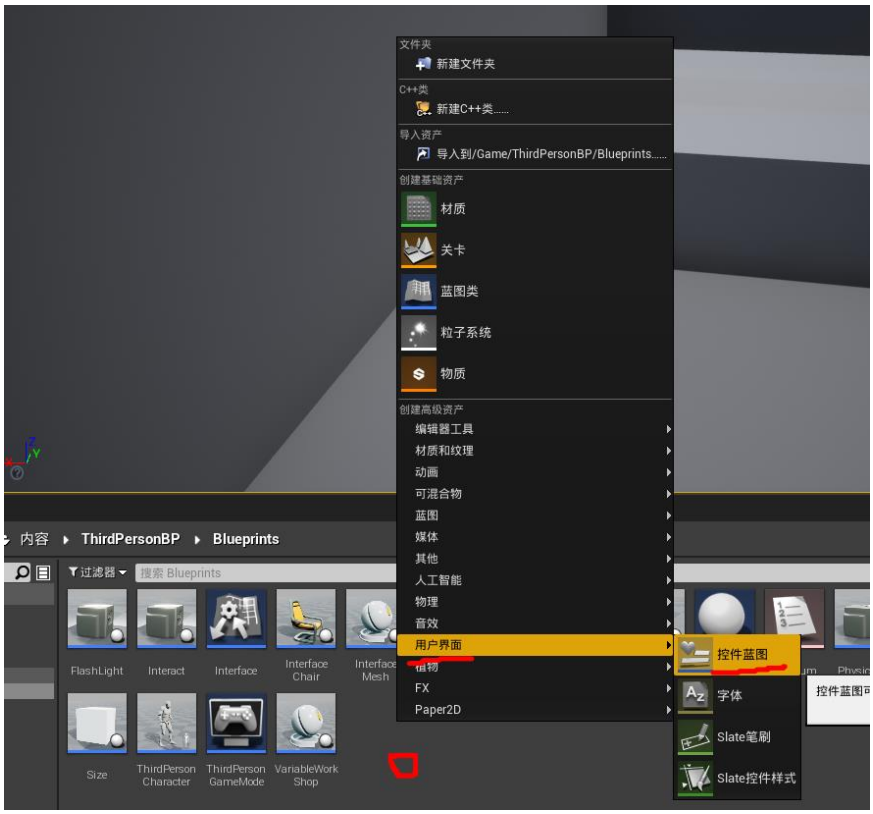
第 11 课 UMG

1、UMG 概述

虚幻运动图形界面设计器（UMG, Unreal Motion Graphics UI Designer）是一款视觉创作工具，可用于实现游戏中的用户 UI 界面，比如游戏里的 HUD（Heads-up Display）、菜单或与界面相关的其他图形。UMG 的核心是控件，用于构成界面，如按钮（Button）、滑块（Slider）、进度条（Progress Bar）。这些控件可以在蓝图中编辑，编辑时用到两个选项卡进行构建：Designer 设计器用于实现界面的视觉布局，Graph 图形用于实现使用控件时提供的功能。

2、控件蓝图（Widget Blueprint）

控件蓝图用于设计 UI 布局样式和编写脚本。本节学习控件蓝图编辑器界面布局及针对控件蓝图的基本操作。继续之前的项目，在内容浏览器鼠标右键，创建“用户界面>>控件蓝图”，命名为 MyUI。



打开 MyUI 蓝图，控件蓝图编辑器布局如下：

UMG UI 设计器由 7 个主要面板组成：

（1）视觉设计器（Designer）：显示放置的控件，可以在此窗口中编辑控件的位置、大小和布局。

（2）细节面板（Details）：显示选中控件的属性。

（3）控制板（Palette）：可以把其中的控件拖放到视觉设计器（Designer）中。

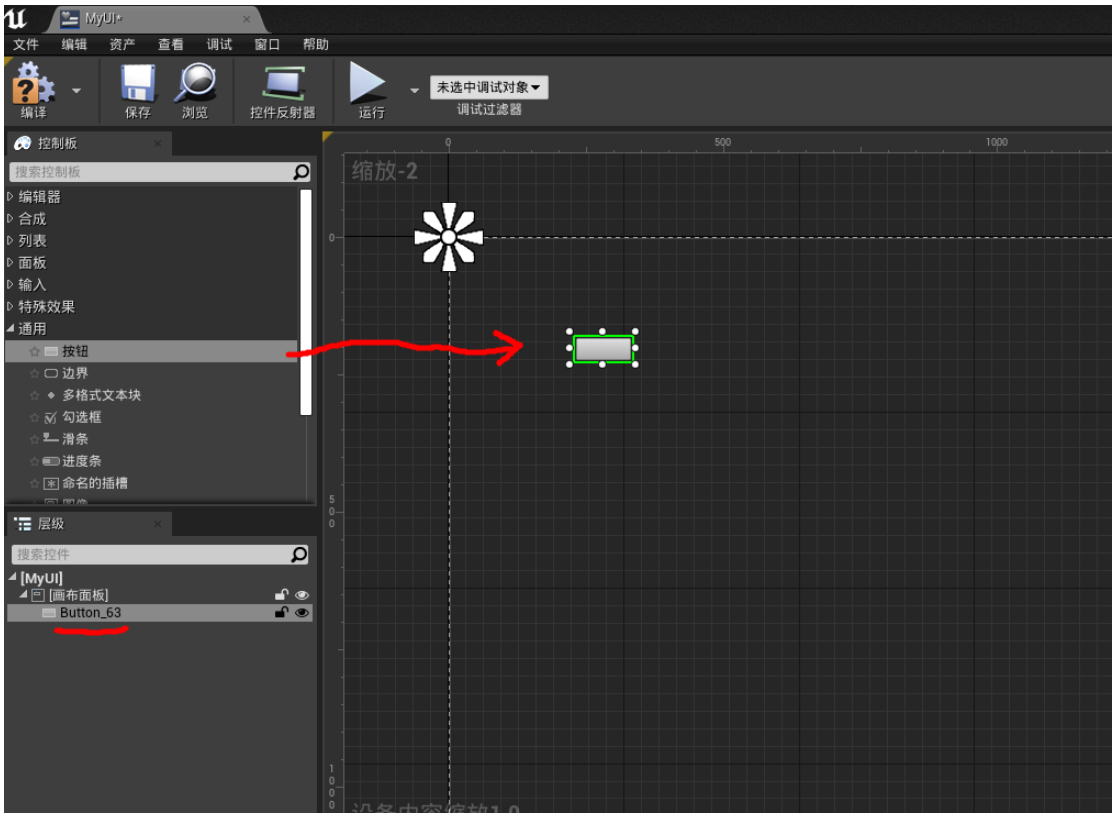
（4）层级（Hierarchy）：显示控件与控件之间的层次关系。

（5）动画（Animations）：用于设计控件的关键帧动画。

（6）时间轴（Timeline）：设置动画的关键帧。

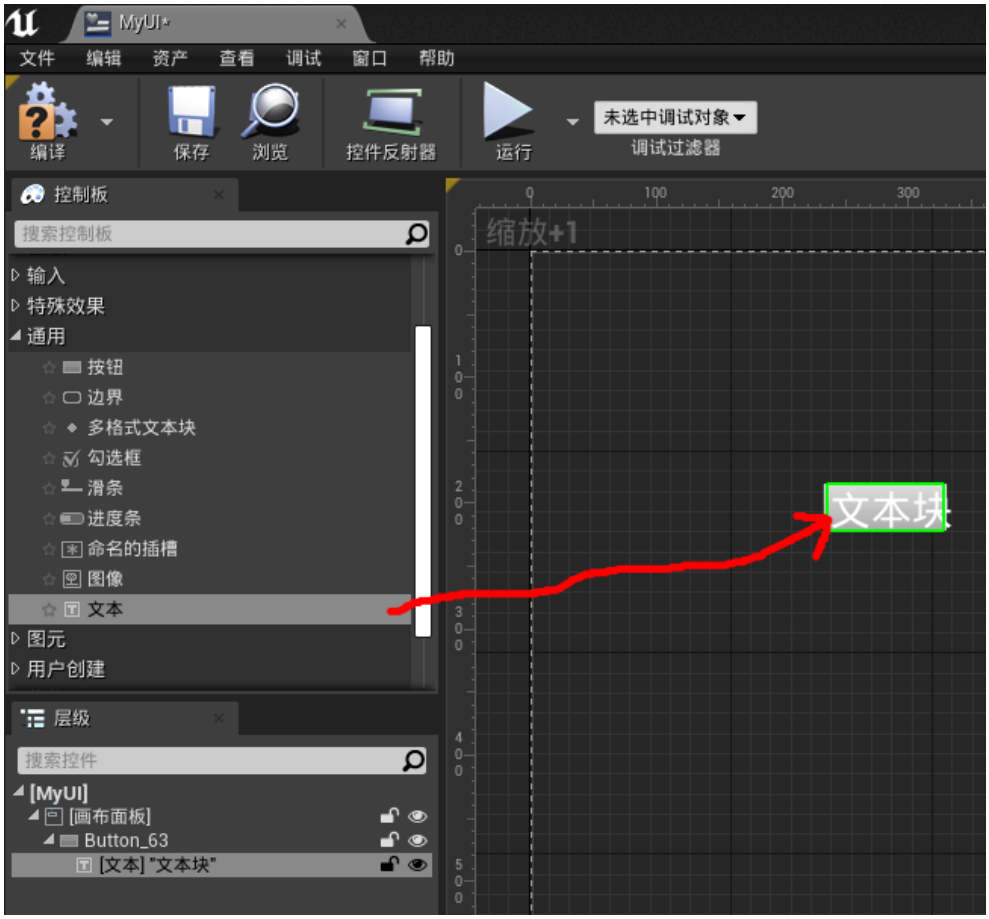
（7）编辑模式（Editor Mode）：在设计器和图表模式之间切换。图表模式与蓝图的事件图表几乎相同。

现在向视觉设计器图表里添加控件。从“控制板（Palette）”中找到“通用（Common）>>按钮（Button）”控件，拖动至视觉设计器中。可以看到“按钮”控件被添加到视觉设计器中，默认的名字叫 Button_63（这个名字会有不同）。



在“层级”面板中，还可以看到“画布面板（Canvas Panel）”控件下增加了一个子控件 Button_63。点击控件，右侧细节面板会显示该控件的一系列属性值。

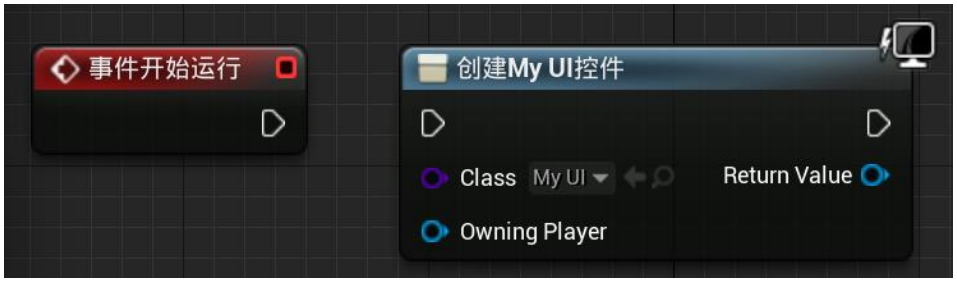
在控制板中找到“通用（Common）>>文本（Text）”控件，拖动至视觉设计器的 Button_63 控件中。



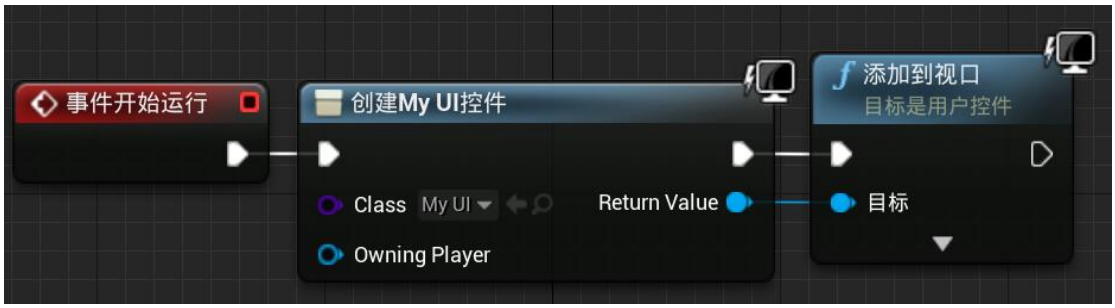
可以看到在 Button_63 控件中嵌入了一个“文本”控件，即“文本”控件成为了 Button_63 控件的子控件，在“层级”面板中可以看到 Button 控件下增加了一个文本子控件。

编译保存 MyUI 蓝图。创建控件蓝图并设计好布局后，如果要令其显示在游戏内，需要在关卡蓝图或者（第一/第三）角色蓝图中调用它。打开关卡蓝图，添加“事件开始运行（Event BeginPlay）”节点和“创建控件（Create Widget）”节点。

“创建控件”节点的 Class 引脚指定为需要创建的控件蓝图（MyUI）。



添加“添加到视口（Add to Viewport）”节点，该节点用于在屏幕上绘制由“目标”引脚指定的控件蓝图，将控件像新窗口一样添加到根窗口。连接连线。



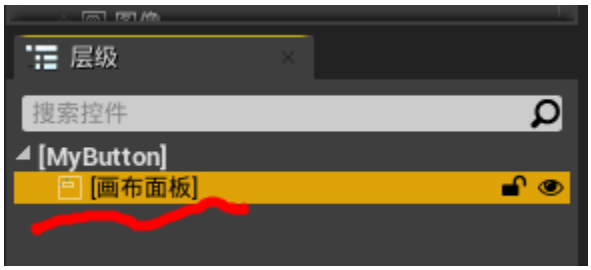
在之前的学习中，可以注意到当游戏运行后，鼠标光标会被隐藏起来，这种情况下玩家无法和 UI 进行交互，因此为了在游戏运行后启用鼠标光标，需要添加“获取玩家控制器”节点，并用该节点找到“设置 Show Mouse Cursor”节点。勾选“Show Mouse Cursor”选项。

如果在添加“获取玩家控制器”节点前，先搜索“设置 Show Mouse Cursor”节点，会无法找到该节点，此时取消“情境关联”的情况下才能找到。但这样的做法并不是好习惯，在不充分了解该节点的使用之前，很可能会导致错误的节点调用（编译报错）。

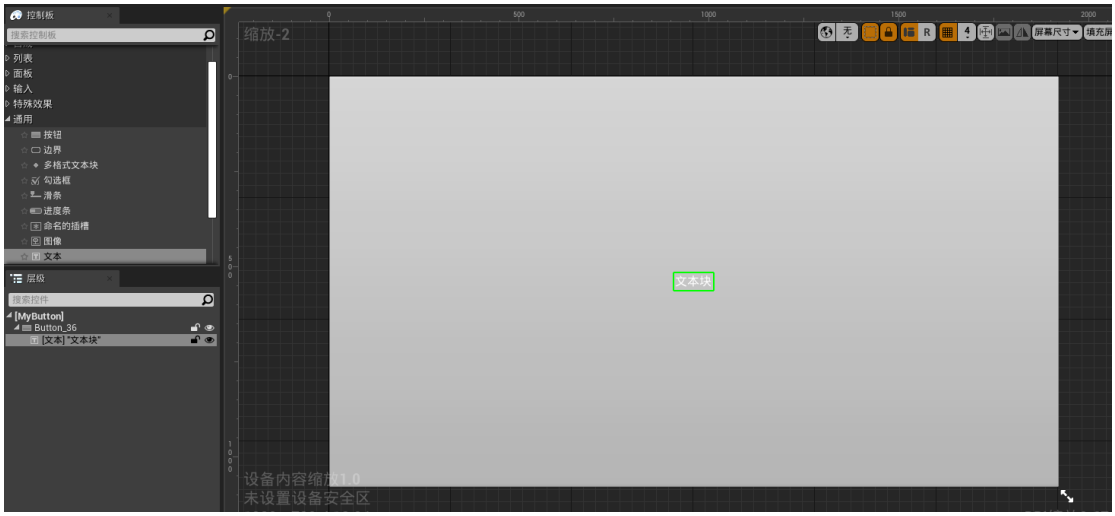
编译保存蓝图，运行游戏，可以看到 MyUI 控件蓝图被添加到了屏幕上，且鼠标光标可见，可以通过鼠标和 UI 控件进行交互（虽然此时还没定义按下 Button 后的事件）。

在编辑控件蓝图时，除了之前从“控制板”向视觉设计器添加系统自带的 UMG 控件外，还可以自定义用户控件，也可以用于视觉设计器中。

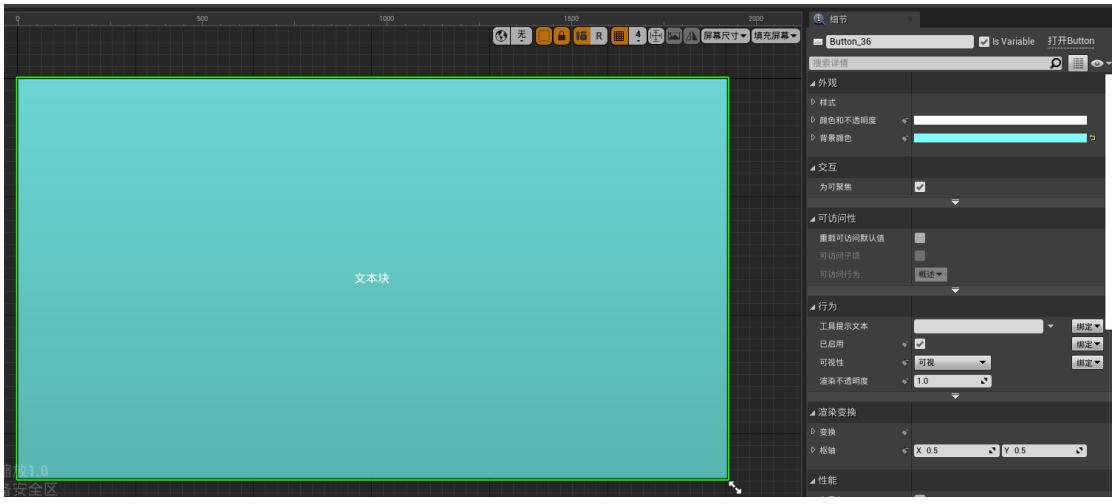
创建一个控件蓝图，命名为 MyButton，准备用于自定义一个用户控件。打开 MyButton 蓝图，删除“层级”面板中默认存在的“画布面板”控件。



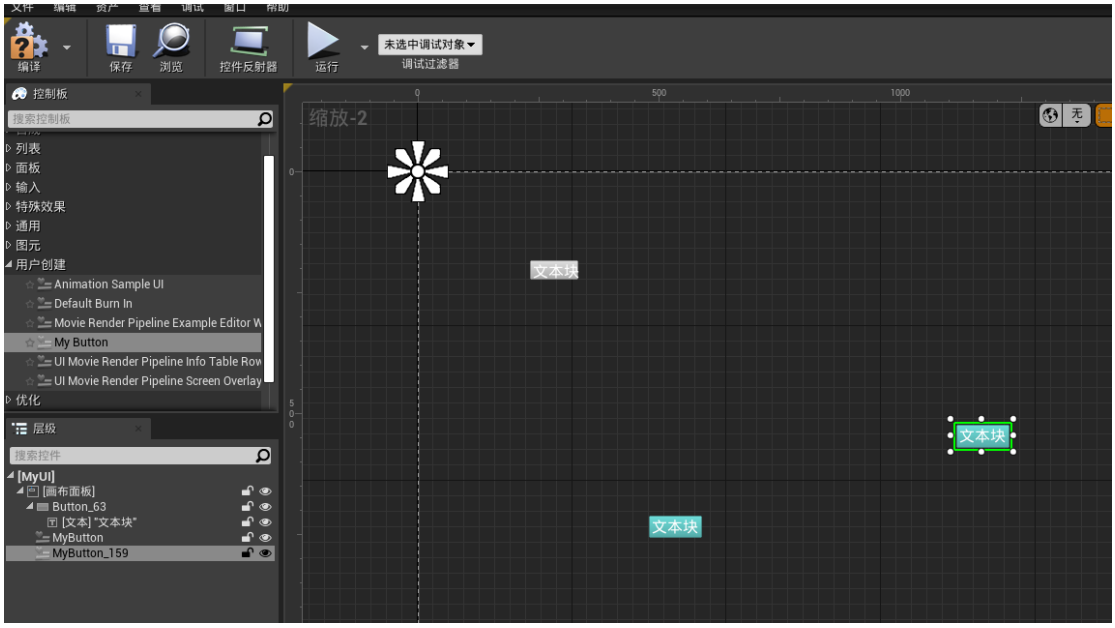
向视觉设计器添加“按钮（Button）”控件，再向“按钮”控件中添加“文本（Text）”控件。



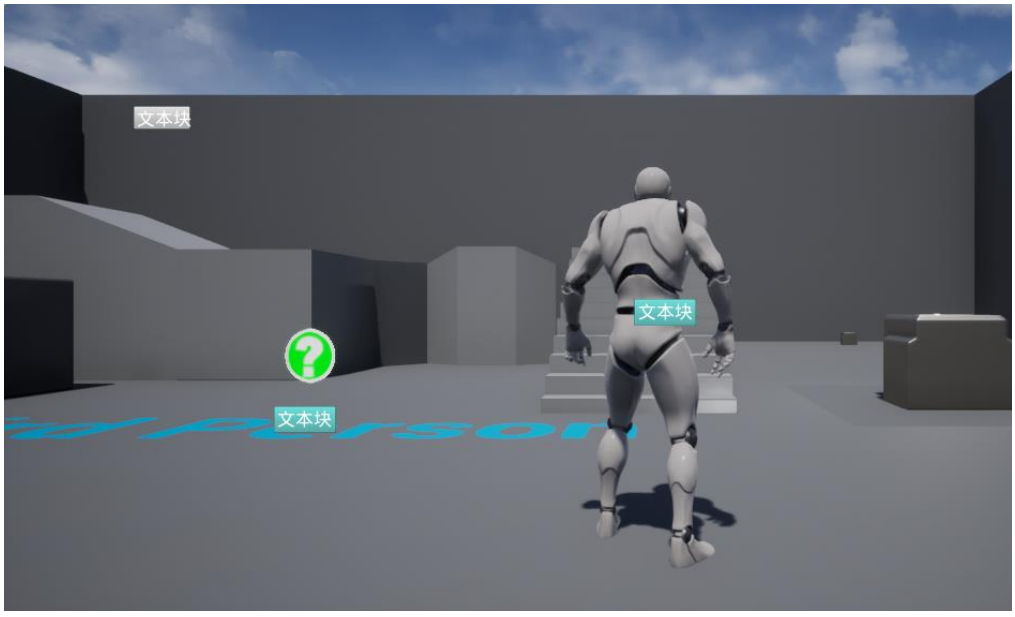
选中已添加的“按钮”控件，在细节面板的“外观（Appearance）>>背景颜色（Background Color）”中可以定义按钮背景色。



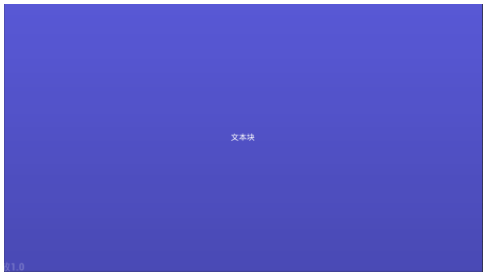
编译保存 MyButton 蓝图，然后回到更早之前定义的 MyUI 蓝图，在控制板中找到“用户创建>>My Button”，这就是刚才自定义的 MyButton 控件。



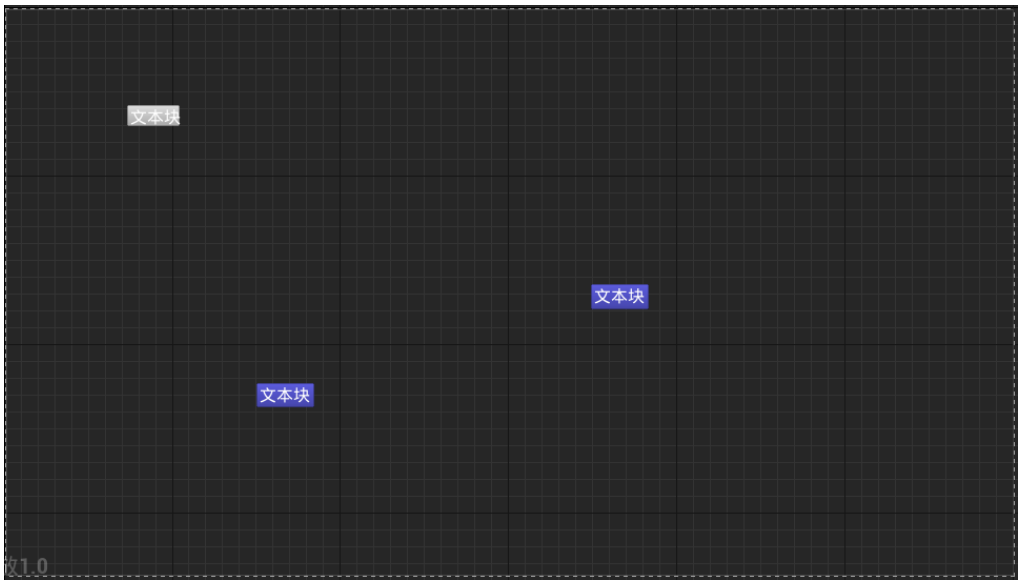
编译保存 MyUI 蓝图，运行游戏，可以看到 UI 效果。



当对自定义用户控件进行改变时，调用该控件的各个控件蓝图中，该控件都会发生同样的变化。例如在 MyButton 控件蓝图中把 Button 的背景色改为紫色，编译保存。

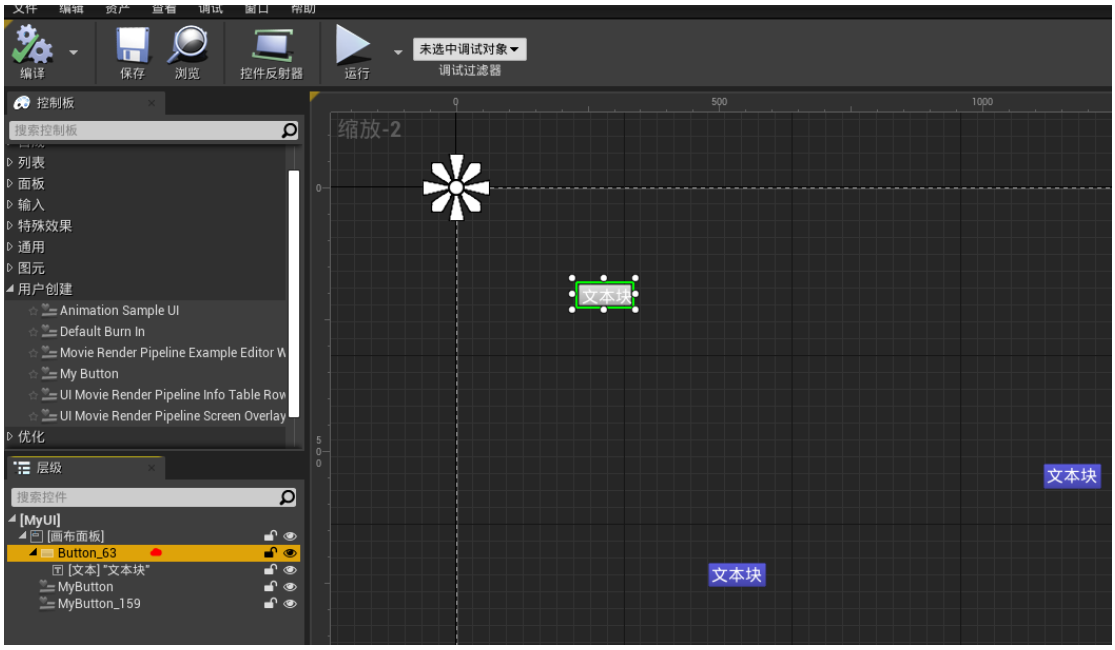


返回 MyUI 蓝图，可以发现蓝图中 My Button 控件的背景色都变成了紫色。



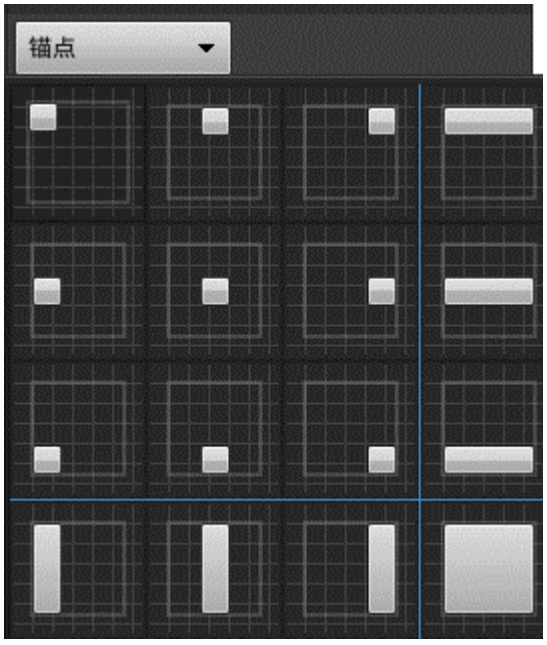
通过 UMG 创建 UI 界面时，排布各种元素的分布仅仅是第一步。对于按钮、状态条、文本框等各类元素，UMG 的细节面板提供了数个选项用于设置显示方式。不同类控件的样式选项各不相同。

在“层级”中选中最早创建的 Button 按钮，可以看到在画布面板控件的左上方出现了一个形似花朵的图标，称为锚点（Anchors），用于定义 UI 控件在画布面板上的基准点，用于在不同屏幕尺寸的情况下维持界面布局。



在 Button 按钮细节面板中可以看到锚点的详细信息。

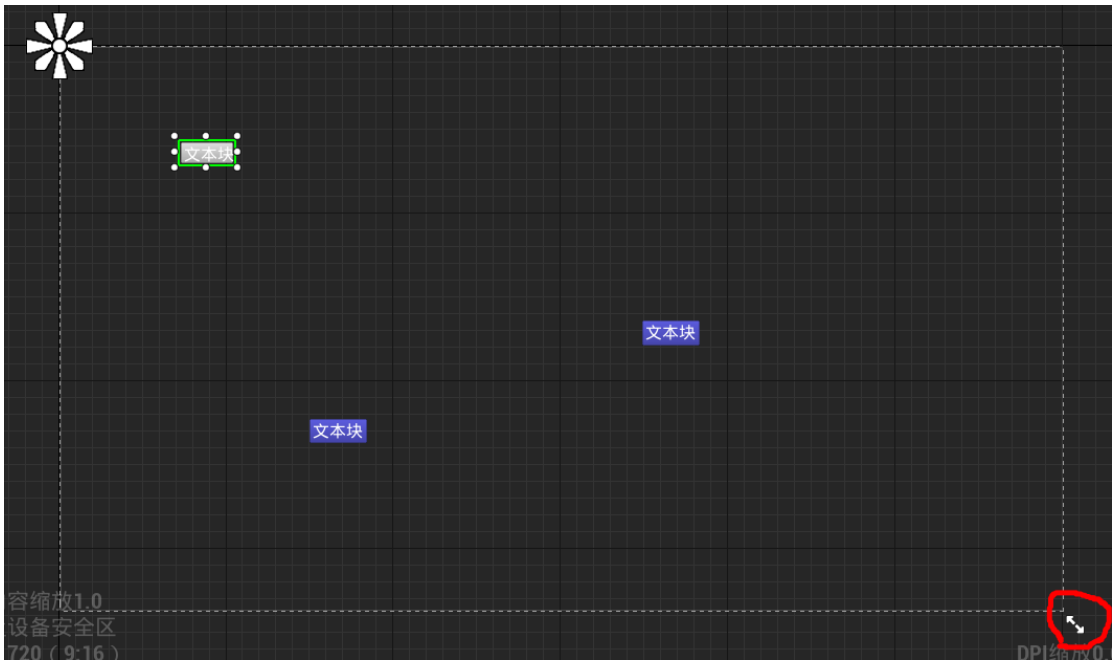
锚点中“最小”和“最大”表示了锚点的范围（相当于左上和右下的位置），X=0 表示最左，X=1 表示最右，Y=0 表示最上，Y=1 表示最下。点开“锚点”可以看到如下系统预设的几种常见的锚点范围。



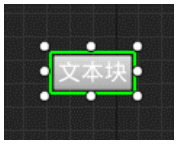
当选择锚点为右上角范围时，“最小”和“最大”的 X 值均为 1，Y 值均为 0；当选择锚点为整个画布面板时，“最小”的 X、Y 均为 0，“最大”的 X、Y 均为 1。

选定锚点后，细节面板中的“位置 X”和“位置 Y”，表示当前按钮控件的位置相对于锚点位置的偏移量，不论屏幕尺寸如何变化，该偏移量都保持不变。“尺寸 X”和“尺寸 Y”表示控件的尺寸。

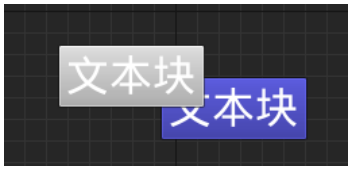
可以看到，当控件的锚点位置设为左上角时，如果屏幕尺寸发生变化，控件的位置不变；当控件的锚点位置设为右下角时，如果屏幕尺寸发生变化，控件的位置也会发生相应变化。



当 Button 按钮的“大小到内容”被勾选时，可以看到按钮的大小调整为与内容相符的大小。



下面看 ZOrder 的作用。左图中 Button_63 控件的 ZOrder 值为 1，My Button 控件的 ZOrder 值为 0。右图中 Button_63 控件的 ZOrder 值为 0，My Button 控件的 ZOrder 值为 1。若两个控件的 ZOrder 值都为 0，则后放置的控件置于上方。也就是说 ZOrder 大的控件在上方，ZOrder 相同时后放的控件在上方。

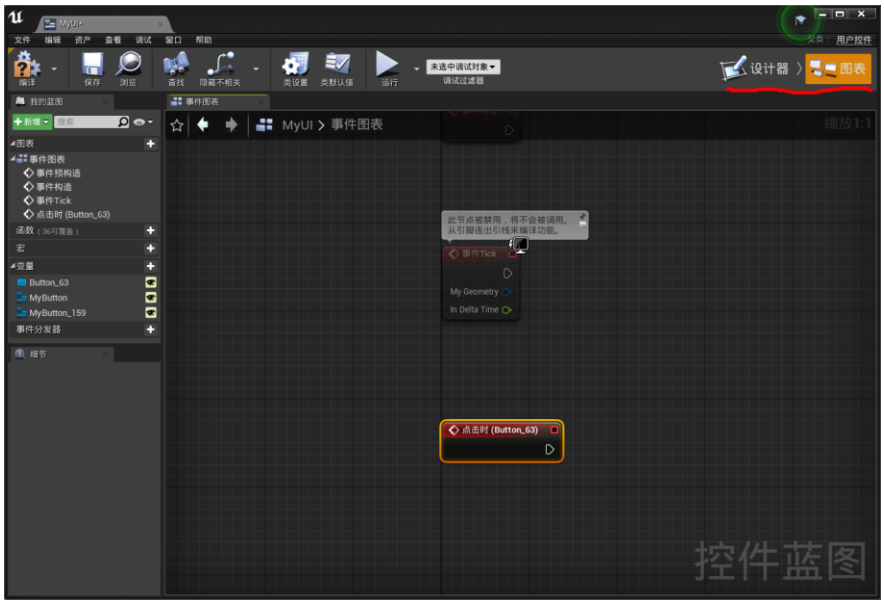


选中 Button_63 控件的“文本”子控件，观察细节面板。

可以看到“文本”控件和“Button_63”控件的插槽类目中的内容完全不同，一个是“按键槽”，一个是“画布面板槽”。

在“Button_63”控件的细节面板中，还有外观、交互、可访问性、行为、渲染变换、性能、裁剪、导航、本地化、事件类目，其中事件（Event）类目中可以为控件添加可绑定事件，例如当点击按钮时触发什么后续节点功能。

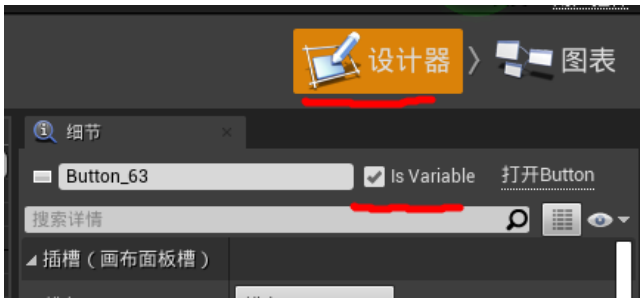
单击“Button_63”控件事件类目的“点击时（OnClicked）”项。控件蓝图编辑器从“设计器（Designer）”切换到“图表（Graph）”模式。



事件图表编辑器中会出现一个名为“点击时（Button_63）”节点，这是该控件被鼠标点击时的入口节点。

左侧“我的蓝图”面板中，当前已有的几个变量均为之前向视觉设计器中添加的控件，可以添加更多的变量。

需要说明，只有当控件被定义为变量时，才可以在“图表”模式下被当作变量来使用。切回到“设计器”模式，点击某个控件，可以看到 Is Variable 默认勾选，说明该控件可以被当作变量使用。



回到“设计器”模式，在“层级”中选择“文本”控件，细节面板中“填充”指围绕该控件的边框大小，“水平对齐”和“垂直对齐”指文本的对齐样式，“文本”用于设置文字内容，“外观”用于设置文字颜色、透明度、字体、阴影等文本样式。

需要注意的是，在“文本”属性后面有一个“绑定（Binding）”功能按钮，该功能可以在控件蓝图上创建一个新函数，它将返回此属性的绑定数据。也就是说把控件的属性（文本内容）绑定到蓝图中的功能或者变量值。

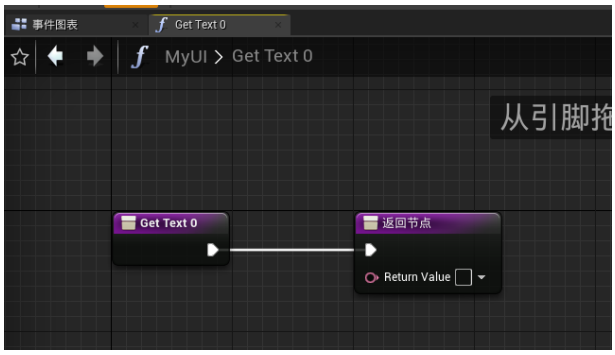
当绑定后，只要调用功能或者更改变量，就会在控件中反应出来。在文本控件中，即体现为修改文本内容。

绑定分为函数绑定和属性绑定。首先通过函数绑定实现 TextBlock_150（数字不一定相同）控件文本的动态变化。点击细节面板“文本”属性的“绑定”按钮，选择“创建绑定”。

此时切换到了“图表”模式，可以发现在“我的蓝图”中新增了与文本内容绑定的“GetText_0”函数。

在“GetText_0”函数的图表编辑器中，默认存在两个节点。一个节点的名字与函数名相同，作为函数的入口节点；另一个名为“返回节点（Return Node）”节点，

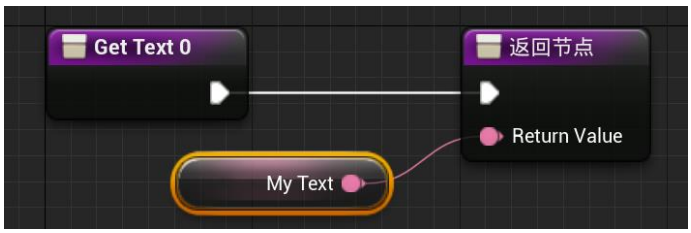
提供函数的返回值。



在 GetText_0 函数的细节面板中还可以添加额外的输入和输出参数。

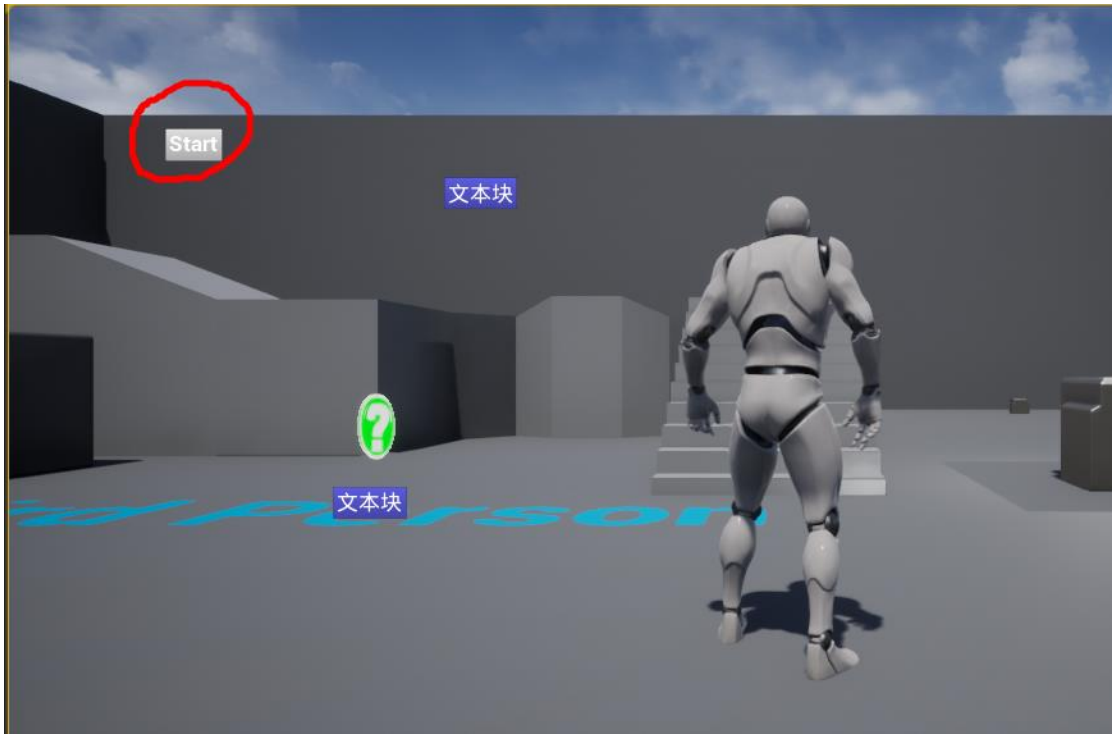
添加文本类型的变量，命名为 My Text，编译后修改默认值为 Start。

编辑蓝图，把 My Text 变量的值传递给 Return Value，编译保存。



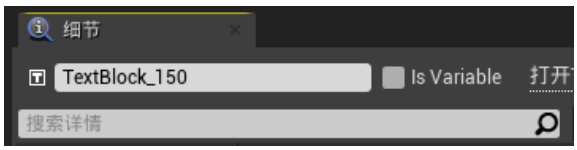
可以在“设计器”模式中看到，“文本”属性与 GetText_0 函数绑定在了一起。

运行游戏，My Text 变量值即为 TextBlock_150 控件的文本内容。

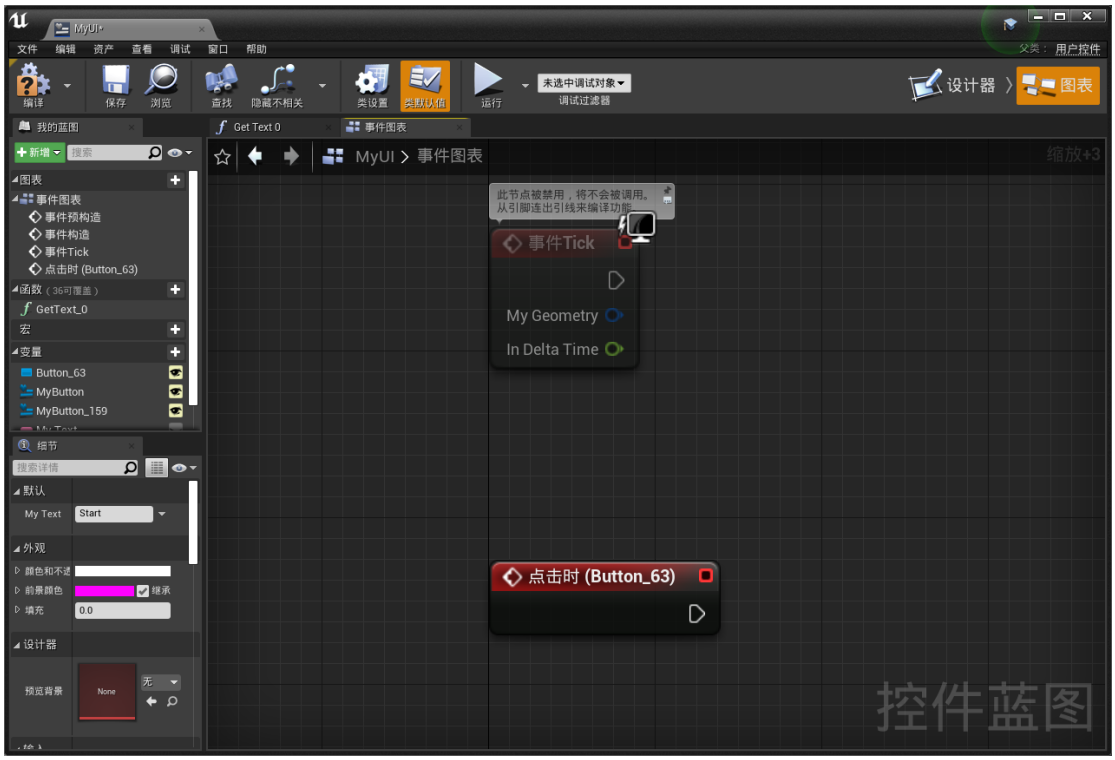


下面要实现的功能是单击 Button_63 时，利用在子控件 TextBlock_150 中定义过的 GetText_0 绑定函数，获取文本内容并打印出来。虽然也可以利用 TextBlock_150 子控件中的“获取文本（Get Text）”节点和“设置文本（Set Text）”节点来获取或者改变文本内容，但当 TextBlock_150 子控件的“Is Variable”没有勾选（事实上默认就是不勾选），无法被当作变量使用时，使用绑定变量就是唯一

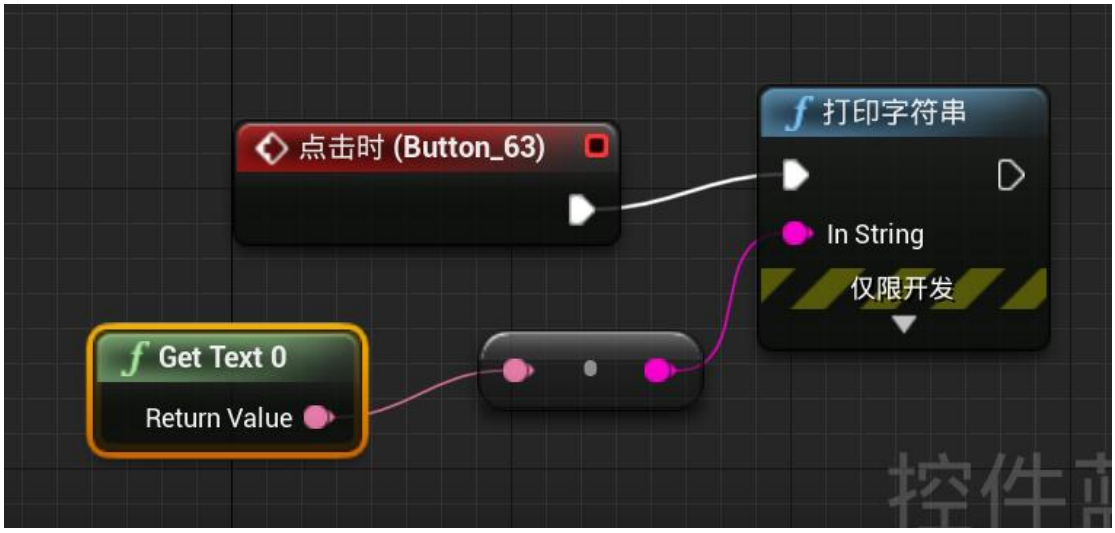
选择。



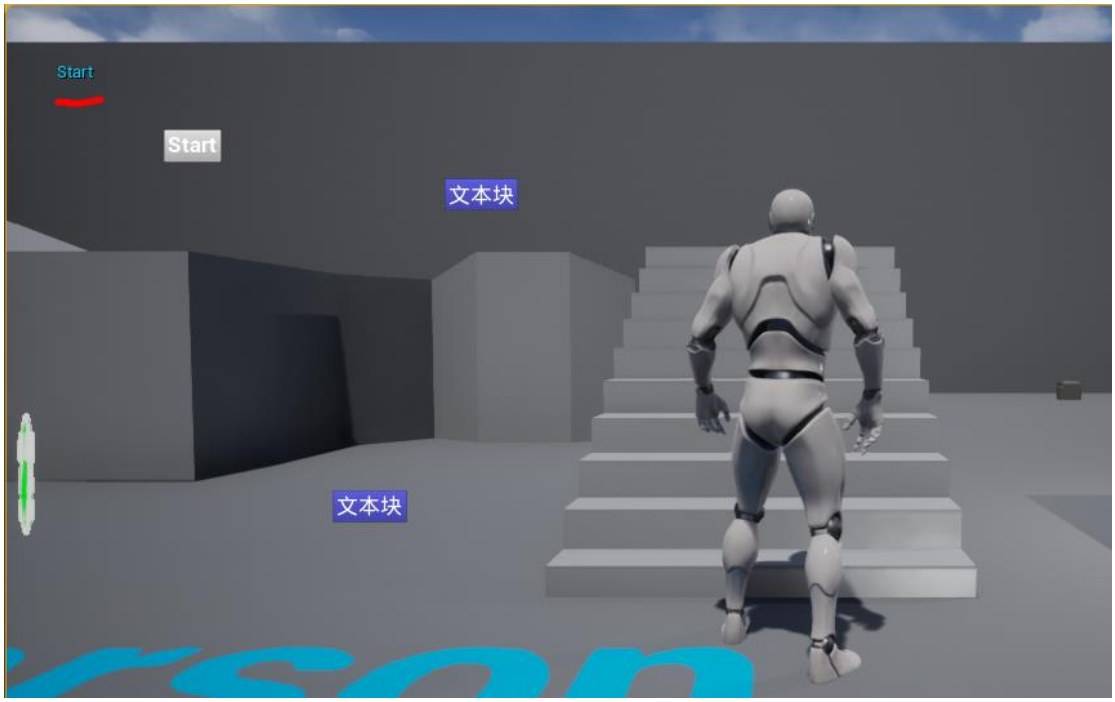
返回 MyUI 蓝图的“图表”模式，在事件图表页面中有之前生成的“点击时（Button_63）”鼠标响应事件节点。



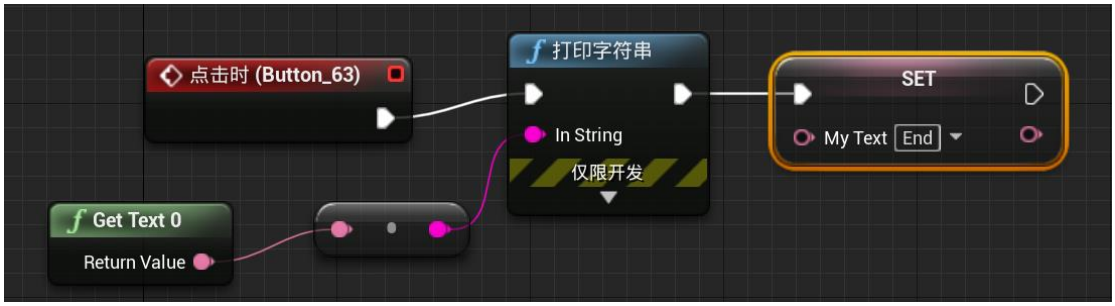
添加 GetText_0 节点和 Print String 节点。



编译保存，运行游戏。TextBlock_150 的文本内容为 MyText 变量值“Start”。单击 Button_63 按钮，文本内容被打印出来。

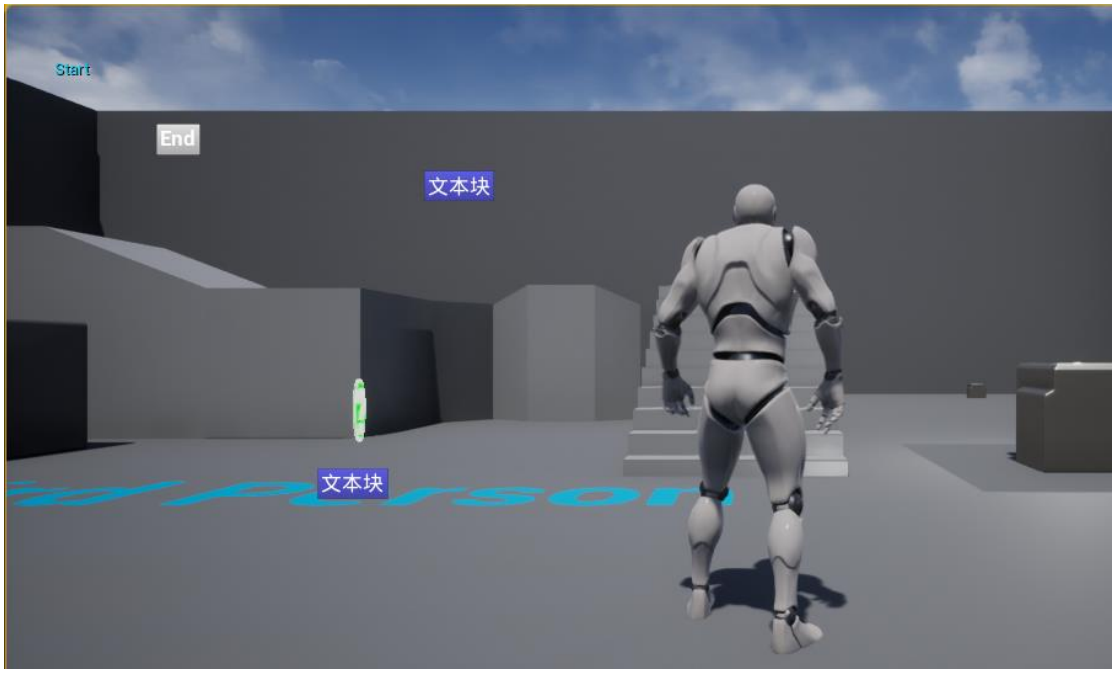


下面要实现的功能是动态改变文本内容。在事件图表中，添加一个“Set My Text”节点，这里的 My Text 是之前在文本控件绑定函数 GetText_0 中添加的变量名。修改“My Text”引脚的值为“End”。



通过点击 Button_63 控件，现在屏幕上输出文本内容“Start”，然后把 My Text 变量的值改为“End”。GetText_0 函数会实时将变量值返回给文本控件，从而改变 TextBlock_150 控件的文本内容。

编译保存，运行游戏，可以看到单击 Button_63 按钮后，TextBlock_150 控件的内容从“Start”变为“End”。


Start
End
文本块
文本块