**UnityEvent**

借助 UnityEvent 可让用户驱动的回调从编辑时间一直持续到运行时，无需进行额外的编程和脚本配置。

UnityEvent 对许多方面都很有用：

* 内容驱动的回调
* 解耦系统
* 持久回调
* 预配置的调用事件

UnityEvent 可添加到任何 MonoBehaviour，并从标准 .net 委托之类的代码中执行。当 UnityEvent 添加到 MonoBehaviour 时，它会出现在 Inspector 中，并可添加持久回调。

UnityEvent 与标准委托有类似的限制。也就是说，它们会保留对目标元素的引用，而这会阻止对目标进行垃圾收集。如果将 UnityEngine.Object 作为目标，而本机表示消失，则不会调用回调。

**使用 UnityEvent**

要在 Editor 中配置回调，需执行以下几个步骤：

0.确保脚本导入/使用 UnityEngine.Events。

1.选择 + 图标为回调添加字段

2.选择要接收回调的 UnityEngine.Object（可使用对象选择器进行选择）

3.选择要调用的函数

4.可为事件添加多个回调

在 Inspector 中配置 UnityEvent 时，支持两种类型的函数调用：

* 静态。静态调用是预配置的调用，具有在 UI 中设置的预配置值。这意味着，在调用回调时，使用已在 UI 中输入的参数调用目标函数。
* 动态。使用从代码发送的参数调用动态调用，并与正在调用的 UnityEvent 类型相关。UI 会过滤回调，仅显示对 UnityEvent 有效的动态调用。

**通用 UnityEvent**

默认情况下，Monobehaviour 中的 UnityEvent 动态绑定到 void 函数。但不一定非得如此，因为 UnityEvent 的动态调用支持绑定到最多包含 4 个参数的函数。为此，您需要定义一个支持多个参数的自定义 UnityEvent 类。此定义十分简单：

[Serializable]

public class StringEvent : UnityEvent <string> {}

通过将此类的实例添加到您的类而不是基本 UnityEvent，即可使回调动态绑定到字符串函数。

然后，可通过调用以 string 为参数的 Invoke() 函数来对其进行调用。

UnityEvent 可在其通用定义中定义最多 4 个参数。

# 什么是 Null 引用异常 (Null Reference Exception)？

尝试访问未引用任何对象的引用变量时，便发生 NullReferenceException。如果引用变量未引用任何对象，则将其视为 null。当变量为 null 时，运行时将通过发出 NullReferenceException 来告知正在尝试访问对象。

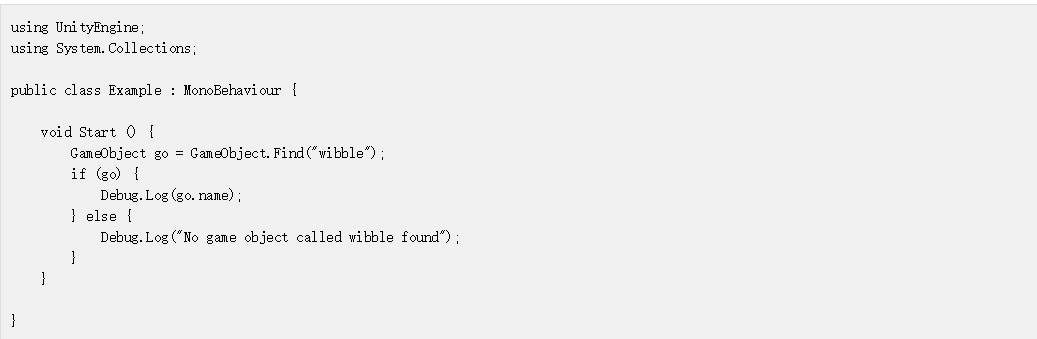
c# 和 JavaScript 中的引用变量在概念上类似于 C 和 C++ 中的指针。引用类型默认为 null，表示未引用任何对象。因此，如果尝试访问被引用的对象而又没有该对象，则将出现 NullReferenceException。

当代码中出现 NullReferenceException 时，意味着在使用变量之前忘记设置该变量。错误消息将如下所示：

此错误消息表明 NullReferenceException 发生在脚本文件 Example.cs 的第 10 行。此外，该消息还指出异常发生在 Start() 函数内。根据这些信息，比较容易查找和修复 Null 引用异常。在此示例中，代码为：

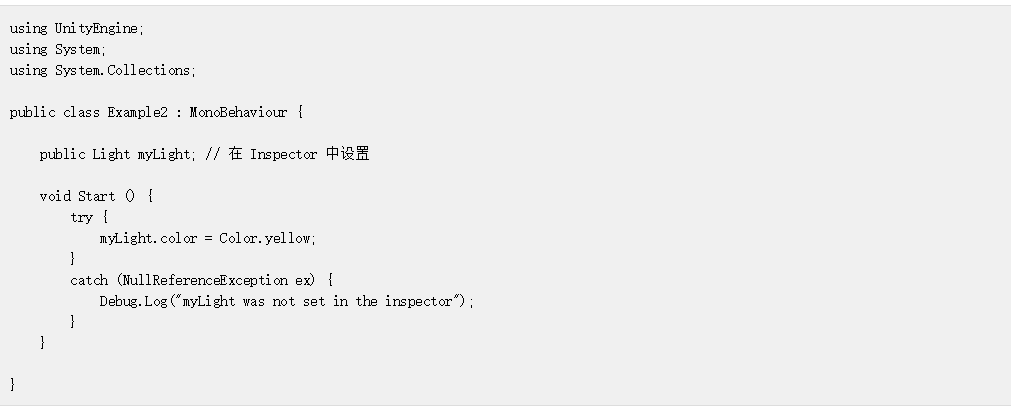
该代码简单地查找一个名为“wibble”的游戏对象。在此示例中不存在该名称的游戏对象，因此 Find() 函数返回 null。在下一行（第 9 行），我们使用 go 变量并尝试打印出其引用的游戏对象的名称。因为我们正在访问一个不存在的游戏对象，所以运行时抛出一个 NullReferenceException

## Null 检查

虽然发生这种情况时令人沮丧，但这只是意味着需要更加注意脚本。这个简单示例中的解决方案是按如下所示的方式更改代码：

现在，在我们尝试对 go 变量执行任何操作之前，我们检查它是不是 null。如果是 null，则显示一条消息。

## Try/Catch 代码块

出现 NullReferenceException 的另一个原因是使用了应该在 Inspector 中初始化的变量。如果忘记这样做，则变量将为 null。处理 NullReferenceException 的另一种方法是使用 try/catch 代码块。例如，以下代码：

在此代码示例中，名为 myLight 的变量是应该在 Inspector 窗口中设置的 Light。如果未设置此变量，则默认为 null。尝试在 try 代码块中改变光源的颜色会导致被 catch 代码块捕获到的 NullReferenceException。catch 代码块将显示一条消息，此消息可能对美术师和游戏设计师更有帮助，并提醒他们在 Inspector 中设置光源。

**总结**

* 脚本代码尝试使用未设置（引用）的变量和对象时，便会发生 NullReferenceException。
* 出现的错误消息可显示有关在代码中何处发生了问题的大量信息。
* 通过编写代码在访问对象之前检查 null 或使用 try/catch 代码块，可避免 NullReferenceException。