Godot学习笔记

命名规则

类、导出变量和方法使用 PascalCase（大驼峰命名法），私有字段使用 _camelCase（前缀下划线的小驼峰命名法），局部变量和参数使用 camelCase（小驼峰命名法）。连接信号时，请务必准确键入方法名称。

脚本

默认成员方法

_EnterTree()

在Godot中，_EnterTree是一种回调函数，用于在节点被添加到场景树中时执行一些操作。

_Process(double delta)

其中变量delta表示时间。

引擎和游戏开发者尽最大努力以恒定的时间间隔更新游戏世界和渲染图像，但在帧的渲染时间上总是存在着微小的变化。这就是为什么引擎为我们提供了这个delta时间值，使我们的运动与我们的帧速率无关。

_ExitTree()

与_EnterTree同理，将节点从场景树中移除的时候调用。

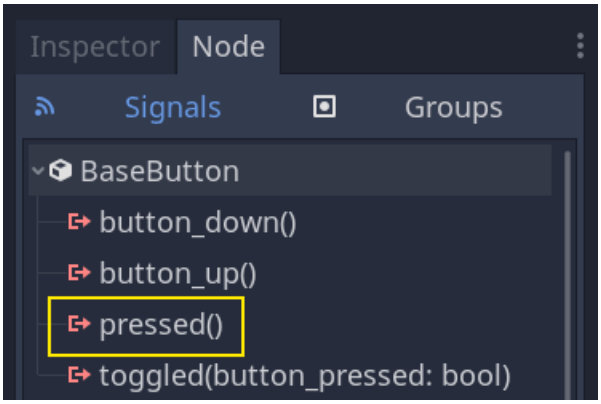
信号

通过界面添加信号

通过c#在界面添加信号目前还有一些bug，需要手动在脚本中添加相关函数，不能自动生成。

以下以官网的例子为例，实现其c#版本。

1. 创建一个sprite节点和一个button节点。
2. 选择 Button 节点，然后在编辑器的右侧，单击检查器旁边名为“节点”的选项卡。双击“pressed”信号，打开节点连接窗口。如下图所示。



3. 点击连接，生成回调方法。按照惯例，这些回调方法命名为“_on_Button_pressed”。

4. 由于Godot目前对c#支持仍有bug，因此需要手动添加该方法。

完整代码如下，预期效果是点击button的时候，Logo停止。

```
public partial class Logo : Sprite2D
{
...
    public void _on_logo_button_pressed() {
        SetProcess(!IsProcessing());
    }
...
}
```

用代码连接信号

1. 创建一个sprite节点并添加Timer节点。
2. 选中Timer节点并勾选 **Autostart**（自动开启）属性。
3. 从 Sprite 获取 Timer 的引用。
4. 将Timer的信号与节点相连。

完整代码如下，预期效果是Logo一秒闪一次。

```
public partial class Logo : Sprite2D
{
    public override void _Ready()
    {
        Timer timer = GetNode<Timer>("Timer");
        timer.Timeout += _on_Timer_timeout;
    }

        private void _on_Timer_timeout() {
        Visible = !Visible;
    }
}
```

自定义信号

没搞懂，以后用的时候再说吧。

Node

CanvasItem

Node2D

CollisionShape2D

- bool **disabled** - 一个有缺陷的碰撞形状在世界上不会造成影响。
- bool **one_way_collision** - 设置碰撞形状是否只检测一侧（顶部或底部）的碰撞。
- Shape2D **shape** - 这个碰撞形状所拥有的实际形状。

Reference

Resource

Curve2D