Introduction to Godot

Jaanus Jaggo

Godot

Godot is a free and open source game engine (MIT licence).

First released to public in 2014.

Platforms: Windows, macOS, Linux, BSD, iOS, Android, UWP, HTML5, WebAssembly (Console ports offered via third party providers)

- 2D / 3D
- GDScript, C#, Visual Script, C++

Scripting

GDScript

Pros:

- Dynamically typed language, similar to Python
- No compilation (fast to start)
- Less code

Cons:

- Less performance
- Code-completion and error detection is more limited

Introduction to GDScript:
https://docs.godotengine.org/en/stable/getting_started/scripting/gdscript/gdscript_advanced.html

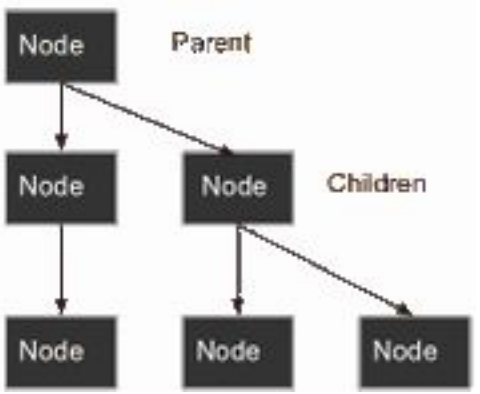
Architecture

- **Nodes & Scenes**
- Instancing
- Referencing
- Resources
- Data structures
- Singletons (AutoLoad)
- Signals
- SceneTree

Nodes

Everything is a **Node**

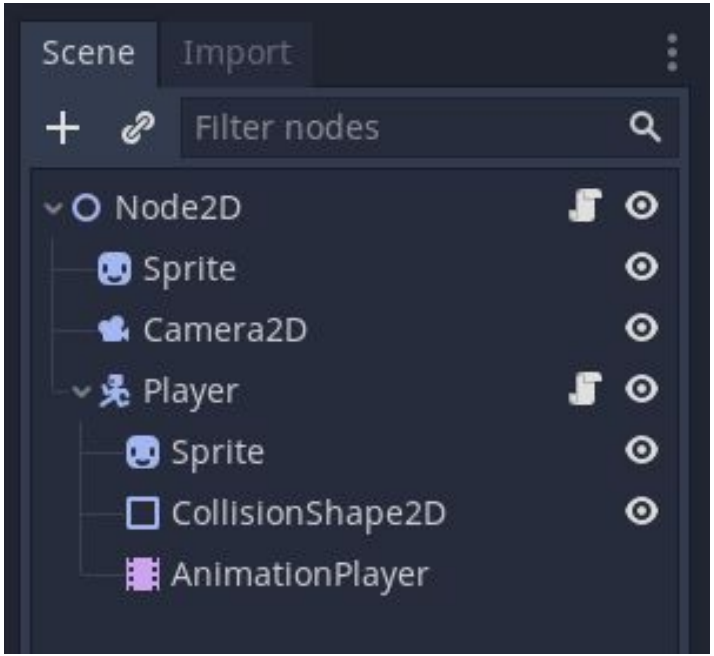
- It has a name
- It has editable properties
- It can receive callback to process every frame
- It can be extended
- It can be added to another node as a child



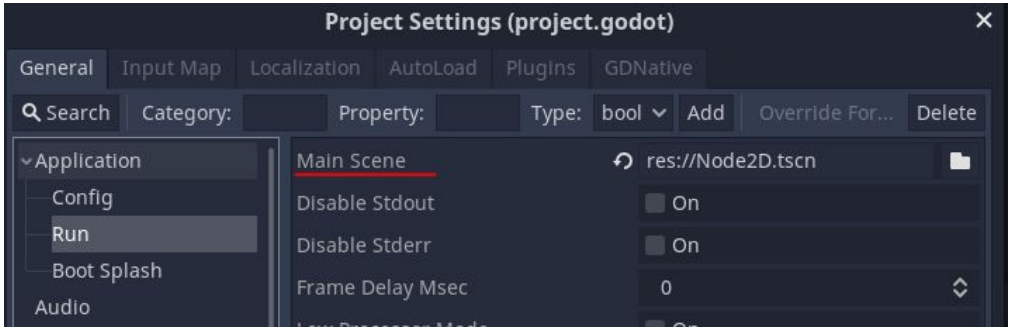
Scenes

Scene is composed of a **group of nodes organized hierarchically**.

- always has one root node
- can be saved to disk and loaded back
- can be instanced



Running a game means running a scene:

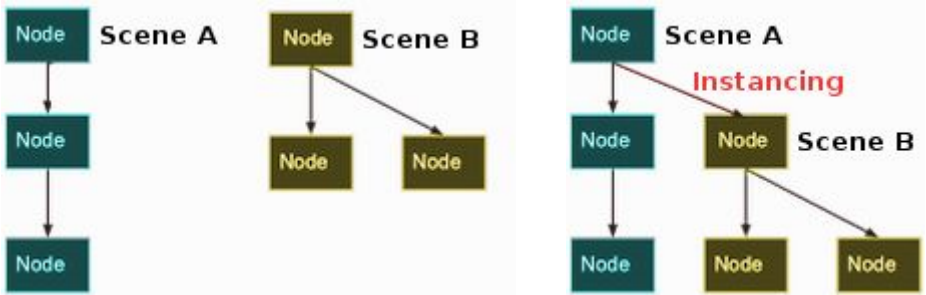


Architecture

- Nodes & Scenes
- **Instancing**
- Referencing
- Resources
- Data structures
- Singletons (AutoLoad)
- Signals
- SceneTree

Instantiating

You can create as many scenes as you want and save them to disk:
Right click node in scene → Save Branch as Scene
This creates **.tscn** file




```
var player = load("res://Player.tscn").instance()
add_child(player)
```

Architecture

- Nodes & Scenes
- Instancing
- **Referencing**
- Resources
- Data structures
- Singletons (AutoLoad)
- Signals
- SceneTree

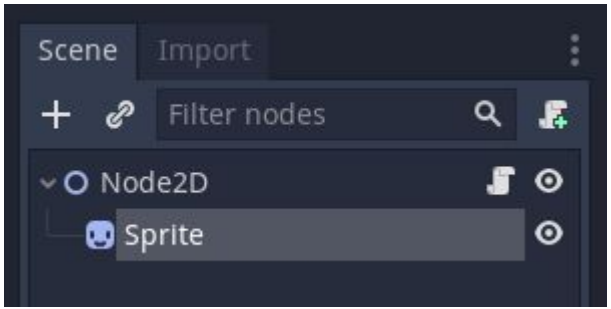
Referencing

Referencing by path

```
onready var sprite = get_node("Sprite")
```

$ is same as **get_node**

```
onready var sprite = $Sprite
```



Referencing with export var

```
export(int) var health = 10
export(Image) var image = null
```

Architecture

- Nodes & Scenes
- Instancing
- Referencing
- **Resources**
- Data structures
- Singletons (AutoLoad)
- Signals
- SceneTree

Resources

Anything godot saves or loads from disk is a resource.

Generic resource file is ***.tres**

```
func _ready():
        var res = load("res://robi.png")
        get_node("sprite").texture = res
```

Scenes are also resources:

```
var playerScene = preload("res://Player.tscn")
func _ready():
    var player = playerScene.instance()
    add_child(player)
```

load vs preload

- **load** - resource is loaded when the line is executed
- **preload** - resource is loaded at compile time

Architecture

- Nodes & Scenes
- Instancing
- Referencing
- Resources
- **Data structures**
- Singletons (AutoLoad)
- Signals
- SceneTree

Data Structures

Array

Serves as: list, set, queue or stack

```
var array = [10, "hello", 40]
array.append(20)
array[2] = 30
array.pop_front()
if 20 in array:
    print(array)
```

Dictionary

Can store different types

```
var d = {"age": 22}
d["name"] = "Joe"
if d.has("age"):
    d.erase("age")
print(d)
```

Data Structures

Array

Serves as: list, set, queue or stack

```
var array = [10, "hello", 40]
array.append(20)
array[2] = 30
array.pop_front()
if 20 in array:
    print(array)
```

[hello, 30, 20]

Dictionary

Can store different types

```
var d = {"age": 22}
d["name"] = "Joe"
if d.has("age"):
    d.erase("age")
print(d)
```

{name:Joe}

Compatible with JSON:

```
var p = JSON.parse('["hello", "world", "!"]')
```

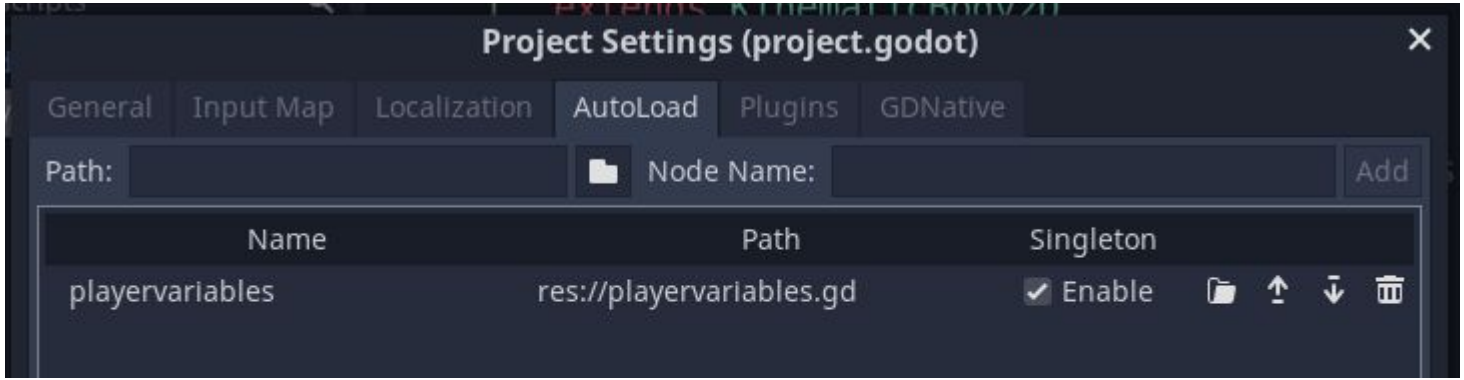
Architecture

- Nodes & Scenes
- Instancing
- Referencing
- Resources
- Data structures
- **Singletons (AutoLoad)**
- Signals
- SceneTree

Singeltons (AutoLoad)

Singeltons are used when presistent information between scenes is needed.
NB! GDScript does not support global variables by design

To add AutoLoad scene or scripts, select Project > Project Settings > AutoLoad tab.



Any script can now access the singleton with:

```
var player_vars = get_node("/root/playervariables")
player_vars.health
```

Architecture

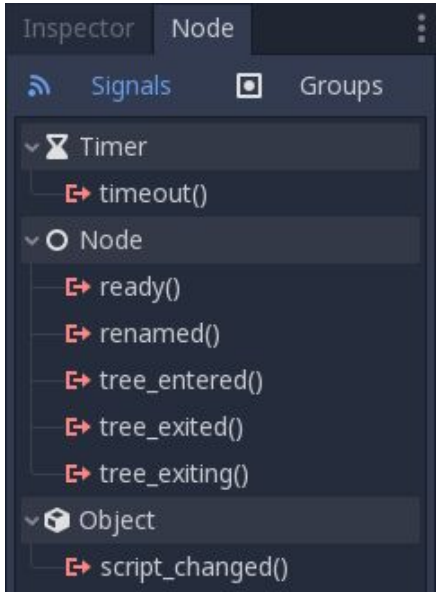
- Nodes & Scenes
- Instancing
- Referencing
- Resources
- Data structures
- Singletons (AutoLoad)
- **Signals**
- SceneTree

Signals

Signals are Godot's version of the observer pattern (Events).

Godot signals can be connected in Node tab, next to Inspector.
This add a corresponding function to the script:

```
func _on_Timer_timeout():
    pass # Replace with function body.
```



Can be also connected in code:

```
func _ready():
    $Timer.connect("timeout", self, "_on_Timer_timeout")


func _on_Timer_timeout():
    $Sprite.visible = !$Sprite.visible
```

Custom signals can be created:

```
signal my_signal(value, other_value)


func _ready():
    emit_signal("my_signal", true, 42)
```

Architecture

- Nodes & Scenes
- Instancing
- Referencing
- Resources
- Data structures
- Singletons (AutoLoad)
- Signals
- **SceneTree**

SceneTree

SceneTree is an automatically created node that manages the hierarchy of nodes. SceneTree is in charge of the game loop.

It contains;

- The root Viewport
- Information about the node groups
- Global state functionality

```
get_tree().get_root() # Access via scene main loop.
get_node("/root") # Access via absolute path.
```

```
get_tree().change_scene("res://levels/level2.tscn")
```

```
yield(get_tree().create_timer(1.0), "timeout")
```