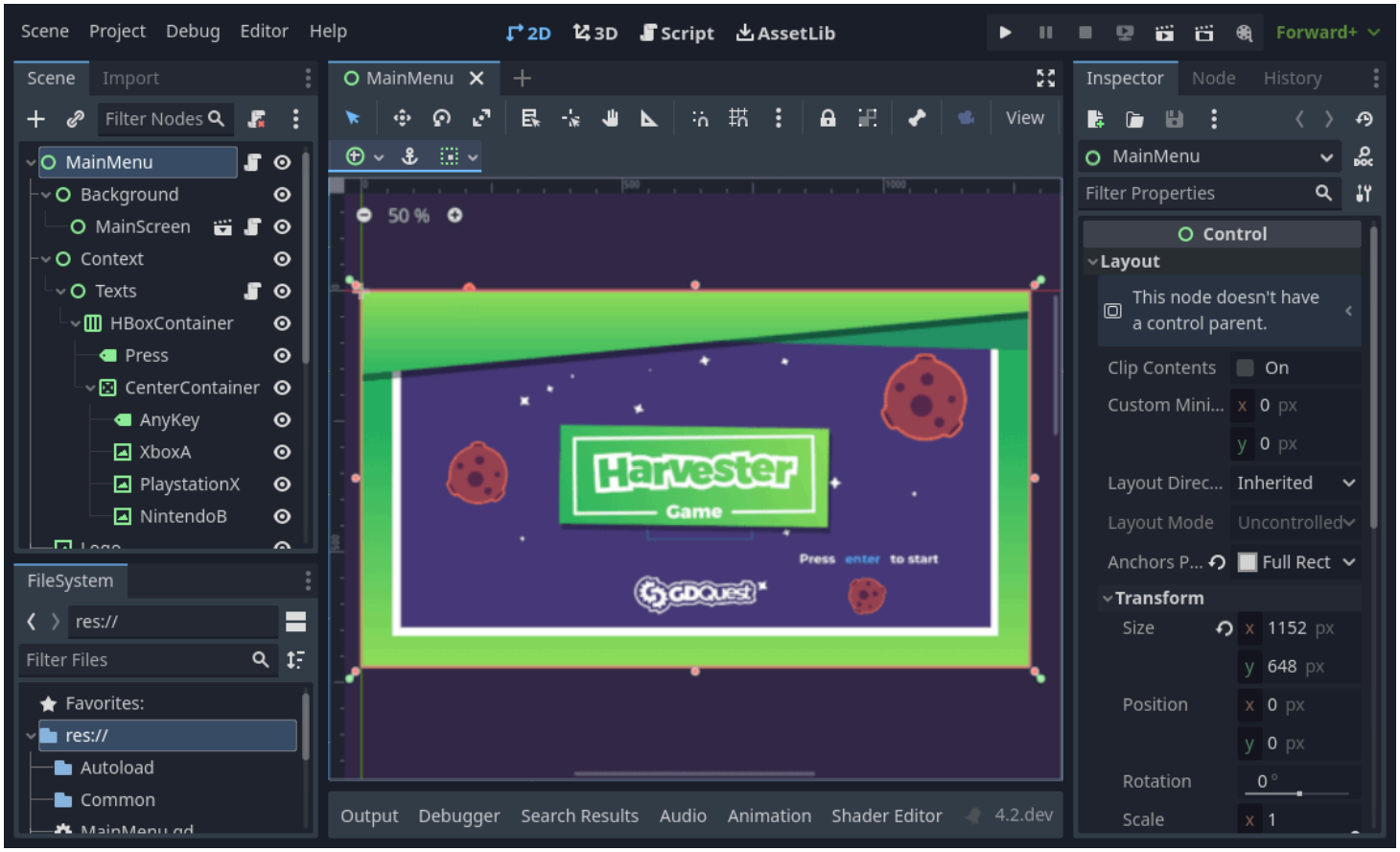
节点，场景和系统函数

在 Godot 中，游戏就是一棵由**节点**构成的树，树又可以结合起来构成**场景**。然后你还可以将这些节点连起来，让它们通过**信号**进行通信。

场景

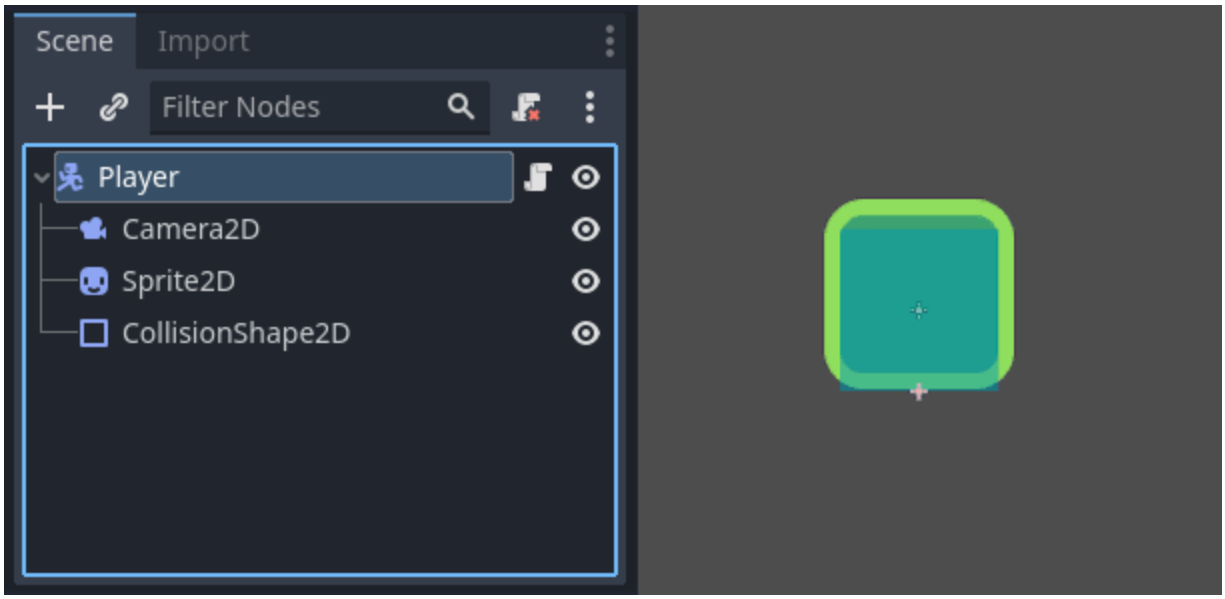
在 Godot 中，你把你的游戏分解成可重复使用的场景。场景可以是一个角色、一件武器、用户界面中的一个菜单、一座房子、整个关卡，或者任何你能想到的东西。Godot 的场景很灵活，既能够充当预制件（一个模板），又能够用作其他游戏引擎中的场景（字面意义上）。
如何新建一个场景



场景是可以在另一个场景中被实例化的（用模板做东西），也就是说，在一个场景中做的人物可以在另一个场景中出现

节点

场景由若干节点组成。节点是你的游戏最小的构件，你将其排列成树状。下面是一个人物节点的例子。



它由名为“Player”的 `CharacterBody2D` 节点、 `Camera2D` 节点、 `Sprite2D` 节点、 `CollisionShape2D` 节点组成。
节点和场景在编辑器中看起来是一样的。当你把一棵节点树保存为场景时，它就显示为一个单一的节点。
游戏的所有场景都汇集在场景树中，字面意思是一棵场景树。由于场景是节点树，因此场景树也是节点树。但是，从场景的角度来考虑你的游戏更容易，因为它们可以代表角色、武器、门或你的用户界面。

系统函数

初始化操作

Godot 的 Node 类提供了一些函数，你可以通过这些函数来在每帧或发生特定事件时更新节点
有两个函数可以用来进行初始化操作及获取节点操作： `_enter_tree()` 和 `_ready()` 。
节点进入场景树时就会激活，引擎会调用其 `_enter_tree()` 方法，此时该节点的子节点可能还未激活，由于你可以从场景树中移除节点然后重新添加这个节点，因此在一个节点的生命周期内，这个函数可能会被调用多次。
在大多数时候，你用的初始化函数其实都是 `_ready()` 。这个函数只会在节点的生命周期中调用一次，且会在 `_enter_tree()` 执行完毕之后调用。 `_ready()` 可以保证所有子节点都已进入场景树。
总结： `_enter_tree` 在每个节点进入场景树时激活，此时该节点的子节点可能还未激活。 `_ready` 在所有节点准备完成后激活一次，之后就不再激活。
另一个有关的回调函数便是 `_exit_tree()` ，在节点退出场景树时，引擎便会调用该节点的该函数。

```
# Called every time the node enters the scene tree.
func _enter_tree():
        pass

# Called when both the node and its children have entered the scene tree.
func _ready():
        pass

# Called when the node is about to leave the scene tree, after all its
# children received the _exit_tree() callback.
func _exit_tree():
        pass
```

空闲处理与物理处理

游戏是通过**循环**来运行的，每一帧都需要先更新游戏世界的状态，然后再把它绘制到屏幕上。Godot 为 Node 类提供了两个方法来完成帧循环处理： `_process(delta)` 和 `_physics_process(delta)` 。 `delta` 指的是两次处理之间的时间。如果你在脚本中定义了这两个函数的其中之一，或者两者都定义了，引擎就会自动进行调用这个（这些）函数。

有两种帧循环处理方式：

1. 空闲处理可以用来执行每帧更新节点的代码，执行频率会尽可能地快（电脑性能越好执行频率越快，但是每次循环间隔的时间是不固定的）。引擎每需要绘制一帧画面，就会调用一次方法 `_process(delta)`
2. 物理处理的执行频率是固定的，默认为每秒 60 次。物理处理和游戏的实际帧率无关，可以让物理平滑执行，故**一切与物理引擎相关的行为都应该用物理处理帧循环函数来进行处理**，如移动可能会与环境相碰撞的实体。对应方法 `_physical_process(delta)`