State Machine | ANTIPIXEL Godot Documentation

Introduction

Installation

Unzip the downloaded package and transfer its contents to your project. By convention, plugins are usually installed inside the `addons` folder.

Some plugins may require you to activate them manually. To do this, go to `Project > Project Settings > Plugins`. If you experience problems, try restarting the editor.

For more information, see the official Godot documentation.

Sample

The `sample` folder includes demo scenes.

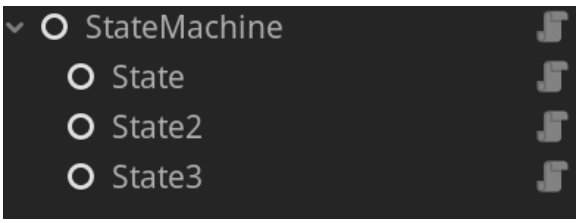
If you no longer need it, you can delete this folder without fear of breaking the plugin.

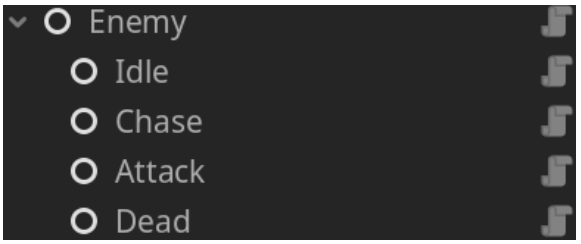
Manual

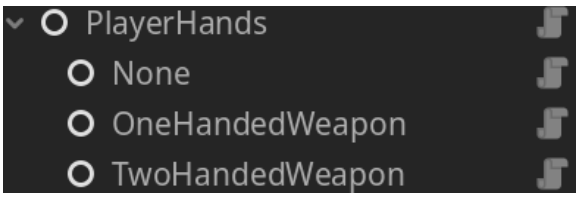
State Machines

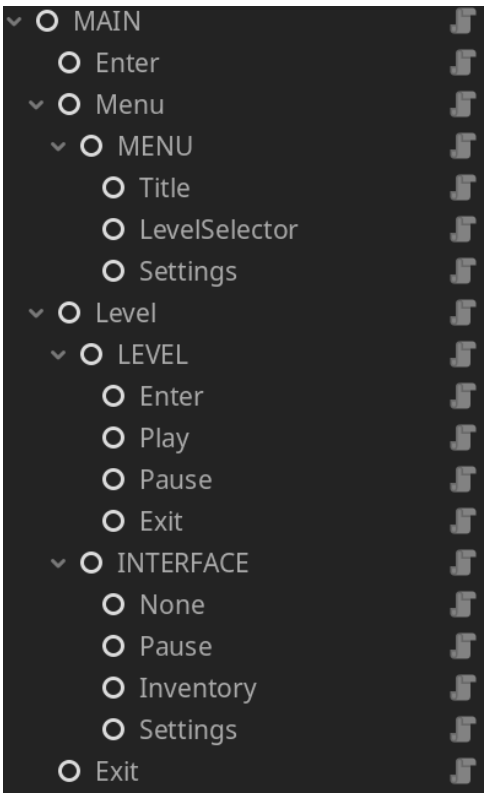
Adds a **State Machine** to the scene and some **States** as children.

Examples



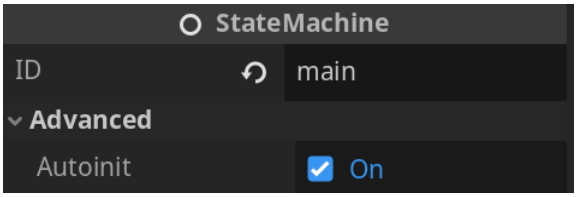






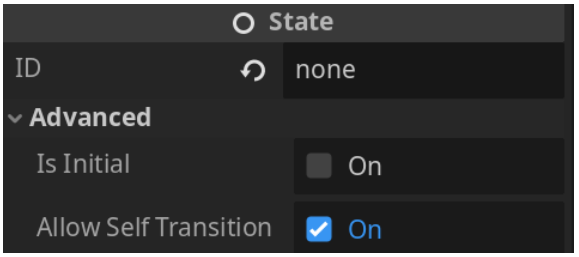
State Machine Properties

ID	State identifier.
Autoinit	Is this the default initial state?



State Properties

ID	State identifier.
Is Initial	Is this the default initial state?
Allow Self Transition	Can the machine change to this state if it is already in this state?



 If no initial state is specified, the first child is chosen by default.

Now we can use the state machine as follows:

Shortcut

```
# Tip: If you are going to use a machine many times in a script,
# you can define a property as a shortcut.
var machine: StateMachine:
    get: return StateMachine.machines["machine_id"]
```

Transition

```
# Access the machine you want and change its state.
machine.change("state_id")
```

ℹ️ When entering a state, all its children will activate their processes and when exiting they will be deactivated.

Data

```
# You can transfer any amount of arguments and of any type.
machine.change("state_id", ["hello", custom_resource, self])
```

Signals

```
# Subscriptions
machine.state_entered.connect(_on_state_entered)
machine.state_exited.connect(_on_state_exited)

func _on_state_entered(state: State, data: Array) -> void:
    print("Enter: ", state.id)

func _on_state_exited(state: State, data: Array) -> void:
    print("Exit: ", state.id)
```

Current State

```
# Checking the current state.
if machine.current.id == "state_id":
    print("Do something")
```

States

This package includes 2 predefined states.

In most cases these two custom states along with the basic one are more than enough for most projects but they can be inherited and overwritten to create more advanced behaviors.

ℹ️ See the API for how to create custom states, initializations and transitions.

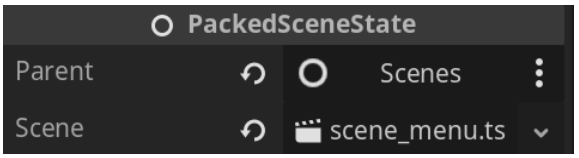
PackedSceneState

Creates a scene on entry and destroys it on exit.

Useful for making transitions between scenes.

Properties

Parent	Where to place the scene?
Scene	Scene to instantiate.



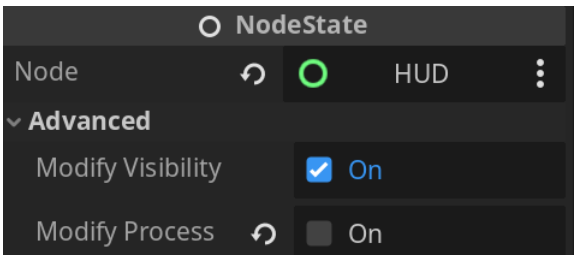
NodeState

Displays a node on entry and hides it on exit.

It also activates and deactivates its processes.

Properties

Node	Target node.
Modify Visibility	Can visibility be turned on or off?
Modify Process	Can _process be turned on or off?



State Components

In some more advanced cases we may want to have certain behaviors only in some states or with some differences.

ℹ️ See the API for how to create components.

Example

In this hypothetical case we can see that the player can move whether he is walking or running. However, we will define in the inspector that the walking speed will be lower than the running speed.

IDs

Identifiers can be used to access their references via dictionary.

> ⚠️ If a machine or state has an identical identifier to a pre-existing one, the old reference will be lost and replaced with the new one. **Keep this in mind, especially when creating multiple instances**.

If you do not specify an ID, one will be assigned automatically based on the node name following this logic: `name.capitalize().to_lower().replace(" ", "_")`

So if your node is called, for example, `GamePause`, its ID would be `game_pause`.

API

State

extends Node

Description

Node that is added to a state machine to represent a condition or behavior of the system at a given time.

Example

```
class_name CustomState extends State

func _enter() -> void:
    print("Enter: ", id)

func _exit() -> void:
    print("Exit: ", id)

# Overwrite get_initial to create advanced initializations.
# Overwrite can_change to create advanced logic for transitions.
```

Properties

id	State identifier.
is_initial	Is this the default initial state?
allow_self_transition	Can the machine change to this state if it is already in this state?
machine	Reference to the state machine to which this state belongs.
components	Contains all the attached components.

Functions

_enter	It is called when the state machine enters this state.
_exit	It is called when the state machine leaves this state.

get_initial	Called if the state machine can be initialized automatically. Overwrite to create advanced initializations.
can_change	Called before changing to this state. Overwrite to create advanced logic for state transitions.
get_components	Returns all components of the specified id.
add_component	Register a new component safely.
remove_component	Remove a component safely.

StateComponent

extends Node

Description

Node that is added to a state to easily represent a part of its qualities.

Example

```
class_name CustomStateComponent extends StateComponent

func _enter() -> void:
    print("Enter: ", state.id)

func _exit() -> void:
    print("Exit: ", state.id)
```

Properties

id	Component indentifier.
state	Reference to the state to which this component belongs.

Functions

_enter	It is called when the state machine enters the state of this component.
_exit	It is called when the state machine leaves the state of this component.

StateMachine

extends Node

Description

Node representing a system that can be in one of several states that it can exchange in response to certain inputs or events.

Signals

state_entering	Invoked before entering a state.
state_entered	Invoked after entered a state.
state_exiting	Invoked before exiting a state.
state_exited	Invoked after exited a state.

Static Properties

machines	Dictionary by ids containing all available state machines.

Properties

id	Machine identifier.
autoinit	Should the machine transition to the initial state automatically?
states	Dictionary by ids containing all the states of the machine.
current_state	Current state.
previous_state	Previous state.
next_state	Next state.
current_data	Current state data.
previous_data	Previous state data.
next_data	Next state data.
is_initialized	Is there already a current state?

Functions

change	Exit the current state and enter a new one.
add_state	Register a new state safely.
remove_state	Remove a state safely.

Support

If you found this asset useful, please consider supporting the creator with a donation. For any suggestions, corrections, or inquiries, please contact the developer. ❤️

antipixelgames@gmail.com