a String.	save
Variables	The variables to keep, by name. Leave the list empty and every variable the machine has is kept. List names when only some belong in the save. On loading, an empty list brings back everything in the file, and a list brings back only the names on it. Delete and Exists ignore it, and so do Save The Scene and Load The Scene, which keep every machine's variables. With nothing picked it also keeps every global variable in the game, such as the score, the lives or a key found, whichever template keeps them, and Load puts them back.	empty
Store Worked In	The name of a variable that is set to true or false. True means the save was written, the save was read, or the save was there. Read it straight afterwards to decide what a menu shows. Empty stores nothing.	empty
Worked Event	The event sent when it worked. Use it to move a menu on to a "Saved" message, or to start the level that was just loaded. Empty sends nothing.	empty
Failed Event	The event sent when it did not work, because there was nothing to load, the slot was not there, or the file could not be written. Use it to grey out a Continue button or to say "No save found". Empty sends nothing.	empty

Write File

Writes words into a text file, or adds them to its end. A plain name is kept in the game's save folder. The file is made if it is not there yet.



Write File as it appears in the Actions tab.

When to use it

To keep a record of every run, append a line such as "Run {run}: {score}" to runs.txt when the player dies, and Read File reads it back later. The game's own files cannot be changed once it is exported, and this never writes among them. A plain name goes into the save folder.

See also

Read File 233, **Save Game** 235, **Change Text In File** 228, **Take Screenshot** 283

Try it

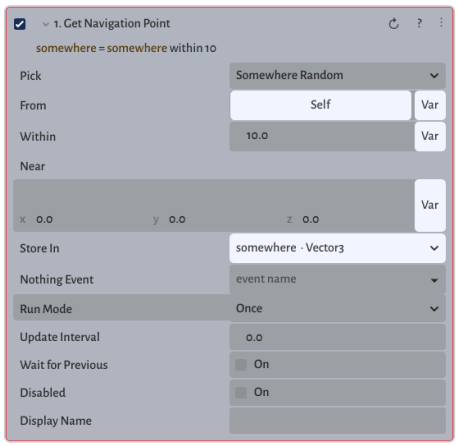
See the action work on its own with **Open Example** in the Add Action window, or open `examples/save_and_files/write_file/write_file.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
File	The file to write, such as notes.txt, kept in the game's save folder. A name with folders in it, such as logs/run.txt, makes the folders too. A full path, such as a folder Choose Folder gave with a name on the end, is written where it points. Put a variable's name in curly brackets, such as {slot}, to drop its value into the name. A file among the game's own is refused, with a warning saying why. Var: a String.	notes.txt
Text	The words to write. Put a variable's name in curly brackets, such as {score}, to drop its value in, or press Var to write the whole of a String variable. Var: a String.	empty
Mode	Write replaces everything the file held with the words. Append adds them to the end, after what is there already. Choices: Write, Append.	Write
Failed Event	An event sent when the file could not be written. Left empty, a warning says what went wrong.	empty
New Line	Only when Mode is Append. On, the default, ends the words with a new line, and each thing added starts on a fresh line.	—

Get Navigation Point

Picks a place on the level's navigation mesh and puts it in a variable. It can be somewhere random to wander to, or the nearest spot that can actually be stood on.



Get Navigation Point as it appears in the Actions tab.

When to use it

A wandering guard is two actions, this one on Somewhere Random, then Navigate To reading the place with Var. The Nearest Spot suits a place that may not be walkable at all, such as where a click landed, and gives the closest piece of floor to it. Like Navigate To it needs walkable ground in the level (NodeWorthy > Bake Navigation Mesh on the right-click menu). With none, it stores nothing, sends its Nothing Event, and the Problems button says so.

> Because (0, 0, 0) in Near means "use the node instead", there is no way to ask for the point nearest the world's exact origin. Put the place in a variable if you really need that.

See also
Navigate To 239, **Navigation Reachable** 240, **Wander** 263, **Random Number** 47

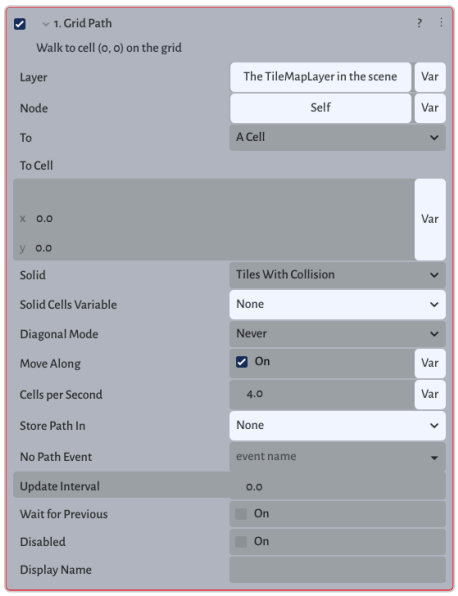
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/navigation/get_navigation_point/get_navigation_point.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Pick	Somewhere Random picks a random point on the mesh within the distance below, for wandering. The Nearest Spot picks the closest point on the mesh to where it is looking, which puts a target back onto walkable ground. Somewhere Random is the default. Choices: Somewhere Random, The Nearest Spot.	Somewhere Random
From	The node whose navigation map to use, and the place it measures from when Near is left alone. Empty, the default, uses the node this Machine works on. Whether that node is a 2D or a 3D one decides which kind of place comes out, so point Navigate To at the same world. Var: a Node.	Self
Within	How far away a random point may be, in metres in 3D or pixels in 2D. It starts at 10, a short step in 3D and barely anything in 2D, where hundreds are usual. Press Var to read it from a variable. The Nearest Spot ignores it. Var: a float.	10.0
Near	Look near this place, not near the node above. It starts at (0, 0, 0), which means the node is used. Press Var to read it from a variable. The Nearest Spot takes the spot on the mesh nearest it, and Somewhere Random takes it as the middle of the circle to wander in. Var: a Vector2 or a Vector3, or a Node for where it stands.	0, 0, 0 in 2D: 0, 0
Store In	The variable the place goes into. It starts at "somewhere". Point Navigate To's To at the same name with Var. Left empty, the place has nowhere to go and a warning asks for a variable name.	somewhere
Nothing Event	Sent when no usable point could be found, because the level has no navigation mesh or it has not been taken in yet. Nothing is stored then, so do not walk to whatever the variable held before. Empty sends nothing.	empty

Grid Path

Finds the shortest way across a grid of tiles, going round the walls, and walks it cell by cell, or only finds it and keeps it in a list. The grid is a TileMapLayer, Godot's grid of tiles.



Grid Path as it appears in the Actions tab.

When to use it

For a unit crossing the board in a tactics game, a mouse in a maze, and click to move on a grid. It moves from cell middle to cell middle at an even pace and finishes on arriving. A level not made of tiles needs Navigate To, which finds its way over a navigation mesh.

> The tiles must join up in rows and columns. Square tiles do, and so do diamond-shaped ones laid out Diamond Right or Diamond Down. Hexagons and offset rows do not, and the Output panel warns about them.

See also

Navigate To 239, **Get Cell** 241, **Set Cell** 244, **Move To** 97

Try it

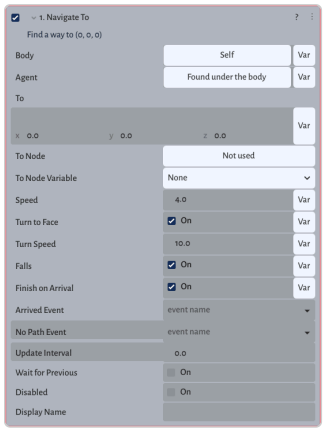
See the action work on its own with **Open Example** in the Add Action window, or open `examples/navigation/grid_path/grid_path.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Layer	The TileMapLayer the grid is. Left empty it uses the node this Machine works on when that is a TileMapLayer, or else the first TileMapLayer in the scene. In a game that has only one, that is the map itself. Var: a Node.	empty
Node	What walks the path, a 2D node or body. Left empty it is the node this Machine works on. The path starts from the cell it is standing in. Var: a Node.	Self
To	Where to go. A Cell, the default, goes to the cell below. A Node goes to the cell a node is in, such as a chest or the enemy. The Mouse goes to the cell the mouse pointer is over, for click to move. Choices: A Cell, A Node, The Mouse.	A Cell
To Cell	Only when To is A Cell. The cell, counted across and down as the tile map counts them, starting at 0, 0. Press Var to read it from a variable, such as the cell Get Cell stored. Var: a Vector2 or a Vector3, or a Node for where it stands.	0, 0
Solid	Which cells are walls it goes round. Tiles With Collision, the default, counts every tile with a collision shape, as walls usually have. Custom Data counts the tiles whose custom data under Data Name is true. Any Tile counts every tile on the layer, for a layer that holds only walls. Empty Cells counts every cell without a tile, for a layer that holds only floor. With Tiles With Collision and Custom Data, a cell with no tile is off the map and never walked through. Choices: Tiles With Collision, Custom Data, Any Tile, Empty Cells.	Tiles With Collision
Solid Cells Variable	A variable holding a list of more cells it may not step on, such as where the other units stand or a crate pushed there. The list may hold nodes too, and the cells they are in are blocked, leaving out the one walking. Left empty nothing more is blocked.	empty
Diagonal Mode	Whether it may step diagonally, in Godot's words. Never, the default, takes only steps across and down, the usual in a tactics game. Always takes diagonal steps whenever they are shorter, even between two walls. At Least One Walkable steps diagonally when at least one of the two cells beside the step is not a wall. Only If No Obstacles steps diagonally only when neither is, so it never cuts a corner. Choices: Always, Never, At Least One Walkable, Only If No Obstacles.	Never
Move Along	On, the default, it walks the path cell by cell and finishes on arriving, and a transition on FINISHED comes when it is there. Off, it only finds the path, keeps it in Store Path In and finishes straight away, for showing a path or counting its steps first. Var: a bool.	on
Cells per Second	Only when Move Along is on. How many cells it crosses a second, starting at 4. At 0 it jumps straight to the end. Var: a float.	4.0
Store Path In	The variable the path is kept in. It is a list of cells, from the one it starts in to the one it ends in, and an empty list when there is no way there. Left empty the path is not kept.	empty
No Path Event	The event sent when there is no way there, because the cell asked for is a wall or walled off. Left empty no event is sent, but a warning in the Output panel still reports that there was no way.	empty
To Node	Only when To is A Node. The node whose cell to go to. Left empty no node is picked, and the Output panel warns about it.	—
Data Name	Only when Solid is Custom Data. The name of the custom data, as the tile set calls it, that is true on a wall, such as solid or wall.	—

Navigate To

Walks the node to a place, or to a node it keeps following, along the level's navigation mesh, going round walls.



When to use it

An enemy crossing a room full of furniture to reach the player finds its way round with this. Move Character walks in a straight line and knows nothing about walls, so keep that for open ground. This needs walkable ground in the level and a NavigationAgent under the thing that moves, and adding it brings both. You get the agent and, the first time, ground baked over the level from its walls and floors. After moving a wall, right-click the level in the scene and choose NodeWorthy > Bake Navigation Mesh.

> With Finish on Arrival off and no event pointed anywhere, the state never completes on its own, so give it another way out.

See also

Get Navigation Point 237, **Navigation Reachable** 240, **Move Character** 134, **Chase** 251

Try it

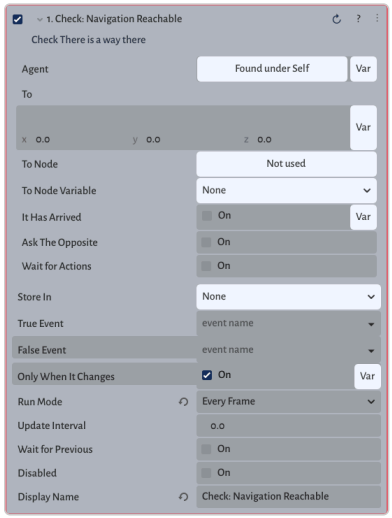
See the action work on its own with **Open Example** in the Add Action window, or open `examples/navigation/navigate_to/navigate_to.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Body	The body to move. Empty, the default, moves the node this Machine works on. Var: a Node.	Self
Agent	The NavigationAgent2D or NavigationAgent3D that works the path out. Empty, the default, uses the first one sitting directly under the body. Put one there and you never fill this in. With no agent found, nothing moves, the action finishes straight away, and a warning says to add one under the body or pick one here. Var: a Node.	empty
To	Where to go, as two numbers in a 2D game and three in a 3D one. Press Var to read the place from a variable, such as the one Get Navigation Point filled in. It is ignored whenever To Node is set. Var: a Vector2 or a Vector3, or a Node for where it stands.	0, 0, 0 in 2D: 0, 0
To Node	Go to where this node is, and keep following it as it moves. The path is worked out again once the node has moved more than a little, not every frame. It wins over To. Left empty, the place in To is used. If the node named is not there, the body stands still and a warning says so.	empty
To Node Variable	Follow the node held in this variable, so one state can chase whoever the target is now, such as the enemy that Look For Target found or the ally being escorted. It wins over To Node as well as over To. Empty, the default, uses the node picked above. If the node it names has gone from the scene, the body stands still and a warning says the variable holds no node.	empty
Speed	How fast it walks, in metres a second in 3D and pixels a second in 2D. It starts at 4, a brisk walk in 3D but almost still in 2D, where speeds are usually in the hundreds. Press Var to read it from a variable, so a chase can be faster than a patrol. Var: a float.	4.0
Turn to Face	On, the default, the body is turned to face the way it is walking. Turn it off for something that keeps its own facing, such as a flat sprite you flip yourself or a turret aiming somewhere else. Var: a bool.	on
Turn Speed	How quickly it swings round to face where it is going. Higher is quicker, and 10 has it round in about a fifth of a second. It starts at 10, a snappy turn. Lower it for something heavy and slow. It does nothing unless Turn to Face is on. Var: a float.	10.0
Falls	On, the default, gravity pulls it down as it walks, for a 3D character or a side-on platform game. Turn it off for a top-down game, where down the screen is not down. A 2D body whose Motion Mode is Floating is seen from above and never falls, even with this ticked. Var: a bool.	on
Finish on Arrival	On, the default, the state finishes as it arrives, so a transition on FINISHED carries on from there. Off, it keeps walking until the state is left, for following a moving node. Var: a bool.	on
Arrived Event	Sent when it arrives, and again each time it catches up with a node it is following after falling behind. Empty sends nothing.	empty
No Path Event	Sent when there is no way through to where it was told to go, because the place is off the mesh or walled off. It waits until the place has been out of reach for a quarter of a second first, so a level still being taken in does not count, and it is sent only once per visit to the state. Point it at a state that gives up and does something else. Empty sends nothing.	empty

Navigation Reachable CHECK

Asks whether the character can walk to a place or a node. Is there a route across the navigation mesh, the walkable floor worked out for the level? With It Has Arrived ticked, it asks whether the character has got there.



Navigation Reachable as it appears in the Actions tab.

When to use it

When the player jumps to a ledge the guard cannot reach, the guard can give up and go back to its post. A patrol can pick only points it can get to. With It Has Arrived ticked, a walking state ends when the character is there, with no guessing at a number of seconds. Ticking Ask The Opposite gives "there is no way through" and stops a character walking into a wall for ever.

> Asking never moves anything. The question is put to the navigation map, not to the agent, and a character already walking somewhere carries on where it was going.

See also

Navigate To 239, **Get Navigation Point** 237, **Move To** 97, **Patrol** 258

Try it

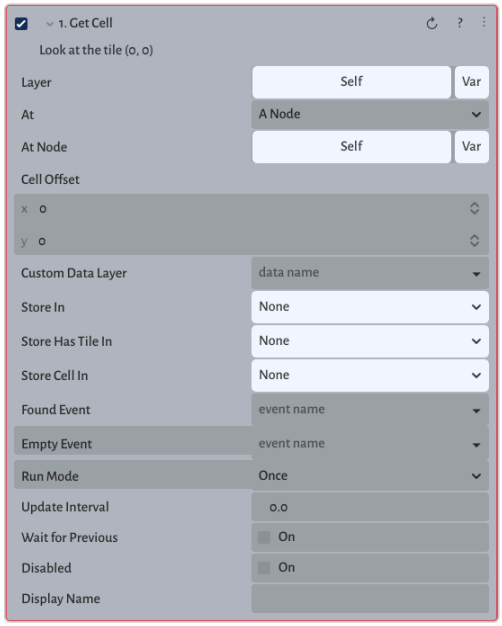
See the action work on its own with **Open Example** in the Add Action window, or open `examples/navigation/navigation_reachable/navigation_reachable.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Agent	The navigation agent to ask, the little helper node that works out routes for a character. Leave it empty and the first navigation agent directly under the node the machine works on is used. With no agent there the answer is always no, and a warning says to add one or pick one under Agent. Var: a Node.	empty
To	The place asked about, as two numbers in a 2D game and three in a 3D one. Left at zero it counts as nowhere named. The answer is then no, and a warning asks you to fill in To or To Node. It is ignored once To Node below is filled in. Var: a Vector2 or a Vector3, or a Node for where it stands.	0, 0, 0
To Node	Asks about wherever this node is standing, not a fixed place, and wins over the place above. This is how to ask "can I get to the player". Leave it empty to use To. If the node named is not there, the answer is no and a warning says it may have been removed.	empty
To Node Variable	Name a variable holding a node and the question is asked about wherever that node is standing. It wins over both To Node and To. An earlier action's find goes here, such as the nearest enemy or the cover point it picked. Empty, the default, leaves it to the two boxes above.	empty
It Has Arrived	Ticked, the question changes. It stops asking whether a route exists and says yes once the character has reached wherever it was last sent, by whichever action sent it. To and To Node are not used then. Leave it off, the default, for the can-I-get-there question. Var: a bool.	off
Ask The Opposite	Turns the answer round into "there is no way there", and the machine can give up and do something else. It turns round the true-or-false answer the Check stores as well, which swapping the True Event and False Event over cannot do.	off
Wait for Actions	Holds the question back until every other action in the state has finished, so it asks about a destination the state's own actions have just chosen. Left off, the default, it is asked as soon as the state runs.	off
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Get Cell

Asks a tile map what is at a place, such as what you are standing on, whether there is floor ahead, or which tile the mouse is over. It tells you what that tile is made of, whether there is a tile there at all, and which square it looked at.



Get Cell as it appears in the Actions tab.

When to use it

Ground that changes how something behaves, such as ice the player slides on. A walker looks at the square ahead and turns round when it is empty, and stays on its ledge. A yes-or-no question on a transition, such as whether the player stands on ice, is the Tile check's job. This action stores the answer or sends an event.

> While Layer does not point at a tile map layer nothing is read, and a warning names the node that is not one. When the answers never change, check Layer first.

See also

Tile 245, **Set Cell** 244, **Grid Map Cell** 242, **Raycast** 116

Try it

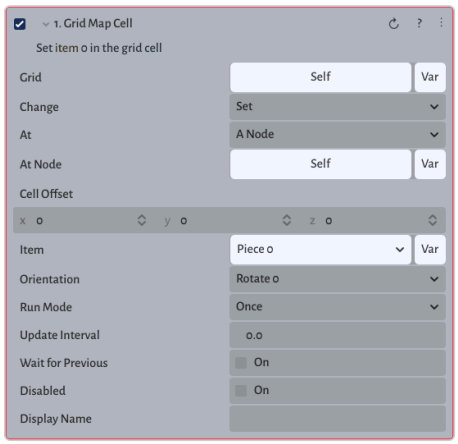
See the action work on its own with **Open Example** in the Add Action window, or open `examples/tilemap/get_cell/get_cell.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Layer	The tile map layer to ask. A tile map is Godot's grid of square tiles, and a layer is one sheet of it. A game often has the ground on one layer and what is drawn in front on another. Leave it empty and the thing this machine works on is asked, but that only works when it is a tile map layer itself. Var: a Node.	Self
At	A Node, where it starts, looks at the square a node stands in. The Mouse looks at the square the pointer is over, for a game where you click on tiles. Choices: A Node, The Mouse.	A Node
At Node	For A Node, the node whose place is looked up. Leave it empty for the thing this machine works on, such as the character asking what it is standing on. Var: a Node.	Self
Cell Offset	How many squares across and down from there to look. 0, 0 is the square the node is in. 0, 1 is the square below, the ground under its feet. 1, 0 is the next one along, for a look ahead before stepping onto it.	0, 0
Custom Data Layer	The name of the marking to read off the tile, spelled as the tile set spells it, such as surface, damage or slippery. The markings are set up once on the tile set, where its author says which tiles are ice and which lava. One action then covers every kind of ground. Leave it empty and nothing useful is stored. A name the tile set lacks gets a warning listing the names it has.	empty
Store In	The name of the variable that marking goes into. It comes back empty when the square is empty or the tile carries no marking of that name, so treat empty as ordinary ground. Empty stores nothing.	empty
Store Has Tile In	The name of a variable set to true or false for whether there is any tile in that square at all. Check this one for a hole in the floor, a ledge or a doorway, where what matters is an empty square. Empty stores nothing.	empty
Store Cell In	The name of a variable that is given the square looked at, as a pair of whole numbers counted across and down the grid. Hand it on to another action to change the square you have just looked at. Empty stores nothing.	empty
Found Event	The event sent when there is a tile in that square. Empty sends nothing.	empty
Empty Event	The event sent when the square is empty, such as the ledge ahead or a gap in the floor. Empty sends nothing.	empty

Grid Map Cell

Puts a piece into one square of a 3D grid, empties a square, or looks at what is in one. The grid here is Godot's GridMap, a three-dimensional grid built out of a library of ready-made shapes.



Grid Map Cell as it appears in the Actions tab.

When to use it

A building game where the player places and knocks out blocks, or a 3D world that changes shape as it is played, such as a floor crumbling away underfoot. It is the 3D twin of Set Cell and Get Cell, which work on a flat tile map.

> With At set to The Mouse and the pointer aimed at nothing, there is no cell that time and the action does nothing. That is not a mistake, and no warning is printed.

See also

Set Cell 244, **Get Cell** 241, **Mouse Pick** 149, **Instantiate** 84

Try it

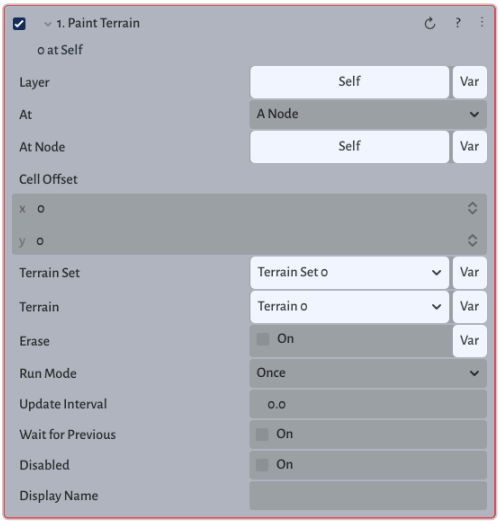
See the action work on its own with **Open Example** in the Add Action window, or open `examples/tilemap/grid_map_cell/grid_map_cell.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Grid	The grid to work on. Leave it empty for the thing this machine works on, but that only works when it is a grid itself. Var: a Node.	Self
Change	Set, where it starts, drops a piece into the square in place of whatever was there. Clear empties the square. Get leaves the grid alone and reports what is in the square. Choices: Set, Clear, Get.	Set
At	A Node, where it starts, works the square out from a node's place. The Mouse uses wherever the pointer is aimed, for a building game. Choices: A Node, The Mouse.	A Node
At Node	For A Node, the node whose square it is. Leave it empty for the thing this machine works on. Var: a Node.	Self
Cell Offset	How many squares across, up and along from there to work on. 0, 0, 0 is the square the node is in, and 0, -1, 0 the one below it, the block under a character's feet.	0, 0, 0
Item	For Set, which shape from the grid's library goes down. The library is set up on the grid and each shape in it has a name, so pick the wall, floor or pillar you have already made. Var: an int.	0
Orientation	For Set, how far round the piece is turned as it goes down. It can be no turn, a quarter, a half, or three quarters of the way round. Turned, one wall piece serves all four sides of a room. Choices: Rotate 0, Rotate 90, Rotate 180, Rotate 270.	Rotate 0
Store Name In	Only when Change is Get. The name of a variable the piece's name goes into, spelled as the library spells it. An empty square leaves it empty. Empty stores nothing.	—
Store Has Piece In	Only when Change is Get. The name of a variable set to true or false for whether the square holds a piece at all. Empty stores nothing.	—
Found Event	Only when Change is Get. The event sent when the square holds a piece. Empty sends nothing.	—
Empty Event	Only when Change is Get. The event sent when the square is empty. Empty sends nothing.	—

Paint Terrain

Paints cells of a tile map with a terrain while the game runs, and picks each tile so its edges join up with the tiles around it, the way the terrain brush in Godot's TileMap panel does. It can also dig cells out and let the tiles around close their edges.



Paint Terrain as it appears in the Actions tab.

When to use it

Dig a tunnel with At on The Mouse and Erase on, and the dirt around it grows proper edges. Build walls and they join at the corners. The terrains come from the layer's tile set, set up in the TileSet panel. Set Cell puts down one particular tile when its edges do not need to join up.

> The tile set needs terrains set up, and its tiles painted with them, in the TileSet panel before anything can be painted.

See also

Set Cell 244, **Get Cell** 241, **Tile** 245, **Grid Map Cell** 242

Try it

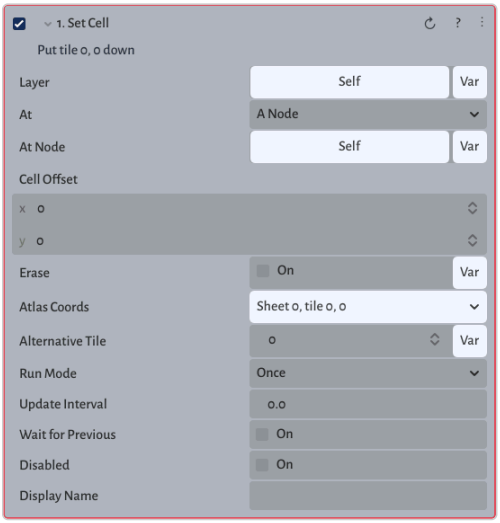
See the action work on its own with **Open Example** in the Add Action window, or open `examples/tilemap/paint_terrain/paint_terrain.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Layer	The TileMapLayer to paint, Godot's grid of tiles for one layer of the map. Left empty it is the node this Machine works on. If it is not a TileMapLayer, nothing is painted and a warning names the node it found. Var: a Node.	Self
At	Where to paint. A Node is the cell a node is in, such as under the player. The Mouse is the cell the mouse pointer is over, for building where the player clicks. A Cell is one cell you give by number. Cells is every cell in a list. Choices: A Node, The Mouse, A Cell, Cells.	A Node
At Node	Only when At is A Node. The node whose cell to paint. Left empty it is the node this Machine works on. Var: a Node.	Self
Cell Offset	Only when At is A Node or The Mouse. Paints the cell this many across and down from there. For example, 0, 1 is the cell just below, the one a character stands on.	0, 0
Terrain Set	Which terrain set of the layer's tile set the terrain is in, counted from 0. Most tile sets have one, numbered 0. Var: an int.	0
Terrain	Only when Erase is off. Which terrain to paint, counted from 0 in the order the TileSet panel lists them. A number the set does not have paints nothing, and a warning lists the set's terrains by name. Var: an int.	0
Erase	On, the cells are dug out, leaving gaps, and the tiles around them change to close their edges. Off, the default, the terrain is painted. Var: a bool.	off
Cell	Only when At is A Cell. The cell to paint, counted in whole tiles across and down from the layer's own starting corner, whatever size the tiles are.	—
Cells	Only when At is Cells. The cells to paint, each written as two numbers across and down, such as 3, 4. Read them from a list variable to paint a shape worked out while the game runs. An empty list paints nothing and a warning says Cells is empty.	—
Mode	Only when At is Cells. Connect joins every cell to all of its neighbours of the same terrain, as a pond or a field does. Path joins each cell only to the one before and the one after it in the list, as a road or a river does, so it does not merge with itself where it runs alongside.	—

Set Cell

Changes one square of a tile map while the game is running. It puts a tile down, or takes one away and leaves a gap. A tile map is Godot's grid of square tiles, and this writes to one layer of it.



Set Cell as it appears in the Actions tab.

When to use it

Dig a tunnel through earth, build a bridge across a gap a square at a time, open a hole under someone standing on a trapdoor, or let the player paint tiles wherever they click. Each run changes one square.

> While Layer does not point at a tile map layer nothing changes, and a warning names the node that is not one. A tile, or a version of one, missing from the layer's tile set is warned about too, and no tile is put down.

See also

Get Cell 241, **Tile** 245, **Grid Map Cell** 242, **Instantiate** 84

Try it

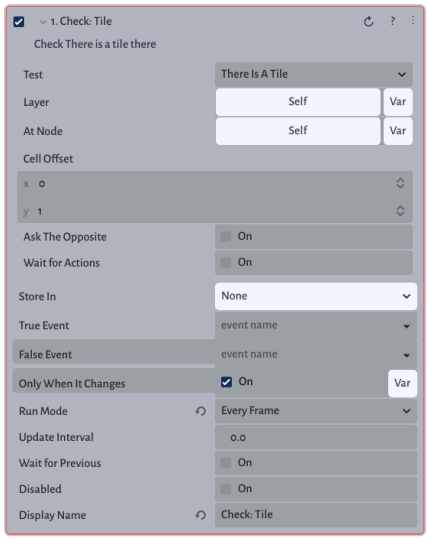
See the action work on its own with **Open Example** in the Add Action window, or open `examples/tilemap/set_cell/set_cell.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Layer	The tile map layer to change. A layer is one sheet of the grid. A game often has the ground on one layer and what is drawn in front on another, and it matters which one you dig into. Leave it empty for the thing this machine works on, but that only works when it is a tile map layer itself. Var: a Node.	Self
At	A Node, where it starts, works the square out from a node's place. The Mouse uses wherever the pointer is, for a building or digging game. Choices: A Node, The Mouse.	A Node
At Node	For A Node, the node whose place decides the square. Leave it empty for the thing this machine works on, such as the digger itself. Var: a Node.	Self
Cell Offset	How many squares across and down from there to change. 0, 0 is the square the node is in. 0, 1 is the square below it, the ground dug out from under someone, and 1, 0 is the one ahead.	0, 0
Erase	Ticked, the tile is taken away and the square left empty, for a hole in the floor, a dug-out tunnel or a broken wall. Unticked, as it starts, a tile is put down. Var: a bool.	off
Atlas Coords	Which tile to put down, chosen by its picture from the layer's tile set, so there is nothing to count or look up. It is not used while Erase is ticked.	0, 0
Alternative Tile	Which version of that tile to use, for a tile set whose author has set up flipped or turned versions of it. Leave it at 0, the plain tile, unless you have made some. A number with no version behind it puts nothing down, and the warning names the tile and the missing version. Var: an int.	0

Tile CHECK

Asks about the tile at a place on a tile map. Is there floor under my feet, is there a gap ahead, am I standing on ice? It can ask whether a tile is there at all, or what the tile set says about the one that is.



Tile as it appears in the Actions tab.

When to use it

A walking enemy turns round at the edge of a platform by asking about the cell one across and one down, with Ask The Opposite ticked, so the answer is yes when the floor runs out. Its Data Is reads the custom data set on the tiles in the tile set, such as a surface of "mud" that slows a character down.

> The second number of the cell offset counts downwards, so a larger number means lower on the screen, not higher.

See also

Get Cell 241, **Set Cell** 244, **Touching** 125, **Raycast** 116

Try it

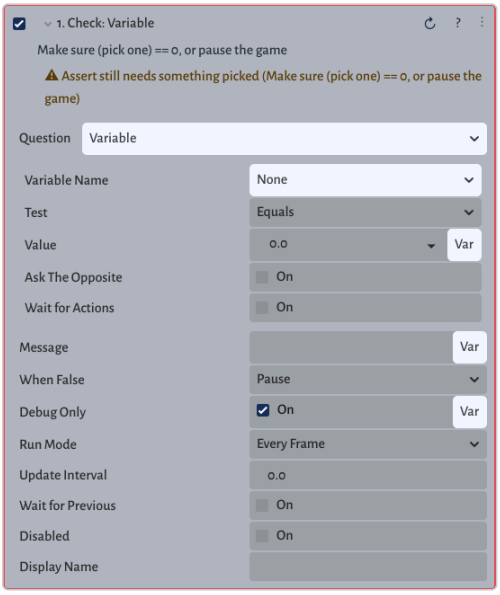
See the action work on its own with **Open Example** in the Add Action window, or open `examples/tilemap/tile/tile.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Test	There Is A Tile, the default, asks only whether a tile sits in that cell. Inverted, it finds a gap, such as a ledge, a doorway or a hole. Its Data Is asks whether the custom data on the tile matches the value below. Choices: There Is A Tile, Its Data Is.	There Is A Tile
Layer	The TileMapLayer to ask, that being the node holding one layer of the tile map. Leave it empty for the node the machine works on, which suits only a machine sitting on the tile map itself. For a character walking over it, point this at the layer. Otherwise nothing is found, the answer is no and a warning asks for the TileMapLayer. Var: a Node.	Self
At Node	The node whose place is looked at. Leave it empty for the node the machine works on, and the character asks about the ground under its own feet. Var: a Node.	Self
Cell Offset	How many cells across and down from there to look. (0, 1), the default, is the cell directly below, the ground you are standing on. (1, 1) is down and to the right, where the edge of a platform shows. (0, 0) is the cell you are in.	0, 1
Ask The Opposite	Turns the answer round into "there is no tile there", such as the edge of a platform, or a doorway to walk through. It turns round the true-or-false answer the Check stores as well, which swapping the True Event and False Event over cannot do.	off
Wait for Actions	Holds the question back until every other action in the state has finished, so it looks at the ground after the character has moved this frame. Left off, the default, it is asked as soon as the state runs.	off
Custom Data Layer	Only when Test is Its Data Is. The name of the custom data to read off the tile, spelled as the tile set calls it, such as "surface" or "damage". A name the tile set does not carry gives no, and a warning lists the custom data it does have.	—
Value	Only when Test is Its Data Is. What the tile's data has to be for the answer to be yes, such as "ice". A number typed as text still matches the same number on the tile, so "3" counts as a match for 3.	—
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Assert

Checks that something which must always be true is true. When it is not, it stops the game and writes a message in the Output panel saying what went wrong and in which state.



Assert as it appears in the Actions tab.

When to use it

Catches mistakes in your designs while you test. Health should never drop below 0, so an Assert asking that it is 0 or more pauses the game on the frame it drops. It asks the same kind of question a Check does, and comes ready to ask one about a variable. On Every Frame, its starting setting, it keeps asking while its state runs and never holds that state up. On Once it asks as the state starts.

> Pausing freezes everything except nodes set to keep running while paused, such as a pause menu. Stop the game from the editor once you have looked.

See also

Print 248, **Pause Game** 222, **Variable** 40, **Compare** 35

Try it

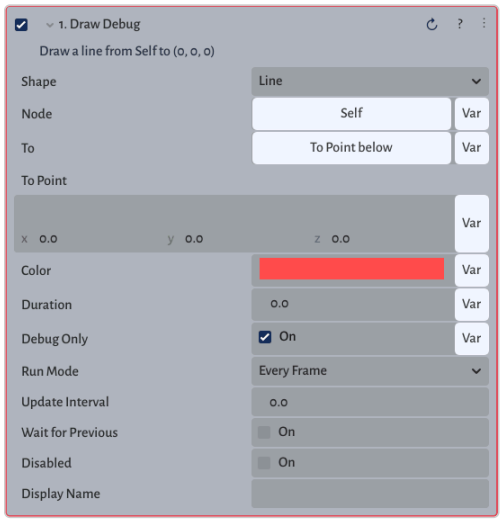
See the action work on its own with **Open Example** in the Add Action window, or open `examples/debug/assert/assert.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Message	What must be true, in your own words, such as The player always has a key here. It is what the Output panel shows when it is not true. Left empty, the question itself is shown. Var: a String.	empty
When False	What happens when it is not true. Pause, the default, freezes the game where it went wrong, and you can look round it in the editor before you stop it. Quit closes the game. Carry On only writes the message and lets the game go on. The message comes once each time it goes wrong, not every frame while it stays wrong. Choices: Pause, Quit, Carry On.	Pause
Debug Only	On, the default, it works while you test the game from the editor and in a debug export, and is skipped in a release export, the kind made for players with Export With Debug turned off. Turn it off only if you want the check in the finished game too. Var: a bool.	on

Draw Debug

Draws a line, a ray, a circle or a box in the running game, over everything else, so you can see what a design is doing while you test it. It works in 2D and 3D.



Draw Debug as it appears in the Actions tab.

When to use it

Shows what the game does not. For example, a Line from a guard to the player it is chasing makes the chase visible while you test. On Every Frame, its starting setting, the drawing follows the node while the state runs. On Once it is drawn where the node is then and stays for Duration seconds. Nothing can touch or hit a drawing. A Ray here only shows a direction, but Raycast is the action that looks along a line and finds what is there.

> On Once with Duration left at 0 the drawing lasts a single frame, too short to see.

See also

Raycast 116, **Print** 248, **Get Property** 80, **Set Property** 91

Try it

See the action work on its own with **Open Example** in the Add Action window, or open `examples/debug/draw_debug/draw_debug.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Shape	Line draws from the node to another node or to a place. Ray draws from the node the way it faces, the same way a RayCast2D or RayCast3D looks. Circle draws a circle round the node, or in 3D a ball made of three rings. Box draws a box round the node, turned the way the node is turned. Choices: Line, Ray, Circle, Box.	Line
Node	Where the drawing is. That is the start of a line or a ray, or the middle of a circle or a box. Left empty it is Self, the node this Machine works on. It must be a 2D node, a Control or a 3D node. Var: a Node.	Self
To	Only when Shape is Line. The node the line goes to, such as the player a guard is chasing, followed as it moves. Left empty, the line goes to To Point. Var: a Node.	empty
To Point	Only when Shape is Line and To is empty. The place the line goes to, two numbers in a 2D game and three in a 3D one. Press Var to use a place kept in a variable, such as where the mouse is. Var: a Vector2 or a Vector3, or a Node for where it stands.	0, 0, 0 in 2D: 0, 0
Color	The colour it is drawn in. A soft red to begin with. Var: a Color.	1, 0.3, 0.3, 1
Duration	How long each drawing stays, in seconds. At 0, the default, it stays for one frame, and on Every Frame that means for as long as the state runs. On Once, give it a number or the drawing is gone before you can see it. Var: a float.	0.0
Debug Only	On, the default, it draws while you test the game from the editor and in a debug export, and draws nothing in a release export, the kind made for players with Export With Debug turned off. Var: a bool.	on
Target Position	Only when Shape is Ray. Where the ray ends, measured from the node and turning with it, just like a RayCast's Target Position. On a 2D node it starts 50 pixels straight down.	—
Radius	Only when Shape is Circle. How far the circle reaches from the node, 0.5 in a 3D game and 32 pixels on a 2D node.	—
Size	Only when Shape is Box. How wide, high and deep the box is, with the node in its middle. It is 1 by 1 by 1 in a 3D game and 64 by 64 pixels on a 2D node.	—

Print

Prints a line in Godot's Output panel, at the bottom of the editor, while the game runs. It is the quickest way to see that a state was reached, and to flag something that has gone wrong.



Print as it appears in the Actions tab.

When to use it

For a machine that seems to do nothing, drop a Print into the state. If the line never appears, the state is never reached. Printing "ammo {ammo}" each time a state is entered shows whether the number is what you thought. Players never see any of it.

> On Every Frame it prints hundreds of lines a second and buries everything else. Leave it to run once unless you mean that.

See also

Set Text 213, **Variable** 40, **Compare** 35, **Send Event** 31

Try it

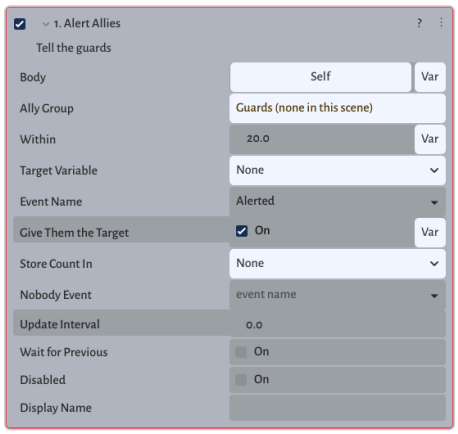
See the action work on its own with **Open Example** in the Add Action window, or open `examples/debug/print/print.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Text	The line to print, Hello to begin with. Write {target} for the name of the thing the machine works on, {state} for the name of the state, and {score}, or any variable's name in curly brackets, for its value. After a colon you can say how to show a number, such as {time:mm:ss} as minutes and seconds or {score:0000} padded with zeros. A name that belongs to no variable is printed just as you typed it. Var: a String.	Hello
Level	Message, as it comes, prints a plain line. Warning and Error print in colour and also show in Godot's Debugger, with the trail of where they came from. Use them for something that should never happen. Choices: Message, Warning, Error.	Message

Alert Allies

Tells every friend nearby that there is trouble. Each of them is handed the thing your character is dealing with, so they know who it is without seeing it, and is sent an event their template can answer.



Alert Allies as it appears in the Actions tab.

When to use it

One guard sees the player and the whole room comes, not one enemy at a time. A gunshot or a pulled alarm can bring friends running the same way.

> Handing the target over only works on friends that have a Perception node under them. The rest still get the event.

See also

Send Event 31, **Look For Target** 256, **Can See** 250, **Chase** 251

Try it

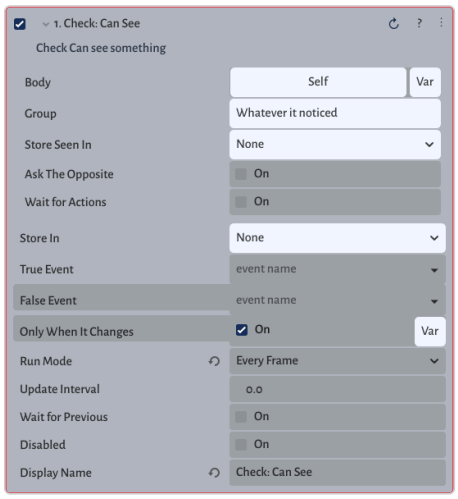
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ai/alert_allies/alert_allies.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Body	The character raising the alarm. Leave it empty for the thing the machine works on. Distances are measured from here, and it never alerts itself. Var: a Node.	Self
Ally Group	Its friends, the ones who hear the alarm, such as every copy of the guard. Press the box and pick it in the scene. It can be one thing by its name, Every Coin for every copy of the coin scene (including ones made while playing), or Everything under a node for a set of markers. NodeWorthy does the rest. Left empty, nobody is told.	guards
Within	How far the alarm carries, in metres in 3D or pixels in 2D. Friends further off than this hear nothing. 0 tells every one of them wherever they are in the level. That is right for a siren but much too far for a shout. Var: a float.	20.0 in 2D: 640 px
Target Variable	The name of a variable holding what to tell them about, in place of what this character's Perception node found. Leave it empty for a guard passing on what it has seen. Fill it in when the alarm is about somebody the machine was handed another way, such as whoever tripped an alarm or fired the shot.	empty
Event Name	The event each friend's machine receives, carrying the thing the alarm is about. Their templates catch it on a transition like any other event, so this name has to match the name on their transition. Left empty, nobody is sent anything and only Give Them the Target does any work.	Alerted
Give Them the Target	On, each friend's Perception node is handed the target, and an alerted friend starts chasing straight away without spotting it first. Turn it off for an alarm that only makes them come and look, such as a gunshot. Then they need no Perception node. Var: a bool.	on
Store Count In	Names a variable to put the number of friends told into, so the template can do something different when only one answered. Left empty the count is thrown away.	empty
Nobody Event	The event sent back to this machine when there was nobody in range to tell, so a lone guard can decide to run, not wait for help that is not coming. Empty sends nothing.	empty

Can See CHECK

Asks whether the character can see the thing it is dealing with right now. That means near enough, inside its view cone, and with nothing in the way. The Perception node on the character does the looking.



Can See as it appears in the Actions tab.

When to use it

A guard that fires only while it has eyes on the player stops shooting the instant a wall comes between them. Tick Ask The Opposite for "it has lost sight of it", when a chase turns into a hunt. Because the answer turns straight away, a player ducking behind a pillar for a fraction of a second breaks off the chase. Target Lost waits a while before it says anything.

> The character needs a Perception node on it. Without one the answer is always no, and a warning names the character.

See also

Target Lost 262, **In Range** 95, **Look For Target** 256, **Search Last Known** 259

Try it

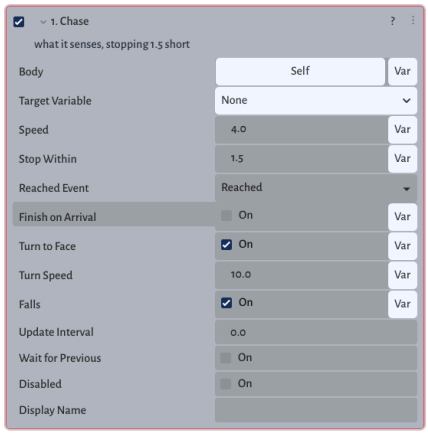
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ai/can_see/can_see.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Body	The character doing the looking. It must carry a Perception node, the node that does the seeing for it. Leave this empty for the node the machine works on. Var: a Node.	Self
Group	Only what is picked here counts, such as the player. Press the box and pick it in the scene. You can pick one thing by its name, Every Coin for every copy of the coin scene (including ones made while playing), or Everything under a node for a set of markers. NodeWorthy does the rest. Whatever It Noticed, the default, accepts whatever the senses have found.	empty
Store Seen In	Name one of the template's variables and the node that was seen goes into it, for an action further down the state to shoot at or walk to. Empty, the default, stores nothing, and the rest of the state cannot tell what was seen.	empty
Ask The Opposite	Turns the answer round. Ticked, the question becomes "it cannot see it", and a chase can break off on it. It turns round the true-or-false answer the Check stores as well, which swapping the True Event and False Event over cannot do.	off
Wait for Actions	Holds the question back until every other action in the state has finished, so a look that should happen after a turn or an animation is not asked on the first frame. Left off, the default, it is asked as soon as the state runs.	off
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Chase

Goes after whatever the character is dealing with and keeps after it as it moves. It stops a little short of it and sends an event on arrival, where an attack usually starts.



Chase as it appears in the Actions tab.

When to use it

A guard that has seen you and is coming. With a NavigationAgent node under the character it walks around walls. Without one it goes straight at its target, and that is fine for an open field.

> Losing what it chases never finishes it. It stands where it is and waits. Look For Target in the same state, or a transition on Target Lost, notices that the target has gone.

See also

Move To 97, **Keep Distance** 255, **Look For Target** 256, **Navigate To** 239

Try it

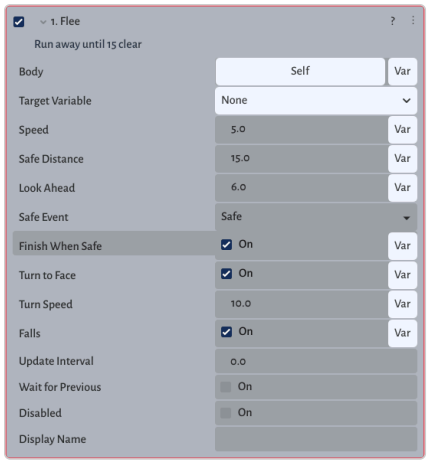
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ai/chase/chase.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Body	The character doing the chasing. Leave it empty for the thing the machine works on. Var: a Node.	Self
Target Variable	The name of a variable holding what to chase, in place of what the character's Perception node found. Leave it empty and it goes after what it can see, which needs a Perception node under the character. Fill it in when something else picked the quarry (an event that carried it in, or the variable Look For Target stores), or when the character has no Perception node.	empty
Speed	How fast it goes, in metres a second in 3D or pixels a second in 2D. The default 4 is a jog. Set it above the player's running speed and it always catches you, but a little below and you can get away. Var: a float.	4.0 in 2D: 128 px
Stop Within	How far short of its target it stops, so it does not shove into what it is chasing. This is also the distance at which Reached Event is sent. The default 1.5 is about arm's length for a human-sized 3D character. In a 2D game it starts at 48 pixels, and about the width of the two sprites is right there. Var: a float.	1.5 in 2D: 48 px
Reached Event	The event sent the first time it gets that close, carrying what it caught up with. Catch it on a transition into your attack state. Empty sends nothing and the chase carries on.	Reached
Finish on Arrival	Off, the default, it keeps chasing for as long as the state runs, closing again each time its target moves away. Turn it on when the state should end on arrival, so a transition on FINISHED carries the template on to the next state. Var: a bool.	off
Turn to Face	On, the character turns to face what it is chasing as it goes, so its sprite or model points the right way. Turn it off when something else decides which way it faces, such as a turret with its own aim. Var: a bool.	on
Turn Speed	How quickly it swings round. Higher is quicker, and 10 has it round in about a fifth of a second. Ten is brisk without looking like a spinning top, while two or three suits something heavy. It does nothing with Turn to Face off. Var: a float.	10.0
Falls	On, gravity pulls it down as it goes. Keep it on for a 3D game or a side-on 2D one. Turn it off for a top-down game, where down the screen is not down. A 2D body whose Motion Mode is Floating is seen from above and never falls, even with this ticked. Var: a bool.	on

Flee

Runs away from whatever the character is dealing with, heading for open ground on the far side, not stepping backwards. Once far enough off it stops, and sends Safe Event.



Flee as it appears in the Actions tab.

When to use it

Anything that should not fight, such as a shopkeeper whose health has run low. Paired with a chase in another state, the same animal runs when you turn on it and follows when you do not.

> With nothing to run from (no Perception node under the character, nothing it has noticed, or a Target Variable holding nothing), it counts itself safe straight away. Safe Event is sent, and with Finish When Safe on it finishes.

See also

Keep Distance 255, **Chase** 251, **Take Cover** 261, **Look For Target** 256

Try it

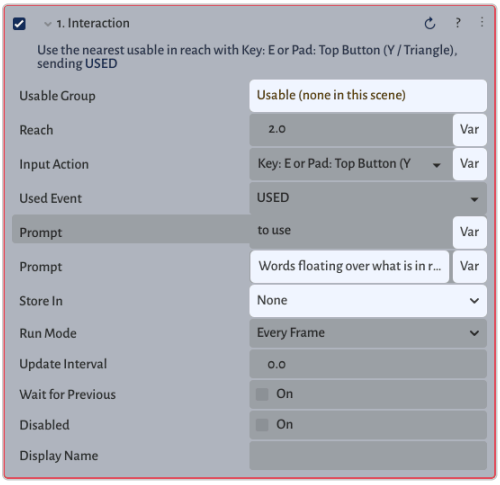
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ai/flee/flee.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Body	The character running away. Leave it empty for the thing the machine works on. Var: a Node.	Self
Target Variable	The name of a variable holding what to run from, in place of what the character's Perception node found. Leave it empty and it runs from what it senses. Fill it in for a villager fleeing whoever just hit it, which it may never have seen, or for anything with no Perception node. A name the template does not have gets a warning naming it, and so does a variable holding something that is not a node. An empty variable counts as nothing to run from.	empty
Speed	How fast it runs, in metres a second in 3D or pixels a second in 2D. The default 5 is quicker than a chase, so a fleeing villager is not caught straight away. Set it below the chaser's speed when you want it caught. Var: a float.	5.0 in 2D: 160 px
Safe Distance	How wide a gap counts as safe. Once it is this far away it stops running and Safe Event goes out. In a 3D game the gap is measured along the ground, so a danger up on a ledge above it is still close. Set it too small and it stops while the danger is still on top of it. Var: a float.	15.0 in 2D: 480 px
Look Ahead	How far ahead it aims each time it looks, on the line straight away from the danger. It is not a distance it travels, only the point it steers for. Larger values give a straighter run, smaller ones make it hug every move the danger makes. Var: a float.	6.0 in 2D: 192 px
Safe Event	The event sent once it is safely away, or once it has forgotten what it was running from. It is sent once each time it becomes safe, not over and over while it stands there. Empty sends nothing.	Safe
Finish When Safe	On, the default, it finishes as soon as it is safe, so a transition on FINISHED takes the character back to whatever it was doing before. Off, it never finishes by itself and the state needs another way out. Either way it keeps watching for as long as the state runs, so danger that comes back is run from again. Var: a bool.	on
Turn to Face	On, it turns to face the way it is running, so it looks like fleeing, not walking backwards. Turn it off if something else is deciding which way it faces. Var: a bool.	on
Turn Speed	How quickly it swings round. Higher is quicker, and 10 has it round in about a fifth of a second. Ten is brisk and looks frightened. Lower numbers make a heavier, slower turn. It does nothing with Turn to Face off. Var: a float.	10.0
Falls	On, gravity pulls it down as it runs, and a 3D game or a side-on 2D one needs this. Turn it off for a top-down game. A 2D body whose Motion Mode is Floating is seen from above and never falls, even with this ticked. Var: a bool.	on

Interaction

Lets the player use things by walking up to them and pressing a key. The nearest usable thing in reach shows a prompt such as "E to open". Pressing the key sends that thing's Machine an event, USED, and its states do the opening, switching or talking.



Interaction as it appears in the Actions tab.

When to use it

One copy on the player serves every chest, lever and door in the level. None of them needs a separate area, hint and key, only a way out on USED. Click To Move sends the same USED when something is clicked, so one chest works with both.

> A usable thing with no Machine cannot be told it was used. A warning names it and says to give it a Machine with a way out on the Used Event.

See also

Click To Move 126, **Event On Input** 141, **Event On Collision** 109, **Send Event** 31

Try it

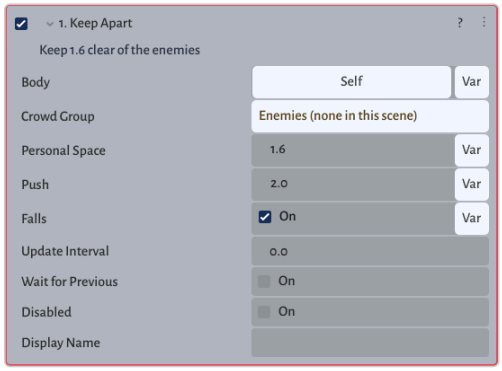
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ai/interaction/interaction.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Usable Group	What can be used, picked in the scene. It can be single chests and levers, or every copy of one. A thing counts only while the state its Machine is in has a way out on the Used Event. A chest that is already open has none, and stops offering itself.	usable
Reach	How near something has to be to be used, measured from the middle of the player to its middle, in metres in 3D and pixels in 2D. The nearest thing inside it gets the prompt. Too small and the player has to stand on top of things. Too big and chests across the room ask to be opened. Var: a float.	2.0 in 2D: 64 px
Input Action	The key or button that uses what is offered. It starts as E, or the top button on a gamepad. The prompt names the key the player last used, so it says the gamepad button when they are playing with a pad. Var: a String.	Key: E or Pad: Top Button (Y / Triangle)
Used Event	The event sent to the used thing's Machine, carrying the player who used it. Click To Move sends USED too when a chest is clicked. Change it only if your things listen for another name.	USED
Prompt	The words shown after the key, so "to use" shows as E to use. To give one thing different words, add a text variable called prompt to its Machine, such as "to open" on a chest or "to talk" on a shopkeeper. Var: a String.	to use
Prompt	A Label to write the prompt in, such as one fixed on the screen. It shows only while something is in reach. Left empty, the words float just above the thing itself, with no setting up. Var: a Node.	empty
Store In	A Node variable given the thing in reach, and emptied while there is none. The player's states can read it to check what they are standing beside. Left empty, nothing is kept.	empty

Keep Apart

Pushes the character gently away from anybody in a group who comes too close, and a crowd spreads out. It only adds a nudge on top of what the rest of the state decided, never fighting the action doing the moving, and it never holds a state up.



Keep Apart as it appears in the Actions tab.

When to use it

Put it in the same state as Chase, Keep Distance or Strafe Around whenever several characters go after one thing. Without it a pack piles into one spot and shoves itself to a standstill. A shoal of fish spreads out the same way.

> It only spaces characters out, so something else in the state still has to decide where they are going.

See also

Keep Distance 255, **Strafe Around** 260, **Chase** 251, **Navigate To** 239

Try it

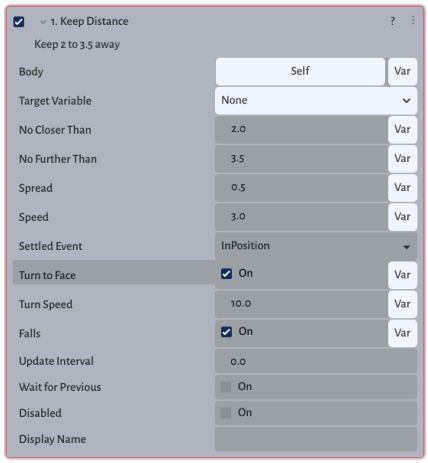
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ai/keep_apart/keep_apart.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Body	The character being kept clear. Leave it empty for the thing the machine works on. Var: a Node.	Self
Crowd Group	What it keeps out of the way of, usually the other enemies as Every Enemy. Press the box and pick it in the scene. It can be one thing by its name, Every Coin for every copy of the coin scene (including ones made while playing), or Everything under a node for a set of markers. NodeWorthy does the rest. They need no machines. Left empty, nothing is pushed.	enemies
Personal Space	How close is too close, in metres in 3D or pixels in 2D, roughly two of them standing side by side. The default 1.6 suits a human-sized 3D character. In a 2D game it starts at 51.2 pixels. 0 turns the push off. Var: a float.	1.6 in 2D: 51.2 px
Push	How hard it is pushed away. The push is firmest when they are on top of each other and fades to 0 at the edge of the personal space. Keep it below the speed the character walks at, or the push wins and nobody gets anywhere. Var: a float.	2.0 in 2D: 64 px
Falls	On, the push is kept level with the ground, so a crowded character is not shoved up into the air or down through the floor. Turn it off for a top-down game. A 2D body whose Motion Mode is Floating is seen from above and never falls, even with this ticked. Var: a bool.	on

Keep Distance

Holds a sensible gap. It closes in when it is too far away, backs off when it is too near, and stands its ground in between. It sends an event the first time it settles, the usual cue to attack, and never holds a state up.



Keep Distance as it appears in the Actions tab.

When to use it

An archer, or a wary animal that will not come within reach. Pair it with Strafe Around. That one decides where round, and this one how far away.

> It stands still while there is nothing to keep away from. Look For Target, or a transition on Target Lost, takes the character out of the state once its target has gone.

See also

Strafe Around 260, **Chase** 251, **Keep Apart** 254, **Can See** 250

Try it

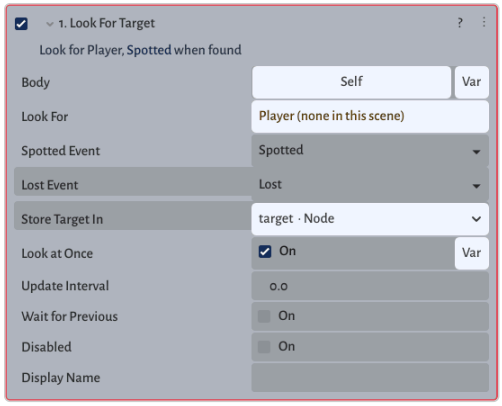
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ai/keep_distance/keep_distance.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Body	The character. Leave it empty for the thing the machine works on. Var: a Node.	Self
Target Variable	The name of a variable holding what to keep away from, in place of what the character's Perception node found. Leave it empty and it holds its distance from what it senses. Fill it in when the template already knows who the fight is with, or when the character has no Perception node.	empty
No Closer Than	Any nearer than this and it backs off. Make it wider than the reach of whatever it is facing, so a swordsman cannot walk straight up to your archer. Both distances are measured along the floor, the way it walks, so something standing up on a ledge is not further off for being higher. Var: a float.	2.0 in 2D: 64 px
No Further Than	Any further than this and it closes in. Between the two numbers it stands its ground. Leave a comfortable margin above No Closer Than, or it fidgets in and out without ever settling. Var: a float.	3.5 in 2D: 112 px
Spread	A random amount, up to this, is added to both distances when the state is entered, different for each character. A group then spreads over several rings, not jostling for one spot. 0 puts every one of them at the same distance, so they line up like a parade and jam. Var: a float.	0.5 in 2D: 16 px
Speed	How fast it moves, closing in or backing off. Keep it below the speed it chases at, because shuffling into position is not a sprint. Var: a float.	3.0 in 2D: 96 px
Settled Event	The event sent the first time it settles into the gap and stops moving, carrying what it is keeping away from. This is usually the cue to start shooting. It is sent again each time it settles after having to move. Empty sends nothing.	InPosition
Turn to Face	On, it keeps facing what it is keeping away from even while walking backwards, so it never turns its back on the thing it is watching. Turn it off if something else is deciding which way it faces. Var: a bool.	on
Turn Speed	How quickly it swings round to face its target. Higher is quicker, and 10 has it round in about a fifth of a second. Ten keeps it looking alert. It does nothing with Turn to Face off. Var: a float.	10.0
Falls	On, gravity pulls it down as it moves, as a 3D game or a side-on 2D one needs. Turn it off for a top-down game. A 2D body whose Motion Mode is Floating is seen from above and never falls, even with this ticked. Var: a bool.	on

Look For Target

Keeps watch for what you pick, such as the player, while the state runs. It sends one event when the character sees it and another when it gives up on it. The Perception node under the character does the looking, and this action passes what it finds to your template.



Look For Target as it appears in the Actions tab.

When to use it

Put it in a state doing something else as well. A guard patrols and watches, for example. It never holds a state up, so it sits beside Patrol or Wander. Spotted leads to the chase state, Lost to a search. A one-off question, whether something is in view right now, is the Can See check's job.

> It looks with the Perception node under the character, which holds how far it sees, how wide and how long it remembers. Picking what to look for adds one if the character has none.

See also

Can See 250, **Target Lost** 262, **Chase** 251, **Search Last Known** 259

Try it

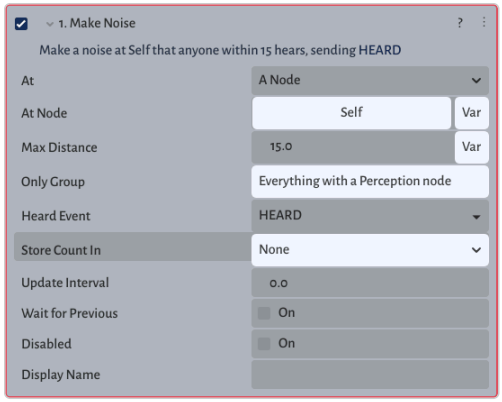
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ai/look_for_target/look_for_target.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Body	The character doing the looking. Leave it empty for the thing the machine works on. It needs a Perception node under it. That is the node holding how far it can see, how wide its view is and how long it remembers. Var: a Node.	Self
Look For	What it looks for, such as the player. Press the box and pick it in the scene. It can be one thing by its name, Every Coin for every copy of the coin scene (including ones made while playing), or Everything under a node for a set of markers. NodeWorthy does the rest. Picking something also gives the character a Perception node to look with, if it has none. It starts on the player. While nothing in the scene is the player, the box says so in warning colour and the problems button tells you.	player
Spotted Event	The event sent as soon as something comes into view, or an ally hands it a target, or what it was dealing with comes back into sight. It carries what was seen. Something it only remembers, out of sight, is not spotted again, so an enemy that has just lost you does not go straight back to chasing from memory. Empty sends nothing.	Spotted
Lost Event	The event sent when what it was watching has been out of sight long enough to give up on, which the Perception node's Remember For setting decides. Catch it to send the character off searching. Empty sends nothing.	Lost
Store Target In	The variable what it noticed is kept in, so the rest of the template can chase it, shoot at it or talk to it. It is emptied again when the target is given up. Leave it as target unless your template already uses another name. An empty box keeps nothing.	target
Look at Once	On, it takes one look as the state starts, without waiting for the Perception node's next look. A guard that begins the game staring at you then reacts with no frame's delay. Off costs a fraction less. Var: a bool.	on

Make Noise

Makes a noise for characters to hear. A character with a Perception node hears it when the noise is inside both Max Distance and that Perception's Hearing Range, through walls and whichever way it faces. Its Machine is sent HEARD, carrying the place the noise came from.



Make Noise as it appears in the Actions tab.

When to use it

A gunshot brings guards running, and a bottle thrown against a far wall lures one away from the door it guards. A guard that hears a noise while it is dealing with nobody remembers the place, and Search Last Known in its next state walks there and looks round.

> Only characters with a Perception node under them hear anything. A Perception with Hearing Range 0 is deaf, and a warning names the characters that would have heard but for that. The noise makes no sound, so play one beside it with Play Sound At Position.

See also

Search Last Known 259, **Alert Allies** 249, **Look For Target** 256, **Play Sound At Position** 176

Try it

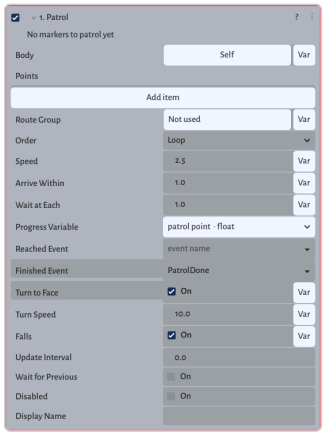
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ai/make_noise/make_noise.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
At	Where the noise is. A Node makes it where a node is, such as the gun going off or the bottle where it lands. A Place makes it at a place you type in or read from a variable, such as where the mouse was clicked. Choices: A Node, A Place.	A Node
At Node	Only when At is A Node. The node the noise comes from. Left empty it is the node this Machine works on. If there is no such node, nobody hears anything and a warning says there was nowhere to make the noise. Var: a Node.	Self
Max Distance	How far the noise carries, in metres in 3D and pixels in 2D, measured the same way as a sound player's Max Distance. A character with a Perception node this close hears it if the noise is also within that Perception's Hearing Range. A gunshot carries a long way and a footstep only a little. At 0 nobody hears it, and a warning points at Max Distance. Var: a float.	15.0 in 2D: 480 px
Only Group	Only these hear it, picked in the scene, such as every copy of the guard. Left empty, every character with a Perception node can hear it, whatever group it is in.	empty
Heard Event	The event each character that hears the noise is sent, carrying the place it came from. A way out on HEARD can put that place in a variable. Whoever makes the noise does not hear it themselves.	HEARD
Store Count In	A number variable given how many characters heard the noise. At 0, the next state knows nobody was near enough. Left empty, nothing is kept.	empty
Place	Only when At is A Place. Where the noise is in the world, as two numbers in a 2D game and three in a 3D one. Read it from a variable to make the noise where something happened, such as where a thrown thing landed.	—

Patrol

Walks a round of markers, waiting at each one. It can go round and round, there and back, or once through. Put empty nodes in the level where you want the character to go and list them here.



When to use it

A guard walking a beat, or a lift calling at floors. Where it had got to is kept in a variable, not in the action, so a guard that breaks off to chase somebody comes back to the right point and does not start the round again.

> If a marker of the round has gone missing, renamed or deleted, the patrol stops there and a warning gives that marker's number.

See also

Wander 263, **Follow Path** 94, **Move To** 97, **Look For Target** 256

Try it

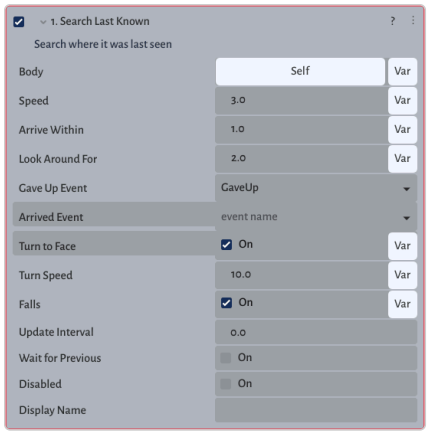
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ai/patrol/patrol.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Body	The character walking the round. Leave it empty for the thing the machine works on. Var: a Node.	Self
Points	The markers to walk between, in the order you list them. Any node will do, such as an empty Node2D or Node3D dropped in the level wherever you want the character to go. Leave the list empty to use Route Group. With neither filled in there is no round to walk. The action finishes straight away, and a warning asks for markers.	empty
Route Group	Instead of listing markers one by one, walk past every one picked here, in the order they sit in the scene. Put a route's markers under one node and pick Everything under it. It is ignored whenever Points has anything in it. Var: a String.	empty
Order	What happens at the end of the list. Loop goes back to the first marker and keeps going for ever. Ping Pong turns round at the end and walks the list backwards. Once stops at the last marker, sends Finished Event and finishes, so a transition on FINISHED can carry on. Choices: Loop, Ping Pong, Once.	Loop
Speed	How fast it walks. The default 2.5 is a patrol pace, slower than a chase, and the change of pace helps a chase read as one. Var: a float.	2.5 in 2D: 80 px
Arrive Within	How close to a marker counts as having reached it. Too small and a wide character never quite arrives and stands there pushing at the marker. The default 1 suits a human-sized 3D character. In a 2D game it starts at 32 pixels. Raise it only for a wide sprite. Var: a float.	1.0 in 2D: 32 px
Wait at Each	Seconds to stand at each marker before moving on. A second or two makes a guard look like it is checking the room. 0 walks the round without ever pausing. Var: a float.	1.0
Progress Variable	The variable that remembers which marker it is heading for, letting a guard that broke off to chase somebody pick up where it left off. Each character keeps separate variables, so several guards may share one template and still be at different points. Under Ping Pong the direction it is walking is kept beside it, in a variable of the same name with _step on the end.	patrol point
Reached Event	The event sent each time it reaches a marker, carrying the marker it reached. Use it to play a looking-around animation, open a door or turn a light on. Empty, the default, sends nothing.	empty
Finished Event	The event sent when a Once round is finished. It is only ever sent under Once, because a round that goes on for ever never reaches an end to announce.	PatrolDone
Turn to Face	On, the character turns to face the way it is walking. Turn it off when something else decides which way it faces, such as a lift that must stay square. Var: a bool.	on
Turn Speed	How quickly it swings round. Higher is quicker, and 10 has it round in about a fifth of a second. Ten looks natural for somebody on patrol. Lower it for something heavy. It does nothing with Turn to Face off. Var: a float.	10.0
Falls	On, gravity pulls it down as it walks. Leave it on for a 3D game or a side-on 2D one, and turn it off for a top-down game. A 2D body whose Motion Mode is Floating is seen from above and never falls, even with this ticked. Var: a bool.	on

Search Last Known

Goes to the place where the character last saw the thing it was chasing, stands and looks around, and then gives up. An enemy that searches where you were seems to be thinking.



Search Last Known as it appears in the Actions tab.

When to use it

The state a Lost or Target Lost transition leads to, between giving up the chase and going back on patrol. With Look For Target in the same state, seeing you again on the way turns the search straight back into a chase.

> If it can hardly move for a second, because somebody is standing on the spot or something is in the way, it counts that as arriving and looks around from where it got to, not pushing for ever.

See also

Look For Target 256, **Target Lost** 262, **Chase** 251, **Patrol** 258

Try it

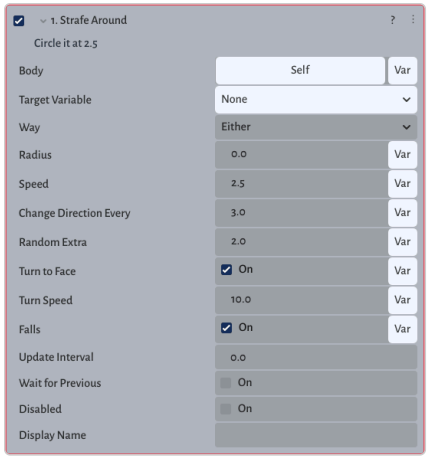
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ai/search_last_known/search_last_known.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Body	The character doing the searching. Leave it empty for the thing the machine works on. The place it searches comes from the Perception node under it, so it needs one. Without one there is no last known place, it gives up straight away, and a warning says the character has no Perception node. With a Perception node but nothing remembered it also gives up straight away. Var: a Node.	Self
Speed	How fast it searches, usually slower than a chase. A hunting pace, not a run, reads as an enemy that has lost you. Var: a float.	3.0 in 2D: 96 px
Arrive Within	How close to the remembered place counts as having got there. The default 1 suits a human-sized 3D character. In a 2D game it starts at 32 pixels. Var: a float.	1.0 in 2D: 32 px
Look Around For	Seconds it stands at the spot before giving up, turning one way and the other as it looks. Two seconds looks deliberate, and longer gives you more time to slip away behind it. 0 gives up as soon as it arrives. Var: a float.	2.0
Gave Up Event	The event sent once, when it has arrived, looked around and found nothing, or when it had nowhere to look in the first place. Catch it to send the character back to its patrol. Empty sends nothing, though the action still finishes.	GaveUp
Arrived Event	The event sent as it reaches the place, before it starts looking around. Use it for a line of dialogue or a torch being raised. Empty, the default, sends nothing.	empty
Turn to Face	On, it faces the way it is going, and sweeps its view from side to side while looking around. Off, it stands and stares straight ahead for the whole search. Var: a bool.	on
Turn Speed	How quickly it swings round while travelling. Higher is quicker, and 10 has it round in about a fifth of a second. Ten suits somebody moving carefully. It does nothing with Turn to Face off. Var: a float.	10.0
Falls	On, gravity pulls it down on the way, for a 3D game or a side-on 2D one. Turn it off for a top-down game. A 2D body whose Motion Mode is Floating is seen from above and never falls, even with this ticked. Var: a bool.	on

Strafe Around

Circles whatever the character is dealing with, turning round and going the other way every few seconds. It never holds a state up.



Strafe Around as it appears in the Actions tab.

When to use it

Two enemies circling opposite ways look like a pack hunting for an opening. Put it beside Keep Distance, which decides how far away while this decides where round, or give it a Radius and it holds the distance itself. It only sets how fast it goes round, and never undoes another movement action in the state.

> It does nothing while there is nothing to circle. Something else in the state has to notice that its target has gone.

See also

Keep Distance 255, **Keep Apart** 254, **Chase** 251, **Look At** 96

Try it

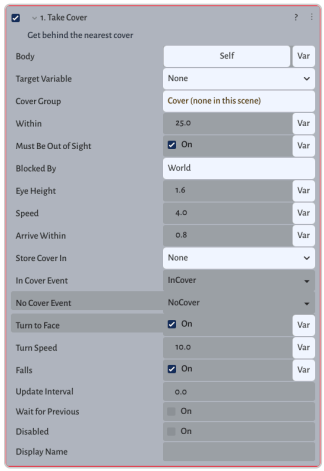
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ai/strafe_around/strafe_around.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Body	The character. Leave it empty for the thing the machine works on. Var: a Node.	Self
Target Variable	The name of a variable holding what to circle, in place of what the character's Perception node found. Leave it empty and it circles what it senses. Fill it in to circle something it cannot see, or to match the variable Keep Distance is using so the two agree on who the fight is with.	empty
Way	Which way round it goes. Either picks a direction at random when the state is entered, so several of them do not all circle the same way. Clockwise and Anticlockwise pin it to one direction, so you can set a pair to circle opposite ways on purpose. Choices: Either, Clockwise, Anticlockwise.	Either
Radius	How far out it circles. 0, the default, keeps whatever distance it is at and leaves the distance to the rest of the state, so it goes well beside Keep Distance. Any other number and it also pulls itself in or out to that distance as it circles. Var: a float.	0.0
Speed	How fast it sidesteps around. The default 2.5 is a wary shuffle. Much higher and it looks like it is skating. Var: a float.	2.5 in 2D: 80 px
Change Direction Every	Seconds before it turns round and circles the other way, so it does not orbit like a moon. Three seconds keeps it restless. 0 keeps it going one way for as long as the state runs. Var: a float.	3.0
Random Extra	Up to this many extra seconds, picked at random, are added to each wait before it turns round, so a pack does not all change direction together. 0 makes them turn in step, as if rehearsed. Var: a float.	2.0
Turn to Face	On, it keeps facing what it is circling while it sidesteps, and that is what makes it look like circling and not running past. Turn it off if something else is deciding which way it faces. Var: a bool.	on
Turn Speed	How quickly it swings round to face the middle. Higher is quicker, and 10 has it round in about a fifth of a second. Ten keeps it looking at its target throughout. It does nothing with Turn to Face off. Var: a float.	10.0
Falls	On, gravity pulls it down as it circles. Keep it on for a 3D game or a side-on 2D one. Turn it off for a top-down game. A 2D body whose Motion Mode is Floating is seen from above and never falls, even with this ticked. Var: a bool.	on

Take Cover

Gets behind something. From the places you have marked as cover it picks the nearest one within reach that really hides it, and goes there. It sends one event when it arrives and another when there is nowhere to hide.



When to use it

A soldier in a firefight ducking behind a crate. Marking cover needs no special node. Put an empty node beside a crate or a wall, and pick those nodes in Cover Group.

> It chooses its hiding place once, as the state starts, and keeps to it on the way. To choose afresh once the danger has moved, send the character into this state again.

See also

Flee 252, **Keep Distance** 255, **Can See** 250, **Move To** 97

Try it

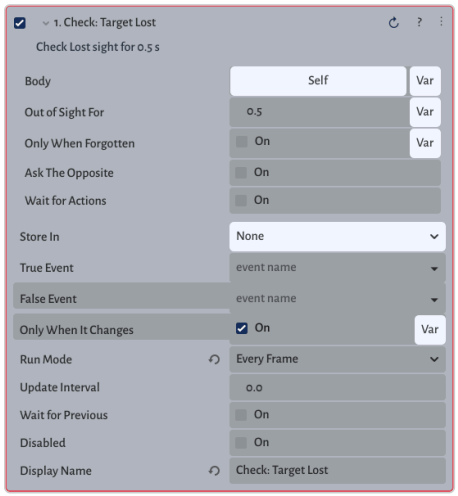
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ai/take_cover/take_cover.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Body	The character taking cover. Leave it empty for the thing the machine works on. Var: a Node.	Self
Target Variable	The name of a variable holding what to hide from, in place of what the character's Perception node found. Leave it empty and it hides from what it senses. With nothing to hide from at all it goes to the nearest marker, as there is nobody for the sight check to test against.	empty
Cover Group	The places it can hide behind. Drop an empty node wherever you want a hiding place, keep them under one node such as CoverPoints, then press this box and pick Everything under CoverPoints. With nothing picked no cover is ever found.	cover
Within	Only markers this close are considered, so it does not sprint the length of the level to hide. 0 considers every marker in the game however far away, which on a large level means long runs across open ground. Var: a float.	25.0 in 2D: 800 px
Must Be Out of Sight	On, a marker only counts when the thing it is hiding from cannot see that spot. The check is the same one a Perception node uses for sight, so another character is never mistaken for cover. Off takes the markers as they are. That is quicker, and you decide by hand what counts. Var: a bool.	on
Blocked By	Which layers count as something solid to hide behind. It starts on World, where the walls, the ground and anything you add yourself stand. Press the box to tick layers by name. If your walls are on a layer that is not ticked here, nothing will ever look like cover. It is only used with Must Be Out of Sight on.	World
Eye Height	How far above the ground the sight check is made, so cover is judged at eye level. Judged at the floor, a low crate would seem to hide a standing figure. Set it to roughly the height of your character's head. It is 3D only and is ignored in a 2D game. Var: a float.	1.6 in 2D: 51.2 px
Speed	How fast it moves to cover. Make it a run, because strolling to cover under fire looks wrong. Var: a float.	4.0 in 2D: 128 px
Arrive Within	How close to the marker counts as being there and safely in cover. The default 0.8 suits a human-sized 3D character. In a 2D game it starts at 25.6 pixels. Var: a float.	0.8 in 2D: 25.6 px
Store Cover In	Names a variable to keep the marker it chose in, so the rest of the template can use it (to lean out from and come back to, say). Empty keeps nothing.	empty
In Cover Event	The event sent when it reaches its cover, carrying the marker it chose. Catch it to start reloading, or to wait a moment before stepping out. Empty sends nothing.	InCover
No Cover Event	The event sent when there is nowhere to hide, so the template can charge or run, not stand in the open. It goes out as the state starts, before the character moves at all. Empty sends nothing and the action finishes.	NoCover
Turn to Face	On, the character turns to face the way it is going. Turn it off if something else is deciding which way it faces, such as an aim held on the danger while it backs into cover. Var: a bool.	on
Turn Speed	How quickly it swings round. Higher is quicker, and 10 has it round in about a fifth of a second. Ten suits somebody moving under fire. It does nothing with Turn to Face off. Var: a float.	10.0
Falls	On, gravity pulls it down on the way. A 3D game or a side-on 2D one needs it. Turn it off for a top-down game. A 2D body whose Motion Mode is Floating is seen from above and never falls, even with this ticked. Var: a bool.	on

Target Lost CHECK

Asks whether the character had something in sight and has not seen it for a moment. A chase turns into a search on it.



Target Lost as it appears in the Actions tab.

When to use it

On a chasing state, a guard starts hunting when the player really has got away, and not every time they step behind a pillar. Tick Only When Forgotten for the later moment, when the character no longer remembers there was anybody and can go back on patrol.

> Without a Perception node on the character the answer is always no and nothing ever gives up. A warning names the character.

See also

Can See 250, **Search Last Known** 259, **Look For Target** 256, **In Range** 95

Try it

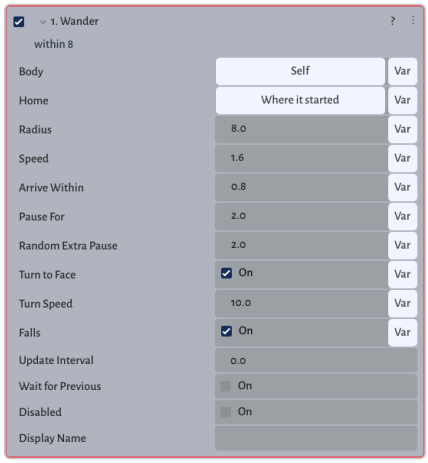
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ai/target_lost/target_lost.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Body	The character being asked about. It carries the Perception node, the node that does its seeing. Leave it empty for the node the machine works on. Var: a Node.	Self
Out of Sight For	How long the thing has to have been out of sight, in seconds, before the answer turns to yes. The default 0.5 lets a chase last past a pillar or a doorway and still feels alert. At 0 it is as twitchy as Can See. At or above the Perception's Remember For it can never be true, and a warning gives both numbers. Var: a float.	0.5
Only When Forgotten	Ticked, it asks a different and much later question. Has the character given up altogether and forgotten? The Perception node's Remember For setting decides when that happens, and Out of Sight For is not used. Leave it off, the default, for "it has lost sight but is still looking". Var: a bool.	off
Ask The Opposite	Turns the answer round into "it still has it", one way to keep a chase going. It turns round the true-or-false answer the Check stores as well, which swapping the True Event and False Event over cannot do.	off
Wait for Actions	Holds the question back until every other action in the state has finished, so it is asked after the chase has moved the character this frame. Left off, the default, it is asked as soon as the state runs.	off
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Wander

Ambles about near a spot, with a pause between strolls, and a village looks lived in. Where it goes depends on the level. With a navigation mesh (the walkable surface laid over a level) it only picks places it can really get to. Without one it picks any point within the circle and walks straight at it. A side-on 2D character, which only walks left and right, picks a place along the ground either side.



Wander as it appears in the Actions tab.

When to use it

Villagers in a market, sheep in a field. With Look For Target in the same state, the idler notices you and gives chase.

> It never finishes by itself, so no FINISHED transition ever follows it. Leave the state on an event, such as the one Look For Target sends when it sees somebody.

See also

Patrol 258, **Move To** 97, **Look For Target** 256, **Get Navigation Point** 237

Try it

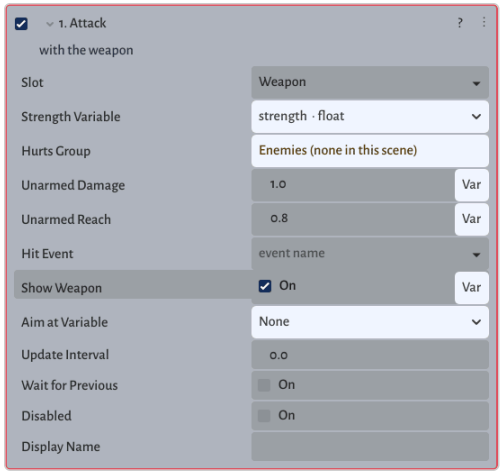
See the action work on its own with **Open Example** in the Add Action window, or open `examples/ai/wander/wander.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Body	The character wandering. Leave it empty for the thing the machine works on. Var: a Node.	Self
Home	The node it stays near. Leave it empty and it wanders around wherever it was standing when the state was entered. Name a node that is not in the scene and there is nowhere to wander around, so the character stands still, and a warning says why. The spot is taken once as the state starts, so moving that node afterwards does not move the wandering. Var: a Node.	empty
Radius	How far from home it may stray, in metres in 3D or pixels in 2D. The default 8 is about a room. In a 2D game it starts at 256 pixels. Var: a float.	8.0 in 2D: 256 px
Speed	How fast it ambles. The default 1.6 is a slow walk. A wanderer moving at chase speed looks panicked, not idle. Var: a float.	1.6 in 2D: 51.2 px
Arrive Within	How close counts as having arrived, after which it stands for a pause. Too small and it fidgets on the spot trying to land on the point. The default 0.8 suits a human-sized 3D character. In a 2D game it starts at 25.6 pixels. Var: a float.	0.8 in 2D: 25.6 px
Pause For	Seconds it stands still between strolls. It waits this long before the first stroll too, so a street full of them does not set off together as the game starts. 0 keeps it walking with no rest. Var: a float.	2.0
Random Extra Pause	Up to this many extra seconds of standing still, picked at random after each stroll, so several of them do not move in step. 0 makes them all set off together, like clockwork. Var: a float.	2.0
Turn to Face	On, the character turns to face the way it is walking. Turn it off if something else is deciding which way it faces. Var: a bool.	on
Turn Speed	How quickly it swings round. Higher is quicker, and 10 has it round in about a fifth of a second. Ten looks natural, and two or three makes a slow, ambling turn that suits a grazing animal. It does nothing with Turn to Face off. Var: a float.	10.0
Falls	On, gravity pulls it down as it walks. That is right for a 3D game or a side-on 2D one. Turn it off for a top-down game. A 2D body whose Motion Mode is Floating is seen from above and never falls, even with this ticked. Var: a bool.	on

Attack

Swings the weapon the character wears, as far in front of them as it reaches and the way they are heading, for the weapon's swing time. Anything it touches that is in the group it hurts and has a Hurt Box takes the weapon's damage plus the character's strength.



Attack as it appears in the Actions tab.

When to use it

Put it in a Swinging state that the player's attack button leads to, with FINISHED leading back. The state waits until the swing is over. A weapon is an item with Damage set, worn in the Weapon slot with Equip Item. A stronger sword or more strength means fewer hits to beat an enemy. With nothing worn it is a punch.

> Each thing is hit once a swing, however long the swing lasts. A top-down 2D character swings the way it last walked. A 3D one, and a side-on 2D one that turns, swing the way they face. The target's Hurt Box takes off its defence, knocks it back as hard as the weapon's Knockback says, and keeps it safe for a moment.

See also

Equip Item 289, **Hit Box** 316, **Hurt Box** 317, **Event On Input** 141

Try it

See the action work on its own with **Open Example** in the Add Action window, or open `examples/combat/attack/attack.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Slot	The equipment slot the weapon is worn in.	Weapon
Strength Variable	A number variable of this machine added to the weapon's damage, as a strength that grows with levels and gear. Left empty, the weapon's damage stands alone. A name this machine has no variable for adds 0, and a warning asks whether it is spelt as in the Variables tab.	strength
Hurts Group	What it hurts. For the player that is every copy of the enemy, and for an enemy that swings it is the player. Press the box and pick it in the scene. You can pick one thing by its name, Every Coin for every copy of the coin scene (including ones made while playing), or Everything under a node for a set of markers. NodeWorthy does the rest. Left empty, it hurts anything with a Hurt Box but the character swinging.	enemies
Unarmed Damage	The damage of a punch, with no weapon worn. The strength is added to it too. Var: a float.	1.0
Unarmed Reach	How far a punch reaches in front when no weapon is worn. It is in metres in 3D, and in pixels in 2D, where it starts at 25.6. Var: a float.	0.8 in 2D: 25.6 px
Hit Event	An event this machine is sent when the swing lands on something, carrying what it hit. Empty sends nothing.	empty
Show Weapon	In 2D, show the weapon's picture swung out in front for the swing, so the player sees it. Var: a bool.	on
Aim at Variable	Swings at whoever this variable names, such as the enemy being fought, when they are within reach, and turns to face them. Otherwise it swings the way the character last moved. Pair it with Click To Move's Store Target In. Left empty, it always swings the way the character last moved.	empty

Bullet Pattern

Fires a pattern of bullets, such as one aimed shot, a shotgun fan, a ring, a spiral, or a wall of shots with a gap to slip through. The bullets are plain scenes with no machines in them, all moved and looked after together, so hundreds can fly at a time. It works in 2D and in 3D.



When to use it

Anything that shoots, from a turret to the player's own gun. The bullets fly, hit, bounce, pierce and disappear by themselves, with no machine built for each one. In a 3D game the pattern lies flat, for a game seen from above. The bullets fly level at the height of From, and angles and directions read as a camera looking down on the floor sees them. A rocket that needs its own states is a different thing. That is a scene with a Machine in it, brought in by Instantiate.

> Keep Gap smaller than Count. A gap as wide as the shot would leave every bullet out, so one fewer is left out and a warning gives the numbers. Bullets fly through tiles that are only stood on from above, the jump-through platforms a character jumps up through, so a hero on a bridge can fire down through it. The first solid tile stops them as usual.

See also

Instantiate 84, **Translate** 105, **Explosion** 110, **Raycast** 116

Try it

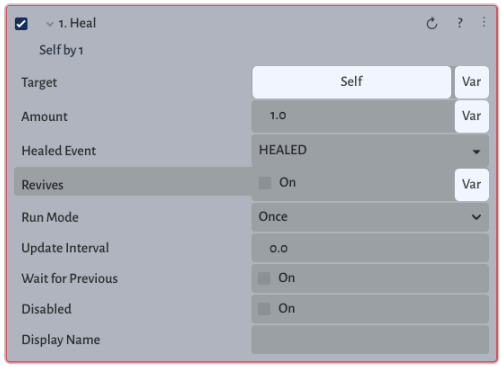
See the action work on its own with **Open Example** in the Add Action window, or open `examples/combat/bullet_pattern/bullet_pattern.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Scene	The bullet itself is a small scene built on an area (a shape that notices what passes through it), with a collision shape and a picture or a model. Use an Area2D in a 2D game and an Area3D in a 3D one. A 3D bullet's model should point along its own forward, the way Godot's models face. Its collision layer says what the bullet is, and its mask what it notices. That means the hurt boxes of the things it can hit, and the walls. With nothing picked here it fires nothing, and a warning asks for a bullet scene beside Scene.	empty
From	The node the bullets start from, such as a marker at the end of a barrel. It also decides which way they go when Aim is The Way It Faces. Leave it empty for Self, the node this Machine works on. Var: a Node.	Self
Aim	Which way the middle of the pattern points. The Way It Faces uses the way From is turned, so a barrel turned by Look At aims the shot. At A Node fires straight at another node, wherever it stands as the shot goes off. Along A Direction uses a direction you type in or read from a variable. An Angle uses degrees, where 0 is right, 90 is down, 180 is left and 270 is up. In 3D, right is east and down is south, as seen from above. Choices: The Way It Faces, At A Node, Along A Direction, An Angle.	The Way It Faces
Count	How many bullets go out in one shot. 1 is a single aimed shot. Raise it for a fan or a ring, and the Spread and Gap boxes appear once it is above 1. 0 fires nothing, and a warning asks for 1 or more. Var: an int.	1
Wobble	Each bullet's direction is nudged at random by up to this many degrees either side, so a gun feels held in a hand, not bolted down. 0 fires dead straight. Two or three degrees feels loose, and twenty is a spray. Var: a float.	0.0
Turn Each Shot	Turn the whole pattern this many more degrees each time it fires. 0 fires the same pattern every time. With Every Seconds set, a ring turned a few degrees a shot becomes a spiral. Var: a float.	0.0
Speed	How fast the bullets fly, in pixels a second in 2D or metres a second in 3D. In 2D, 120 is a slow shot the player can weave through and 400 or more a quick one. In 3D, 10 is slow and 40 is quick. 0 leaves them hanging where they were fired. Var: a float.	120.0
Start Out	Start each bullet this far out from From, along the way it is going, in pixels in 2D or metres in 3D. At 0 a ring starts at From, inside the boss that fired it. Forty pixels or about a metre starts it just outside. Var: a float.	0.0
Every Seconds	Keep firing another shot this often, in seconds, for as long as the state runs. 0.2 is five shots a second, a gun's fire rate. 0, the default, fires one shot and stops. Var: a float.	0.0
Lifetime	How many seconds a bullet flies before it disappears by itself. Four seconds crosses most screens. Shorten it to keep a busy screen clear, or lengthen it for a slow bullet that has a long way to go. Var: a float.	4.0
Hits Group	What these bullets hurt. That is the player for an enemy's bullets, or every copy of the enemy for the player's. Press the box and pick it in the scene. It can be one thing by its name, Every Coin for every copy of the coin scene (including ones made while playing), or Everything under a node for a set of markers. NodeWorthy does the rest. A Hurt Box tucked inside counts. Left empty, the bullets hurt nothing and only something solid stops them, and a warning asks what they should hit.	empty
Hit Event	The event sent to the machine of whatever a bullet hits, carrying Damage with it. It starts as HIT, so give the thing being shot a transition waiting for that name. Something with a Hurt Box is hurt through that. Its health goes down by Damage, less its defence, and its machine hears the Hurt Box's HURT or DIED, not this event. Empty sends nothing, but a Hurt Box is still hurt.	HIT
Damage	The number the hit event carries. The machine that was hit can drop it into a variable as the transition is taken and take it off its health. Something with a Hurt Box has it taken off its health for it, less its defence. Var: a float.	1.0
Pierce	How many more things one bullet may pass through after the first thing it hits. 0, the default, stops it at the first. 2 lets a single shot go through three enemies standing in a row. Var: an int.	0
Bounces	How many times a bullet bounces off walls before a wall finally stops it. 0 stops it at the first wall. Two or three makes a shot that rattles round a small room. Var: an int.	0
Homing	Turn each bullet toward the nearest thing in Hits Group by up to this many degrees a second. 0, the default, flies dead straight. Thirty curves gently and can still be outrun, while ninety is hard to dodge. Var: a float.	0.0
Sound	A sound played each time it fires. However many guns fire it together, only a few copies play at a time, and a room full of enemies never turns into a roar. Leave it empty to fire silently.	empty
Target	Only when Aim is At A Node. The node the middle of the pattern points at when the shot goes off, usually the player. Left empty, or with the target standing on From's spot, the shot falls back to the way From faces.	—
Direction	Only when Aim is Along A Direction. The direction the middle of the pattern points, typed in as two numbers or read from a variable with Var. The player's aim from Get Input Vector drops straight in. In 3D the two numbers are across and forward, as Get Input Vector gives them. A direction of 0, 0 falls back to the way From faces.	—
Angle	Only when Aim is An Angle. The direction in degrees, where 0 is right, 90 is down, 180 is left and 270 is up. Remember that in 2D the numbers grow downwards, so 90 points at the bottom of the screen. In 3D the angle is on the floor, so 0 is east and 90 is south, toward the bottom of a screen looking down.	—
Spread	Only when Count is more than one, or is read from a variable, as a gun that gains shots from power-ups does. How wide the pattern is in degrees, measured from the first bullet to the last. 30 is a shotgun fan, 90 a wide sweep, and 360 spaces them evenly all the way round as a ring. 0 sends them all along one line, on top of one another.	—
Gap	Only when Count is more than one, or is read from a variable. Leaves out this many bullets side by side, at a different place in the pattern each shot, so a wall of bullets gets a gap to slip through. 0 leaves none out.	—
Sound Bus	Only when a Sound is picked. The mixer bus the sound plays through. A bus is a named channel with one volume. Pick Effects and an Options menu's effects slider reaches every gun in the game.	—

Heal

Gives health back through a Hurt Box, to the same health a hit takes from, and never above that health's Max. The healed thing's Machine is sent HEALED, carrying how much it was given.



Heal as it appears in the Actions tab.

When to use it

A potion that gives back 25, or a campfire that mends a little every second. Set Variable could change the number too, but Heal stops at the Max, keeps whole-number health whole, and brings back something that had run out only when Revives is on.

> Health with no Max in the Variables tab can rise for ever. A warning names the health variable, once.

See also

Hurt 267, **Set Variable** 49, **Modify Stat** 268, **Use Item** 300

Try it

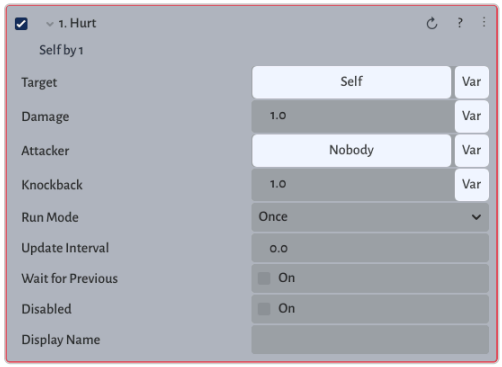
See the action work on its own with **Open Example** in the Add Action window, or open `examples/combat/heal/heal.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Target	What to heal. It needs a Hurt Box under it, where the name of its health variable is kept. Left empty it is the node this Machine works on. With nothing there, or no Hurt Box, nothing is given and a warning says which is missing. Var: a Node.	Self
Amount	How much health to give back. It must be above 0. Whatever would go over the health's Max, set in the Variables tab, is not given. Health kept as a whole number is given whole numbers, at least 1. Var: a float.	1.0
Healed Event	The event sent to its Machine when health is given, carrying the amount, which comes out below Amount near the Max. It is not sent when the target was already full. Left empty, nothing is sent.	HEALED
Revives	Off, the default, something whose health has already run out stays beaten, and a warning suggests turning Revives on. On, it is given health all the same and gets up again as a respawn does, safe from hits for a moment before it can be hurt again. Var: a bool.	off

Hurt

Takes health off something through its Hurt Box, with nothing touching it. It is hurt as a Hit Box would hurt it. Its defence comes off first, it blinks and is safe from another hit for a moment, and its Machine is sent HURT, or DIED when its health runs out.



Hurt as it appears in the Actions tab.

When to use it

Damage with no weapon behind it, such as poison that eats away a little every second, lava underfoot, a trap that goes off, or a scripted hit in a cut scene. The thing hurt needs a Hurt Box under it, the same one that takes hits from a weapon. A weapon swung at whatever it touches is Attack's job.

> As with any hit, one that comes while it is still safe from the last does nothing. For poison, put it on Every Frame with an Update Interval longer than the Hurt Box's Invincibility Time, such as 1 second.

See also

Attack 264, **Modify Stat** 268, **Explosion** 110, **Bullet Pattern** 265

Try it

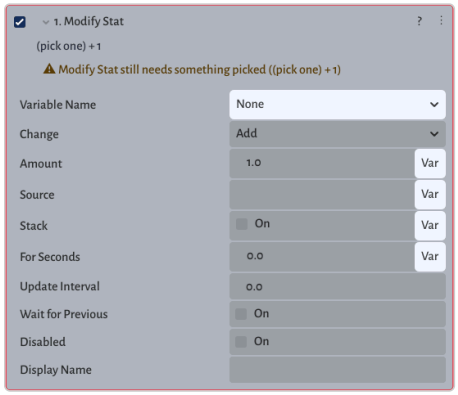
See the action work on its own with **Open Example** in the Add Action window, or open `examples/combat/hurt/hurt.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Target	What to hurt. Left empty it hurts the node this Machine works on. Pick another, such as the player, for a trap's Machine to hurt whoever sets it off. Its Hurt Box may be on it or on the character it is part of. Var: a Node.	Self
Damage	How much health to take off, before its defence, starting at 1. A hit always takes at least 1, and whole-number health takes whole hits. At 0 or less no health is taken, and the Output panel prints a warning. Var: a float.	1.0
Attacker	Who did it, if anyone. Picked, the one hurt is knocked back away from them, they are kept in its Hurt Box's Store Attacker In variable when it has one, and they are given the XP it is worth if this beats it. Left empty it is nobody, so nothing is knocked anywhere and nobody is rewarded. Var: a Node.	empty
Knockback	How hard it is knocked back from the attacker, times the Hurt Box's own Knockback. 1, the default, is a usual hit, 0 is no push at all and 2 twice as hard. Nothing is knocked back without an attacker. Var: a float.	1.0

Modify Stat

Changes one of the machine's number variables under a name, so that one change can be taken away again later and leave every other change alone. The number always holds what it started as, plus every addition, times every multiplication, whatever order they came and went in.



Modify Stat as it appears in the Actions tab.

When to use it

A potion that adds 50 to speed for ten seconds is Add, with Amount 50, Source "potion" and For Seconds 10. A poison that halves damage until it is cured, or a shrine that makes the player faster for good, works the same way. Items make their own changes when Give Item hands them over and need none of this.

> If something else sets the variable in between, the new number is taken to include the changes still named. Taking one away later then leaves the right number, with no jump.

See also

Set Variable 49, **Give Item** 290, **Math** 46, **Timer** 68

Try it

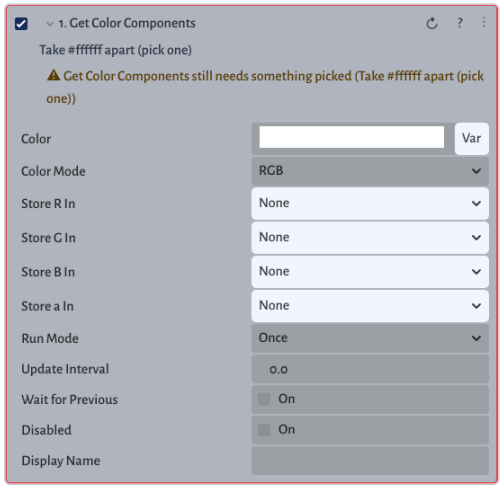
See the action work on its own with **Open Example** in the Add Action window, or open `examples/combat/modify_stat/modify_stat.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Variable Name	The number variable being changed, such as speed or fire_delay. Left empty, nothing is changed and a warning asks for one. With Remove, empty takes the changes made under Source off every stat.	empty
Change	Add puts Amount on top of the number. Multiply scales it, so 1.5 is half as much again and 0.5 is half. Remove takes away every change ever made under the Source name and leaves the rest standing. Add is the default. Choices: Add, Multiply, Remove.	Add
Amount	How much to add, or what to multiply by. A negative number takes some off when Adding. It starts at 1, which adds one or multiplies by one, so set it to what the boost should be. It is not shown when Change is Remove, which needs no amount. Var: a float.	1.0
Source	The name the change is made under, such as the item or the boost it comes from, for example "rage potion". Taking it away later uses this same name, so the two must match. Left empty the change is still made, under a blank name, and every other unnamed change on that variable then replaces it. A warning suggests naming it after the item or boost. Var: a String.	empty
Stack	On, a second change under the same name grows the first, so two damage-ups add up and two halvings halve twice. Off, the default, the second change replaces the first, so picking up the same boost twice is still one boost. It is not shown when Change is Remove. Var: a bool.	off
For Seconds	Take the change away by itself after this many seconds, as a power-up that runs out does. At 0, the default, it stays until something takes it away. Made again under the same Source before the time is up, the change lasts from the second time, so picking up the same boost again tops it up. A paused game holds the time still. It is not shown when Change is Remove. Var: a float.	0.0

Get Color Components

Takes a colour apart into numbers, either red, green, blue and alpha, or hue, saturation, value and alpha, each put into a separate float variable. Every part is a number from 0 to 1.



Get Color Components as it appears in the Actions tab.

When to use it

Math, a Check or a label sometimes needs one part of a colour as a number. A fading ghost's alpha, taken out into a float, lets a Check tell when it has dropped below 0.1.

> With every Store variable left empty, nothing is kept and a warning asks for one beside Store A In or one of the others.

See also

Set Color Components 272, **Get Vector Components** 50, **Sample Gradient** 271, **Math** 46

Try it

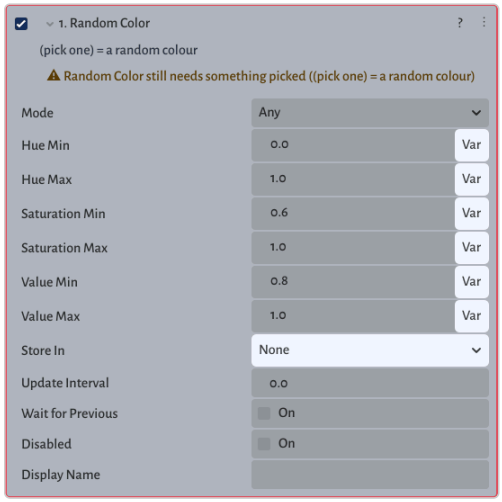
See the action work on its own with **Open Example** in the Add Action window, or open `examples/color/get_color_components/get_color_components.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Color	The colour to take apart. Most often you press Var beside it and pick a Color variable. Var: a Color.	1, 1, 1, 1
Color Mode	RGB takes it apart into red, green and blue, and HSV into hue (where it is round the colour wheel), saturation (how strong the colour is) and value (how bright). Alpha comes out in both. Choices: RGB, HSV.	RGB
Store R In	Only with RGB. The float variable its red goes into, from 0 (no red) to 1 (full red). Left empty, the red is not kept.	empty
Store G In	Only with RGB. The float variable its green goes into, from 0 to 1. Left empty, the green is not kept.	empty
Store B In	Only with RGB. The float variable its blue goes into, from 0 to 1. Left empty, the blue is not kept.	empty
Store a In	The float variable its alpha goes into, where 0 is invisible and 1 solid. Left empty, the alpha is not kept.	empty
Store H In	Only with HSV. The float variable its hue goes into. That is where it sits round the colour wheel, from 0 to 1, with 0 red, about 0.33 green and about 0.67 blue. Left empty, the hue is not kept.	—
Store S In	Only with HSV. The float variable its saturation goes into, where 0 is grey and 1 the strongest the colour goes. Left empty, it is not kept.	—
Store V In	Only with HSV. The float variable its value goes into, where 0 is black and 1 as bright as it goes. Left empty, it is not kept.	—

Random Color

Puts a random colour into a Color variable. It can pick from the colour wheel, within ranges you set, or from a Gradient you draw, taking one of the colours at its points or anywhere along it.



Random Color as it appears in the Actions tab.

When to use it

Each slime can come out a different colour when it is made, and confetti can take every colour there is. To choose from a few colours you pick, draw them as points on a Gradient and tick Points Only. Show the colour with Set Modulate, Set Light or Set Material Color, reading the variable with Var.

> The variable must hold a colour. One holding a number or words is left as it was, and a warning says what it holds.

See also

Random Number 47, **Sample Gradient** 271, **Set Color Components** 272, **Random Seed** 48

Try it

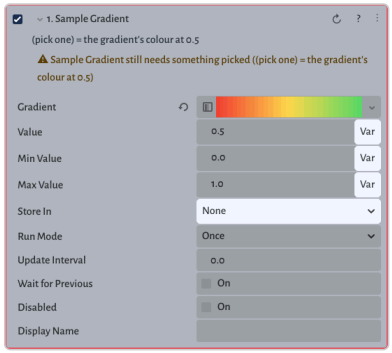
See the action work on its own with **Open Example** in the Add Action window, or open `examples/color/random_color/random_color.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Mode	Any picks from the colour wheel within the ranges below. From Gradient picks from the Gradient you draw. Choices: Any, From Gradient.	Any
Hue Min	Only with Any. The lowest hue, where it is round the colour wheel. 0 is red, about 0.33 green, about 0.67 blue and 1 red again. A lowest above the highest goes round through red, so 0.9 to 0.1 gives only reds. Var: a float.	0.0
Hue Max	Only with Any. The highest hue. A lowest of 0 and a highest of 1, as it starts, allow every hue on the wheel. Var: a float.	1.0
Saturation Min	Only with Any. The lowest saturation, where 0 allows greys and 1 only the strongest colours. It starts at 0.6, so the colours come out strong. Var: a float.	0.6
Saturation Max	Only with Any. The highest saturation, up to 1. Var: a float.	1.0
Value Min	Only with Any. The lowest value, meaning how bright. 0 allows black and 1 only the brightest. It starts at 0.8, so the colours come out bright. Var: a float.	0.8
Value Max	Only with Any. The highest value, up to 1. Var: a float.	1.0
Store In	The Color variable the colour goes into. Left empty, the colour is lost and a warning asks for a Color variable beside Store In.	empty
Gradient	Only with From Gradient. The colours to pick from. It starts with four plain colours in steps. Click on it to add a point and pick its colour, and right-click a point to take it out.	—
Points Only	Only with From Gradient. On, it picks one of the colours at the Gradient's points, each as likely as the others, so the Gradient becomes a list of colours to choose from. Off, it picks anywhere along it, the blends between the points too.	—

Sample Gradient

Picks a colour along a Gradient you draw, by a number, and puts it into a Color variable. The number is placed between a lowest and a highest, the way a progress bar places its value.



Sample Gradient as it appears in the Actions tab.

When to use it

A health bar can run from green through yellow to red as health falls. Set Min Value to 0, Max Value to the most health there is, and read Value from the health variable with Var. On Every Frame the colour follows the number as it changes. Show it with Set Modulate, Set Light or Set Material Color, reading the Color variable with Var.

> Min Value and Max Value must differ. With the same number in both, the variable is left as it was and a warning gives the number they share.

See also

Random Color 270, **Get Color Components** 269, **Ease Variable** 44, **Set Value** 215

Try it

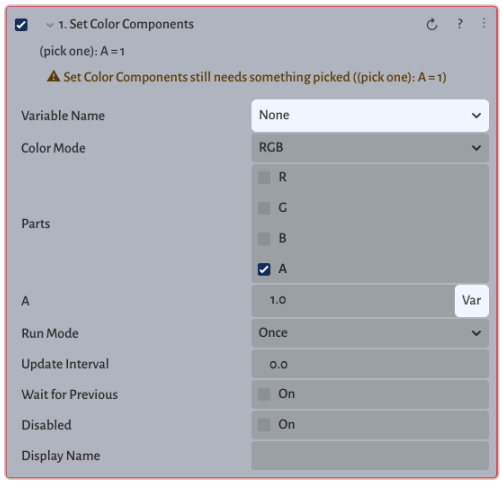
See the action work on its own with **Open Example** in the Add Action window, or open `examples/color/sample_gradient/sample_gradient.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Gradient	The colours to pick along. It starts as red through yellow to green, which suits a health bar from empty to full. Click on it to add a point and pick its colour, and right-click a point to take it out.	red to yellow to green
Value	The number to look up, such as health. It is most often read from a variable with Var. Var: a float.	0.5
Min Value	The number that gives the colour at the Gradient's left end, such as no health at all. A Value below it gives that end's colour. Var: a float.	0.0
Max Value	The number that gives the colour at the Gradient's right end, such as full health. A Value above it gives that end's colour. With Min Value at 0 and Max Value at 1, Value is how far along the Gradient to look. Var: a float.	1.0
Store In	The Color variable the colour goes into. Left empty, the colour is lost and a warning asks for a Color variable beside Store In.	empty

Set Color Components

Changes the parts of a Color variable you tick and leaves the others as they are. Each part can be typed in or read from a float variable with Var.



Set Color Components as it appears in the Actions tab.

When to use it

A ghost fades when only its alpha is set, say to 0.3, and a gem changes colour when only its hue moves round the wheel. It can also build a colour from numbers worked out elsewhere. Show the result with Set Modulate, Set Light or Set Material Color, reading the variable with Var.

> A variable holding something other than a colour is left as it was, and a warning names the variable and what it holds.

See also

Get Color Components 269, **Set Vector Components** 52, **Set Variable** 49, **Random Color** 270

Try it

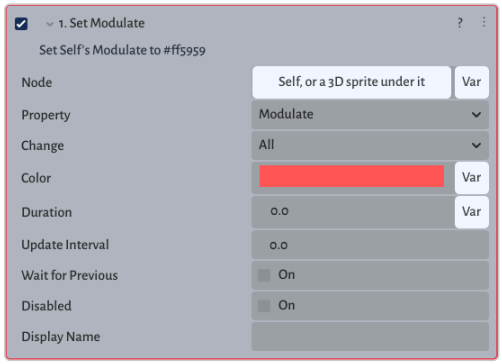
See the action work on its own with **Open Example** in the Add Action window, or open `examples/color/set_color_components/set_color_components.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Variable Name	The Color variable to change. A name no variable has yet is made on the spot, starting from white.	empty
Color Mode	RGB changes its red, green and blue, and HSV its hue, saturation and value. Alpha can be changed in both. The ticks under Parts are named for the mode picked. Choices: RGB, HSV.	RGB
Parts	Which parts to change. The others keep what they hold. Only the alpha is ticked to start with, ready for a fade.	A
A	Only with A ticked. The alpha it becomes. 0 is invisible, 0.5 half see-through and 1 solid. Var: a float.	1.0
R	Only with RGB and R ticked. The red it becomes, from 0 to 1.	—
G	Only with RGB and G ticked. The green it becomes, from 0 to 1.	—
B	Only with RGB and B ticked. The blue it becomes, from 0 to 1.	—
H	Only with HSV and H ticked. The hue it becomes, from 0 to 1 round the colour wheel, with 0 red, about 0.33 green and about 0.67 blue. A number past 1 goes on round the wheel, so 1.25 is the same as 0.25.	—
S	Only with HSV and S ticked. The saturation it becomes, where 0 is grey and 1 the strongest the colour goes.	—
V	Only with HSV and V ticked. The value it becomes, where 0 is black and 1 as bright as it goes.	—

Set Modulate

Tints or fades anything drawn, such as a sprite, a shape, a panel or button, or a 3D sprite or label. It changes the node's Modulate, the colour it is drawn through, straight away or eased over a number of seconds.



Set Modulate as it appears in the Actions tab.

When to use it

Flash a character red when it is hit, or grey out a button that cannot be pressed yet. A 3D mesh has no colour to be drawn through. Its material paints it, and Set Material Color changes that.

> A newer Set Modulate on the same node stops one still easing it, so two never pull the colour two ways.

See also

Set Visible 216, **Tween Property** 106, **Set Material Color** 276, **Set Shader Parameter** 204

Try it

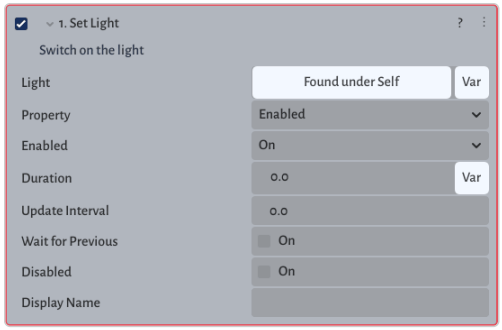
See the action work on its own with **Open Example** in the Add Action window, or open `examples/color/set_modulate/set_modulate.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node to tint. Left empty it uses the node this Machine works on when that node is drawn itself, as anything 2D or on the screen is. On a 3D character it uses the first 3D sprite, or failing that the first 3D label, anywhere under it. Var: a Node.	empty
Property	Modulate, the default, tints the node and everything under it, as a whole character. Self Modulate tints only the node itself, so a name floating above a character, or a picture inside a panel, keeps its own colour. A 3D sprite or label only has Modulate, which already leaves what is under it alone. Choices: Modulate, Self Modulate.	Modulate
Change	All, the default, sets both the colour and how see-through it is, as the Color below is picked. Color changes only the colour and keeps how see-through it is now. Alpha changes only how see-through it is and keeps its colour, for a fade in or out. Choices: All, Color, Alpha.	All
Color	Only when Change is All or Color. The colour to tint it, starting as a soft red. White is the colour it was drawn, so tinting back to white undoes a tint. The alpha at the bottom of the colour picker is how see-through it is, with 0 invisible, and is used only when Change is All. Var: a Color.	1, 0.35, 0.35, 1
Duration	How many seconds the change takes. At 0, the default, it changes straight away, and above 0 it eases there, as a fade does. Var: a float.	0.0
Alpha	Only when Change is Alpha. How see-through to make it, where 1 is solid, 0.5 half see-through and 0 invisible. It starts at 1.	—
Curve	Only when Duration is above 0. The shape of the change. Linear, the default, keeps one speed all the way. Sine, Quad, Cubic and Expo speed up or slow down more and more. Elastic wobbles past the end and back, Back goes a little past and comes back, and Bounce bounces at the end.	—
Easing	Only when Duration is above 0. Where the change is slow. Ease Out, the default, slows at the end, as a thing coming to rest. Ease In slows at the start, Ease In Out at both ends and Ease Out In in the middle.	—
Wait for End	Only when Duration is above 0. On, the default, the action finishes only when the change is over, and a transition on FINISHED comes after the fade. Off finishes straight away and lets the change play on by itself, even after the state is left.	—

Set Light

Changes a light while the game runs. It switches it on or off, or sets its colour, its energy, how far it reaches, or whether it casts shadows. Each change is made straight away or eased over a number of seconds, and a light switched on or off over a time fades.



Set Light as it appears in the Actions tab.

When to use it

A torch can dim as it burns down, or an alarm turn the room's light red. It works on a 2D light (a PointLight2D or a DirectionalLight2D) and on a 3D one (an OmniLight3D, a SpotLight3D or a DirectionalLight3D). Each Set Light changes one thing, so to switch a light on and colour it too, put two in the state.

> A 2D light is switched with its Enabled setting and a 3D one by being shown or hidden, so a 3D light switched off here is hidden, not merely dark.

See also
Screen Effect 202, **Set Modulate** 273, **Tween Property** 106, **Set Property** 91

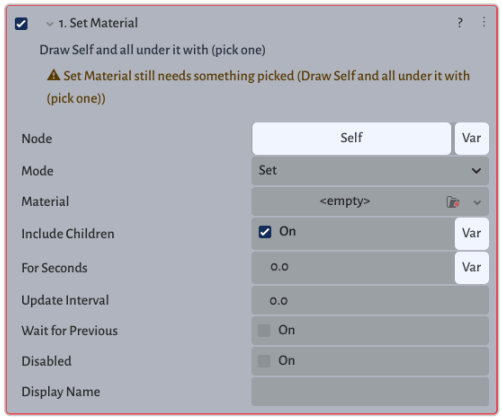
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/lights/set_light/set_light.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Light	The light to change. Left empty it uses the node this Machine works on when that is a light, or else the first light anywhere under it, whatever it is called. A Machine on a lamp or a torch finds the light it carries. Var: a Node.	empty
Property	What to change. Enabled, the default, switches it on or off. Color is the colour of its light, Energy is how bright it is, and Range is how far it reaches. Shadow Enabled is whether what it lights casts shadows. Only the setting for the one picked is shown below it. Choices: Enabled, Color, Energy, Range, Shadow Enabled.	Enabled
Enabled	Only when Property is Enabled. On switches it on, Off switches it off, and Toggle flips it from how it is now, turning one button into a light switch. Over a Duration, Off fades it down to 0 and then switches it off, and On switches it on at 0 and fades it up to the brightness it had. Either way it keeps its brightness for next time. Choices: On, Off, Toggle.	On
Duration	Not shown for Shadow Enabled. How many seconds the change takes. At 0, the default, it changes straight away. Above 0 it eases there, and switching on or off becomes a fade. Var: a float.	0.0
Color	Only when Property is Color. The colour of its light, starting as a warm orange-white. White is plain light, orange a fire, blue moonlight.	—
Energy	Only when Property is Energy. How bright it is. 1, the default, is as bright as a new light starts, 0 gives no light at all, and 2 is twice as bright.	—
Light Range	Only when Property is Range. How far it reaches, starting at 3. For a 3D light it is in metres. For a 2D light it is how many times the size of its light picture, so 2 lights twice as far. A directional light lights everything wherever it is and has no range, and the Output panel warns if you try to set one.	—
Shadow Enabled	Only when Property is Shadow Enabled. On, the default, whatever it lights casts shadows. Off, there are no shadows. Shadows cost more to draw, and a light far away can do without them. This one always changes straight away.	—
Curve	Only when Duration is above 0. The shape of the change. Linear, the default, keeps one speed all the way. Sine, Quad, Cubic and Expo speed up or slow down more and more. Elastic wobbles past the end and back, Back goes a little past and comes back, and Bounce bounces at the end.	—
Easing	Only when Duration is above 0. Where the change is slow. Ease Out, the default, slows at the end. Ease In slows at the start, Ease In Out at both ends and Ease Out In in the middle.	—
Wait for End	Only when Duration is above 0. On, the default, the action finishes only when the change is over, and a transition on FINISHED comes after the fade. Off finishes straight away and lets the change play on by itself.	—

Set Material

Swaps how something is drawn for another look, a material, and puts the original back later. The material is a resource that says how a surface is drawn, made in the FileSystem or picked from one already made.



Set Material as it appears in the Actions tab.

When to use it

A white flash for a moment when something is hit, a red glow while an enemy is enraged, or a burnt skin once a car is wrecked. In 3D the material becomes the node's Material Override, drawn in place of what its mesh is painted with. In 2D it becomes the node's Material, which needs a shader to change how it looks. Changing only a colour is simpler with Set Material Color or Set Modulate.

> The action always finishes straight away, even with For Seconds set, and the original comes back by itself later.

See also

Set Material Color 276, **Set Shader Parameter** 204, **Set Modulate** 273

Try it

See the action work on its own with **Open Example** in the Add Action window, or open `examples/material/set_material/set_material.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The node whose look to swap, such as a mesh, a sprite, or a whole character. Left empty it uses the node this Machine works on. Var: a Node.	Self
Mode	Set, the default, draws it with the material below. Put Back gives back the material it had before it was first swapped, however many swaps came in between, including having none at all. Choices: Set, Put Back.	Set
Material	Only when Mode is Set. The material to draw it with, picked from the FileSystem. That is a StandardMaterial3D or a ShaderMaterial in 3D, or a ShaderMaterial or a CanvasItemMaterial in 2D. Left empty, no look is swapped and a warning says no material is picked. A 3D material cannot draw a 2D node, nor a 2D one a 3D node. Those parts are left as they are, and a warning follows if no part could take it.	empty
Include Children	On, the default, it swaps the look of everything drawn under the node as well, and a character made of several parts all changes. Under a 3D node only meshes and CSG shapes count, so a name floating over a character keeps its words. Off changes the node alone. Var: a bool.	on
For Seconds	Only when Mode is Set. Put the original back by itself after this many seconds, for a flash when something is hit. It keeps counting after the state is left, and a flash is never left on. At 0, the default, the new look stays until a Put Back. Var: a float.	0.0

Set Material Color

Changes the colour or the glow of a 3D thing. That is its Albedo Color, the colour a surface is painted, or its Emission, the glow it gives off by itself. It is set straight away or eased over a number of seconds. Only this one thing changes, never everything else painted with the same material.



Set Material Color as it appears in the Actions tab.

When to use it

A door panel can turn from red to green once it is unlocked, or a crystal glow when it is charged. It works on meshes painted with a StandardMaterial3D. A 2D sprite is tinted with Set Modulate, and a mesh painted with a shader is changed with Set Shader Parameter.

> The first time a mesh is coloured, each material it is painted with is copied and the copy given to that mesh alone, so every other copy of the same enemy keeps its colour. A mesh painted with no material is given a plain white one to colour.

See also

Set Material 275, **Set Modulate** 273, **Set Shader Parameter** 204

Try it

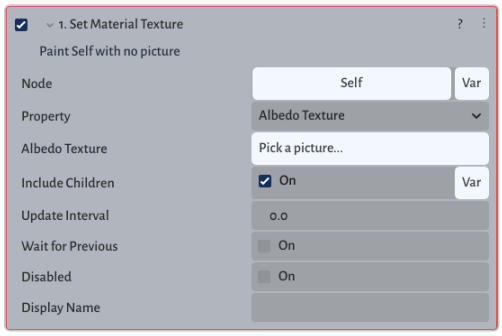
See the action work on its own with **Open Example** in the Add Action window, or open `examples/material/set_material_color/set_material_color.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The 3D thing to colour, either a mesh or a whole character. Left empty it uses the node this Machine works on. Var: a Node.	Self
Property	Albedo Color, the default, is the colour it is painted. Emission is the glow it gives off by itself, which shows even in the dark. The material's glow is switched on for it as the action runs. Choices: Albedo Color, Emission.	Albedo Color
Color	The colour, starting as a bright green. For Albedo Color, white shows its picture as it was made, and a colour that is less than solid, lower alpha at the bottom of the picker, makes it see-through. Its material is then set to draw see-through for it to show. For Emission, black is no glow. Var: a Color.	0.35, 1, 0.45, 1
Include Children	On, the default, it colours every mesh under the node as well, and a character made of several parts all changes. A 3D sprite or 3D label under it is left alone. Off colours the node alone. Var: a bool.	on
Duration	How many seconds the change takes. At 0, the default, it changes straight away, and above 0 it eases there. Var: a float.	0.0
Emission Energy Multiplier	Only when Property is Emission. How strongly it glows. 1, the default, is the usual strength, and 4 glows brightly.	—
Curve	Only when Duration is above 0. The shape of the change. Linear, the default, keeps one speed all the way. Sine, Quad, Cubic and Expo speed up or slow down more and more. Elastic wobbles past the end and back, Back goes a little past and comes back, and Bounce bounces at the end.	—
Easing	Only when Duration is above 0. Where the change is slow. Ease Out, the default, slows at the end. Ease In slows at the start, Ease In Out at both ends and Ease Out In in the middle.	—
Wait for End	Only when Duration is above 0. On, the default, the action finishes only when the change is over. Off finishes straight away and lets the change play on by itself.	—

Set Material Texture

Changes the picture painted on a 3D thing, or slides that picture across its surface. It changes the Albedo Texture (the picture a StandardMaterial3D paints) or the UV1 Offset (where that picture sits), on a copy of the material that belongs to that thing alone.



Set Material Texture as it appears in the Actions tab.

When to use it

A conveyor belt set to Scroll at 0.5 moves its picture half a width each second, and water or lava can creep along the same way. Swapping the Albedo Texture suits a screen or a poster that changes. Materials are often shared between copies of a scene, but this changes the one you pick and leaves the other belts alone. A 2D sprite's picture is changed with Set Texture.

> Scroll never finishes, so its state does not send FINISHED, so leave it on another event. A mesh painted with a ShaderMaterial is not changed, and a warning points to Set Shader Parameter.

See also

Set Texture 214, **Set Material Color** 276, **Set Material** 275, **Set Shader Parameter** 204

Try it

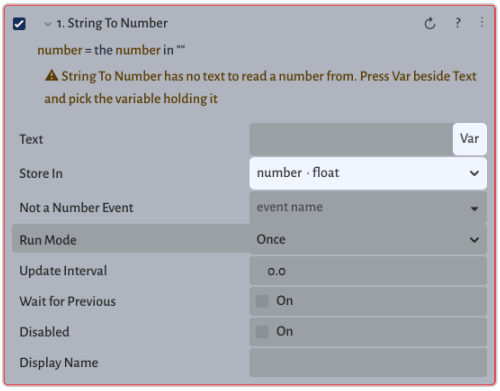
See the action work on its own with **Open Example** in the Add Action window, or open `examples/material/set_material_texture/set_material_texture.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Node	The 3D thing to change, either a mesh or a whole character with meshes under it. Left empty, it is the thing the machine works on. Var: a Node.	Self
Property	Albedo Texture swaps the picture. UV1 Offset slides the picture to a place on the surface. Scroll keeps sliding it at a speed for as long as the state is active. Choices: Albedo Texture, UV1 Offset, Scroll.	Albedo Texture
Albedo Texture	Only with Albedo Texture. The picture to paint it with, picked from the FileSystem. Left empty, it takes the picture away and leaves only the material's colour.	empty
Include Children	On, everything drawn under the node changes too, so a character made of several meshes all changes. Off changes the node alone. Var: a bool.	on
UV1 Offset	Only with UV1 Offset. Where the picture sits, measured in picture widths. 0.5 across slides it half its width, and 1 brings it all the way round to where it began.	—
Scroll Speed	Only with Scroll. How far the picture slides each second, in picture widths, so 0.5 across slides it half its width a second. A minus number slides it the other way.	—

String To Number

Reads a number out of some text into a number variable, such as the 12 in "12 coins".



String To Number as it appears in the Actions tab.

When to use it

An amount a player types into a LineEdit (a box for typing into) reaches the game as text. This turns it into a number you can do sums with. The first number in the text is read, and words and spaces round it do not matter.

> A thousands comma counts as the end of the number, so 1,000 reads as 1.

See also
Format String 54, **String Compare** 57, **Set Variable** 49, **Set Text** 213

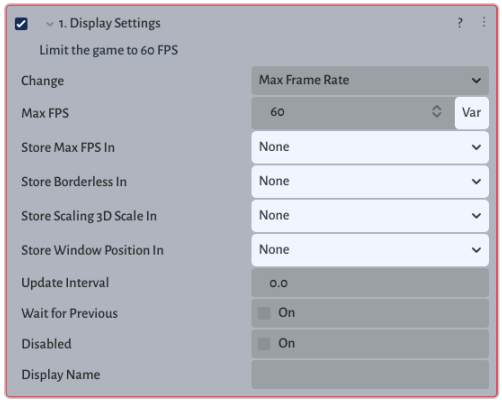
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/convert/string_to_number/string_to_number.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Text	The text to read. Type it in, or more usefully press Var and pick the variable holding it, such as the words an Event On Signal stored when the player pressed Enter in a box. Var: a String.	empty
Store In	The variable the number goes into. It starts as number. An int variable gets a whole number, with anything after the point dropped, so 7.9 gives 7. A float gets the number as it is. A variable holding text, or anything else that is not a number, is left alone with a warning.	number
Not a Number Event	Sent when the text has no number in it at all, such as a player typing lots. The variable is left as it was. With no event set, a warning is printed.	empty

Display Settings

Changes the rest of what an Options menu does to the display. That covers the most frames a second the game draws, a window with no border, the resolution a 3D world is drawn at, and where the window sits on the screen. It can also put how each of these is set now into variables.



Display Settings as it appears in the Actions tab.

When to use it

A laptop player who wants a quiet fan can cap the game at 30 frames a second from an Options menu. Fullscreen, the window's size and V-Sync live in Screen Settings, not here. Set Change to Nothing at the start of a menu to fill its boxes with how things are now.

> The border and the window's place cannot change while the game is fullscreen or running inside the editor's Game tab. A warning says why. Run the game in its own window to see them change.

See also

Screen Settings 225, **Set UI Scale** 282, **Save Game** 235, **Run When Unfocused** 281

Try it

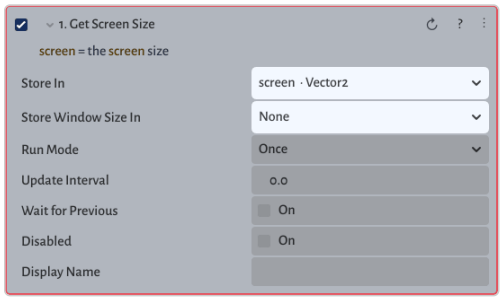
See the action work on its own with **Open Example** in the Add Action window, or open `examples/application_and_display/display_settings/display_settings.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Change	What to change. Nothing changes nothing and only fills in the variables below. Max Frame Rate, Borderless, Resolution Scale and Window Position each use the setting of the same name below. Centre Window puts the window in the middle of the screen. Choices: Nothing, Max Frame Rate, Borderless, Resolution Scale, Window Position, Centre Window.	Max Frame Rate
Max FPS	Only when Change is Max Frame Rate. The most frames a second the game draws. 30 or 60 save power and keep a computer quiet, and 0 means no limit at all. V-Sync, in Screen Settings, can hold the game lower still. Var: an int.	60
Store Max FPS In	A variable to put the frame rate limit into, as a whole number, after the change, where 0 means no limit. Leave it empty to store nothing.	empty
Store Borderless In	A variable to put true or false into, for whether the window has no border now, after the change. Leave it empty to store nothing.	empty
Store Scaling 3D Scale In	A variable to put the resolution scale into, as a number, after the change. Leave it empty to store nothing.	empty
Store Window Position In	A Vector2 variable to put the window's place on the screen into, after the change, so Save Game can keep it and the window can go back there next time. Leave it empty to store nothing.	empty
Borderless	Only when Change is Borderless. On takes away the window's border and title bar, and off puts them back.	—
Scaling 3D Scale	Only when Change is Resolution Scale. The share of the window's resolution a 3D world is drawn at, from 0.25 to 2. 1 is full sharpness, and 0.5 draws a quarter of the pixels and is much quicker but softer. A number outside that range is held at its end, with a warning. It changes nothing in a 2D game.	—
Window Position	Only when Change is Window Position. Where the window's top left corner goes, in pixels from the top left of the screen.	—

Get Screen Size

Puts the size of the part of the game that can be seen, and the size of the window, into variables that hold two numbers (Vector2), in pixels. X is how wide and Y how tall.



Get Screen Size as it appears in the Actions tab.

When to use it

An enemy that should come in just off the right side needs the screen's width, the X of the stored size, and a marker in a corner needs both numbers. With Run Mode on Every Frame both sizes keep up while the player resizes the window. It only measures, and Screen Settings changes the window.

> It measures the screen, not the level. With a camera that moves or zooms, the screen shows a different part of the level as it goes.

See also

Screen Settings 225, **Mouse Position** 150, **Off Screen Indicator** 209, **Get Vector Components** 50

Try it

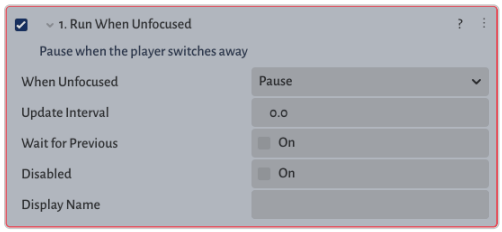
See the action work on its own with **Open Example** in the Add Action window, or open `examples/application_and_display/get_screen_size/get_screen_size.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Store In	The variable the size of the visible game area goes into. It starts as screen, which is made when the game runs if the template does not have it. The area is measured the way the game draws it. With the project's stretch settings on, it stays near the size the game was made for while the window grows and shrinks. Inside a SubViewport it is that viewport's size. Left empty, it is not stored.	screen
Store Window Size In	The variable the size of the window goes into, in the screen's own pixels. These are what grow and shrink when the player drags the window's corner. Left empty, it is not stored.	empty

Run When Unfocused

Says what the game does while the player has switched to something else, such as another program or a message. It can keep running, pause until they come back, or keep going at a lower frame rate. Set it once and it holds for the rest of the game.



Run When Unfocused as it appears in the Actions tab.

When to use it

Put it in the state a game starts in. A single-player game set to Pause waits while the player reads an email, and they lose no lives while away. A game played online with others usually keeps running. To show a pause menu as well when the player switches away, add a transition waiting for the system event Application Focus Out.

> A game already paused when the player switches away, by its own pause menu, is left paused when they come back, so the menu stays up.

See also

Pause Game 222, **Display Settings** 279, **Set Time Scale** 66

Try it

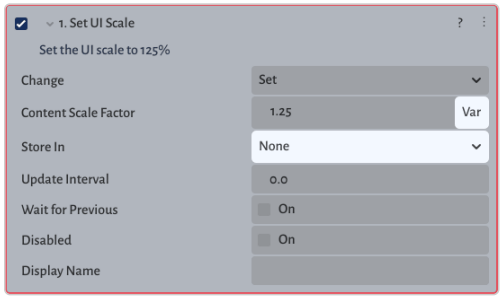
See the action work on its own with **Open Example** in the Add Action window, or open `examples/application_and_display/run_when_unfocused/run_when_unfocused.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
When Unfocused	What happens while the game's window is not in front. Keep Running carries on as if nothing happened, as a game does by default. Pause freezes the game until the player comes back to it. Lower Frame Rate keeps it going but draws only a few frames a second. Choices: Keep Running, Pause, Lower Frame Rate.	Pause
Unfocused Max FPS	Only when When Unfocused is Lower Frame Rate. The most frames a second the game draws while the player is elsewhere. 5 to 15 is plenty for a game nobody is watching. 0 would mean no limit, so the action refuses it with a warning and changes nothing.	—

Set UI Scale

Makes the interface bigger or smaller. Menus, buttons, labels and bars grow or shrink together and keep their places at the screen's edges.



Set UI Scale as it appears in the Actions tab.

When to use it

A player on a sofa across the room from a big television needs larger text than one at a desk. A Bigger button in an Options menu, with Step at 0.1, grows the interface a tenth at a time. In a 3D game only the interface changes size. In a 2D game the world is drawn bigger too, because everything flat in the window is scaled together.

See also

Display Settings 279, **Screen Settings** 225, **Save Game** 235

Try it

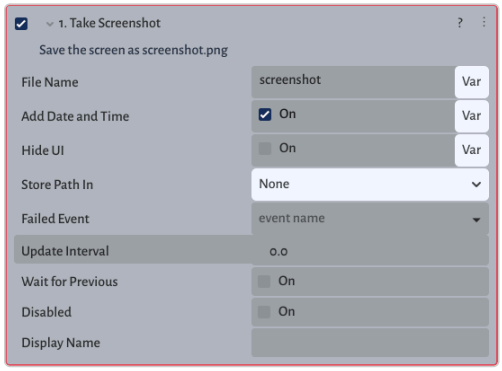
See the action work on its own with **Open Example** in the Add Action window, or open `examples/application_and_display/set_ui_scale/set_ui_scale.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Change	Set makes the scale the number in Content Scale Factor. Bigger and Smaller change it by Step each time, and stop quietly at the smallest and biggest sizes allowed. Reset puts it back to 1, the size the game was made at. Choices: Set, Bigger, Smaller, Reset.	Set
Content Scale Factor	Only when Change is Set. How big everything is drawn. 1 is the size the game was made at, 1.5 half as big again and 0.75 three quarters. It can go from 0.5 to 8, and a number outside that is held at its end and a warning is printed. Var: a float.	1.25
Store In	A variable to put the scale into after the change, as a number where 1 is the size the game was made at, so a menu can show it and Save Game can keep it. Leave it empty to store nothing.	empty
Step	Only when Change is Bigger or Smaller. How much the scale changes each time, where 0.1 is a tenth of the size the game was made at.	—

Take Screenshot

Saves what is on the screen as a picture, a PNG file in the screenshots folder of the game's own save folder, and puts the file's path into a variable. It can hide the interface for the picture.



Take Screenshot as it appears in the Actions tab.

When to use it

A small picture beside each save slot shows the player where they were. Take it with Add Date And Time off and File Name set to {slot}, and keep the path in a variable to show the picture in the load menu.

> The picture is taken once the next frame has been drawn, and the state carries on a moment later. A game running with no screen at all has nothing to take a picture of, and prints a warning.

See also

Save Game 235, **Write File** 236, **Choose Folder** 229

Try it

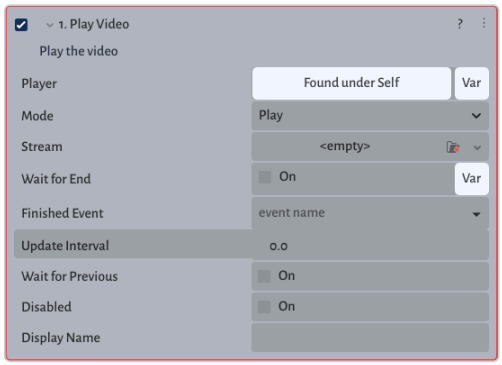
See the action work on its own with **Open Example** in the Add Action window, or open `examples/application_and_display/take_screenshot/take_screenshot.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
File Name	The picture's file name, without .png. Put a variable's name in curly brackets, such as {slot}, to name a save slot's picture after the slot. Characters a file name cannot have, such as a colon, are changed to a line along the bottom (_). Var: a String.	screenshot
Add Date and Time	On, the default, adds the date and time to the name, and each new picture is kept beside the others. Turn it off for a save slot's picture, which should replace the old one. Var: a bool.	on
Hide UI	On hides the interface for the picture, the menus, bars and labels, and shows it again straight afterwards, leaving only the game in the picture. A background drawn behind the game stays. Var: a bool.	off
Store Path In	A String variable to put the saved picture's path into. Leave it empty to store nothing.	empty
Failed Event	An event sent when no picture could be taken or saved. Leave it empty to send nothing. The state carries on either way.	empty

Play Video

Plays a video on a VideoStreamPlayer, or pauses, resumes, stops or jumps through it. It can wait for the video to end and send an event when it does.



Play Video as it appears in the Actions tab.

When to use it

An intro before the title screen plays on a VideoStreamPlayer, a node that shows a video. Stretch it over the screen with its Expand setting on and the video fills the window. The videos it plays are Ogg Theora files, ending .ogv. A video in another format has to go through a video converter first.

> Wait For End only lets the state finish once every other action in it has finished too. In a state that also listens for a key, leave it off and send Finished Event. A video set to Loop never ends.

See also

Play Sound 175, **Play Animation** 162, **Change Scene** 220

Try it

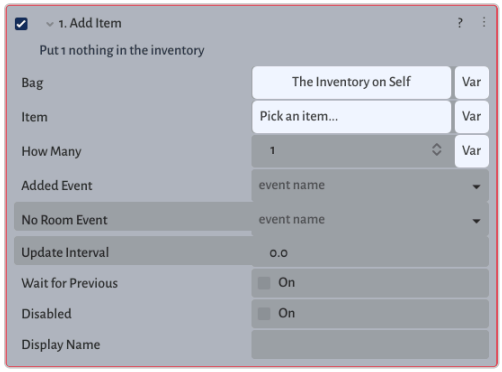
See the action work on its own with **Open Example** in the Add Action window, or open `examples/video/play_video/play_video.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Player	The VideoStreamPlayer. Left empty it uses the node this Machine works on if that is one, or the first VideoStreamPlayer anywhere under it, whatever it has been named. Var: a Node.	empty
Mode	Play starts the video from its beginning. Pause holds it where it is, Resume plays on from there, Stop ends it and goes back to the start, and Seek jumps to Stream Position. Choices: Play, Pause, Resume, Stop, Seek.	Play
Stream	Only when Mode is Play. The video to play, an .ogv file. Left empty, the player plays the video it already has. With neither, the action warns that there is no video to play.	empty
Wait for End	Only when Mode is Play or Resume. On suits a cutscene, because the state waits while the video plays and carries on once it has played to its end. Off, the default, the state carries on straight away and the video plays by itself. Var: a bool.	off
Finished Event	Only when Mode is Play or Resume. An event sent when the video plays to its end, even after the state has been left. Leave it empty to send nothing.	empty
Stream Position	Only when Mode is Seek. How far into the video to jump, in seconds from its start. It works on a video that is playing or paused.	—

Add Item

Puts something in an inventory, and says whether it all fitted.



Add Item as it appears in the Actions tab.

When to use it

Loot going into the player's pack, stock onto a shop shelf and a key into a chest all go in with this. One event answers when everything went in and another when it did not, so a template can say "your pack is full" and the thing is not lost. Piles already in the inventory are topped up before a new one is started, so twenty arrows take one slot, not twenty. The inventory is an Inventory node. With Bag left empty, the one on the node this machine works on is used, as for the player's pack.

> How much room there is depends on the item as well as the inventory. An inventory with every slot in use can still take more of something it already has, if that pile is not full. Max Stack on the item says how many sit in one pile.

See also

Remove Item 296, **Move Item** 292, **Count Item** 286, **Inventory Has** 291

Try it

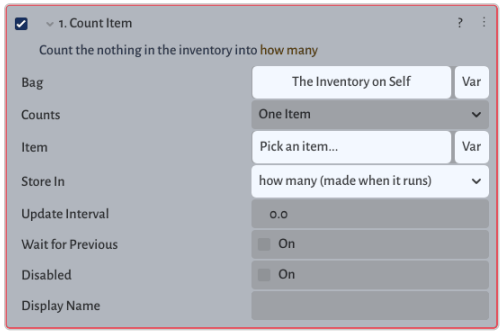
See the action work on its own with **Open Example** in the Add Action window, or open `examples/inventory/add_item/add_item.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Bag	The inventory to put it in. Leave it empty for the Inventory on the node this machine works on. Point it at a chest, a shelf or another character to fill theirs instead. Var: a Node.	empty
Item	The item. Press Var to read it from a variable, such as the one Pick Item chose. Var: a Resource.	empty
How Many	How many of it. Press Var to read the number from a variable, for loot that comes in different amounts. Var: an int.	1
Added Event	Event sent when every one of them went in. Empty sends nothing.	empty
No Room Event	Event sent when some or none of them did, because the inventory had no room. This is the event to show "your pack is full" on. Empty sends nothing.	empty

Count Item

Puts how many of something an inventory is carrying into a variable.



Count Item as it appears in the Actions tab.

When to use it

For an arrow counter on the HUD, count the arrows into a variable for Set Text to show, and a Check can ask whether any are left. Counting Stacks Used gives the 12 in "12 of 20" on a panel.

> Counting one item counts every pile of it, so three potions in one pile and two in another answer five.

See also

Inventory Has 291, **Add Item** 285, **Remove Item** 296, **Set Text** 213

Try it

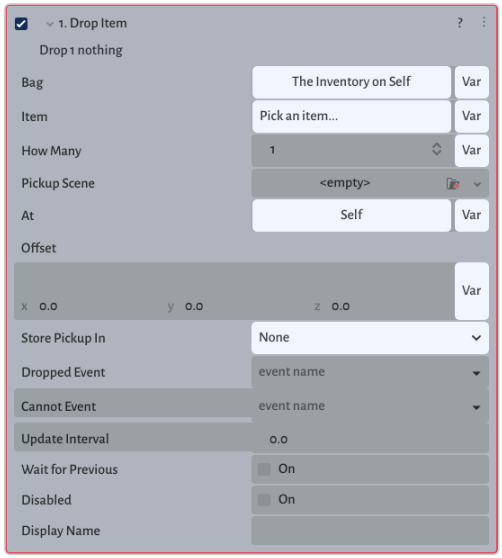
See the action work on its own with **Open Example** in the Add Action window, or open `examples/inventory/count_item/count_item.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Bag	The inventory to count in. Leave it empty for the Inventory on the node this machine works on. Var: a Node.	empty
Counts	What to count. One Item counts the item below. Stacks Used counts how many slots are in use, and Slots Free how many are still empty. An inventory with no limit set answers -1 to that one, because it never runs out. Choices: One Item, Stacks Used, Slots Free.	One Item
Item	Only when counting one item. The item to count. Press Var to read it from a variable. Var: a Resource.	empty
Store In	The variable the answer goes into, as a whole number.	how many

Drop Item

Puts something down in the world where it can be picked up again.



Drop Item as it appears in the Actions tab.

When to use it

A thrown-away weapon, or what a defeated enemy was carrying, lands in your own pickup scene, which needs an Inventory to hold it. Picking it up again is a Move Item from that inventory into the player's, and a dropped pile of twenty arrows is still twenty arrows.

> With no Pickup Scene, or a scene with no Inventory on it, the item stays where it was, Cannot Event is sent, and a warning says what is missing, because an item put into a scene that cannot hold it would be lost.

See also

Move Item 292, **Add Item** 285, **Instantiate** 84, **Inventory** 319

Try it

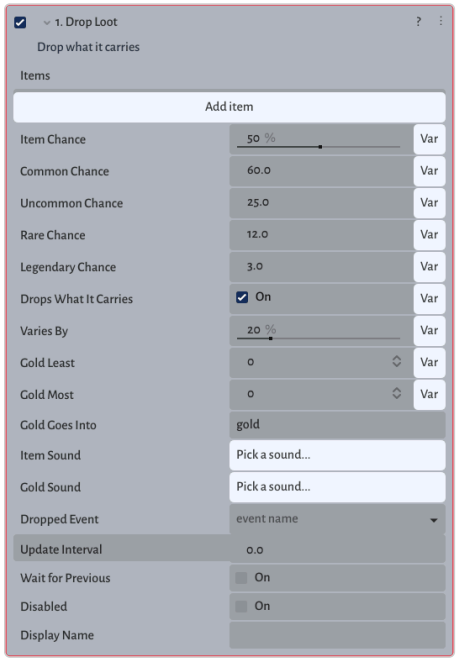
See the action work on its own with **Open Example** in the Add Action window, or open `examples/inventory/drop_item/drop_item.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Bag	The inventory to take it out of. Leave it empty for the Inventory on the node this machine works on. Var: a Node.	empty
Item	The item to put down. Press Var to read it from a variable, such as the one a panel was showing. Var: a Resource.	empty
How Many	How many of it. Zero or less drops every one there is. Press Var to read the number from a variable. Var: an int.	1
Pickup Scene	The scene put into the world to hold it. It is your own pickup, with an Inventory on it and whatever picture and area it needs. Without one there is nowhere for the item to go and nothing is dropped.	empty
At	Where it lands. Leave it empty to drop it at the node this machine works on. Var: a Node.	empty
Offset	How far from there it lands, in the world's directions, so a drop appears beside a character, not inside it. Two numbers in a 2D game, three in a 3D one. Var: a Vector2 or a Vector3, or a Node for where it stands.	0, 0, 0
Store Pickup In	Store the dropped pickup in this variable, so a later action can move it, watch it or take it away again. Empty stores nothing.	empty
Dropped Event	Event sent once it is down, carrying the item. Empty sends nothing.	empty
Cannot Event	Event sent when it could not be dropped, because it is not being carried, there is no scene to put it in, or that scene has no Inventory on it. Empty sends nothing.	empty

Drop Loot

Drops loot on the floor, as a beaten enemy, a broken pot or an opened chest does. Now and then there is an item from a list, rarer ones less often, and a handful of gold. What drops lies on the floor with a glow and its name in the colour of how rare it is, and a player walking over it picks it up.



Drop Loot as it appears in the Actions tab.

When to use it

An enemy's Dead state is the usual place, so enemies leave something behind, or a chest's Open state. Items go into the player's Inventory, gold onto their gold variable. For an item that always drops, give the dropper an Inventory and tick Drops What It Carries.

> An item is picked up only by someone with an Inventory that has room for it. Otherwise it stays on the floor. It works in 2D and 3D alike, and in 3D the loot's picture and name turn to face the camera.

See also

Give Item 290, **Drop Item** 287, **Inventory** 319, **Add Item** 285

Try it

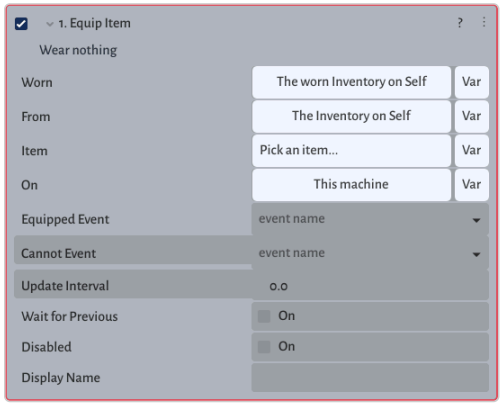
See the action work on its own with **Open Example** in the Add Action window, or open `examples/inventory/drop_loot/drop_loot.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Items	The items it can drop. Each has its own rarity, and rarer ones drop less often, as the chances below say. With none, only gold and what it carries drop.	empty
Item Chance	How likely it is to drop an item at all, from 0 (never) to 100 (always). 50 drops one about every other time. Var: a float.	50.0
Common Chance	How likely a Common item is, weighed against the other three chances. The four need not add up to 100, because they are shares. Var: a float.	60.0
Uncommon Chance	How likely an Uncommon item is, against the others. Var: a float.	25.0
Rare Chance	How likely a Rare item is, against the others. Var: a float.	12.0
Legendary Chance	How likely a Legendary item is, against the others. Keep it small, so finding one feels special. Var: a float.	3.0
Drops What It Carries	Whatever the dropper carries in its Inventory drops too, every time, beside what drops by chance, so a boss carrying its crown always drops it. Each can carry something different, set in its Inventory's Starts With. Var: a bool.	on
Varies By	Each item dropped gets its own numbers, up to this much better or worse than the item it was made from. Set to 20 it is a fifth either way. 0 drops them as made. Var: a float.	20.0
Gold Least	The least gold it drops. With Gold Most at 0, no gold drops. Var: an int.	0
Gold Most	The most gold it drops. Each drop is somewhere between the two. Var: an int.	0
Gold Goes Into	The number variable the gold is added to on whoever picks it up, such as the player's gold. It belongs to them, not to this machine, so type its name as their template has it.	gold
Item Sound	The sound played when an item it dropped is picked up. Empty plays none.	empty
Gold Sound	The sound played when its gold is picked up. Empty plays none.	empty
Dropped Event	Sent to this machine as things drop, carrying what dropped. It goes once for the item picked by chance, and once for each kind of item it carried, however many of it there were. Gold sends it, carrying the amount, only when no item dropped with it. Empty sends nothing.	empty

Equip Item

Wears something out of an inventory, and makes its stat changes while it is on.



Equip Item as it appears in the Actions tab.

When to use it

A sword going into the Weapon slot comes out of the pack, goes in the slot named in its Slot, and adds its 2 damage while it is worn. The bonus stops as soon as it comes off. Whatever was in that slot comes off first and goes back into the pack. The character needs two Inventories, one with Slot Names filled in for what it wears and an ordinary one it carries things in. With both left empty, it finds them on this machine's node.

> If there is no room in the inventory for the thing being taken off, NOTHING happens at all and Cannot Event is sent. Swapping a sword should never lose the old one, and a template cannot put back something the game has already dropped.

See also

Unequip Item 299, **Modify Stat** 268, **Inventory Has** 291, **Move Item** 292

Try it

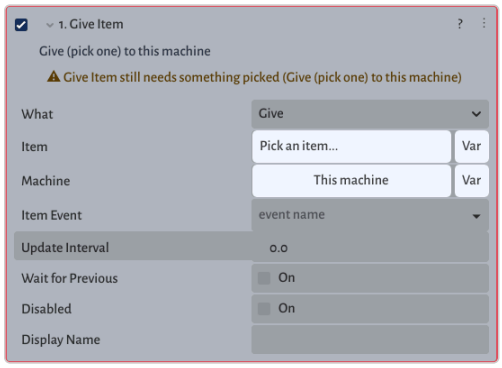
See the action work on its own with **Open Example** in the Add Action window, or open `examples/inventory/equip_item/equip_item.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Worn	The inventory with Slot Names, holding what is worn. Leave it empty for the worn Inventory on the node this machine works on. Var: a Node.	empty
From	The inventory the item is carried in. Leave it empty for the ordinary Inventory on the same node. Var: a Node.	empty
Item	The item to wear. Press Var to read it from a variable, such as the one a panel is showing. Var: a Resource.	empty
On	The machine whose stats the item changes. It can be a Machine node or a node with one, such as the player. Empty changes this machine's stats. Var: a Node.	empty
Equipped Event	Event sent once it is worn, carrying the item, so a HUD can name what was put on. Empty sends nothing.	empty
Cannot Event	Event sent when it cannot be worn, because it is not being carried, it goes in no slot that inventory has, or what was worn there had nowhere to go. Empty sends nothing.	empty

Give Item

Gives an item to a machine, usually the player's. Every number the item changes is changed there, the item counts as held for this run, and it goes into the collection of everything ever found. Remove undoes that item's changes again.



Give Item as it appears in the Actions tab.

When to use it

When the player takes something off a pedestal or buys it from a shop shelf, this goes in the state that follows. Remove is for an item swapped out, sold or lost, and it puts back only what that one item changed, whatever else has been given or taken since. Pick Item chooses the item, and this hands it over.

> Each of the item's changes names a number variable on the receiving machine (damage, speed, fire delay), so that machine needs those variables for the item to be worth anything.

See also

Pick Item 295, **Show Item** 297, **Modify Stat** 268, **New Run** 293

Try it

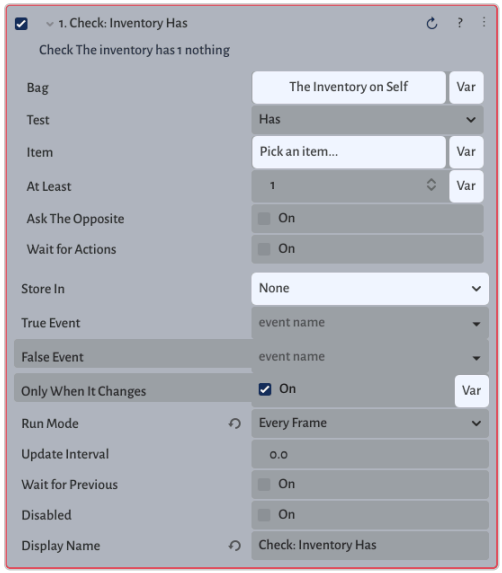
See the action work on its own with **Open Example** in the Add Action window, or open `examples/inventory/give_item/give_item.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
What	Give hands the item over and makes its changes. Remove undoes that item's changes and stops it counting as held, though it stays in the collection. Give is the default. Choices: Give, Remove.	Give
Item	The item to give. An item is a file you make once. In the FileSystem dock, right-click a folder, choose New Resource... and pick the item type, then fill in its name, the line saying what it does, its picture, its rarity, its price, and the numbers it changes. Press Var to give the item held in a variable, such as the one Pick Item has just chosen. With no item here, nothing is given and a warning says there was nothing to give. Var: a Resource.	empty
Machine	The machine that receives it, as a Machine node or a node with one, such as the player. Empty, the default, gives it to this machine, for when the player's machine does the picking up. Point it at a node with no machine on it or above it and nothing is handed over. The item is not marked as held, none of its numbers change, and a warning says there is nobody to give it to. Var: a Node.	empty
Item Event	Shown only when What is Give. An event sent to this machine straight afterwards, carrying the item, so another state can put its name and its line on the screen with Show Item. Empty sends nothing.	empty

Inventory Has CHECK

Asks what an inventory is carrying. Has it enough of something, is it full, is it empty?



Inventory Has as it appears in the Actions tab.

When to use it

A locked door's transition asks Has, with the key as the item and At Least 1. A pickup can check Is Full first and show "your pack is full". Like every question it can also be asked by a Check action inside a state, to store the answer or send an event.

> An inventory can be full and still have room for more of something it already has, if that pile is not full. Is Full means every slot is in use, not that nothing more will fit.

See also

Count Item 286, **Add Item** 285, **Remove Item** 296

Try it

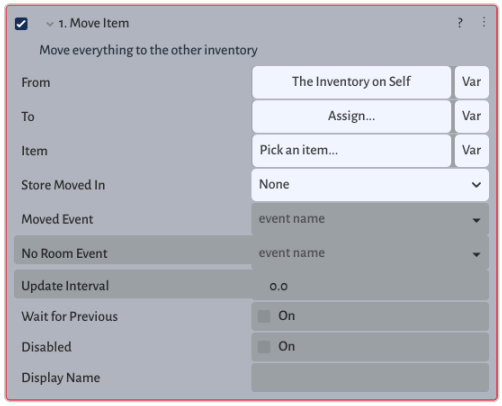
See the action work on its own with **Open Example** in the Add Action window, or open `examples/inventory/inventory_has/inventory_has.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Bag	The inventory to ask about. Leave it empty for the Inventory on the node this machine works on. Var: a Node.	empty
Test	What to ask. Has looks for the item below. Is Full asks whether every slot is in use. Is Empty asks whether it is carrying nothing at all. Choices: Has, Is Full, Is Empty.	Has
Item	Only when asking Has. The item to look for. Press Var to read it from a variable. Var: a Resource.	empty
At Least	Only when asking Has. How many of it there must be for the answer to be yes. Var: an int.	1
Ask The Opposite	Turns the answer round, so Has becomes "it has not got enough" and Is Full "there is a slot free". It turns round the true-or-false answer the Check stores as well, which swapping the True Event and False Event over cannot do.	off
Wait for Actions	Holds the question back until every other action in the state has finished, so it asks after the state's own actions have added or taken items. Left off, the default, it is asked as soon as the state runs.	off
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Move Item

Moves something from one inventory to another, without ever making or losing any.



Move Item as it appears in the Actions tab.

When to use it

Taking loot out of a chest is a Move Item from the chest's Inventory to the player's. What the other inventory has no room for stays where it was, and Store Moved In says how many really moved. A chest that could not be emptied still holds the rest. Name no item and everything in the inventory goes. Name one with How Many at 0 to move every one of that item.

> The count is taken out of the first inventory before it is put in the second, and anything the second turns out not to take goes straight back. Nothing is ever created or destroyed by a move, whatever the two inventories' limits are. Moving everything asks for each kind in turn. When the pack has room for only some, the rest stays where it was.

See also

Add Item 285, **Remove Item** 296, **Count Item** 286, **Inventory Has** 291

Try it

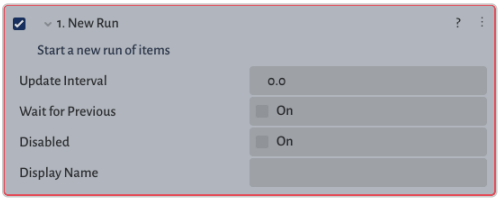
See the action work on its own with **Open Example** in the Add Action window, or open `examples/inventory/move_item/move_item.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
From	The inventory it comes from. Leave it empty for the Inventory on the node this machine works on. Var: a Node.	empty
To	The inventory it goes to. This one has to be filled in, because there is no sensible guess for where something should end up. Var: a Node.	empty
Item	The item. Press Var to read it from a variable. Leave it empty, with no variable either, and everything in the inventory moves, one kind at a time. That is how you empty a chest, take what a dropped pile holds, or strip what a defeated enemy was carrying. How Many is hidden while it is empty. Var: a Resource.	empty
Store Moved In	The variable how many really moved goes into. Empty stores nothing.	empty
Moved Event	Event sent when every one of them moved. Empty sends nothing.	empty
No Room Event	Event sent when some or none of them did, because the other inventory had no room. Empty sends nothing.	empty
How Many	How many of it. Zero or less moves every one there is. Press Var to read the number from a variable.	—

New Run

Starts a new run. Every item can turn up again, and nothing counts as held any more. The collection of everything ever found is left untouched.



New Run as it appears in the Actions tab.

When to use it

A game played in runs, such as a dungeon crawl that starts over when the player dies, needs this in the state that sets a fresh run going, or behind the menu button that starts one. Pick Item can then offer every item again, and the list of what the player holds is empty.

> It clears the lists only. Numbers that items already changed on a machine still standing stay changed, so start a run with a fresh scene.

See also

Pick Item 295, **Give Item** 290, **Change Scene** 220, **Save Game** 235

Try it

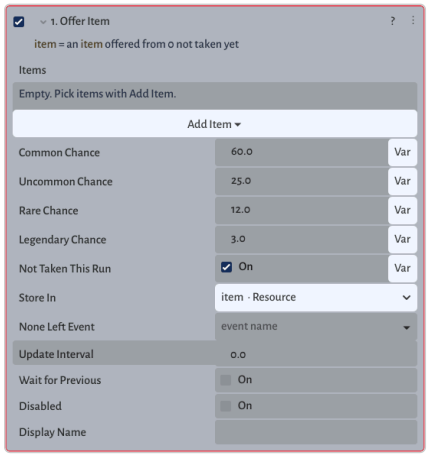
See the action work on its own with **Open Example** in the Add Action window, or open `examples/inventory/new_run/new_run.tscn`.

Its settings

This one has no settings of its own beyond the four every action has.

Offer Item

Chooses an item to show without spending it, for a pedestal the player may walk away from, a shop shelf or a chest. Rarer items come up less often, an item the player has already been given this run is left out, and the one chosen goes into a variable for Show Item to put on screen and Give Item to hand over.



Offer Item as it appears in the Actions tab.

When to use it

The player looks at a shop shelf and leaves without buying. Offering costs nothing, and only Give Item uses an item up, so what was passed over can come round again later in the same run. When every pedestal should show something new whether or not the player takes it, Pick Item does that.

> Because offering spends nothing, a pedestal that never runs Give Item can keep offering the same item over and over.

See also

Pick Item 295, **Give Item** 290, **Show Item** 297, **New Run** 293

Try it

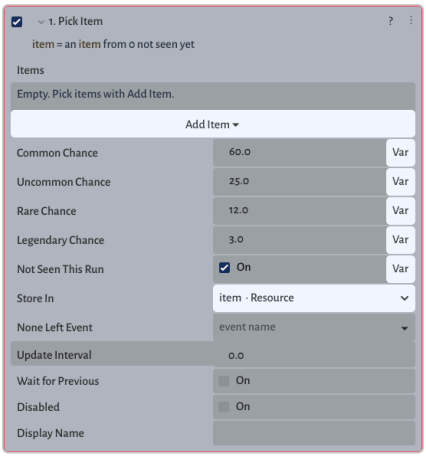
See the action work on its own with **Open Example** in the Add Action window, or open `examples/inventory/offer_item/offer_item.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Items	The items to choose from, a row for each with its picture and name. Press Add Item and pick one, your own items first. An item whose Unlocked By names a true-or-false global variable is only offered once that variable is true, so one list can hold things the player has not earned yet.	empty
Common Chance	How likely a Common item is against the others. The four chances are weights rather than percentages, so only their sizes next to each other matter. It starts at 60, so against the other three defaults a Common item comes up most of the time. Set it to 0 to leave Common items out altogether. Var: a float.	60.0
Uncommon Chance	How likely an Uncommon item is, weighed against the other three. It starts at 25, a little under half as often as a Common one. Set it to 0 to leave Uncommon items out. Var: a float.	25.0
Rare Chance	How likely a Rare item is, weighed against the other three. It starts at 12, about a one in eight chance with the other defaults. Set it to 0 to leave Rare items out. Var: a float.	12.0
Legendary Chance	How likely a Legendary item is, weighed against the other three. It starts at 3, so one hardly ever turns up. Set it to 0 to leave Legendary items out. Var: a float.	3.0
Not Taken This Run	On, the default, an item the player has already been given this run is left out, so nothing they are already carrying is offered again. Off, anything on the list can be offered however often it has been taken. Only being given puts an item out of reach, and being looked at and left does not. New Run clears the record. Var: a bool.	on
Store In	The Resource variable the chosen item goes into. It starts at "item". Point Show Item and Give Item at the same name. Left empty, the chosen item is lost and a warning asks for a Resource variable beside Store In.	item
None Left Event	Sent when nothing could be chosen, because the list is empty, the player already has everything on it, or nothing is unlocked yet. The variable is emptied as well, so send this somewhere that closes the pedestal or puts out something else. Empty sends nothing.	empty

Pick Item

Chooses an item at random from a list. Rarer items come up less often, an item that has already turned up this run is not offered again, and the one chosen goes into a variable for Give Item and Show Item to use.



Pick Item as it appears in the Actions tab.

When to use it

Put it in the state that dresses a reward, such as a pedestal deciding what stands on it, a chest deciding what is inside, or a shop stocking its shelves. It only chooses. Show Item puts the picture and the words on screen, and Give Item hands the thing over when the player takes it. Choosing uses an item up, so every pedestal shows something new whether or not the player takes it. Where the player may walk away and what they left should come round again, use Offer Item.

> An item counts as having turned up as soon as it is chosen, not when it is given, so an item on a pedestal the player walks past is used up for the rest of that run.

See also

Offer Item 294, **Give Item** 290, **Show Item** 297, **New Run** 293

Try it

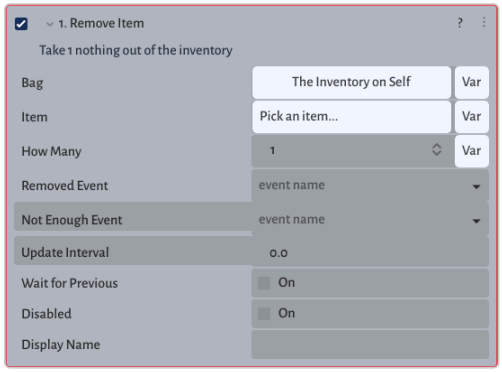
See the action work on its own with **Open Example** in the Add Action window, or open `examples/inventory/pick_item/pick_item.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Items	The items to choose from, a row for each with its picture and name. Press Add Item and pick one, your own items first. An item whose Unlocked By names a true-or-false global variable is only offered once that variable is true, so one list can hold things the player has not earned yet.	empty
Common Chance	How likely a Common item is against the others. The four chances are weights rather than percentages, so only their sizes next to each other matter. It starts at 60, so against the other three defaults a Common item comes up most of the time. Set it to 0 to leave Common items out altogether. Var: a float.	60.0
Uncommon Chance	How likely an Uncommon item is, weighed against the other three. It starts at 25, a little under half as often as a Common one. Set it to 0 to leave Uncommon items out. Var: a float.	25.0
Rare Chance	How likely a Rare item is, weighed against the other three. It starts at 12, about a one in eight chance with the other defaults. Set it to 0 to leave Rare items out. Var: a float.	12.0
Legendary Chance	How likely a Legendary item is, weighed against the other three. It starts at 3, so one hardly ever turns up. Set it to 0 to leave Legendary items out. Var: a float.	3.0
Not Seen This Run	On, the default, an item that has already turned up this run is left out, so the same thing never appears twice in one run. Off, anything on the list can come up again and again. New Run clears the record of what has turned up. Var: a bool.	on
Store In	The Resource variable the chosen item goes into. It starts at "item". Point Give Item and Show Item at the same name. Left empty, the chosen item is lost and a warning asks for a Resource variable beside Store In.	item
None Left Event	Sent when nothing could be chosen, because the list is empty, everything on it has turned up already, or nothing is unlocked yet. The variable is left holding nothing, so send this somewhere that closes the pedestal or puts out a different reward. Empty sends nothing.	empty

Remove Item

Takes something out of an inventory, or says there was not enough.



Remove Item as it appears in the Actions tab.

When to use it

A key spent on a door, or coins paid at a shop, come out with this. If there is not enough it takes NOTHING at all and sends Not Enough Event, since a template could not undo a price half paid.

> It asks before it takes, so the two events are exclusive. One of them always happens and never both.

See also

Add Item 285, **Inventory Has** 291, **Move Item** 292, **Count Item** 286

Try it

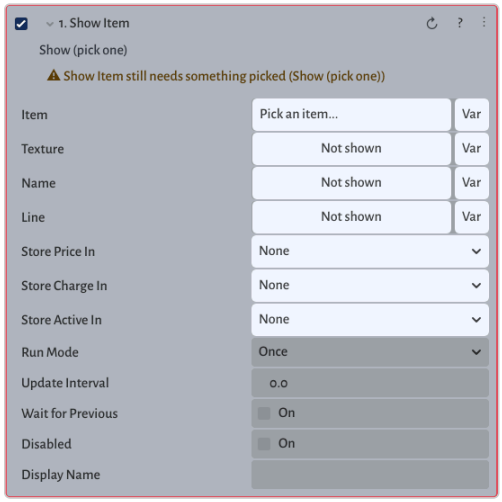
See the action work on its own with **Open Example** in the Add Action window, or open `examples/inventory/remove_item/remove_item.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Bag	The inventory to take it from. Leave it empty for the Inventory on the node this machine works on. Var: a Node.	empty
Item	The item. Press Var to read it from a variable. Var: a Resource.	empty
How Many	How many of it. Press Var to read the number from a variable, for a price that changes. Var: an int.	1
Removed Event	Event sent when every one of them came out. Empty sends nothing.	empty
Not Enough Event	Event sent when there were not enough, with none taken. Leave the door shut on this one. Empty sends nothing.	empty

Show Item

Shows an item, with its picture on a sprite or a picture box, its name in one label and its line in another. It can also copy the item's price, its charge and whether it is an active item into variables. With no item, the picture and the labels are cleared.



Show Item as it appears in the Actions tab.

When to use it

A pedestal shows what is on it, a shop shelf the price beside it, a popup the name of what was just picked up, a heads-up display the active item's slot. On Once, the Run Mode it starts with, it writes as the state starts, so put it in a state the game comes back to whenever the item changes.

> On Every Frame it keeps up with the item its variable holds. As the player moves the highlight along a shelf, the name, the line and the price follow without the state being left and entered again. On Once, as it starts, it shows one item and stops.

See also

Pick Item 295, **Give Item** 290, **Set Text** 213, **Set Texture** 214

Try it

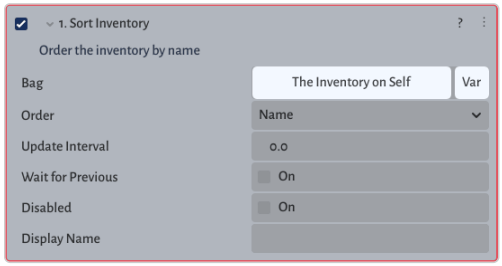
See the action work on its own with **Open Example** in the Add Action window, or open `examples/inventory/show_item/show_item.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Item	The item to show. Press Var to show the one held in a variable, such as the one Pick Item has just chosen. With no item, the picture and both labels are cleared. That empties a pedestal once its item has been taken. Var: a Resource.	empty
Texture	The Sprite2D, Sprite3D or TextureRect that shows the item's picture. Empty shows no picture. Any other kind of node shows nothing, and a warning names it and says which kinds work. Var: a Node.	empty
Name	The label that shows the item's name. A Label, a RichTextLabel, a Label3D standing in the world, or a Button. Empty shows no name. Var: a Node.	empty
Line	The label that shows the item's line, the sentence saying what it does. The same kinds of node as above. Empty shows no line. Var: a Node.	empty
Store Price In	Copies the item's price into this variable, so a shop can compare it with the player's coins before letting them buy. Empty stores nothing, and with no item it stores 0.	empty
Store Charge In	Copies how many rooms an active item takes to charge into this variable, so a bar on the screen knows how full to draw. Empty stores nothing, and with no item it stores 0.	empty
Store Active In	Copies whether this is an active item (one the player sets off with a button) into a true-or-false variable, so a display can show the slot only when there is something in it. Empty stores nothing.	empty

Sort Inventory

Puts an inventory's piles in order, so a panel draws them tidily.



Sort Inventory as it appears in the Actions tab.

When to use it

Sort when a panel opens or after a fight. Every frame is too often. Nothing is added or taken away, and a panel draws the piles in the new order.

> Piles holding nothing sink to the end, out of the way.

See also

Add Item 285, **Count Item** 286, **Move Item** 292

Try it

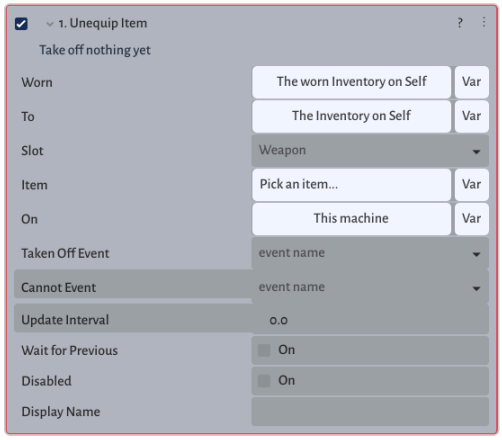
See the action work on its own with **Open Example** in the Add Action window, or open `examples/inventory/sort_inventory/sort_inventory.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Bag	The inventory to put in order. Leave it empty for the Inventory on the node this machine works on. Var: a Node.	empty
Order	What to order them by. Name is A to Z. Type groups weapons with weapons and potions with potions, using the Type written on each item, and orders by name inside each group. Rarity puts the rarest first. Choices: Name, Type, Rarity.	Name

Unequip Item

Takes off what is worn in a slot, back into the inventory, with its stat changes undone.



Unequip Item as it appears in the Actions tab.

When to use it

Putting a sword away takes its 2 damage off again. With no room in the inventory it stays on and Cannot Event is sent. It is never dropped where nobody can reach it.

> Equip Item takes the old thing off by itself when it wears a new one in the same slot. This one is for putting something away.

See also

Equip Item 289, **Modify Stat** 268, **Add Item** 285, **Inventory Has** 291

Try it

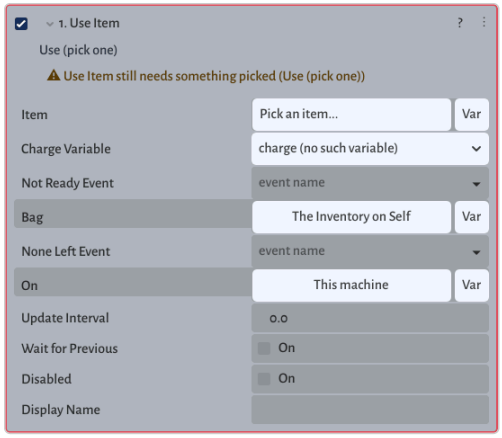
See the action work on its own with **Open Example** in the Add Action window, or open `examples/inventory/unequip_item/unequip_item.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Worn	The inventory with Slot Names, holding what is worn. Leave it empty for the worn Inventory on the node this machine works on. Var: a Node.	empty
To	The inventory it goes back into. Leave it empty for the ordinary Inventory on the same node. Var: a Node.	empty
Slot	The slot to empty, picked from the worn inventory's Slot Names, such as Weapon. Leave it empty and name the item below instead.	empty
Item	The item to take off, when no slot is named above. It comes off whichever slot is named in its Slot. Press Var to read it from a variable, since an Inventory Grid hands over the item the player chose, not the slot it sits in. Var: a Resource.	empty
On	The machine whose stats the item was changing. Empty changes this machine's stats. Var: a Node.	empty
Taken Off Event	Event sent once it is off and back in the inventory, carrying the item. Empty sends nothing.	empty
Cannot Event	Event sent when nothing is worn in that slot, or there is no room in the inventory for it. Empty sends nothing.	empty

Use Item

Uses an item. A potion, food or a scroll (an item made Used Up When Used) comes out of the inventory and makes the changes set under its When Used, wearing off after its For Seconds if it has them. An active item, set off with a button once it has charged, sends its Use Event to this machine and its charge starts again from 0.



Use Item as it appears in the Actions tab.

When to use it

Wire it to the key that drinks a potion, eats food or sets off the player's active item. Pick the item here, or pick a variable holding it when the item changes during the game, such as the active item the player last picked up. The state that catches an active item's Use Event does what that item does, such as setting off a bomb, so one button serves every active item in the game.

> An item's page says which way it is used, so once an item is picked only the settings for its way show.

See also
Give Item 290, **Show Item** 297, **Modify Stat** 268, **Send Event** 31

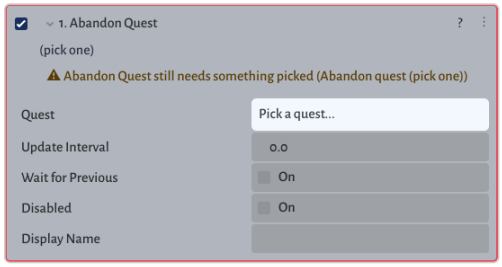
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/inventory/use_item/use_item.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Item	The item to use, picked from your items or made with New item.... Used when no variable is picked above. With neither, nothing is used and a warning says no item is picked. Var: a Resource.	empty
Charge Variable	For an active item. It is the number variable holding how much charge has built up since it was last used, and it starts at "charge". It is counted in rooms cleared and set back to 0 each time the item is used. It does not fill by itself, so add 1 to it when a room is cleared.	charge
Not Ready Event	For an active item. It is sent when there is nothing to use, or when the charge has not filled yet, so play a dull click or flash the slot red. Empty sends nothing.	empty
Bag	For an item used up when used, this is the Inventory it comes out of. Empty uses the Inventory on this machine's node. With no Inventory there, a warning says to add one. Var: a Node.	empty
None Left Event	For an item used up when used, this is sent when there are none left in the inventory, and nothing is used. Empty sends nothing.	empty
On	Who uses the item and is changed by it. It can be a Machine, or a node with one, such as the player drinking a potion chosen in an inventory. Empty is this machine. Var: a Node.	empty

Abandon Quest

The player gives a quest up. It shows as abandoned, its steps stop counting, no reward is given, and its Abandoned Event is sent.



Abandon Quest as it appears in the Actions tab.

When to use it

Wire it to whatever lets the player walk away from a quest, such as a Give Up button in a journal or a "never mind" answer to whoever gave it.

> Start Quest begins an abandoned quest again from its first step. Quest Is can ask whether a quest was abandoned, so whoever gave it can say so.

See also

Start Quest 306, **Fail Quest** 303, **Quest Is** 304

Try it

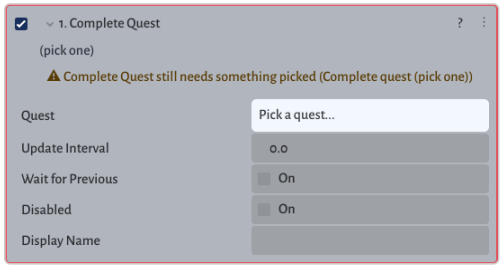
See the action work on its own with **Open Example** in the Add Action window, or open `examples/quests/abandon_quest/abandon_quest.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Quest	The quest given up. Left empty, or set to a quest that is not under way, it does nothing, and a warning says why.	empty

Complete Quest

Marks a quest done now and gives its reward, whatever its steps say.



Complete Quest as it appears in the Actions tab.

When to use it

Some quests end when the game says so, not when their steps are done. Perhaps the player is thanked by whoever gave the quest, or beats a boss in a way no step counts. A quest that was never started is started and finished in one go.

> Every step is ticked off, so the Quest List shows it finished. The reward is given once only, so completing a quest that is already done does nothing.

See also

Start Quest 306, **Fail Quest** 303, **Quest Is** 304

Try it

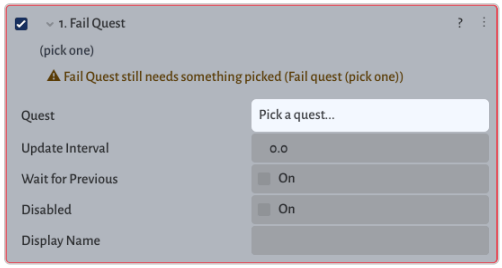
See the action work on its own with **Open Example** in the Add Action window, or open `examples/quests/complete_quest/complete_quest.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Quest	The quest that is done. Left empty, nothing happens and a warning says there is no quest to complete.	empty

Fail Quest

Fails a quest under way. It shows as failed in the Quest List, its reward is not given, and its Failed Event is sent.



Fail Quest as it appears in the Actions tab.

When to use it

Put it where a quest is lost, such as the state a Timer leads to when the prisoner has not been freed in time.

> Only a quest under way can fail. One not begun, or already done, is left alone, and a warning names it and says where it stands. Start Quest begins a failed quest again from its first step, for a second try.

See also

Start Quest 306, **Complete Quest** 302, **Quest Is** 304, **Timer** 68

Try it

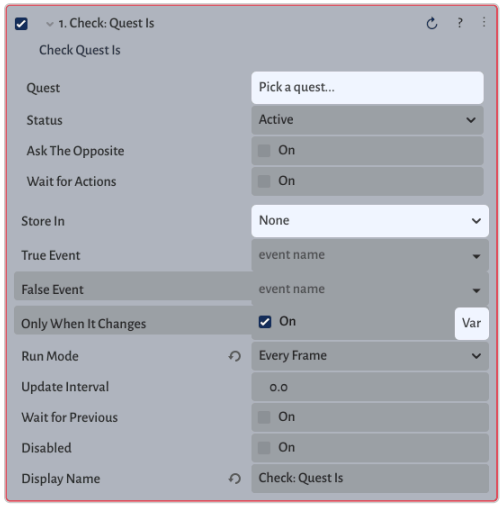
See the action work on its own with **Open Example** in the Add Action window, or open `examples/quests/fail_quest/fail_quest.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Quest	The quest that has failed. Left empty, nothing happens and a warning says there is no quest to fail.	empty

Quest Is CHECK

Asks where a quest has got, whether not started, under way, done or failed.



Quest Is as it appears in the Actions tab.

When to use it

Whoever gives a quest sorts their answers with it. They offer the quest while it is Not Started, ask how it is going while it is Active, and say thank you once it is Done. A door can wait for Done, and a guard turn hostile on Failed.

> A quest's status lasts the whole game, across scenes, and Save Game keeps it.

See also

Start Quest 306, **Complete Quest** 302, **Fail Quest** 303

Try it

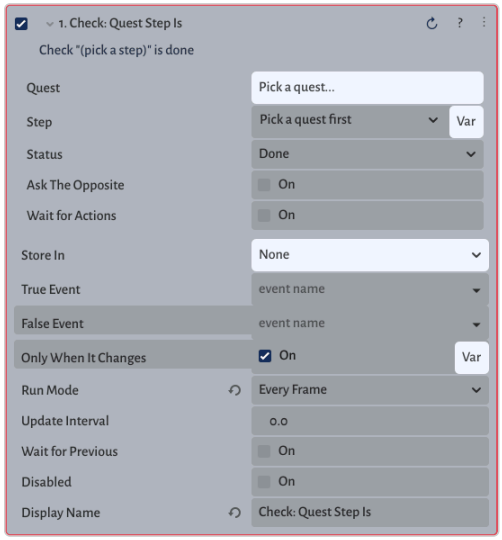
See the action work on its own with **Open Example** in the Add Action window, or open `examples/quests/quest_is/quest_is.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Quest	The quest to ask about. With none picked the answer is always no, and a warning asks you to pick one.	empty
Status	What to ask about, one of Not Started, Active (under way), Done, Failed, or Abandoned (given up by the player). It starts on Active. Choices: Not Started, Active, Done, Failed, Abandoned.	Active
Ask The Opposite	Turns the answer round, so Done becomes "it is not done yet". It turns round the true-or-false answer the Check stores as well, which swapping the True Event and False Event over cannot do.	off
Wait for Actions	Holds the question back until every other action in the state has finished, so it asks after the state's own actions have started, finished or failed the quest. Left off, the default, it is asked as soon as the state runs.	off
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Quest Step Is CHECK

Asks where one step of a quest has got, whether it is done, not done, or the one to do now.



Quest Step Is as it appears in the Actions tab.

When to use it

Whoever gave the quest can say "you found the mushrooms, now bring them to me" once the first step is done, and a door can open once its step is the one to do now. Both ask about a step, not the quest.

> A quest not begun has no step done and none to do now.

See also
Quest Is 304, **Start Quest** 306

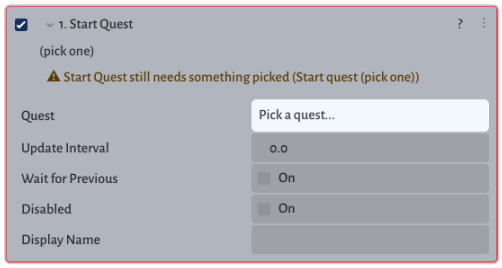
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/quests/quest_step_is/quest_step_is.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Quest	The quest to ask about.	empty
Step	Which of its steps, picked by what it says. Choices: Pick a quest first. Var: any kind.	Pick a quest first
Status	What to ask. The choices are Done, Not Done, or To Do Now (its quest under way, the step not done and, with In Order, every step before it done). Choices: Done, Not Done, To Do Now.	Done
Ask The Opposite	Turns the answer round, so asking whether a step is done becomes "that step is not done yet". It turns round the true-or-false answer the Check stores as well, which swapping the True Event and False Event over cannot do.	off
Wait for Actions	Holds the question back until every other action in the state has finished, so it asks after the state's own actions have moved the quest on. Left off, the default, it is asked as soon as the state runs.	off
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Start Quest

Begins a quest. It appears in the Quest List, and its steps start ticking themselves off as the game is played. When they are all done the quest is done and its reward is given.



Start Quest as it appears in the Actions tab.

When to use it

Put it in the state where the game hands a quest out, such as the one a notice board leads to when it is read. A quest given in a dialogue needs no action, because the answer's Start Quest setting begins it.

> A quest already under way or done is left as it is, so the action can sit in a state that is visited again and again. A failed quest begins again from its first step.

See also

Complete Quest 302, **Fail Quest** 303, **Quest Is** 304, **Quest List** 325

Try it

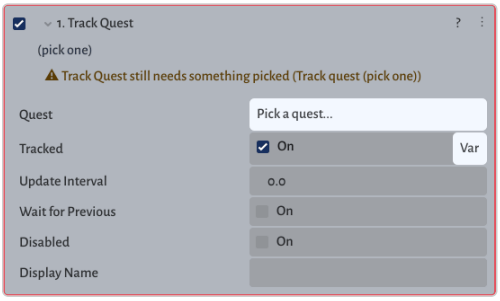
See the action work on its own with **Open Example** in the Add Action window, or open `examples/quests/start_quest/start_quest.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Quest	The quest to begin, a file made with Create New and NodeWorthy Quest in the FileSystem dock. Left empty, nothing begins and a warning says Start Quest has no quest to begin.	empty

Track Quest

Tracks a quest, or stops tracking it. Tracked quests are marked on the Minimap, and a Quest List set to Only Tracked shows only them.



Track Quest as it appears in the Actions tab.

When to use it

A player with many quests picks the ones to follow. Put it behind a Track button in a journal, or take an old quest off the map once another begins. Every quest is tracked as it begins, so a game with a few quests never needs it.

> A quest not begun cannot be tracked, and a warning says to start it with Start Quest first. Tracking lasts until the game ends, and Save Game keeps it.

See also

Start Quest 306, **Quest List** 325, **Minimap** 322

Try it

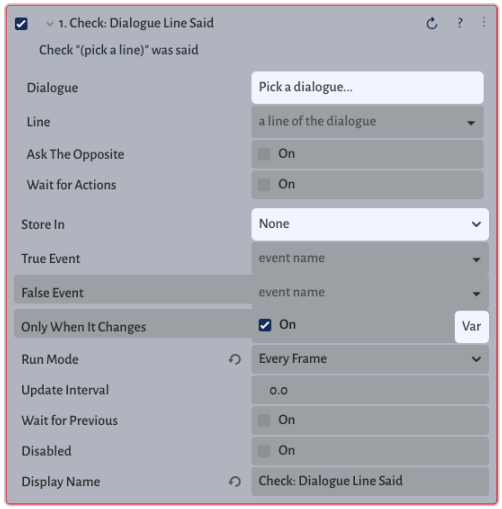
See the action work on its own with **Open Example** in the Add Action window, or open `examples/quests/track_quest/track_quest.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Quest	The quest to track, or to stop tracking. It must have begun.	empty
Tracked	On tracks it, and off stops tracking it. Var: a bool.	on

Dialogue Line Said CHECK

Asks whether a line of a dialogue has been said this game.



Dialogue Line Said as it appears in the Actions tab.

When to use it

Characters who remember. A villager greets the player differently the second time, and a door opens once a guard has told the player the password.

> Every line said and every answer picked is remembered for the rest of the game, and Save Game keeps it.

See also
Start Dialogue 309, **Quest Is** 304

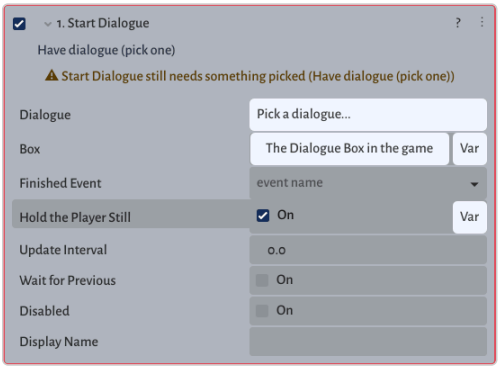
Try it
See the action work on its own with **Open Example** in the Add Action window, or open `examples/dialogue/dialogue_line_said/dialogue_line_said.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Dialogue	The dialogue to ask about.	empty
Line	Which of its lines, picked from the list of its lines.	empty
Ask The Opposite	Turns the answer round into "that line has not been said yet", and an answer can be offered only until it has been heard. It turns round the true-or-false answer the Check stores as well, which swapping the True Event and False Event over cannot do.	off
Wait for Actions	Holds the question back until every other action in the state has finished. Left off, the default, it is asked as soon as the state runs.	off
True Event	Set on the Check that asks this. The event sent when the answer is yes. Empty sends nothing.	empty
False Event	Set on the Check. The event sent when the answer is no. Empty sends nothing.	empty
Store In	Set on the Check. A bool variable the answer goes into, true or false. Empty keeps nothing.	empty
Only When It Changes	Set on the Check, for while it is asked on Every Frame. It sends an event only when the answer changes, not every frame it stays the same.	on

Start Dialogue

Plays a dialogue in the scene's Dialogue Box. Each line comes in turn, a letter at a time, with the player's answers as buttons, following the arrows of the dialogue until it ends.



Start Dialogue as it appears in the Actions tab.

When to use it

Anyone the player talks to. Put it in a Talking state and go back on FINISHED. Write the dialogue as a graph first. In the FileSystem dock, Create New and NodeWorthy Dialogue make one, and it opens in the NodeWorthy panel with a box for each line and an arrow for each answer.

The state waits until the dialogue ends, so nothing else moves it on. A variable's name in curly brackets in the words or the answers, such as {coins}, is filled in as the line is said. An answer's own settings, opened with the cog beside it in the dialogue's graph, can offer it only when a check is true, send an event and change a variable, all on this machine. An event that takes the machine to another state ends the dialogue there. Leaving the state early closes the box. A second dialogue begun while one is still going takes the box over, and the first one ends.

See also

Show Choices 217, **Set Text** 213, **Dialogue Line Said** 308, **Set Visible** 216

Try it

See the action work on its own with **Open Example** in the Add Action window, or open `examples/dialogue/start_dialogue/start_dialogue.tscn`.

Its settings

SETTING	WHAT IT DOES	STARTS AS
Dialogue	The dialogue to play, a file made with Create New and NodeWorthy Dialogue. With none, or one with no lines yet, the action finishes straight away and a warning says which.	empty
Box	The Dialogue Box to show it in. Leave it empty, as it comes, for the one in the game, and pick one only when there is more than one. With none in the scene, a warning says how to add one. Right-click the level in the Scene dock and pick NodeWorthy > Add To Screen > Dialogue Box. Var: a Node.	empty
Finished Event	An event sent when the dialogue ends, as well as the state completing. Empty, as it comes, sends nothing.	empty
Hold the Player Still	On, as it comes, keeps the player where they are while the dialogue lasts, paused until it ends, as most games do. Off lets them walk away mid-dialogue. Var: a bool.	on

Bar NODE

A bar across the screen that follows a character's number, with its name over it. It suits a boss's health along the top of the screen, a shield, or a rage meter. Hits drain it smoothly, with a pale strip showing what was just lost.



Bar in a running game.

When to use it

A boss fight, or any number big enough to deserve its own bar. Pick the character and the number. The number's Max is a full bar and nothing else needs setting. For a health bar over an enemy's head, tick Over Its Head, or right-click the enemy and choose Give It, Health Bar.

> It goes when the character is gone, so it needs nothing to hide it after a win.

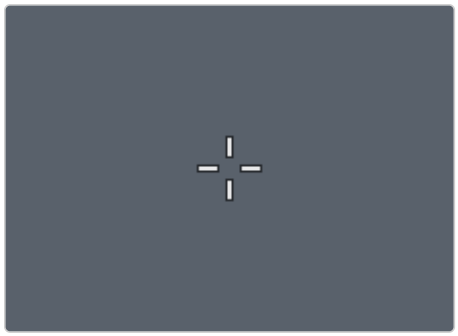
See also
Globe 315, **Icon Meter** 318, **Hurt Box** 317

Its settings, in the Inspector

SETTING	WHAT IT DOES	STARTS AS
Character	Whose number it shows, picked in the scene. Empty is the player, the node in the player group.	empty
Shows	The character's number variable it follows, such as health, picked from the numbers your templates keep.	health
Title	The name written over it, such as the boss's name. Empty writes none.	empty
Color	The colour of the bar.	0.72, 0.1, 0.1, 1
Width	How wide the bar is, in pixels.	640.0
Thickness	How tall the bar is, in pixels.	18.0
Appears Within	Shows it only once the player comes this close to the character, in pixels in 2D or metres in 3D, so a boss's bar comes up as the player walks into its hall. Once up, it stays up. 0 shows it always.	0.0
Shows Number	Writes the number on the bar, such as 180 / 250.	off
Over Its Head	Off to begin with, and the bar stays where it is put on the screen. Ticked, it rides just above its Character in the game, in 2D and 3D, and hides while the character is off the screen. Give It, Health Bar on a character adds a small bar with this ticked.	off

Crosshair NODE

The aiming mark in the middle of a shooter's screen. It opens out and closes in to show how steady the aim is.



Crosshair in a running game.

When to use it

Any game where the player aims at the middle of the screen. A machine changes what it shows with Set Property, such as Spread up while running, Bloom kicked open for each shot, Color red over an enemy, and Hit set when a shot lands. Read its Opening with Get Property and hand it to a Raycast as Spread, and shots scatter as widely as the mark shows.

> It fills the screen and draws in the middle of it, so put it under a CanvasLayer (a node that keeps things drawn on the screen, not in the world).

See also
Raycast 116, **Recoil** 193, **Set Property** 91, **Get Property** 80

Its settings, in the Inspector

SETTING	WHAT IT DOES	STARTS AS
Look	The shape of the mark. Choices: Cross, Cross And Dot, Dot, Circle.	Cross
Color	Its colour. Change it from a machine to say what is under it, red over an enemy and white over nothing.	1, 1, 1, 0.9
Length	How long each mark is, in pixels.	9.0
Thickness	How thick the marks are, in pixels.	2.0
Gap	The space left in the middle when the mark is fully closed, in pixels.	5.0
Spread	How far the marks open beyond the gap, in pixels. Set it from a machine, more while running or jumping and less while standing still. The marks ease to the new number by themselves.	0.0
Spread Scale	How much of Spread is shown, as a share. Aiming down the sights can set 0.3 to keep the marks tight without touching the Spread that walking and running set.	1.0
Spread Speed	How quickly the marks follow Spread. Higher snaps open and shut, and lower drifts.	14.0
Bloom	A kick open on top of Spread, in pixels. Set it from a machine on each shot. It closes again by itself, so holding fire keeps it open and letting go lets it settle.	0.0
Bloom Recovery	How fast Bloom closes again, in pixels a second.	40.0
Outline	Draws a dark edge round the marks so they can still be seen against a bright wall or sky.	on
Outline Color	The colour of that edge.	0, 0, 0, 0.6
Hit	Shows the hit marks for this many seconds. Set it from a machine when a shot lands, and it counts down to 0 by itself.	0.0
Hit Color	The colour of the hit marks.	1, 1, 1, 1

Data Node NODE

Gives one thing its own values from a Data Definition, such as this goblin's Health and Damage or that chest's Gold. Its fields show at the top of the thing's Inspector.



Data Node in a running game.

When to use it

Put one under anything that needs its own copy of a kind of data. Add it from the right-click menu under NodeWorthy > Give It > Data Node. Two goblins of one kind can then each lose their own Health. Get Data, Set Data and the Data check reach it with From set to Data On A Node.

> Values it never sets follow the Data Definition's Starts As, or the record picked in Starts As. Changes made while playing are not saved into any file.

See also
Get Data 63, **Set Data** 64, **Data** 61

Its settings, in the Inspector

SETTING	WHAT IT DOES	STARTS AS
Definition	The Data Definition its fields come from, such as Enemy Type. Until one is picked it has no fields, and a warning on the node says so.	empty
Starts As	A Data record of the same Definition to start from, such as a Goblin file. Its values are used for every field this node has not set itself. A record of another Definition gives nothing, and the node's warning sign says so.	empty

Dialogue Box NODE

Where a dialogue is shown. It shows who is speaking, their picture, and their words a letter at a time, with the player's answers as buttons under them.



Dialogue Box in a running game.

When to use it

Wherever characters talk. Write what the guard at the gate says as a dialogue, a graph of lines and answers, and the Start Dialogue action plays it in this box. The right-click menu puts it along the bottom of the screen, and you can drag it anywhere.

> It is hidden until a dialogue uses it, and hides again when the dialogue ends. Enter, Space, a click or a gamepad's bottom button (A / Cross) shows the rest of the words straight away, then moves on. The arrows, the D-pad or a stick move between the answers, round from the last to the first and never off them. While a dialogue goes on the box has the focus, so a button or an Inventory Grid elsewhere on the screen cannot take those presses. Whatever had it has it back when the dialogue ends. It takes its panel, font and buttons from the game's look, set in the Theme tab.

See also
Start Dialogue 309, **Show Choices** 217, **Set Text** 213, **Stat Panel** 328

Its settings, in the Inspector

SETTING	WHAT IT DOES	STARTS AS
Letters per Second	How fast the words appear, in letters each second. 0 shows every letter straight away.	40.0
Hide When Done	Hide the box when the dialogue ends. Turn it off to leave the last line showing.	on
Portrait Size	How big the speaker's picture is drawn, in pixels. A line with no picture shows none.	128, 128

Experience NODE

Experience and levels for the character it is under. Enough XP takes them up a level, their stats grow, and every machine is told.



Experience in a running game.

When to use it

Put one under the player in any game where they grow stronger. NodeWorthy > Give It > Experience adds it, and gives the player's template the xp and level variables it counts in where it has none. Anything that adds to the XP variable counts, such as a beaten enemy's XP Worth, a quest's reward or a Set Variable. A state waiting for LEVEL_UP can show the message and play a sound.

> It works with no numbers changed. Level 2 takes 100 XP, each level after takes half as much again, and each level gives +5 max health and +1 strength where the character keeps those numbers, with health filled up. Enough XP for several levels counts each one, and sends LEVEL_UP for each. The Stat Panel shows the XP out of what the next level needs, XP 40 / 150.

See also
Hurt Box 317, **Stat Panel** 328, **Set Variable** 49, **Start Quest** 306

Its settings, in the Inspector

SETTING	WHAT IT DOES	STARTS AS
XP Variable	The number variable of the character's machine that XP goes into.	xp
Level Variable	The number variable that holds the level, from 1.	level
First Level XP	The XP it takes to go from level 1 to level 2.	100.0
Each Level More	How much more each level after that needs. 1.5 is half as much again as the one before, and 1 the same every time.	1.5
Gains Each Level	What each level adds to the character's stats. Each names a variable and adds to it, or multiplies it.	empty
Heal on Level Up	Fill the health variable up to its Max on every level, as a reward.	on
Health Variable	The health variable filled up on a level.	health
Level Up Event	The event every machine in the scene is sent on a level, carrying the new level.	LEVEL_UP
Top Level	The highest level there is. 0 means there is no top.	0

Globe NODE

A glass globe that fills with a number, the way action role-playing games show health and mana. Red liquid shows health and blue shows mana, draining as it goes down and gently swaying, with the number written across it.



Globe in a running game.

When to use it

Put one in each bottom corner of the screen for the player's health and mana. Pick the number in Shows and it follows by itself, its Maximum too when the variable has a Max.

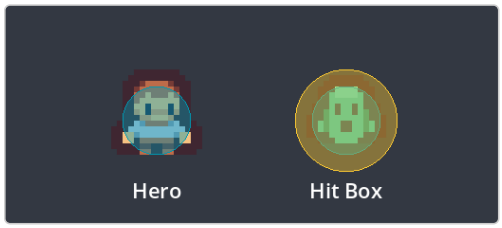
See also
Bar 310, **Icon Meter** 318, **Skill Bar** 327

Its settings, in the Inspector

SETTING	WHAT IT DOES	STARTS AS
Character	Whose number it shows. Leave it empty for the player, the node in the player group.	empty
Shows	The character's number variable it follows, such as health or mana, picked from the numbers your templates keep. Empty shows Value below, as a machine sets it with Set Property.	health
Value	The number shown, when Shows is empty and a machine sets it.	100.0
Maximum	The most the number can be, which fills the globe. It follows the variable's Max by itself when Shows is picked.	100.0
Liquid Color	The colour of what fills it, such as red for health and blue for mana.	0.78, 0.08, 0.1, 1
Diameter	How wide the globe is, in pixels.	150.0
Shows Number	Writes the number across it, such as 72 / 100.	on

Hit Box NODE

The part that does the hurting, such as a slime's body, a sword's edge or a spike trap. Whatever it touches that has a Hurt Box takes its damage.



Hit Box in a running game.

When to use it

Put one under anything that hurts to touch and set it to hurt the player's group. A sword's is switched on only for the swing, and hits each thing once per swing.

> It never hurts its own character. A body that always hurts hits again whenever the one it hurt is no longer safe, so standing on spikes keeps hurting at the pace of the victim's safety. Its area is drawn gold when Debug > Visible Collision Shapes is on.

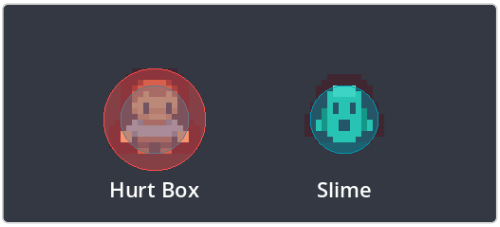
See also
Hurt Box 317, **Set Property** 91, **Wait** 69

Its settings, in the Inspector

SETTING	WHAT IT DOES	STARTS AS
Damage	How much a hit takes off, before the target's defence.	1.0
Damage Variable	Reads the damage from this number variable of the character's machine, for a strength that grows. Empty uses Damage.	empty
Hurts Group	What it hurts, such as the player or every copy of the enemy. Press the box and pick it in the scene. You can pick one thing by its name, Every Coin for every copy of the coin scene (including ones made while playing), or Everything under a node for a set of markers. NodeWorthy does the rest. Anything hurts anything with a Hurt Box but its own character.	empty
Hurts Group Variable	Reads what it hurts from this variable of the character's machine, letting the character's own Starting Value for it, set in the Inspector, decide. An Enemy set on other enemies leaves the player alone. Empty uses Hurts Group.	empty
On	Whether it is hurting now. Switch it off and on, with Set Property, for a swing.	on
Hit Again	When Safe hits again whenever the one it hurt is no longer safe, for a body that always hurts. Next Swing hits each thing once until it is switched off and on again, for a sword. Choices: When Safe, Next Swing.	When Safe
Knockback	How hard it knocks back what it hits. 1 is a good push, 0 none, 2 twice as hard. The Hurt Box's own Knockback is multiplied by it.	1.0
Hit Event	An event its own machine is sent when it lands a hit, carrying what it hit. Empty sends nothing.	empty
Shape	Its shape, either the same as its character's own body or a circle. Choices: Same As Body, Circle.	Same As Body
Radius	Only for Circle. How big it is, in pixels in 2D and metres in 3D.	16.0
Offset	Only for Circle. How far it sits from the character's middle, such as a sword's reach in front.	0, 0, 0

Hurt Box NODE

Makes something able to be hurt, such as the player, an enemy or a barrel. A Hit Box touching it takes its damage, less the defence, off its health, and its machine is told.



Hurt Box in a running game.

When to use it

Anything that can be hurt gets one from the right-click menu's NodeWorthy > Give It > Health. Its machine is sent HURT, carrying how much was taken, and DIED when its health reaches 0. When DIED comes, a barrel's machine can break it apart and drop its loot.

> It takes its shape from the character's collision shape, a little bigger to touch whatever bumps into the character, and needs no setting up. After a hit it is knocked back away from whatever hit it, and is safe for a moment, blinking. Given health again, as a respawn does, it can be hurt again after two seconds' safety. Its area is drawn red when Debug > Visible Collision Shapes is on.

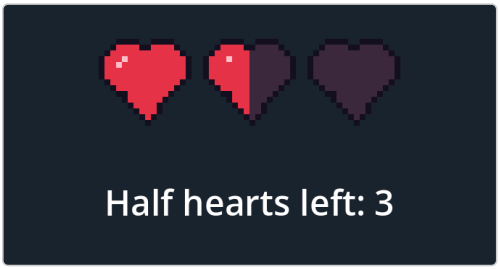
See also
Hit Box 316, **Stat Panel** 328, **Modify Stat** 268, **Icon Meter** 318

Its settings, in the Inspector

SETTING	WHAT IT DOES	STARTS AS
Health Variable	The number variable of the character's machine a hit takes off, such as health.	health
Defence Variable	A number variable taken off each hit first, as armour does. Empty takes nothing off. A hit always does at least 1.	empty
Invincibility Time	How long it is safe from another hit afterwards, in seconds.	0.6
Knockback	How hard a hit knocks it back. 1 is a good push, 0 none, 2 twice as hard. It carries a character moved by Move Character or a Hero along.	1.0
Blink	Blink while it is safe, so the player can see it was hit.	on
Hurt Event	The event its machine is sent when it is hurt. It carries how much was taken.	HURT
Died Event	The event its machine is sent when its health reaches 0.	DIED
XP Worth	The experience whoever beats it is given, when they have an Experience node or an xp variable. A slime in Mossbrook is worth 35.	0.0
Store Attacker In	A Node variable given whatever hit it, for a machine that turns to face its attacker. Empty keeps nothing.	empty
Flash Color	In 2D, the colour it flashes for a moment when a hit lands, so the hit is felt. Clear, with no colour at all, does not flash.	1, 0.35, 0.35, 1
Shows Damage	Shows how much each hit takes as a number rising and fading over it.	off
Damage Color	The colour of those numbers. White over an enemy and red over the player read well.	1, 1, 1, 1
Hurt Sound	A sound played where it is when it is hurt, such as a grunt or bones knocking. Empty plays none.	empty
Died Sound	A sound played where it was when it dies, in place of the hurt sound. Empty plays the hurt sound.	empty
Shape	Its shape, either the same as its character's own body or a circle. Choices: Same As Body, Circle.	Same As Body
Radius	Only for Circle. How big it is, in pixels in 2D and metres in 3D.	16.0
Offset	Only for Circle. How far it sits from the character's middle.	0, 0, 0

Icon Meter NODE

A row of icons that shows a number, such as hearts for health, skulls for lives or shells for ammunition. It draws plain hearts until you give it pictures of your own, and it follows a character's number or takes one from a machine.



Icon Meter in a running game.

When to use it

For a number that reads better as pictures than as a bar or as text. Pick a character's number in Shows and it follows by itself. A number kept anywhere else goes in a variable, and one Set Property action writes it into Value on Every Frame.

> With no pictures at all it draws plain hearts. Once any of the three pictures is set the plain hearts stop, so set Full Icon as well, or the full icons are not drawn.

See also
Set Property 91, **Set Value** 215, **Machine Variable** 27, **Clamp** 42

Its settings, in the Inspector

SETTING	WHAT IT DOES	STARTS AS
Full Icon	The picture of a full icon, such as a whole heart.	empty
Half Icon	The picture of a part-filled icon, such as half a heart. Left empty, any part of an icon shows as a full one.	empty
Empty Icon	The picture of an empty icon. Left empty, nothing is drawn past the number, so lost hearts vanish and do not turn grey.	empty
Character	Whose number it shows. Leave it empty for the player, the node in the player group.	empty
Shows	The character's number it follows by itself, such as health, picked from the numbers your templates keep. Its Maximum follows too when the variable has a Max. Leave it empty for a machine to set Value with Set Property.	empty
Value	The number shown. A machine sets this one.	6.0
Maximum	The most the number can be. With Per Icon it decides how many icons there are, so a Maximum of 6 at 2 each is three hearts.	6.0
Per Icon	How much of the number one icon stands for. At 2 every heart is two halves, so 5 out of 6 shows two and a half hearts. Use 1 for lives or shells.	2.0
Icon Size	How big each icon is drawn, in pixels.	32, 32
Gap	The space between icons, in pixels.	4.0
Per Row	Starts a new row after this many icons, for a long health bar. 0 keeps them all on one row.	10

Inventory NODE

What something is carrying. Put one under anything that holds things and the item actions work on it.



Inventory in a running game.

When to use it

The player, a chest, a shop shelf, a guard, a stash. A scene can have as many as you like, each carrying separate things, so a chest and a pack hold different loot. What it starts with is filled in on the node, not built by a machine. Drag items into Starts With and say how many of each. Those settings belong to each copy, so one chest scene dropped into a level twenty times can hold something different in every copy.

> An inventory that fills itself does so ONCE, as it enters the scene, and what it ends up holding is what a saved game keeps. One that rolled its loot afresh every time it loaded would hand the player something different each time they came back to it.

See also

Add Item 285, **Remove Item** 296, **Move Item** 292, **Inventory Has** 291

Its settings, in the Inspector

SETTING	WHAT IT DOES	STARTS AS
Slots	How many piles it can hold. 0 holds as many as it needs, so leave it there unless the game is about running out of room. A pile is one slot however many are in it, so twenty arrows take one.	0
Starts With	What it is carrying to begin with, one pile a row, each an item and how many of it. The rows are copied as the game starts, so playing never changes what the scene says it holds.	empty
Slot Names	The slots this inventory wears things in, such as Weapon, Armour, Trinket, or whatever your game wears. Leave it empty and the inventory is an ordinary one, holding piles of anything. Name some and it wears ONE thing in each, and takes only items whose Slot matches. A character that both carries and wears things has two Inventories. Equip Item looks for the one with slots and Add Item for the one without, so both work with nothing filled in.	empty
Fills From	Take items from this set as it enters the scene, as well as anything in Starts With. For a chest whose loot is not decided yet. Leave it empty for an inventory whose contents you decide yourself.	empty
Takes	How many to take from that set. Rarer items turn up less often, the same way Pick Item weighs them.	1
Fill Chance	How likely each item taken is really there, from 0 to 100. A ready-made Enemy's Loot Inventory takes 1 at 35, so it drops an item about one time in three.	100.0
Not Seen This Run	Leave out anything that has already turned up this run, so two chests do not hold the same thing. It starts off, so every chest and enemy takes from its own list each time the level starts. Tick it for a game played in runs and begin each run with New Run, or a list runs dry and stays dry until the game is closed.	off

Inventory Grid NODE

An inventory drawn as a grid of slots, for the player to look through and choose from.



Inventory Grid in a running game.

When to use it

The player's pack, a chest they have opened, a shop's shelf, what they are wearing. It shows what an Inventory is carrying, with a count on any pile of more than one, and lets the player move about it and choose. Keyboard and gamepad work with no setup, through Godot's own arrow and accept controls. With the mouse, a click moves the highlight and a second click on the same slot chooses it. At an edge the arrows pass the focus to whatever is beside the grid, and the accept button on an empty slot is left for anything else waiting for it, such as a Dialogue Box moving on.

> With When Chosen at Use Or Wear, as a new grid is, choosing uses or wears the item with no template at all. Moving the highlight sends Selection Changed and pressing accept sends Chosen, both carrying the item. Set When Chosen to Nothing and a template decides what choosing means (buy it, drop it, sell it). Catch either signal with an Event On Signal, whose event carries the item into a variable that Equip Item or Remove Item can be pointed at. A worn inventory draws one slot for each of its Slot Names, filled or not, so an empty weapon hand still has a place on screen.

See also
Inventory 319, **Equip Item** 289, **Item Collection** 321, **Event On Signal** 76

Its settings, in the Inspector

SETTING	WHAT IT DOES	STARTS AS
When Chosen	What choosing an item does. With Use Or Wear, a potion is drunk, a sword put on (chosen where it is worn, taken off again), and an item used with a button used, all for whoever the inventory belongs to. The first thing worn makes them a Worn Inventory to wear it in. With Nothing, choosing does nothing by itself, for a template to decide, as a shop that sells what is chosen does. Chosen is sent either way. Choices: Use Or Wear, Nothing.	Use Or Wear
Bag	The inventory to show. Point it at the node holding the Inventory, or at the Inventory itself. Leave it empty to show the Inventory on this panel's parent or, with none there, the player's pack.	empty
Cell Size	How big each slot is drawn.	56, 56
Gap	The space between slots. How many fit across is worked out from the width you give the panel, so a wider panel shows more in a row.	8.0
Across	How many slots across. 6 makes a grid six wide, wherever the panel sits, and 0 fits as many as its width holds.	0
Cell Color	The colour of an empty slot.	1, 1, 1, 0.06
Picked Color	The colour of the slot the highlight is on.	1, 0.85, 0.42, 0.85
Count Color	The colour the count on a pile is written in. A pile of one is not numbered.	1, 1, 1, 0.9

Item Collection NODE

A grid of item pictures for a collection page. The items ever found show in full colour, and the rest as dark shapes still to find. Hover over one to read its name and line.



Item Collection in a running game.

When to use it

A menu page of everything the player has discovered across all their runs. With Show set to Held This Run it becomes the part of a pause menu listing what they carry now. It looks the items up by itself. Anything that has gone into the player's Inventory counts as found, and so does anything Give Item has handed over.

> What has been found is kept after the game closes, so an item found while trying your game out is still found the next time you run it.

See also

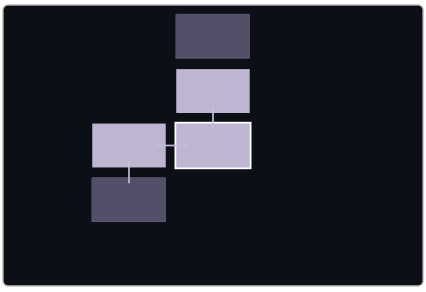
Give Item 290, **Show Item** 297, **New Run** 293, **Offer Item** 294

Its settings, in the Inspector

SETTING	WHAT IT DOES	STARTS AS
Show	Every Item shows all of them, found or not. Held This Run shows only the items given to the player this run, in the order they were given. Choices: Every Item, Held This Run.	Every Item
Items	The items to show, in order. When there are many, leave it empty and name a Folder.	empty
Folder	Shows every item file in this folder as well, and in the folders under it, so a new item appears on the page without being added here.	empty
Cell Size	How big each item is drawn, in pixels.	48, 48
Gap	The space between items, in pixels.	8.0
Not Found Color	The colour an item is drawn in before it has been found. It is dark, so only its shape shows.	0.05, 0.04, 0.08, 0.85
Cell Color	The colour of the square behind each item.	1, 1, 1, 0.06

Minimap NODE

A small map in the corner of the screen. It is a radar marking the enemies, people and items near the player, the way out, and what the tracked quests ask for now, over a picture of the level from above if one was taken, or, on a floor Build Dungeon made, the dungeon's map.



Minimap in a running game.

When to use it

Put one on the HUD of any level and it is a radar. The player sits in the middle, its arrowhead pointing the way it goes. The enemies, the people to talk to, what there is to pick up and the way out near it are marked, each in a colour of its own, and each tracked quest marks, in its colour, the place a step sends the player to, whoever plays the dialogue a step waits for, and each enemy a step asks to be beaten. A quest's mark or the way out further off than the map reaches sits at its edge, pointing the way, and an enemy, person or item that far off is not marked. Press Take Map Picture at the top of its Inspector and the level itself, pictured from above, slides under the marks as the player walks. On a floor laid out by Build Dungeon it draws the dungeon. Rooms visited are filled in, rooms next to them shown faintly, the player's room marked, and the boss, treasure, shop and secret rooms coloured once found.

> A level you laid out by hand shows the radar, and a floor Build Dungeon made shows the dungeon's map. Only tracked quests under way are marked, never hidden ones, and only the steps to do now.

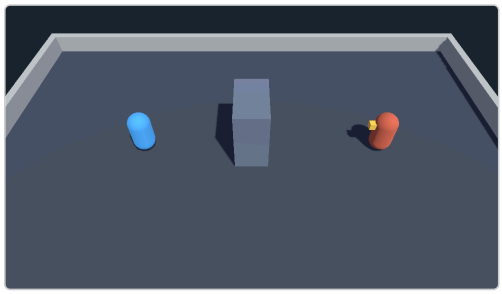
See also
Build Dungeon 219, **Quest List** 325, **Track Quest** 307

Its settings, in the Inspector

SETTING	WHAT IT DOES	STARTS AS
Player	The player. The room it is in is the one marked, and every room it enters is filled in. Leave it empty for the node in the player group.	empty
Shows Quests	Mark what the tracked quests ask for now. Off, no quest is marked, but the enemies, people, items and way out still are.	on
Reaches	How far the radar reaches from the player, in pixels in a 2D game and metres in 3D. 0 reaches 600 pixels, or 40 metres.	0.0
Backdrop Color	The colour behind the radar.	0.05, 0.06, 0.1, 0.6
Player Color	The player's mark in the middle.	1, 1, 1, 1
Shape	Round, as a radar is, or Square. Choices: Round, Square.	Round
Shows Enemies	Ticked, the enemies near the player are marked as dots in Enemy Color. An enemy further off than the map reaches is not marked. Untick it for a map that keeps them a surprise.	on
Shows People	Ticked, the people to talk to near the player are marked in People Color, such as a Villager.	on
Shows Items	Ticked, what there is to pick up or open near the player is marked in Item Color, such as an Item Pickup, loot on the ground or a Chest. Something picked up is no longer marked.	on
Shows Way Out	Ticked, the way out of the level is marked in Way Out Color. Further off than the map reaches, it sits at the edge of the map, pointing the way.	on
Enemy Color	The colour of an enemy's mark. Red to begin with.	0.95, 0.28, 0.25, 1
People Color	The colour of the mark for someone to talk to. Green to begin with.	0.4, 0.9, 0.45, 1
Item Color	The colour of the mark for something to pick up or open. Gold to begin with.	1, 0.82, 0.25, 1
Way Out Color	The colour of the way out's mark. White to begin with.	1, 1, 1, 1
Map Texture	A picture of the level from above, drawn under the marks. Take Map Picture at the top of the Inspector takes one of the level as it is, and Use My Own Picture... puts in one of your own. Empty, it is a plain radar.	empty
Map Covers	The part of the level the picture shows, in pixels in a 2D game and metres in 3D. Take Map Picture fills it in. Change it to line a picture of your own up with the level.	Rect2(0, 0, 0, 0)
Shows	Around The Player slides the picture under the player as they move. The Whole Level shows all of it at once, with the player's mark moving over it. Choices: Around The Player, The Whole Level.	Around The Player
Turns with Player	Turn the picture so the way the player faces is always up the map, the way the camera looks in 3D. Off, north stays up and the player's mark turns instead.	off
Room Size	How big each room is drawn on the map, in pixels. Keep it the same shape as your real rooms.	18, 11
Gap	The space between rooms on the map, in pixels.	3.0
Visited Color	The colour of rooms the player has been in.	0.78, 0.75, 0.86, 0.95
Known Color	The colour of rooms the player knows about but has not entered.	0.45, 0.42, 0.55, 0.7
Current Color	The colour of the outline round the room the player is in.	1, 1, 1, 1
Boss Color	The mark on the boss room.	0.9, 0.25, 0.35, 1
Treasure Color	The mark on the treasure room.	0.98, 0.8, 0.3, 1
Shop Color	The mark on the shop.	0.4, 0.8, 0.5, 1
Secret Color	The mark on the secret room, once it has been found.	0.6, 0.45, 0.95, 1

Perception NODE

What a character can notice. Put one under anything that should react to something else, and the AI actions and checks read it to learn who it can see, where it last saw them, and whether it has given up.



Perception in a running game.

When to use it

A guard watching for the player needs one, and so does anything watching for a group. It looks on a timer, and Look For Target, Chase, Can See and Target Lost all read that one result, so they stay simple and agree on one target. It also remembers where it last saw its target. That way an enemy can see you, lose you, search where you were and then give up.

> It faces the way its parent faces, which in 3D is away from the screen and in 2D is to the right. Turn the parent, not the Perception.

See also

Look For Target 256, **Chase** 251, **Can See** 250, **Target Lost** 262

Its settings, in the Inspector

SETTING	WHAT IT DOES	STARTS AS
Watch Group	What it looks for, such as the player. Press the box and pick it in the scene. You can pick one thing by its name, Every Coin for every copy of the coin scene (including ones made while playing), or Everything under a node for a set of markers. NodeWorthy does the rest. Look For Target sets it to the action's Look For while its state runs.	player
Sight Range	How far it can see, in metres in a 3D game or pixels in a 2D one. The 20 it starts with suits 3D. A 2D game needs a few hundred, and a Perception added with NodeWorthy > Give It > Perception in a 2D level starts at 400.	20.0
FOV	How wide its view is, in degrees, centred on the way it faces. 360 sees all round, and 120 is roughly what a person notices.	120.0
Needs Line of Sight	Anything behind a wall stays unseen. Untick it for a game with no walls to hide behind, which saves the check.	on
Blocked By	Which layers block its sight, usually World (the walls and ground). The ready-made player and enemies stand on separate layers and do not block it.	1
Eye Height	How far above the node its eyes are, so it does not look along the floor. It is for 3D only. In 2D it looks from the middle of its shape, or near the top of it seen from the side, and seeing someone's middle or head is seeing them.	1.6
Hearing Range	It notices anything this close whichever way it faces and whatever is in the way, as if it heard it. 0 switches hearing off.	0.0
Check Every	Seconds between looks. Looking every frame is wasted work, and a fifth of a second is quick enough for anything that walks.	0.2
Remember For	Seconds it keeps after a memory once sight is lost, before giving up. 0 forgets as soon as sight is broken.	8.0

Performance Overlay NODE

A small panel in the corner of the running game that shows how fast it runs, and how much of each frame the machines take.



Performance Overlay in a running game.

When to use it

Add one while you build to see whether the game is smooth. It shows frames a second, the 1% low (the speed of the slowest frame in every hundred, where stutter shows up), the worst frame, the time the machines take, and how many machines are running. A game that averages 60 with a 1% low of 30 feels rough. In a smooth one the two numbers sit close together.

> Take it out before you ship the game.

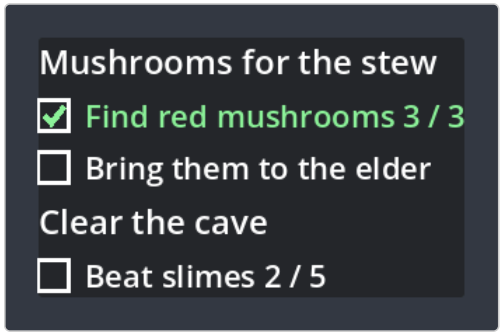
See also
Set Time Scale 66, **Enable Machine** 18

Its settings, in the Inspector

SETTING	WHAT IT DOES	STARTS AS
Toggle Input	The key that shows and hides the panel while playing, picked from the list of inputs. Empty leaves it as it is. Either key listed does it. On a Mac the F keys work the screen's brightness and its windows unless the keyboard is set otherwise, so F3 on its own never reaches the game there.	Key: F3 or Key: Backtick
Frames Kept	How many of the latest frames the numbers are worked out from. More is steadier, and fewer follows a change more quickly. Changed while the game runs, the numbers start afresh.	600

Quest List NODE

The quests under way, and their steps with a tick against each one done, such as Find mushrooms 2 / 3.



Quest List in a running game.

When to use it

Small in a corner it is a quest tracker. With descriptions and finished quests shown it becomes a journal on a pause screen. It reads the quests straight from the game and keeps itself up to date.

> It hides itself while there is nothing to show. The ticks are drawn as shapes, not typed, so they show in any font. It takes its panel and font from the game's look, set in the Theme tab.

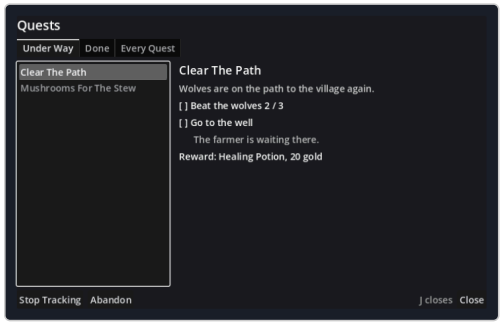
See also
Start Quest 306, **Quest Is** 304, **Stat Panel** 328, **Inventory Grid** 320

Its settings, in the Inspector

SETTING	WHAT IT DOES	STARTS AS
Show	Which quests it shows. Active shows only those under way, as a tracker, Active And Done adds the finished ones, and Every One shows failed ones too, as a journal. Choices: Active, Active And Done, Every One.	Active
Only Tracked	Show only the quests the player is tracking. Every quest is tracked as it begins, and Track Quest stops or starts tracking one. Off, every quest under way shows.	off
Show Descriptions	Show each quest's line saying what it is about under its title, and each step's description under the step.	off
Font Size	How big the words are. Leave it at 0 for the text size of the game's look, set in the Theme tab.	0
Done Color	The colour of a step done, and of a quest done.	0.55, 0.95, 0.6, 1
Failed Color	The colour of a quest failed.	1, 0.45, 0.45, 1

Quest Log NODE

The player's quest log. It is a screen opened with a key, with the quests down one side and the one picked set out beside them, and buttons to track it or give it up.



Quest Log in a running game.

When to use it

Put it on the HUD of a game with quests. Q opens it over the game, which stands still while the player reads. The quests under way, those done, or every one are a tab away. The one picked shows its description, each step ticked with its count and its description, and its reward. It reads the quests from the game and keeps itself up to date.

> Hidden quests are never in it. Track and Stop Tracking choose which quests the Minimap marks and a Quest List set to Only Tracked shows. It sends Opened and Closed as it opens and closes, and Quest Picked carrying the quest, for an Event On Signal.

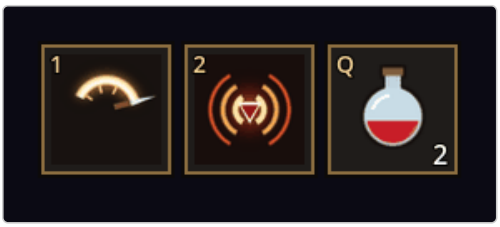
See also
Quest List 325, **Minimap** 322, **Track Quest** 307, **Abandon Quest** 301

Its settings, in the Inspector

SETTING	WHAT IT DOES	STARTS AS
Open Key	The key, or pad button, that opens the log and closes it again, picked from the list of inputs. Escape, or a pad's back button, closes it too.	Key: Q or Pad: Select / Back
Pauses the Game	Stop the game while the log is open, so nothing happens to the player while they read.	on
Can Abandon	Let the player give a quest up from the log, with an Abandon button that asks twice.	on
Font Size	How big the words are. 0 uses the text size of the game's look.	0
Done Color	The colour of a step done, and of a quest done.	0.55, 0.95, 0.6, 1
Failed Color	The colour of a quest failed or given up.	1, 0.45, 0.45, 1

Skill Bar NODE

A row of skills along the bottom of the screen, each with its picture and its key. Press the key or click it, and when it is ready and there is mana enough, the mana is paid, its event is sent to the player's machine, and it darkens and sweeps back as it comes ready again.



Skill Bar in a running game.

When to use it

A hero's special moves. What a skill does is up to the template, as with any state, so a state waiting for the cleave's event swings the axe. A skill the mana cannot pay for is shown dimmed, and one that uses an item, such as a potion, shows how many are carried.

> A skill's event does nothing until a state in the character's template waits for it.

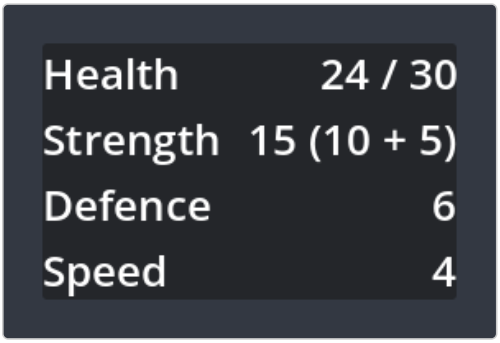
See also
Globe 315, **Use Item** 300, **Explosion** 110, **Attack** 264

Its settings, in the Inspector

SETTING	WHAT IT DOES	STARTS AS
Character	Whose skills they are. That means whose mana pays and whose machine is sent the events. Leave it empty for the player, the node in the player group.	empty
Mana Variable	The character's number variable skills are paid from, such as mana. Empty makes every skill free.	mana
Mana Back per Second	How much mana comes back by itself each second, up to the mana variable's Max. 0 brings none back.	3.0
Skills	The skills, from left to right. Each has its picture, its key, the event it sends, its cost, how long it takes to come ready again, and the item it uses, if any.	empty
Slot Size	How big each slot is drawn, in pixels.	64, 64
Gap	The space between slots, in pixels.	8.0

Stat Panel NODE

A character's stats on the screen, such as Strength 15, Defence 6, Health 24 / 30. It reads them straight from the character's template and keeps itself up to date.



Stat Panel in a running game.

When to use it

Any game where numbers matter to the player, such as an RPG's stat sheet. As soon as it is added it shows the player's numbers, in the editor as they start and in the game as they change. A stat that gear or an effect is changing shows where it came from, so a sword that adds 5 to Strength shows 15 (10 + 5). Every change flashes green going up and red going down.

> It finds the player by itself as the node in the player group, which the ready-made players join. A stat limited by another, such as health that is Max max_health, shows as one line, Health 24 / 30. It is a panel for the HUD, so the right-click menu puts it under a CanvasLayer, and in the editor it shows in the 2D view, even in a 3D game.

See also
Modify Stat 268, **Equip Item** 289, **Set Variable** 49, **Icon Meter** 318

Its settings, in the Inspector

SETTING	WHAT IT DOES	STARTS AS
Character	Whose stats, such as the player, an enemy or a shopkeeper. Leave it empty for the player, the node in the player group.	empty
Stats	The stats shown, top to bottom, by the names of the character's variables, such as strength, defence, health. Leave it empty for every number the character's template has that is not one of its settings.	empty
Show Maximum	Show a stat whose Max is another stat with it, as Health 24 / 30, and leave the other off its own line.	on
Show Changes	Show what gear and effects are adding, as 15 (10 + 5).	on
Font Size	How big the words are. Leave it at 0 for the text size of the game's look, set in the Theme tab.	0
Up Color	The colour a stat flashes when it goes up.	0.55, 0.95, 0.6, 1
Down Color	The colour a stat flashes when it goes down.	1, 0.45, 0.45, 1

A to Z

Every action, check and node by name. Checks and nodes are marked.

A

Abandon Quest QUESTS 301
Add Item INVENTORY 285
Add Node NODE 70
Alert Allies AI 249
Animation Event ANIMATION 160
Apply Force PHYSICS 107
Array ARRAY 58
Assert DEBUG 246
Attack COMBAT 264

B

BarNODE NODES TO ADD 310
Bool Operator LOGIC 33
Build Dungeon SCENE 219
Bullet Pattern COMBAT 265

C

Call Method NODE 71
Call Static Method NODE 72
Camera Follow CAMERA 184
Camera Limits CAMERA 185
Camera Shake CAMERA 186
Camera Zoom CAMERA 187
Can SeeCHECK AI 250
Cancel Delayed Events STATE MACHINE 17
Carry TRANSFORM 92
ChanceCHECK LOGIC 34
Change Scene SCENE 220
Change Text In File SAVE & FILES 228
Chase AI 251
Choose Folder SAVE & FILES 229
Clamp MATH 42
Click To Move CHARACTER 126
CompareCHECK LOGIC 35
Complete Quest QUESTS 302
Copy Folder SAVE & FILES 230
Count Item INVENTORY 286
CrosshairNODE NODES TO ADD 311
Crouch CHARACTER 127

D

Dash CHARACTER 128
DataCHECK DATA 61
Data NodeNODE NODES TO ADD 312
Device Motion INPUT 139
Dialogue BoxNODE NODES TO ADD 313
Dialogue Line SaidCHECK DIALOGUE 308
Dictionary ARRAY 59
Display Settings APPLICATION & DISPLAY 279
DistanceCHECK TRANSFORM 93
Drag INPUT 140
Draw Debug DEBUG 247
Drive PHYSICS 108
Drop Item INVENTORY 287
Drop Loot INVENTORY 288
Duplicate NODE 73

E

Ease Along Curve MATH 43
Ease Variable MATH 44
Emit Particles EFFECTS 199
Emit Signal NODE 74
Enable Machine STATE MACHINE 18
Enable Node NODE 75
Equip Item INVENTORY 289
Event On Collision PHYSICS 109
Event On Input INPUT 141
Event On Signal NODE 76
ExperienceNODE NODES TO ADD 314
Explosion PHYSICS 110
Expression MATH 45

F

Fade Screen CAMERA 188
Fail Quest QUESTS 303
File ExistsCHECK SAVE & FILES 231
Find Child NODE 77
Find Row DATA 62
Flee AI 252
Flicker EFFECTS 200
Flip Sprite ANIMATION 161
Follow Path TRANSFORM 94
For Each LOGIC 36
Format String STRING 54
Format Time STRING 55
Forward Events STATE MACHINE 19
Freeze PHYSICS 111

G

Get Action Strength INPUT 142
Get Cell TILEMAP 241
Get Children NODE 78
Get Color Components COLOR 269
Get Contact PHYSICS 112
Get Current Camera CAMERA 189
Get Data DATA 63
Get Event Data STATE MACHINE 20
Get Focus UI 205
Get Input Vector INPUT 143
Get Last Collision CHARACTER 129
Get Last Event STATE MACHINE 21
Get Navigation Point NAVIGATION 237
Get Overlapping PHYSICS 113
Get Parent NODE 79
Get Previous State STATE MACHINE 22
Get Property NODE 80
Get Scene SCENE 221
Get Screen Size APPLICATION & DISPLAY 280
Get State STATE MACHINE 23
Get Text UI 206
Get Time TIME 65

Get Value UI 207
Get Vector Components VECTOR 50
Get Velocity PHYSICS 114
Get Water Depth CHARACTER 130
Give Item INVENTORY 290
GlobeNODE NODES TO ADD 315
Go To Previous State STATE MACHINE 24
Go To State STATE MACHINE 25
Grab Focus UI 208
Grid Map Cell TILEMAP 242
Grid Path NAVIGATION 238
Group NODE 81
Group Switch NODE 82

H

Has Child Of TypeCHECK NODE 83
Head Bob CAMERA 190
Heal COMBAT 266
High Score Table ARRAY 60
Hit BoxNODE NODES TO ADD 316
Hurt COMBAT 267
Hurt BoxNODE NODES TO ADD 317

I

Icon MeterNODE NODES TO ADD 318
In RangeCHECK TRANSFORM 95
In StateCHECK STATE MACHINE 26
In WaterCHECK CHARACTER 131
InputCHECK INPUT 144
Input Combo INPUT 145
Instantiate NODE 84
Interaction AI 253
InventoryNODE NODES TO ADD 319
Inventory GridNODE NODES TO ADD 320
Inventory HasCHECK INVENTORY 291
Is Child OfCHECK NODE 85
Item CollectionNODE NODES TO ADD 321

J

Joypad Connected INPUT 146
Jump CHARACTER 132

K

Keep Apart AI 254
Keep Distance AI 255

L

Last Device INPUT 147
Leave Marks EFFECTS 201
Ledge Grab CHARACTER 133
Look At TRANSFORM 96
Look For Target AI 256

M

Machine Variable STATE MACHINE 27
Make Noise AI 257
Math MATH 46
MinimapNODE NODES TO ADD 322
Modify Stat COMBAT 268
Mouse Look CAMERA 191
Mouse Motion INPUT 148
Mouse Pick INPUT 149
Mouse Position INPUT 150
Move Character CHARACTER 134
Move Item INVENTORY 292
Move To TRANSFORM 97
Move To Bin SAVE & FILES 232
Mute AUDIO 171

N

Navigate To NAVIGATION 239
Navigation ReachableCHECK NAVIGATION 240
New Run INVENTORY 293
Next Frame Event STATE MACHINE 28
NodeCHECK NODE 86

O

Off Screen Indicator UI 209
Offer Item INVENTORY 294
On ScreenCHECK PHYSICS 115
Option Button UI 210
Orbit Camera CAMERA 192

P

Paint Terrain TILEMAP 243
Path Progress TRANSFORM 98
Patrol AI 258
Pause Game SCENE 222
Pause Sound AUDIO 172
PerceptionNODE NODES TO ADD 323
Performance OverlayNODE NODES TO ADD 324
Pick Item INVENTORY 295
Pinch INPUT 151
Play Animation ANIMATION 162
Play Music AUDIO 173
Play Random Sound AUDIO 174
Play Sound AUDIO 175
Play Sound At Position AUDIO 176
Play Sprite Animation ANIMATION 163
Play Video VIDEO 284
PlayingCHECK ANIMATION 164
Pointer Over UICHECK UI 211
Print DEBUG 248
Procedural Placement SCENE 223
PropertyCHECK NODE 87
Punch TRANSFORM 99

Q

Quest IsCHECK QUESTS 304
Quest ListNODE NODES TO ADD 325
Quest LogNODE NODES TO ADD 326
Quest Step IsCHECK QUESTS 305
Queue Free NODE 88
Quit Game SCENE 224

R

Random Color COLOR 270
Random Event STATE MACHINE 29
Random Number MATH 47
Random Seed MATH 48
Random Vector VECTOR 51
Raycast PHYSICS 116
Raycast All PHYSICS 117
Read File SAVE & FILES 233
Rebind Input INPUT 152
Recoil CAMERA 193
Remove Item INVENTORY 296
Reparent NODE 89
Repeat LOGIC 37
Rotate TRANSFORM 100
Rotate To TRANSFORM 101
Run Program SAVE & FILES 234
Run Template STATE MACHINE 30
Run When Unfocused APPLICATION & DISPLAY 281

S

Sample Gradient COLOR 271
Save Game SAVE & FILES 235
Scale To TRANSFORM 102
Screen Effect EFFECTS 202
Screen Orientation INPUT 153
Screen Settings SCENE 225
Screen To World CAMERA 194
Search Last Known AI 259
Seek Animation ANIMATION 165
Send Event STATE MACHINE 31
Sequence LOGIC 38
Set Animation Speed ANIMATION 166
Set Background Color CAMERA 195
Set Body Properties PHYSICS 118
Set Bus Effect AUDIO 177
Set Cell TILEMAP 244
Set Collision PHYSICS 119
Set Color Components COLOR 272
Set Data DATA 64
Set Disabled UI 212
Set Environment EFFECTS 203
Set Gravity PHYSICS 120
Set Joint PHYSICS 121
Set Language SCENE 226
Set Light LIGHTS 274
Set Material MATERIAL 275
Set Material Color MATERIAL 276
Set Material Texture MATERIAL 277
Set Modulate COLOR 273
Set Mouse Mode INPUT 154
Set Pitch AUDIO 178
Set Process Mode NODE 90
Set Property NODE 91
Set Shader Parameter EFFECTS 204
Set Sound Loop AUDIO 179
Set Text UI 213
Set Texture UI 214
Set Time Scale TIME 66
Set Tree Parameter ANIMATION 167
Set UI Scale APPLICATION & DISPLAY 282
Set Value UI 215
Set Variable MATH 49
Set Vector Components VECTOR 52
Set Velocity PHYSICS 122
Set Visible UI 216
Set Volume AUDIO 180
Shape Cast PHYSICS 123
Show Choices UI 217
Show Item INVENTORY 297
Skill BarNODE NODES TO ADD 327
Sort Inventory INVENTORY 298
Stamina CHARACTER 135
Start Dialogue DIALOGUE 309
Start Quest QUESTS 306
Stat PanelNODE NODES TO ADD 328
Step Up CHARACTER 136
Stop Animation ANIMATION 168
Stop Music AUDIO 181
Stop Sound AUDIO 182
Stop Sprite Animation ANIMATION 169
Strafe Around AI 260
String STRING 56
String CompareCHECK STRING 57
String To Number CONVERT 278
Sub State FinishedCHECK STATE MACHINE 32
Swim CHARACTER 137
Swing PHYSICS 124
Switch LOGIC 39
Switch Camera CAMERA 196
System SCENE 227

T

Take Cover AI 261
Take Screenshot APPLICATION & DISPLAY 283
Target LostCHECK AI 262
Text To Speech AUDIO 183
TileCHECK TILEMAP 245
TimeoutCHECK TIME 67
Timer TIME 68
To Global TRANSFORM 103
To Local TRANSFORM 104
Toggle Button UI 218
Touch Count INPUT 155
Touch Gesture INPUT 156
TouchingCHECK PHYSICS 125
Track Quest QUESTS 307
Translate TRANSFORM 105
Travel To ANIMATION 170
Tween Property TRANSFORM 106

U

Unequip Item INVENTORY 299
Use Item INVENTORY 300

V

VariableCHECK LOGIC 40
Variable ChangedCHECK LOGIC 41
Vector VECTOR 53
Vibrate INPUT 157
Virtual Joystick INPUT 158

W

Wait TIME 69
Wall Cling CHARACTER 138
Wander AI 263
Warp Mouse INPUT 159
Weapon Sway CAMERA 197
World To Screen CAMERA 198
Write File SAVE & FILES 236