

文件说明

Assets存放的是资源集，有装饰物，敌人，塔防等图片

Bullet存放的是子弹场景后缀(.tscn)

Decorations存放的是场景装饰 比如树，花等

Global下存放的是存放全局变量与函数的脚本

Mobs下存放的是怪物场景

MobsMove下存放的是怪物移动的场景

Resources下存放的是升级卡牌资源后缀(.tres)

Scripts下存放的是脚本

Tiles下存放的是瓦片集拼成的地图

Towers下存放的是塔和布置塔面板的场景

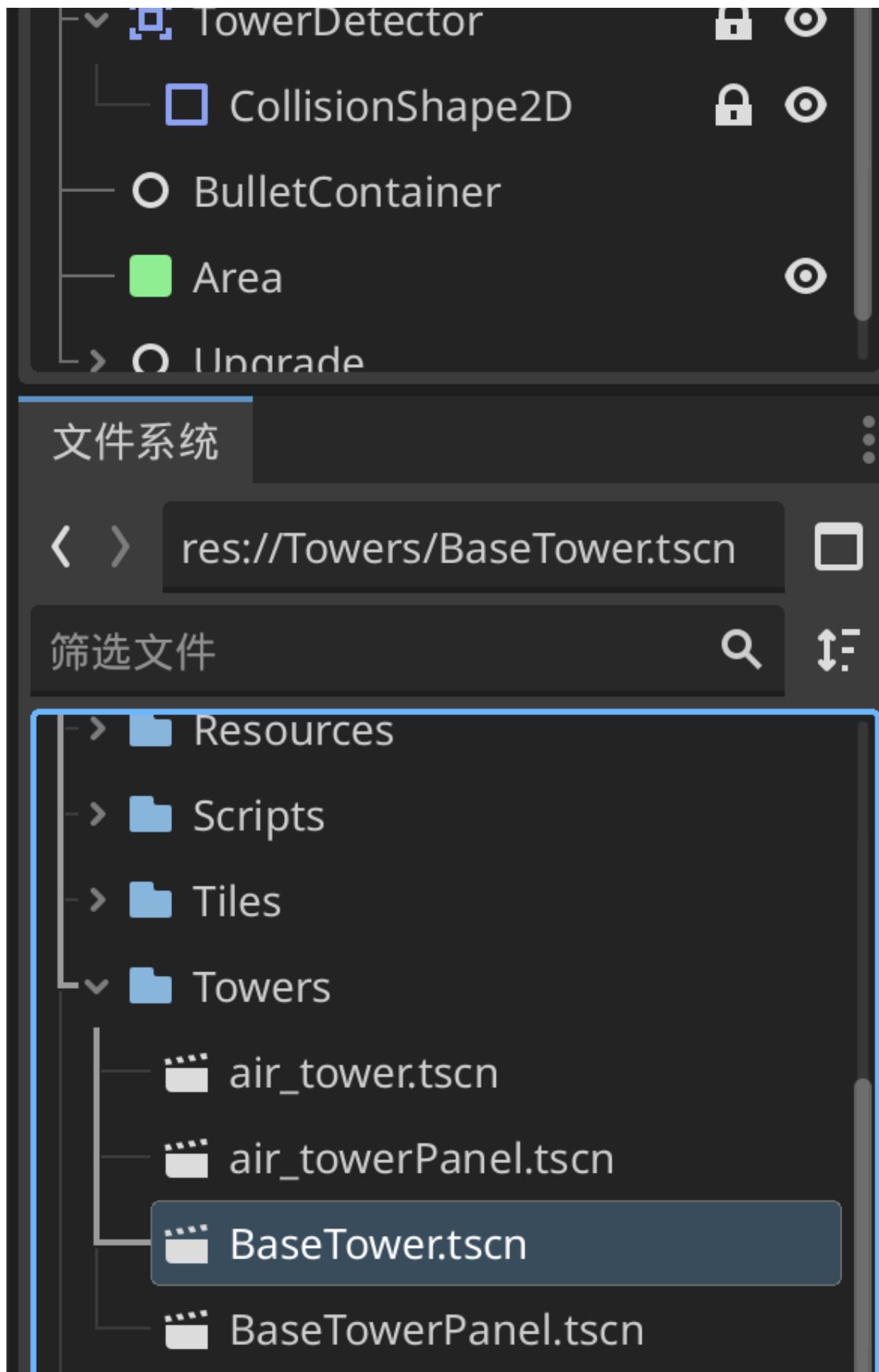
UI下存放的是UI场景

如何创建新的场景

下面以塔做演示

首先点击带有Base前缀的场景



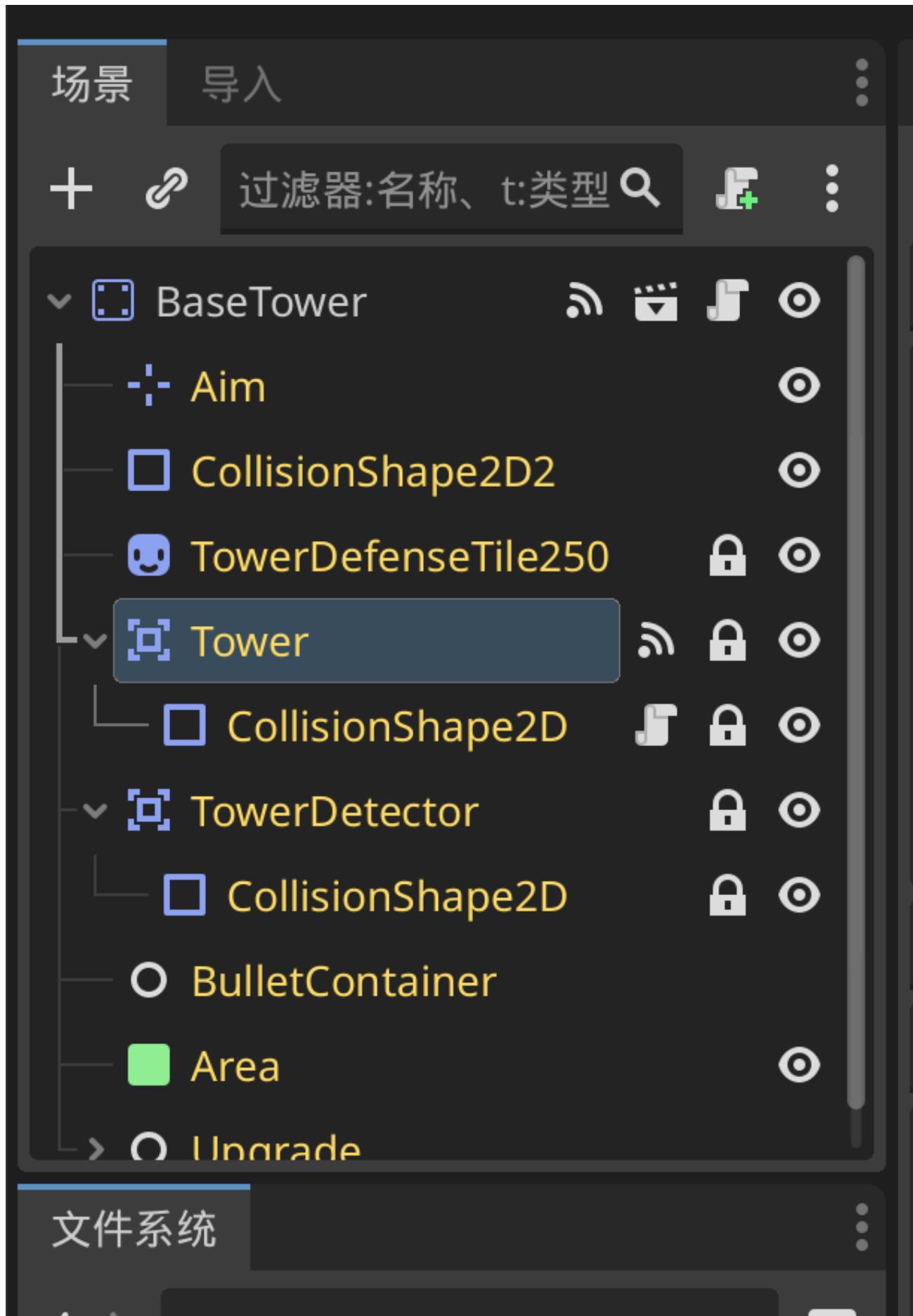


右键场景 选择新建继承场景

- CollisionS
 - TowerDetector
 - CollisionS
 - BulletContact
 - Area
 - Upgrade
- 系统
- res://Towers
 - 文件
 - Resources
 - Scripts
 - Tiles
 - Towers
 - air_tower:
 - air_towerl
 - BaseTower
 - BaseTower
 - UI
-  新建继承场景
 -  设为主场景
 -  实例化
 -  编辑依赖...
 -  查看所有者...
 -  新建
 -  复制路径 Shift+Command+C
 -  复制绝对路径 Option+C
 -  复制 UID Shift+Option+C
 -  重命名...
 -  复制为...
 -  移动/复制到...
 -  删除 Command+Delete
 -  添加到收藏
 -  在终端中打开文件夹 Option+Command+Shift+O
 -  在外部程序中打开 Option+Command+Shift+E



新建的继承场景节点为黄色



如果只是需要更换贴图，在右侧监视找到材质更换



TowerDefenseTile250



筛选属性



Sprite2D

Texture



64×64 RGBA8

无 Mipmap

内存 : 16.00 KiB

场景

导入



过滤器:名称、 t:类型



BaseTower

Aim

CollisionShape2D

TowerDefenseTile250

Tower

CollisionShape2D

再修

TowerDetector

CollisionShape2D

BulletContainer

Area

Upgrade

文件系统

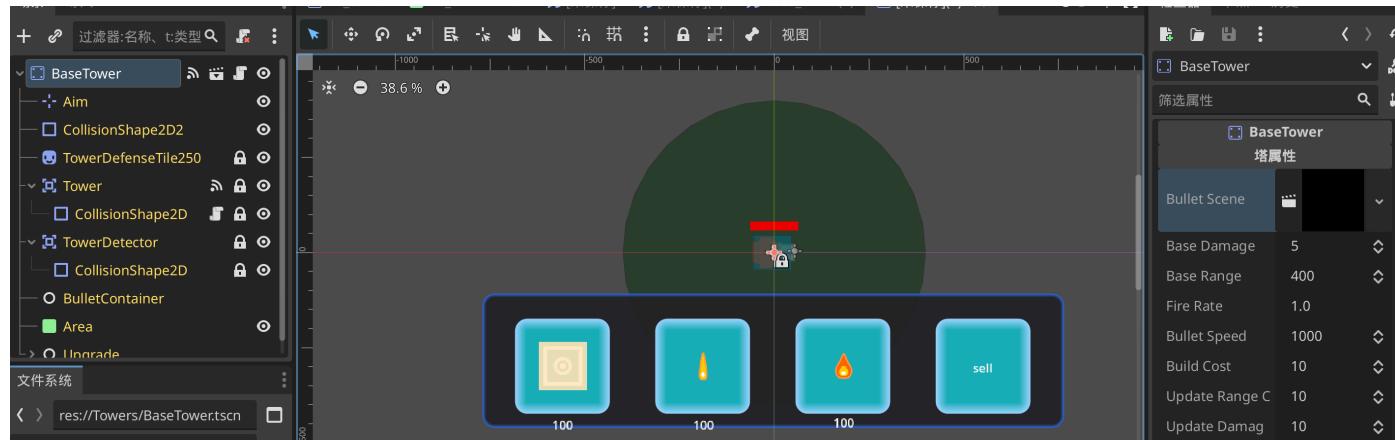


res://Towers/BaseTower.tscn

改collisionShape2d参数调整 碰撞体积

这里2D2是塔的体积。Tower下的2D是判断敌人进入进出的体积，TowerDetector是放塔时能否放塔的体积，检测体积内是否有其他物体。

如需更新数值点击母节点，即可看见右侧可更改的数值



如需要添加更多需要继承的数值修改

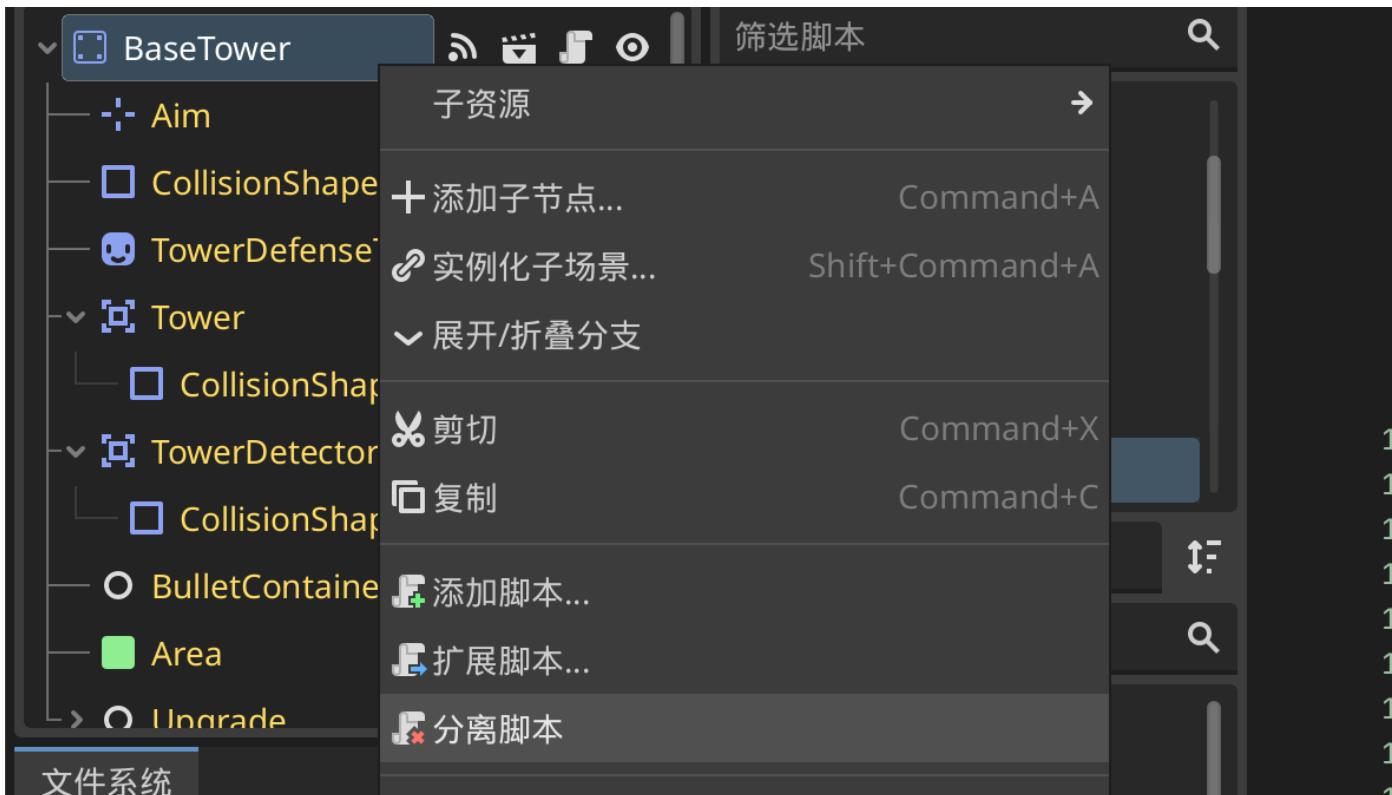
到basetower对应的脚本处更改

```
1 extends StaticBody2D
2 class_name BaseTower
3 @export_category("塔属性")
4 @export var bullet_scene: PackedScene = preload("res://Bullet/")
5 @export var base_damage: int = 5
6 @export var base_range: int = 400
7 @export var fire_rate: float = 1.0 # 射击间隔 (秒), 替代原本的 rate
8 @export var bullet_speed: int = 1000#子弹飞行速度
9 @export var build_cost: int = 10#建造花费
10 @export var update_range_cost: int = 10
11 @export var update_damage_cost: int = 10
12 @export var update_fire_rate_cost: int = 10
13
```

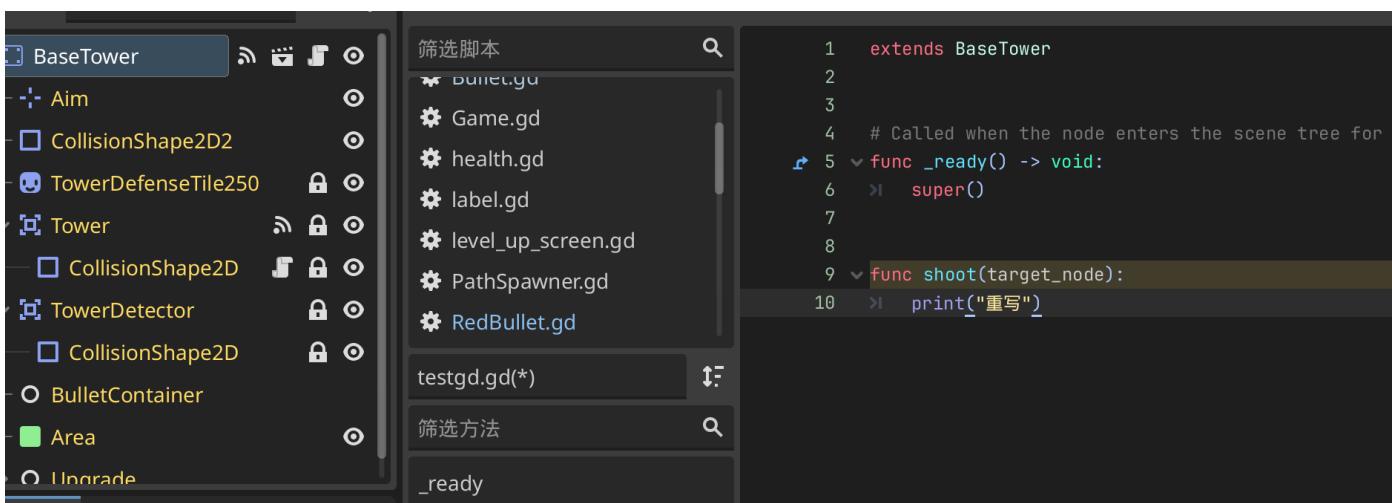
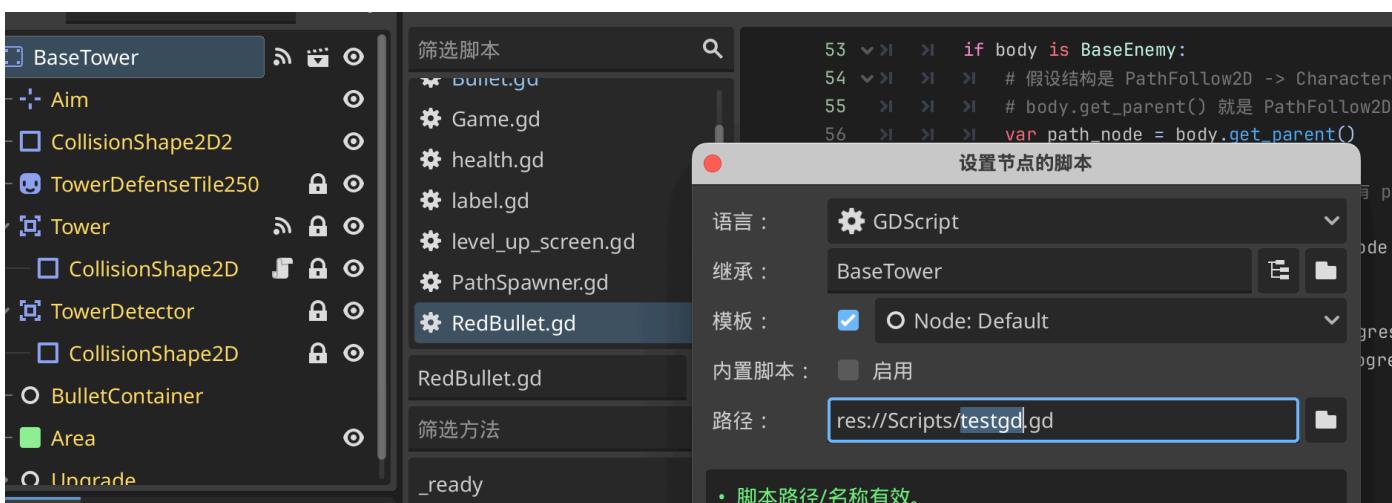
export的属性就是新建继承场景后可填写更改的数值

实现新功能

这里要注意 继承场景的脚本都是跟原场景的一样，如果直接修改就会更改原场景脚本，影响其他节点的继承。如果继承场景需要新功能实现需先进行分离脚本



后添加脚本，继承找到要继承的类



ready中调用父亲初始化 通过super调用父类方法，或是重写原方法.

```
1  extends BaseTower
2
3
4  # Called when the node enters the scene tree for
5  ↳ 5  func _ready() -> void:
6    >I  super()
7
8
9  ↳ 9  func shoot(target_node):
10   >I  print("重写")
11   >I  super.Shoot(target_node)
```

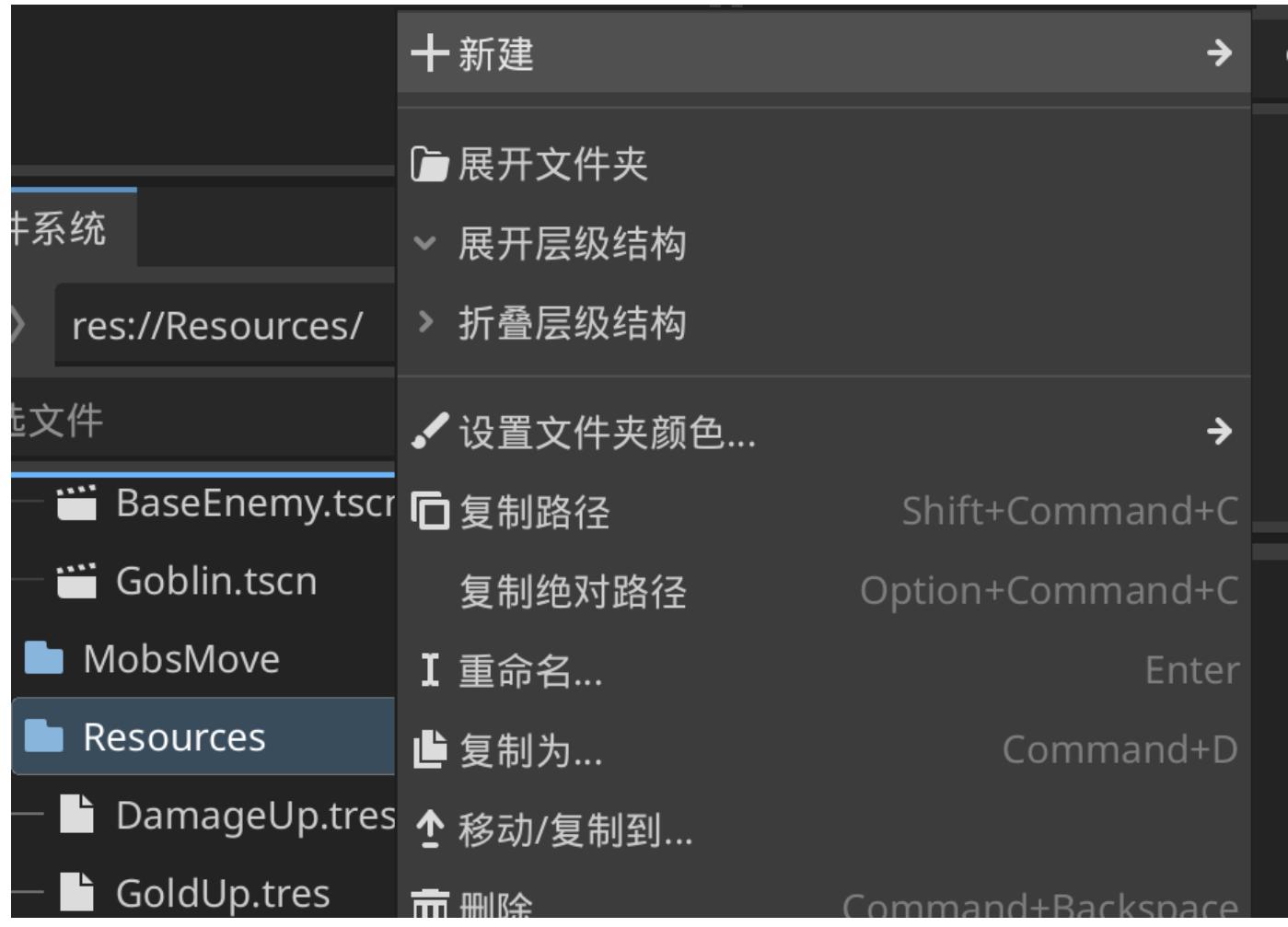
升级卡片逻辑

upgrade_item.gd脚本中定义的继承后需要填写的属性

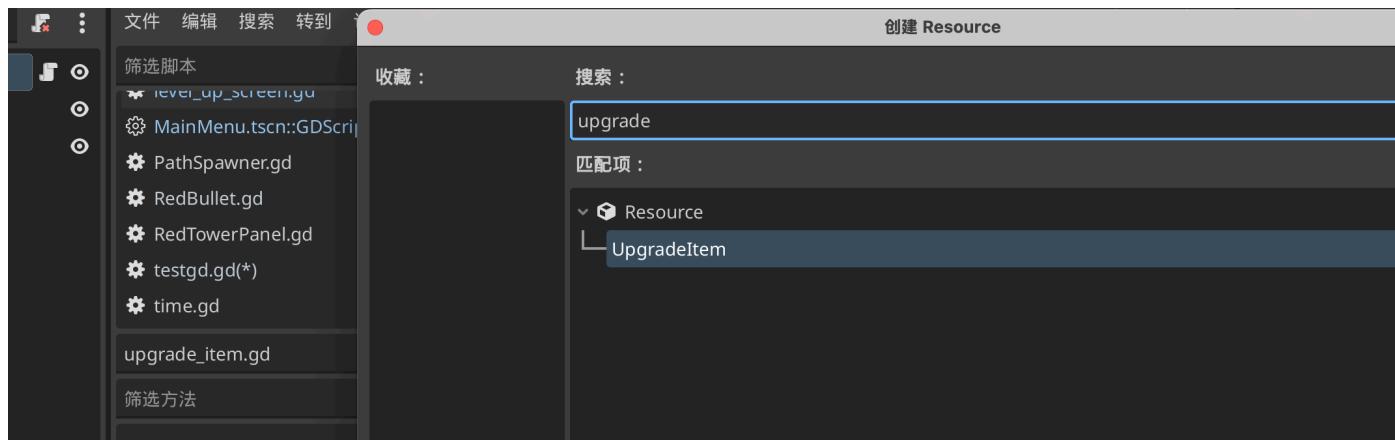
```
1  extends Resource
2  class_name UpgradeItem
3
4  @export_group("显示信息")
5  @export var title: String = "升级名称"
6  @export_multiline var description: String = "升级描述"
7  @export var icon: Texture2D # 用来存图标 (png/jpg)
8
9  @export_group("游戏数据")
10 @export_enum("damage", "speed", "gold", "health") var id: String
11 @export var value: int = 0.0
12
```

@export_enum("damage", "speed", "gold", "health") var id: String

对应了逻辑实现



要生成一张，需要在Resources文件夹右键新建资源选择UpgradeItem类



在右侧填写数值

UpgradeItem

▼ 显示信息

Title

升级名称

Description

升级描述



Icon

<空>



▼ 游戏数据

ID

damage



Value

0



Resource

选择后的处理逻辑在level_up_screen.tscn场景的脚本中

```
0  ↘ func _on_card_selected(item: UpgradeItem):  
1    ↗   print("选择了升级: ", item.title)↗  
2    ↗   apply_upgrade_effect(item)  
3    ↗   visible = false  
4    ↗   get_tree().paused = false  
5    ↗   Game.gain_research(0)  
6  
7  ↘ func apply_upgrade_effect(item: UpgradeItem):  
8    ↗   match item.id:  
9      ↗   "damage":  
10     ↗   ↗   Game.global_damage_bonus += item.value  
11      ↗   "speed":  
12     ↗   ↗   Game.global_speed_bonus += item.value  
13      ↗   "gold":  
14     ↗   ↗   Game.gain_gold(int(item.value))  
15
```