## 参考文档

<https://www.bilibili.com/video/BV1JuCrYcEvJ?spm_id_from=333.788.videopod.sections&vd_source=79fbe818ff96aae3677e123f0374bd91>

首先导入资源

资源地址：<https://pan.quark.cn/s/16f45b684f0e#/>

解压后得到一个叫AssetBundle的文件夹，直接把这个文件夹拖动到Godot编辑器的文件系统中

## 场景

创建2D场景，然后创建一个Sprite2D节点，将资产包中的森林背景图赋值给这个Sprite2D的材质选项

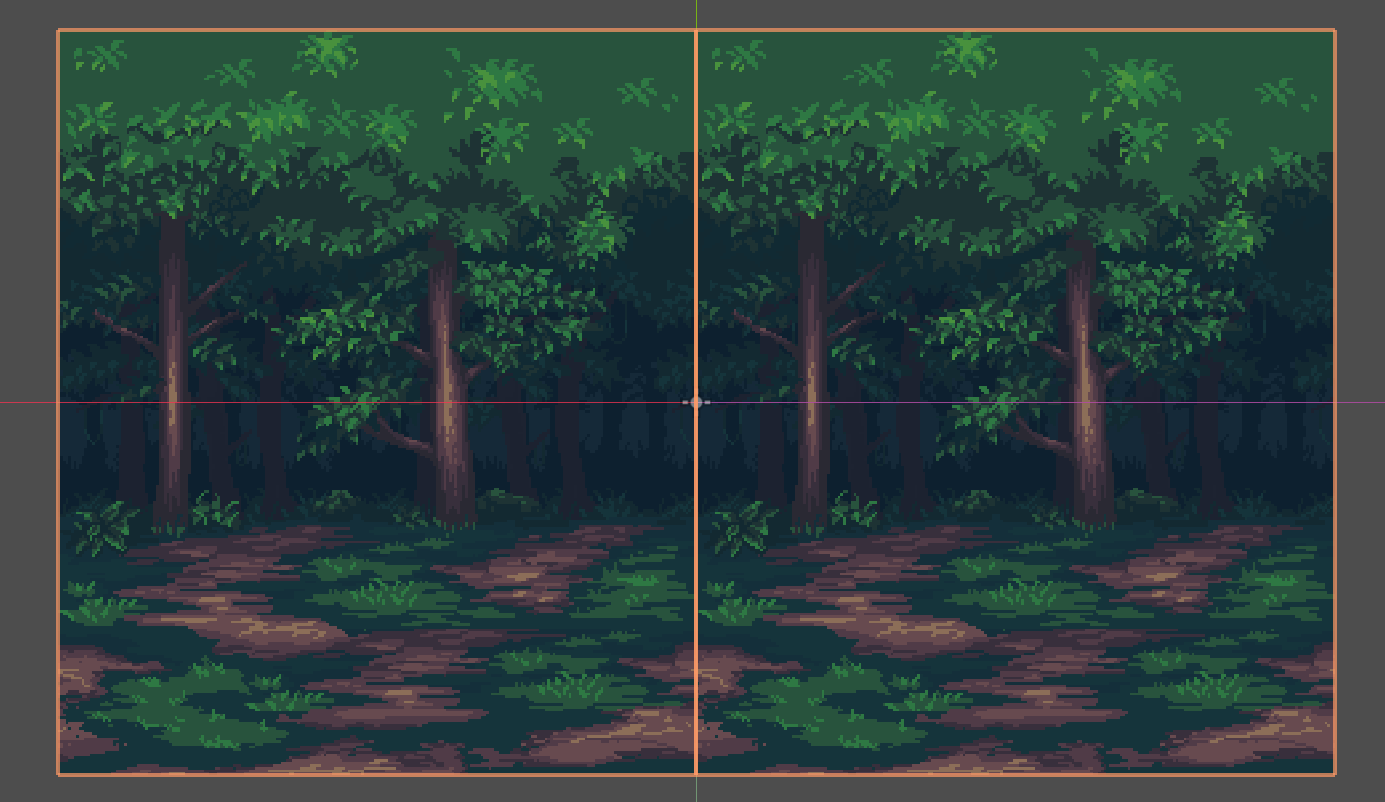
因为想做一个像素游戏，所以需要到项目设置中，“渲染--纹理”，将画布纹理的默认纹理过滤设置为Nearest

然后将场景树中的Sprite2D重命名成Background1。在视口中将这个图片移动到Y轴左侧的位置

此时我们已经有了一张背景，但我们还想让它更大一点。

用Ctrl D快速复制一个Background1，命名为Background2

然后控制Background2的位置往右移动，直到这两张图片拼接成一个背景



然后Ctrl S保存场景，将场景命名为Game.tscn，保存路径为res://Scenes

因为Godot在游戏中只渲染视图范围内的画面，我们又想把森林渲染在正中央，还需要添加一个摄像机，也就是Camera节点

我们在场景中Ctrl A创建一个Camera2D节点，然后就可以在视口中看到摄像机节点的额粉色方框

修改摄像机渲染范围的方法是在属性面板中修改Zoom的值

## 玩家角色

玩家角色也是需要以结点的形式添加进游戏场景，而一般玩家角色是一个独立的场景，然后在游戏场景中调用玩家场景。这是oop的思想。

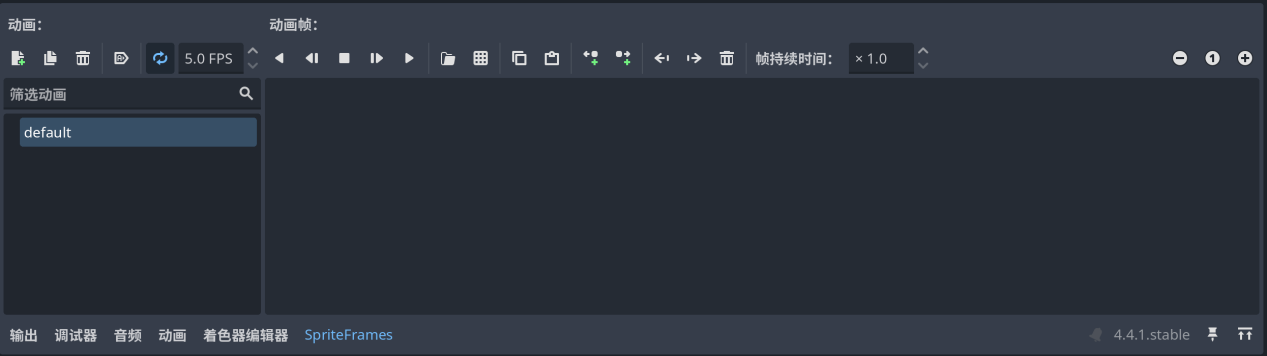
我们来创建一个新场景



这次的根节点为CharacterBody2D节点，命名为Player，场景保存命名为Player.tscn

下面给玩家添加图片，给玩家创建AnimatedSprite2D类子节点

然后给这个子节点在属性的“Animation--SpriteFrame”中新建一个SpriteFrame，然后就可以在下面打开它

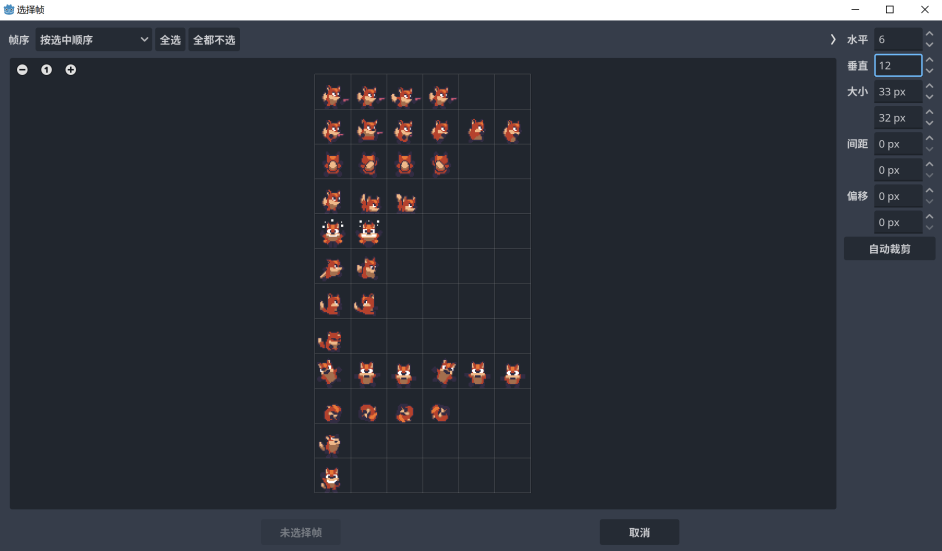


接下来添加帧



选择Asset中的Foxy.png

在游戏中，角色的动画一般都是被存放在一种叫SpriteSheet图片表中的。图片表会保存玩家每一帧下的不同的动作

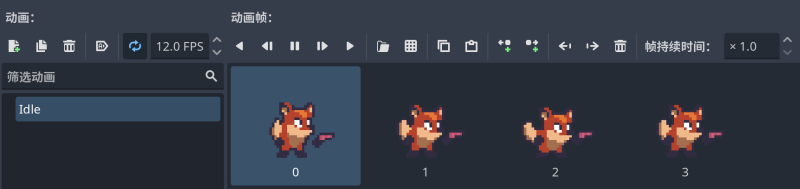


然后设置右侧的数据

水平为6个，垂直为12个，这样就会自动切割。

第一行是玩家角色的Idle动作，我们可以分别按照顺序点击他们，然后就会选择这四张图片

现在我们只添加这4帧



然后可以点击播放来预览，也可以设置帧率

然后双击动画名称命名为Idle

然后选择加载后自动播放



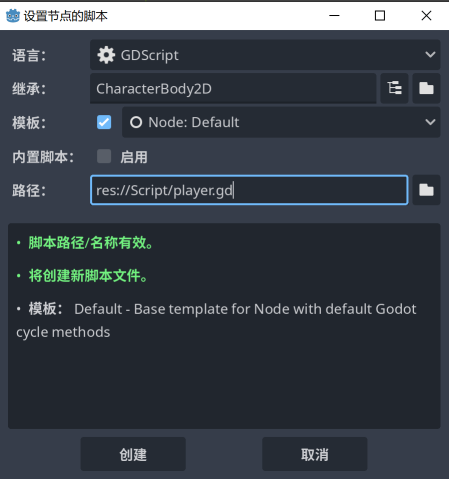
然后回到Game场景，来将角色链接放到主场景中



## GDScript

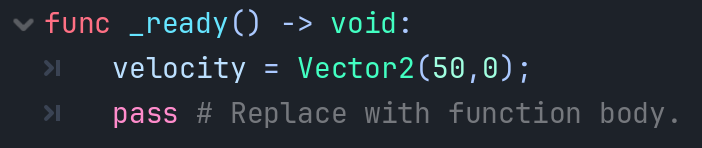
进入Player场景，来给Player这个CharacterBody2D创建一个脚本





## 角色移动

角色自身有一个变量velocity，表示角色的速度向量。想让角色移动起来，首先要让她有一个速度



然后在EventTick中调用move\_and\_slide函数。这个函数可以根据velocity来移动该物体。

## 键盘控制角色移动

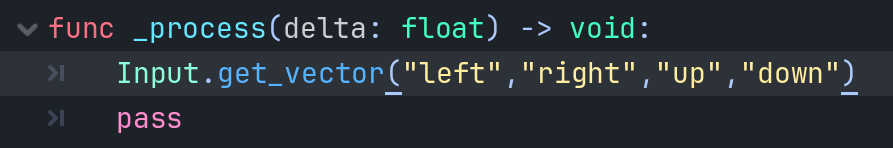
在项目设置的菜单中找到输入映射。玩家的每一个输入信号在Godot中都被称为动作Action。

首先添加新动作，然后给动作分配键位



完成之后即可关闭。

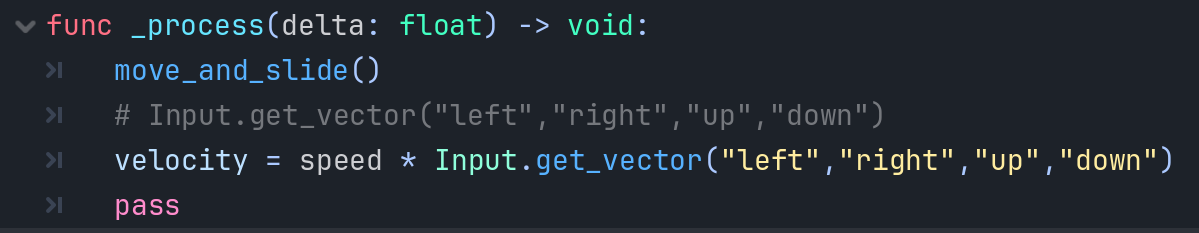
然后在游戏运行过程中实时检测玩家在WASD上的输入，并将其转换成角色的移动方向。



这里这个函数的作用是确定输入映射的方向。基本上就是按照这么设定的。函数的参数是前面定义的输入映射的动作名称，顺序不能变，理由可以Ctrl看函数的定义。

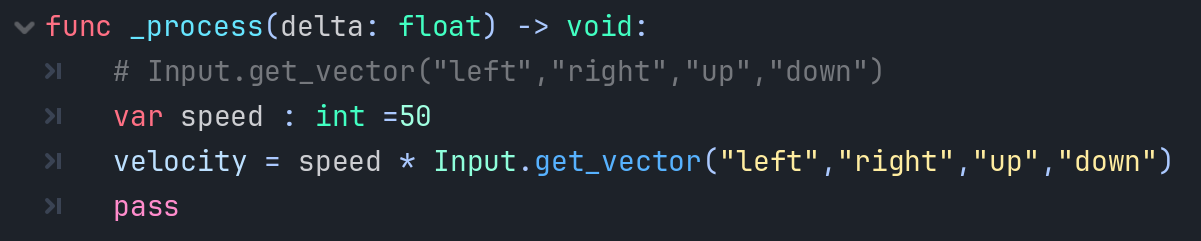
也可以通过打印该函数的返回值来看会输出什么。

想控制玩家的速度，就需要修改Character的velocity参数



如果前面使用打印看了函数的返回值，就会发现，这样设置的速度，设置的值是1。

所以我们需要给这个赋值提供一点倍数



（这个speed应该是全局参数）

最后，因为不同的电脑的帧率不一样，所以我们应该锁帧，也就是\_physics\_process函数，默认锁定运行频率是每秒60次

## 变量

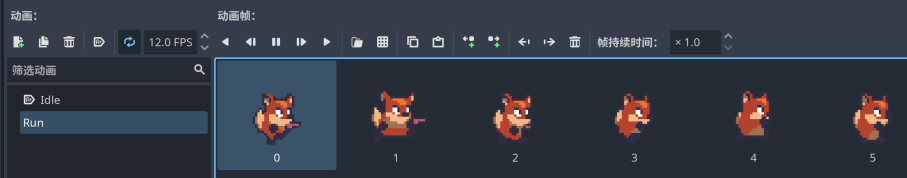
将变量暴露给编辑器，只需要在前面加上@export关键字即可



## 跑步动画

打开AnimateSprite2D节点的动画帧编辑器

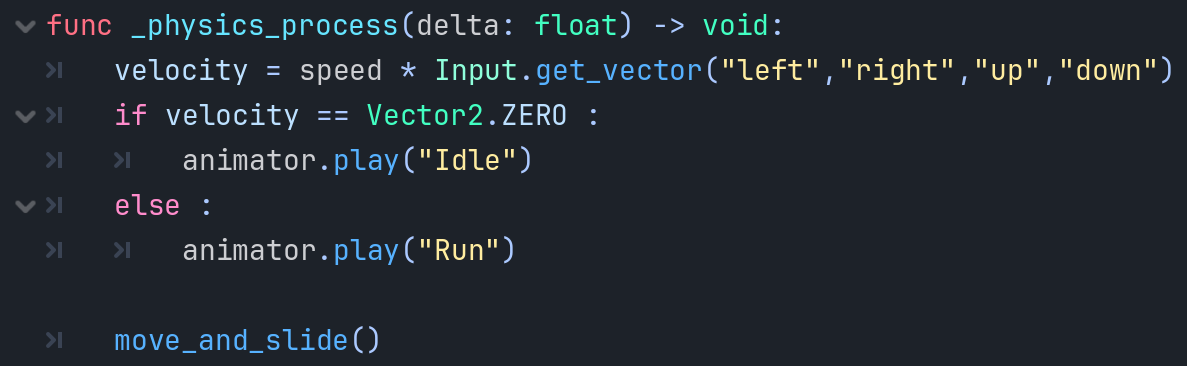
首先创建一个新的动画



然后获取角色的AnimateSprite2D节点

首先定义变量，然后在Tick中判断速度是不是为0，如果是0就Idle动画，否则就跑步动画





## StaticMesh

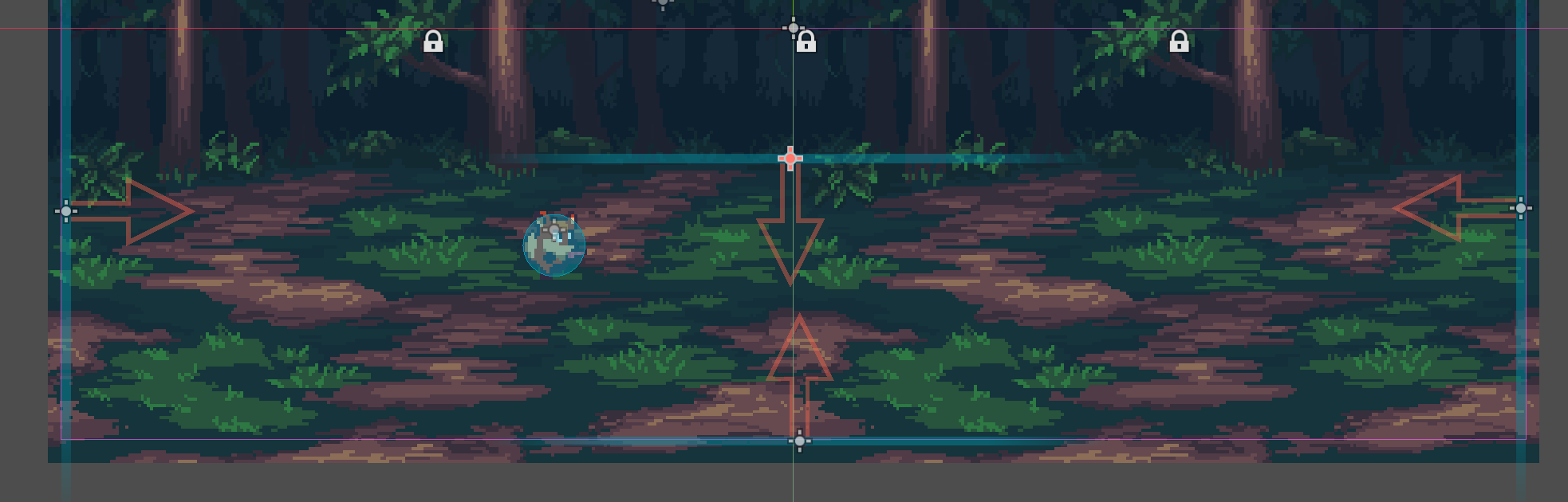
UE中的StaticMesh在godot中叫StaticBody2D。既可以用来搭建地图静止部分，也可以实现空气墙

StaticBody需要一个CollionShape，来定义形状。这样才能参与碰撞。创建完后，在属性面板中定义他的形状

我们使用



这是一个无限延伸的墙



注意一下空气墙的方向奥

我们也需要同时给角色添加一个CollisionShape。角色的碰撞体可以选择球形碰撞。

## 创建敌人

史莱姆敌人最好也有单独的场景

这里，敌人节点的根节点类型为Area2D

这是一个物理节点，他可以检测是否有别的物理节点碰到了它

然后创建AnimateSprite2D子节点和CollisionShape2D节点

然后进入到主场景中，我们将史莱姆添加到场景的根节点下，不作为任何节点的子节点。

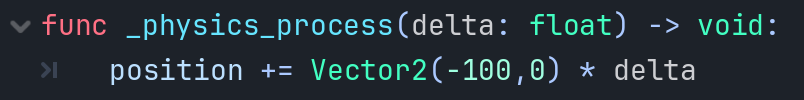
通过实例化子场景来将史莱姆链接到场景下

创建敌人的脚本，继承自敌人的根节点Area2D



史莱姆是没有velocity参数的，因为根节点是Area2D而不是CharacterBody2D

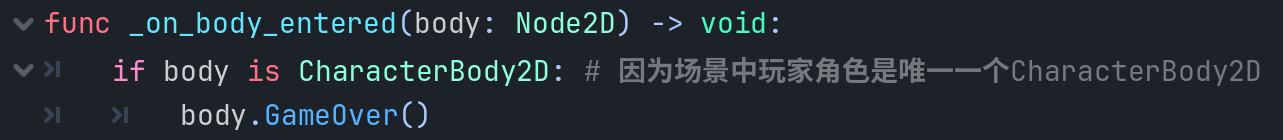
所以想移动的话，可以直接修改敌人的世界坐标参数position



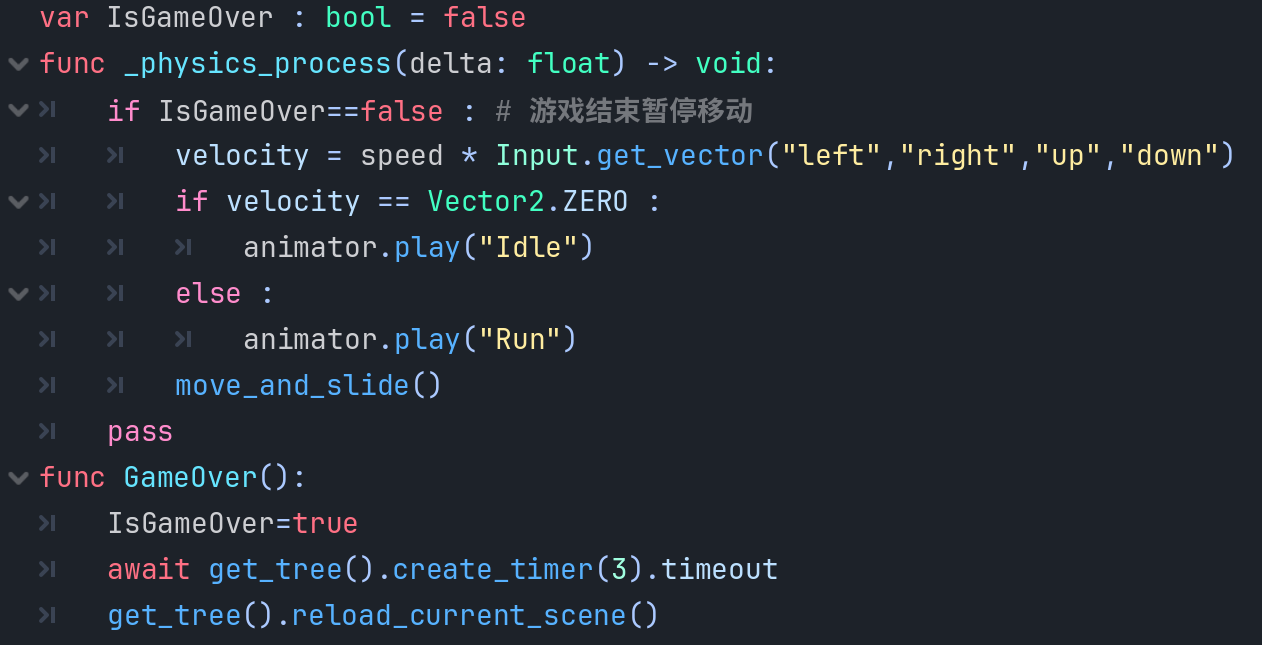
## 检测物理碰撞

在节点的属性面板中找到节点面板，里面有各种各样的事件

我们使用body\_entered事件，类似于OnComponentBeginOverlap



（GameOver是玩家角色的一个函数，用于重启场景）



最后添加玩家角色的死亡动画

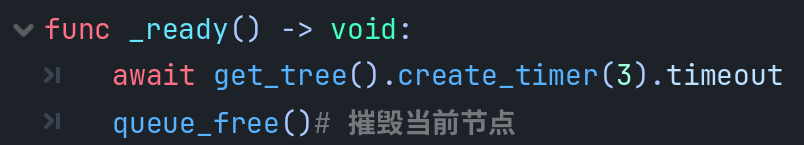
## 发射子弹攻击史莱姆

子弹也需要有自己的场景

子弹的根节点选择Area2D

然后添加代码，自动移动

子弹需要在一段时间后，自动销毁

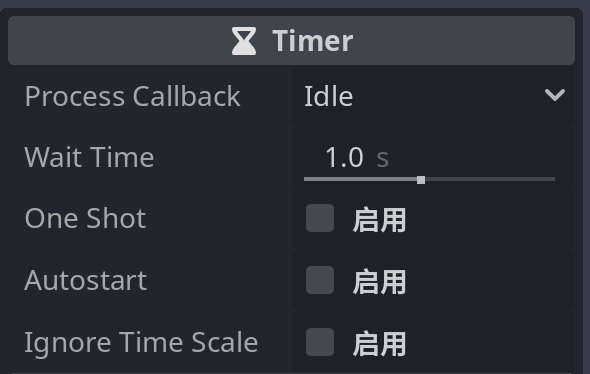


## 循环执行

用Timer倒计时来循环执行一个动作，来让玩家角色周期性的发射子弹

在玩家角色类中，添加Timer节点

然后看他的细节面板



WaitTime表示时长，OneShot表示仅触发一次

AutoStart表示自动开始

然后看Timer的事件

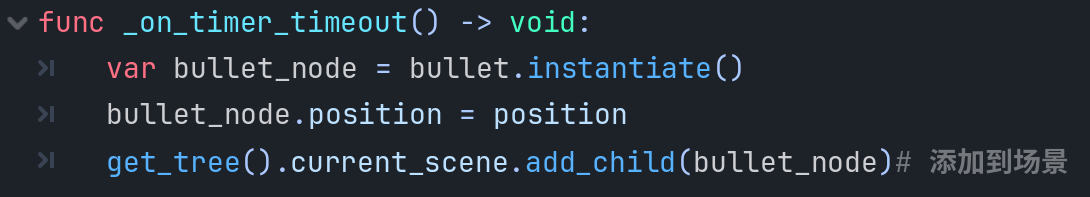
Timer有一个专属的timeout事件，我们通过这个事件，每秒发射一颗子弹

我们需要在角色类中定义子弹变量



PackedScene是指以文件形式保存在我们项目里的场景

然后就可以生成子弹节点了



## 史莱姆死亡

死亡动画

给史莱姆的动画列表中加上一个动画序列

这个动画需要取消动画循环。就在动画列表里面就有按钮

死亡判断

给史莱姆加一个信号

首先找到史莱姆的Area2D的area\_entered事件，创建函数

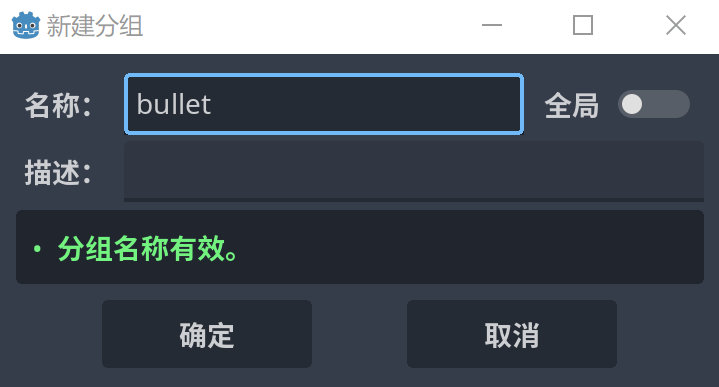
因为有多个史莱姆，而且史莱姆和子弹的根节点都是Area2D，所以我们需要判断的是史莱姆是否与子弹类的Area2D发生碰撞，而不是Area2D与Area2D发生碰撞。

这里会用到godot的分组group功能

进入Bullet场景，在节点属性列表中有分组选项

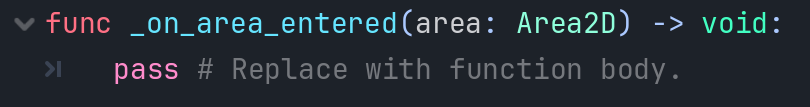


首先创建一个叫bullet的分组



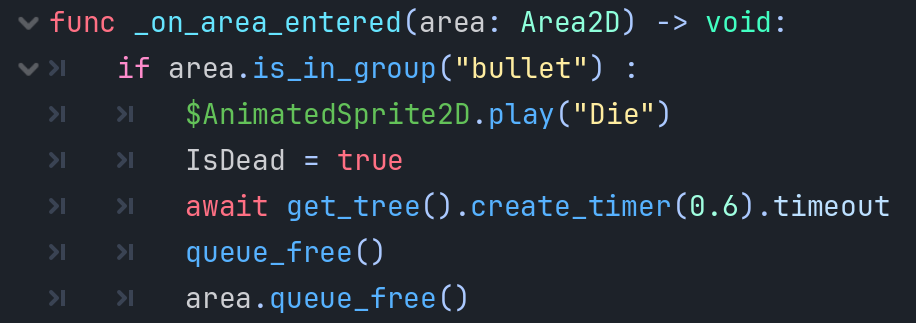
然后godot就会自动把 子弹节点 分配给 bullet组了

回到史莱姆节点



这个area参数就是撞到本史莱姆身上的area蓝图（area节点），也就是子弹蓝图

我们可以通过判断它是否是在bullet分组的，来执行死亡行为



## 源源不断的史莱姆

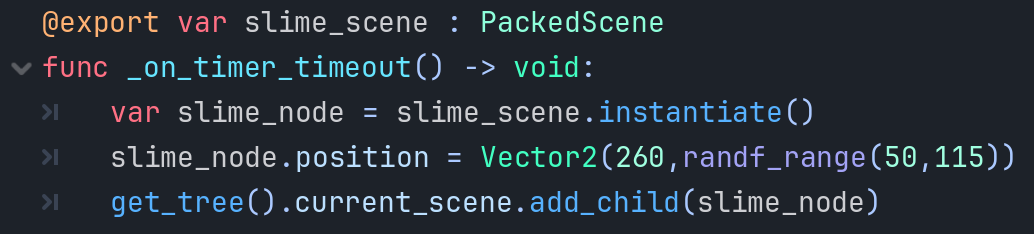
可以使用一个Timer

我们可以新建一个GameManager

在主场景中给根节点新建一个GDS，命名为GameManager.gd

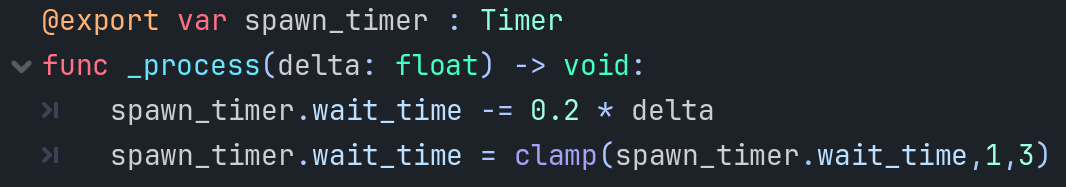
并在主场景下新建一个Timer节点

设置Timer的相关属性



修改生成速率

方法就是按帧设置Timer的时间



## UI

记录击败了多少史莱姆，位置是在GameManager中

实现给GameManager中的分数变量加分是在enemy类中实现

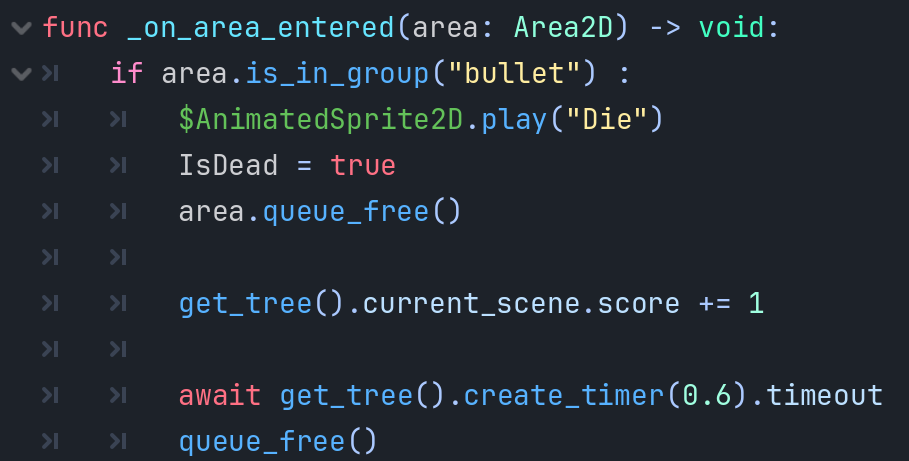
我们需要在enemy.gd中获取GameManager蓝图

跨场景获取变量并不是一个好的方法

但是有一个更好的方法

GameManager.gd是被绑定到Game主场景的根节点上的，所以如果我们可以获取主场景的根节点，也就相当于获取了GameManager的代码

在史莱姆的代码中，使用get\_tree.current\_scene来获取当前场景的根节点



显示UI

在主场景中添加CanvasLayer节点

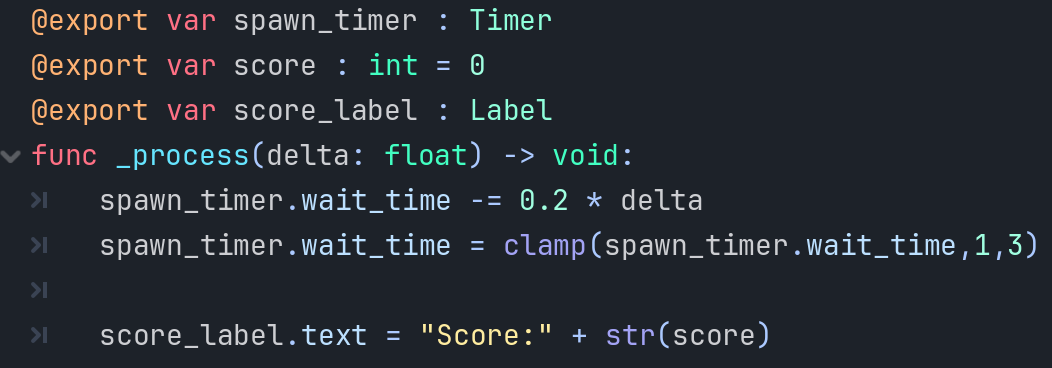
然后在Game场景的视口中拉远窗口，可以看到一个很大的方框，这就是CanvasLayer的工作区。

然后给CanvasLayer添加子节点Label类节点，来显示文字

然后设置一下字体和颜色

在细节面板中找到“Theme Overrides”，在这里设置字体和颜色

然后设置内容

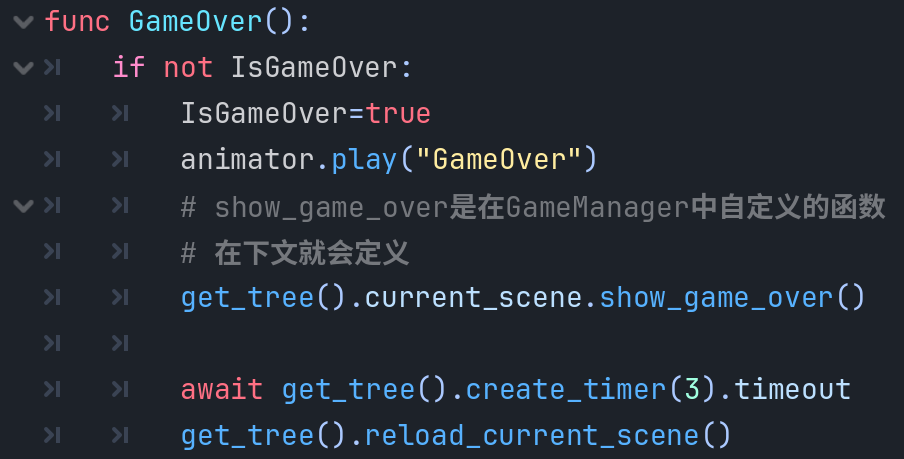


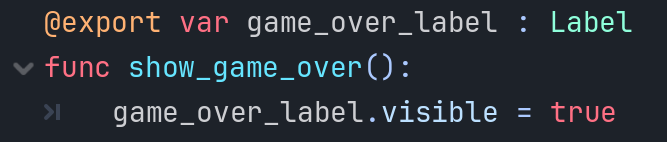
最后创建游戏结束UI

给CanvasLayer加一个Label类节点，命名为GameOverUI

GameOverUI默认为隐藏

然后进入player脚本，通过玩家蓝图来告诉GameManager游戏结束了，然后再由GameManager来显示这个文本Label





## 播放声音

播放声音由AudioStreamPlayer节点实现。

播放声音就是使用节点的play（）函数

停止播放就是使用stop（）函数

跑步声音

记得启用循环



这个时候，因为跑步音效是循环播放的，所以不能多次调用play函数，方法就是检测这个Audio的playing参数是否为真

## 声音源

AudioStreamPlayer播放的是没有左右声道区分的声音

AudioStreamPlayer2D播放的是有左右声道区分的声音

一般用在耳机和音响上

## 固定加载的类（蓝图）（场景）

在运行游戏的时候，一般是加载该场景的根节点。默认所有的东西都是这个场景的根节点的子节点

如果有一种类，比如全局bgm，想让他不受战斗场景的影响，可以将这个bgm单独保存为一个类（单独保存为一个场景），然后在项目设置中，在“全局”菜单下添加自动加载的类（场景）

## 导出游戏

导出到PC和浏览器

不同分辨率下的窗口缩放

在项目设置中找到“常规--显示--窗口”

“拉伸--模式”选择canvas item

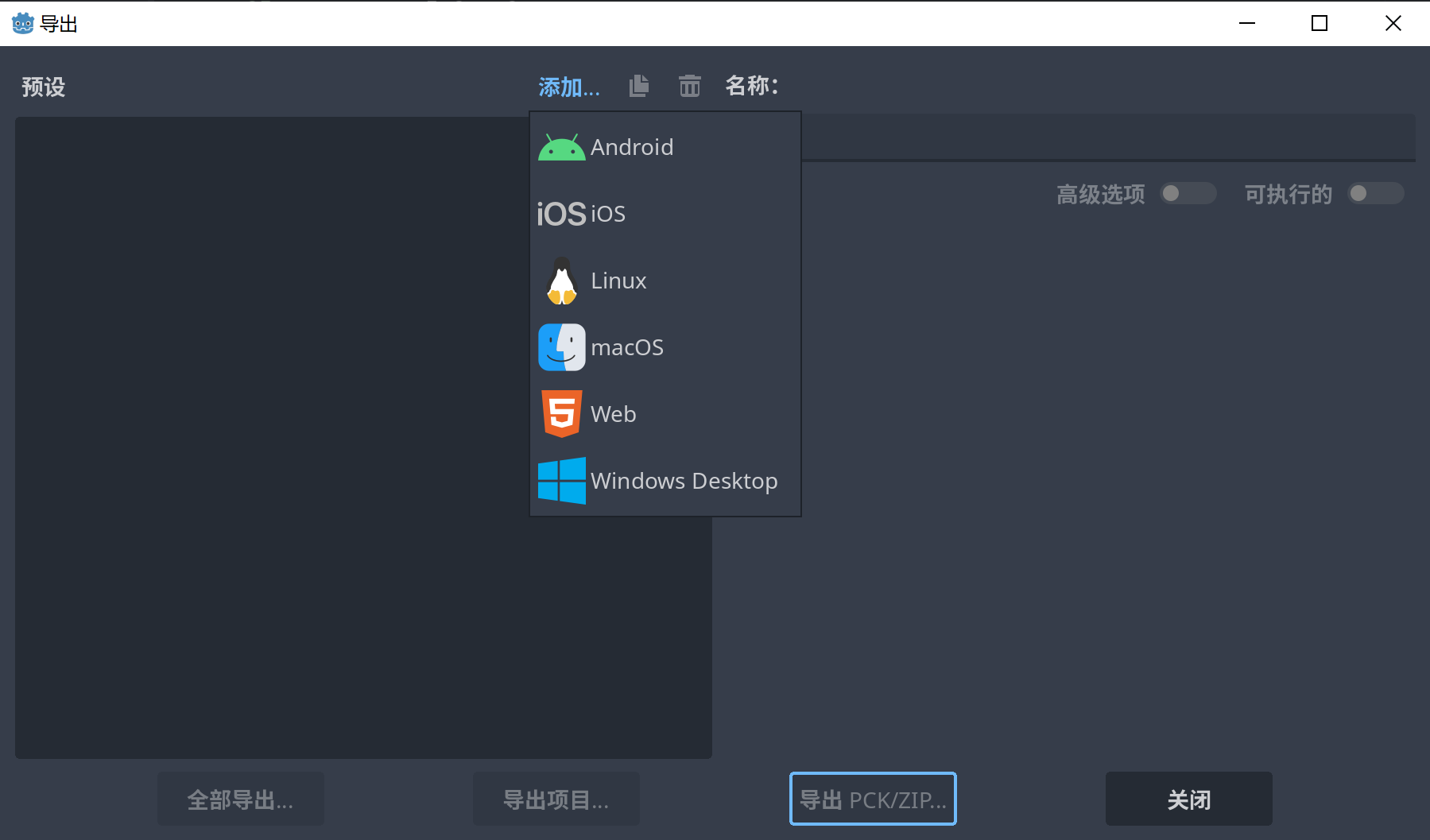
“大小--模式”选择Exclusive Fullscreen

导出模板

在godot的菜单栏中选择“编辑器--管理导出模板”，首先在这里下载导出模板

然后点击godot的菜单栏中“项目--导出”

我们可以添加导出预设



以windowsDesktop为例，导出设置基本不用改，不过可以改一下“二进制格式--内嵌PCK”为真，这样导出的就直接是exe格式

也可以导出到网站

导出后将导出的文件夹压缩成一个zip

网站这里选择itch.io

如果想让游戏在itch.io上游玩，需要导出的文件名称为index.html

然后进入itch.io

在这里面上传html格式的游戏时，需要上传压缩包

然后勾选“该文件将在浏览器中运行”

在这里windows版也可以上传，直接上传exe就可以了。记得勾选平台

在itch.io中有可能不能运行godot游戏，解决办法是在上传的时候勾选“SharedArrayBuffer support -(Experimental)……”这个选项