CastTo节点要尽可能地少用，会增加内存占用

大型程序项目设计中，尽可能的解耦

有这样一种情况：有一个Widget显示剩余角色数量，每当杀死敌人时更新数据。还有一个门，只有当所有敌人都死亡时他才会打开。所有对象之间都需要互相通信：Widget、门、Level，甚至在Enemy和Player之间也需要发送。如何正确的实现这一切呢？

## 观察者模式（发布者/订阅者模式）

问题的重点就是，游戏中不同对象之间的通信与信息共享。我们该如何创建一种可扩展的、易于维护的方式去实现。

就从上面说的这个例子，我们该如何实现角色死亡时动态更新剩余角色数量？

一种错误的方法是直接获取对Widget的引用。虽然能实现功能，但是这样就耦合了角色和Widget。

观察者模式是基于事件发生时进行信号广播的模式。游戏中其他接收到广播信号的对象，就会对这个事件进行反应。这意味着你可以拥有一个对象，在某处广播一个信号，却不需要考虑会发生什么。游戏中的其他对象可以监听这个事件，并对其作出不同的反应。

在蓝图中，这是非常容易实现的，只需要定义一个事件分发器EventDispatcher就可以实现。

在C++中，则需要定义Delegate。

这里，我们只关注蓝图中的实现。C++中的思路类似，只是实现方式略微不同。

通常情况下，事件分发器会命名为OnXXX，因为是一个触发器。

## 发布者

打开Enemy蓝图。现在我们想把Enemy类作为发布者来广播一个信号，来广播已经死亡。

在Enemy类中创建事件分发器命名为OnDeath，创建自定义事件Death。

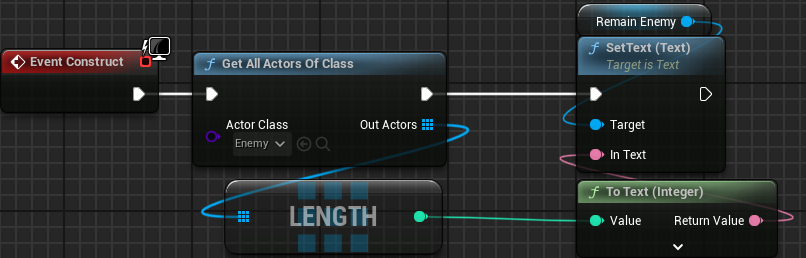


如果想通过事件分发器传递参数，在事件分发器的细节面板最上面的输入中加上参数。这里我们并不需要参数。

发布者的行为已完成

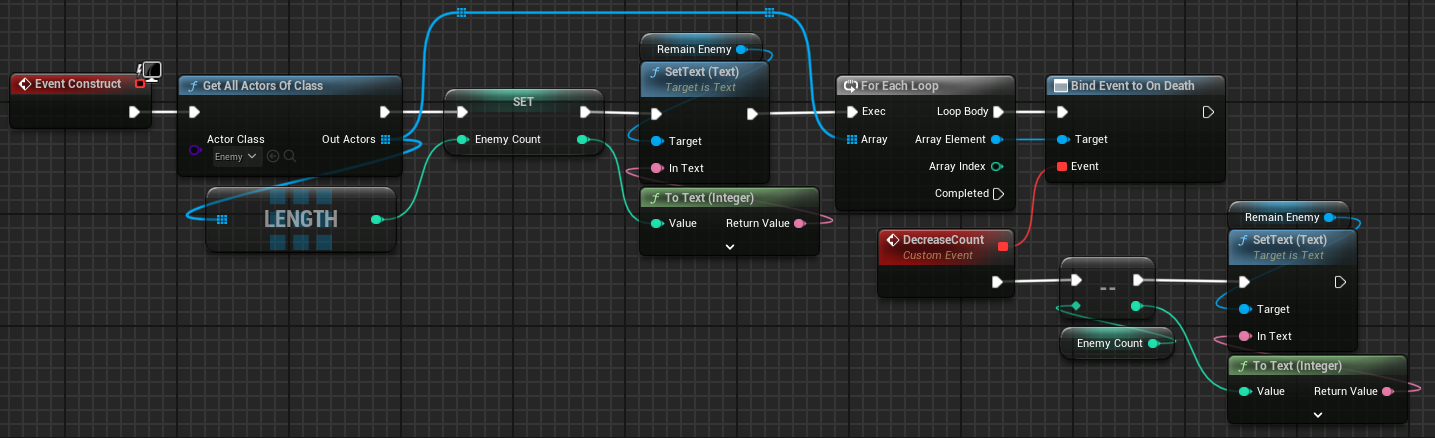
## 订阅者

在WBP\_ReaminEnemy中，



接下来我们需要在WBP中订阅事件

为了订阅事件，需要使用bind关键字



完成功能

观察者模式的好处就是可以用多个订阅者订阅同一个事件去完成不同的任务。

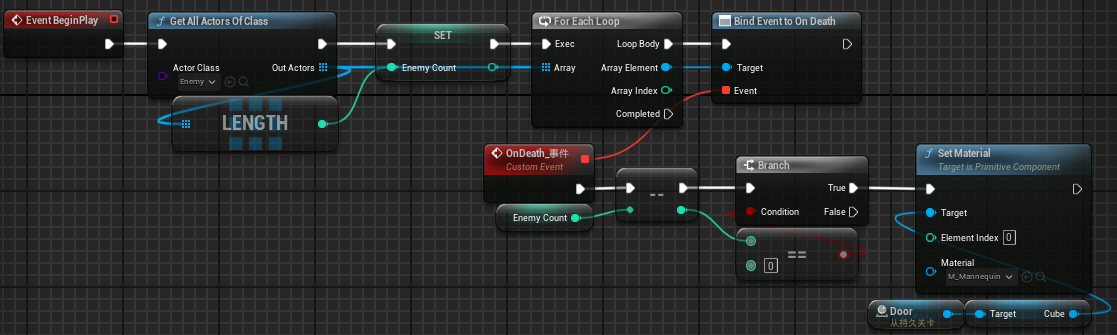
## 订阅者（开门）

游戏中有很多门，他们的开关原因可能是不同的。所以我们可以在关卡蓝图中实现。

在关卡蓝图中添加门对象的引用



我们设计：当所有敌人都死亡的时候，设置门的材质，代替开门的功能。



完成功能。

C++项目下的观察者模式：

Ali Elzoheiry 的虚幻引擎中的设计模式系列教程：观察者模式 https://www.youtube.com/watch?v=YFtLd-bKl-U

--------------------------------------------------------------------------------

设计模式就是如何编写良好代码的指南，无论您使用蓝图还是 C++，了解良好的软件实践都是必须的！

我希望本系列的第一个视频能够介绍何为设计模式，如何使用观察者模式。

这会通过「事件分发器」的案例来完成，虚幻引擎中的事件分发器使用Delegate，是观察者模式的实现。

通过蓝图，可以调用事件分发器，但有某些限制，本系列将讨论这些限制。

下载项目文件并在 Patreon 上支持我：

https://www.patreon.com/c/Alizoh/home

加入我的 Discord 服务器： https://discord.com/invite/AEnM4wvYzN

额外资源：

https://refactoring.guru/design-patterns/observer https://gameprogrammingpatterns.com/observer.html https://dev.epicgames.com/documentation/en-us/unreal-engine/delegates-and-lamba-functions-in-unreal-engine?application\_version=5.2