## 容器

用于装填控件或是容器的UI元素，一般容器会具备布局特性，常用布局，自由布局，参考线布局，格状布局，横向布局，纵向布局等。

## 控件

用于进行用户交互操作中，收集用户输入，反馈信息等用途，具有接收传递信息的作用。例如按钮，输入框，菜单，复选框，滑块等。

## 分类

Common 一般控件，提供基础控件，功能单一，是所有复杂控件奠基石

1. Border 一般用来做布局容器的圈定，内部只能填入单一控件，布局方式为非自由式，可按照设定进行横纵向填充或是横纵向靠左，靠右，居中布局
2. Button用来接收用户点击事件，是交互的基础元素。UE中的Button只具备按钮基本特性，不具备描述，如需要加入描述可与控件Text配合使用
3. CheckBox，选择组件，用于构建单选操作，常用来询问用户是否同意某一操作
4. Named Slot 当将UserWidget作为控件时，NameSlot将被保留，用于在外部进行控件添加对齐使用。像是一个预留插槽，专用于在外部插入新的控件到内部布局中。
5. Iamge 用来展示一张贴图或是材质，例如Icon，头像
6. ProgressBar 进度条
7. Slider 滑块，用于在游戏中调整一个区间数据，例如音量大小，游戏效果，视距等
8. Text用来展示文本信息，注意这是非编辑文本信息，运行中无法让用户进行输入操作。
9. TextBox 与Text相反，允许用户进行输入编辑

Extra 特殊的一个控件，内放一个可以提供弹出下拉框的控件

用于进行构建一个具有Head（标题），Body（内容的组件，一般如下拉展示框）

Input 信息输入组件

1. ComboBox 下拉选择框 一般用于构建例如选择性别，地区等。即给予用户参考数据进行选择
2. Editable Text 给予一个无样式，可以进行用户编辑信息的窗口，供给用户进行信息输入。用于构建简单的信息输入收集。（注意，单行输入）
3. Editable Text（Mul）给予一个无样式，可以进行用户编辑信息的窗口，供给用户进行信息输入。用于构建例如意见反馈，联系GM，发送邮件信息等，大段文本信息使用。（注意，可以换行）
4. Spin Box 滚动信息选择框，用于给予用户给定信息，进行选择
5. Text Box（Mul） 多行信息用户输入，用于收集输入文本，可换行。

Optimization

1. Retainer Box 将此控件下的所有内容当作一张贴图传输给材质中的贴图参数，一般我们可以使用此控件构建一不规则UI，例如记住一张半透明贴图构建一界面贴图，最常见的，圆形小地图，吃鸡中的指南针。
2. InvalidationBox失效框，用于进行子控件渲染封装进行缓存，以加快UI的渲染速度。UI是逐帧绘制，使用缓存可以降低刷新绘制成本。一般我们会将不变化的控件封装在此容器中。用于UI的优化处理

CRT**缓存相对转换** 缓存子绘制元素相对于无效框的位置，会在逐帧绘制它们时增加额外开销。但是，在无效框的位置会逐帧发生变化的情况下，这会节省很多资源。

I V**易变** 如果为真，则会防止控件或其子控件的图形以及布局信息进行缓存。如果控件会逐帧发生改变，但您仍希望该控件处于无效面板中，则应该将其设置为易变，而不是逐帧对其进行无效化处理（后者会导致无效面板无法缓存任何内容）。

Panel 容器

1. CanvasPanel 自由布局容器
2. Grid Panel 格子布局（不会强制修正子控件大小）
3. H B 横向布局容器
4. Overlay 容器内部控件会堆叠，布局方式参照对堆叠容器插槽
5. Safe Zone 安全区域
6. Scale Box 容器可以保证在尺寸进行比例调整时，不影响内部控件大小（不刻意拉伸控件内容），一般我们可以使用此方法进行背景图展示，背景图是不希望被拉伸的
7. SizeBox 按照给定的尺寸圈定一个容器（只能填充一个子控件）
8. Uniform Grid Panel 标准格子容器，填充的内容大小必须相同，一般我们可以用来制作背包
9. Vertical Box 垂直容器
10. Widget Switecher 标签页容器
11. Wrap Box 标准容器，不会修正子控件大小，子控件按照填入内容横向排列，当横向满则换行继续填充。并且不会向缝隙填入控件，他的布局总是规整的。

Uncategorized

Window Title Bar Area 用于用户在桌面上可以拖动游戏窗口，一般用于如果我们编译后的游戏隐藏了窗口状态栏，用户无法将窗口进行拖动，使用此组件用户可以继续拖动窗口

## 锚

用来定义控件在画布面板上的预期位置，并在不同的屏幕尺寸上维持这一位置。

控件的锚点受制于控件的父级容器特点，并不是所有控件都会具备锚点特性

##### 锚点预设方式

自由容器插槽方式

9方向对齐

横向三拉伸

纵向三拉伸

双方向拉伸

堆叠容器插槽方式

左对齐，右对齐，中对齐，横向填充

上对其，下对齐，中对齐，纵向填充

## 层级关系

UMG编辑器中遵循层级关系确定子父级关系

## DPI

每英寸的像素数量

UMG支持与分辨率无关的UI设计，即UI可以不依照分辨率的变更而变得无法使用。

###### DPI的缩放规则

最短边 - 该选项将基于窗口的最短边来评估缩放曲线（最常用的设置）。

最长边 - 基于窗口的最长边来评估缩放曲线。

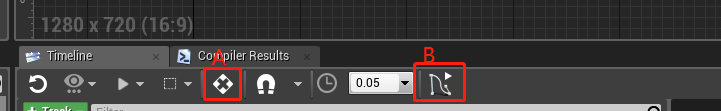
水平 - 基于窗口的 X 轴来评估缩放曲线。

垂直 - 基于窗口的 Y 轴来评估缩放曲线

## 数据驱动方式

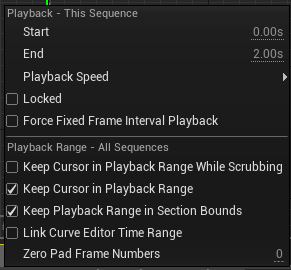
1. 函数驱动
2. 属性绑定 绑定成员对象的一个属性
3. 事件驱动（最优方式）

###### 动画

1. 先要添加动画资源名称
2. 选取当前动画中针对变化的控件
3. 动画帧编辑，在控件属性前面有则表明可以添加动画帧记录
4. 

A为自动录制，需要先手动将参数进行帧添加，并且只能在参数帧区间进行自动记录

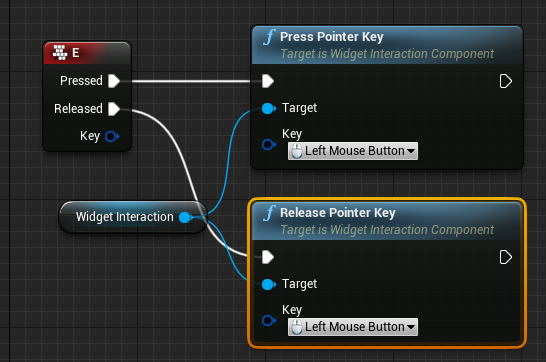
B为动画曲线，只有属性被记录了，才可以进行曲线调整



Locked 后无法在时间轴调整动画时间

Start/End可以直接调整动画时间

###### 3D游戏UI的交互方式



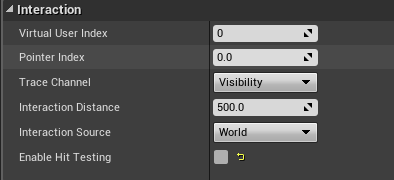
借助控件交互组件，将其他输入事件转换为UMG输入事件，由于Widget组件只能接收WidgetInsteraction输入的事件，所以，所有的输入均必须使用WI进行输入模拟，包括键盘事件

WidgetInsteraction组件

按键点击模拟 Press Pointer Key/ Release Pointer Key

按键输入模拟 Press Key/ Release Key 为输入框进行输入模拟 如同Send Key Char这不是当作敲击来使用，而是当作字符输入来使用

输入字符 Send Key Char



VUI，虚拟用户索引，用来绑定当前组件输入内容传递给哪个虚拟用户，这样做的好处可以将多个WidgetInteraction组件输入内容绑定到一个虚拟用户中

制作模版控件

添加控件

创建控件

移除控件

注意：

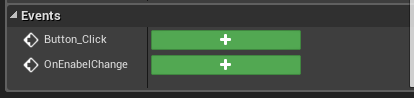
模版控件在被添加到其他控件中时，模版控件中的Dispatcher会被当作事件出口

例如A空间中加入了Dispatcher，在B控件中加入了A，则A的细节面板下端可以看到Dispatcher

A中的Dis

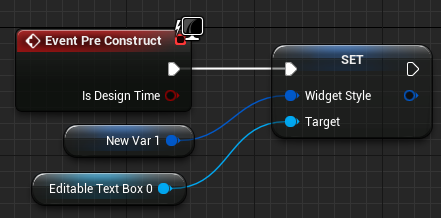


A被添加到B中



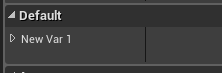
如果想要在自定义控件中将某个组件的样式暴露到外部进行调整，可以使用下面的方法

A中创建一个样式变量，并开启外部编辑

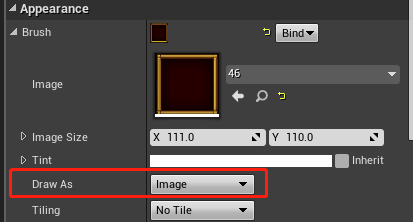




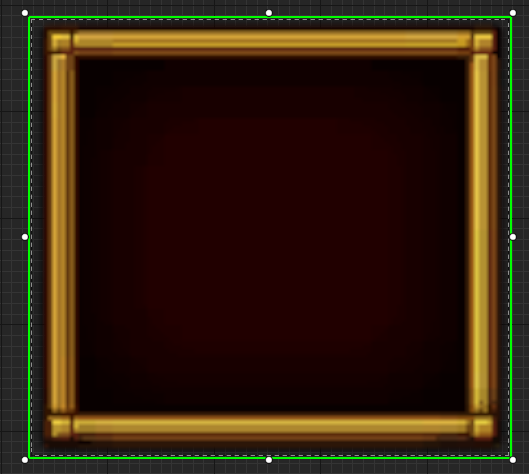
在B控件中添加A后，在细节面板中看到



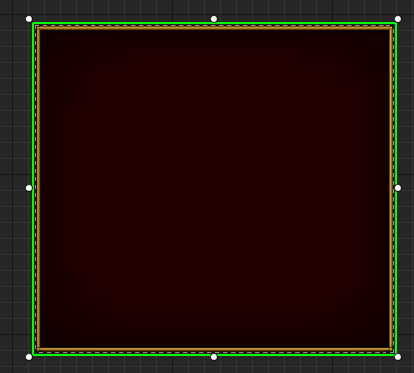
样式中背景图铺设方法（9切法）



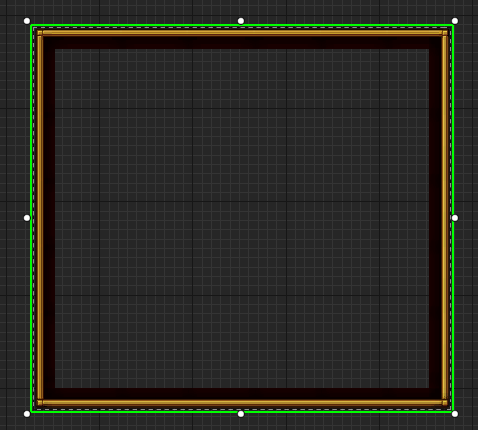
Image效果



Box效果 Margin是百分比



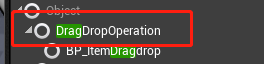
Border效果



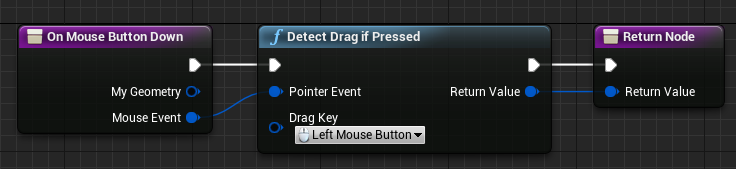
UI拖拽方式实现

步骤：

1. 创建蓝图类DragdropOperation，用于在拖拽过程中进行拖拽交互，拖拽交互数据均来自此类

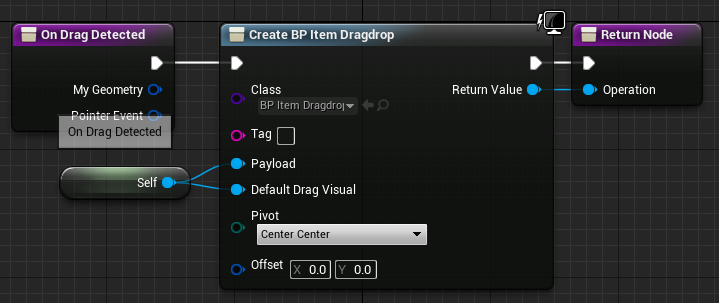


1. 在被拖拽的控件中重写OnMouseButtonDown



目的：当控件收到点击事件时，进行检测拖拽，如果使用的是鼠标左键，则响应拖拽事件

1. 在被拖拽的控件中重写OnDrag Detected

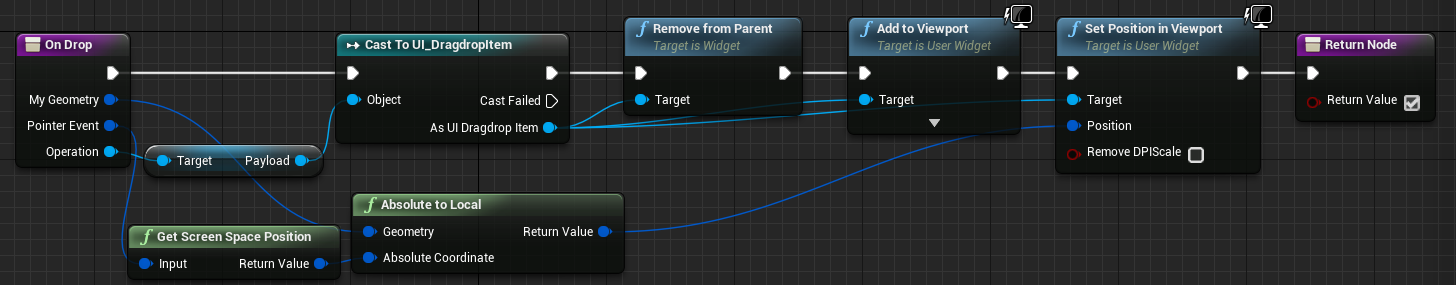


Payload 用于在响应抬起时，进行数据交互，Visual是拖拽时您想要显示的虚拟体

1. 在希望响应拖拽结束的控件中，找到上层容器，开启容器的接受鼠标拖拽结束事件



1. 在接受拖拽完毕的控件中，重写如下的函数

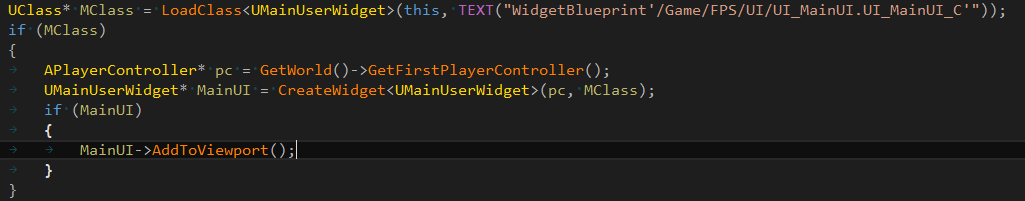


注意：由于DPI影响，GetScreenSpacePostion是有缩放的（基准比率是1920\*1080，是1：1），所以屏幕小的时候，获取的位置会被放大，使用ATL转换到当前控件坐标，由于转换过程中没有忽略DPI，所以在设置的时候需要去掉Remove DPIScale选项。

## C++中操作UMG

创建类，继承自UUserWidget，在构建完成类后，在编辑器中构建蓝图，继承自自定义控件类，然后和编辑普通UI一样编辑

在C++中创建UI并添加到视图窗口

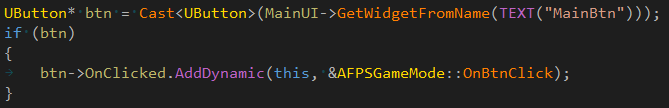


CreateWidget函数是全局函数（调用的是UserWidget的静态函数进行编辑），一共有三个重写函数

监听按钮事件

通过Userwidget成员函数GetWidgetFromName获取到UW中的控件，返回是UUserwidget控件，进行转换即可

Button绑定的函数需要使用UFUNCTION标记，属于动态绑定



动态绑定的另一种写法

FScriptDelegate Del;

Del.BindUFunction(**this**, "OnBtnChangeImgClick");

btn**->**OnClicked.Add(Del);

在C++中获取到动画

两种办法，第一种较为简单，直接通过在C++中暴露成员属性（类型UWidgetAnimation）方法，在蓝图中进行设置即可



第二种较为复杂，需要遍历整个对象类的成员属性（动画添加本身就是向自己加入了一个成员属性）

