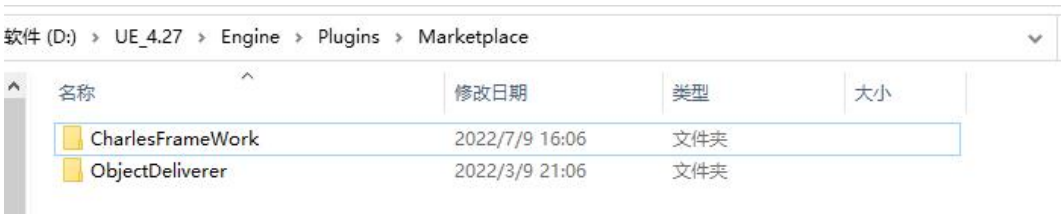
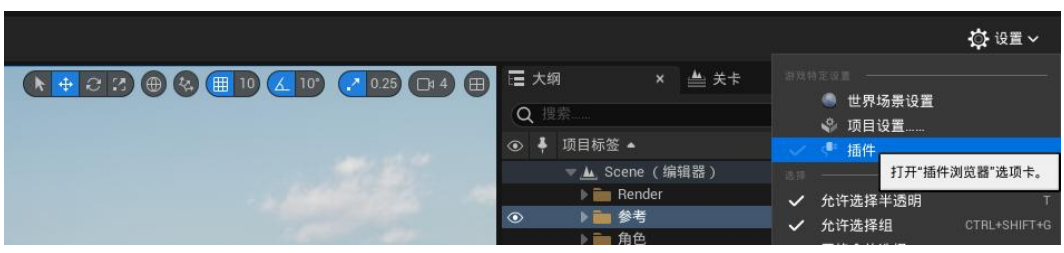
TechnologyArt-License

● 安装

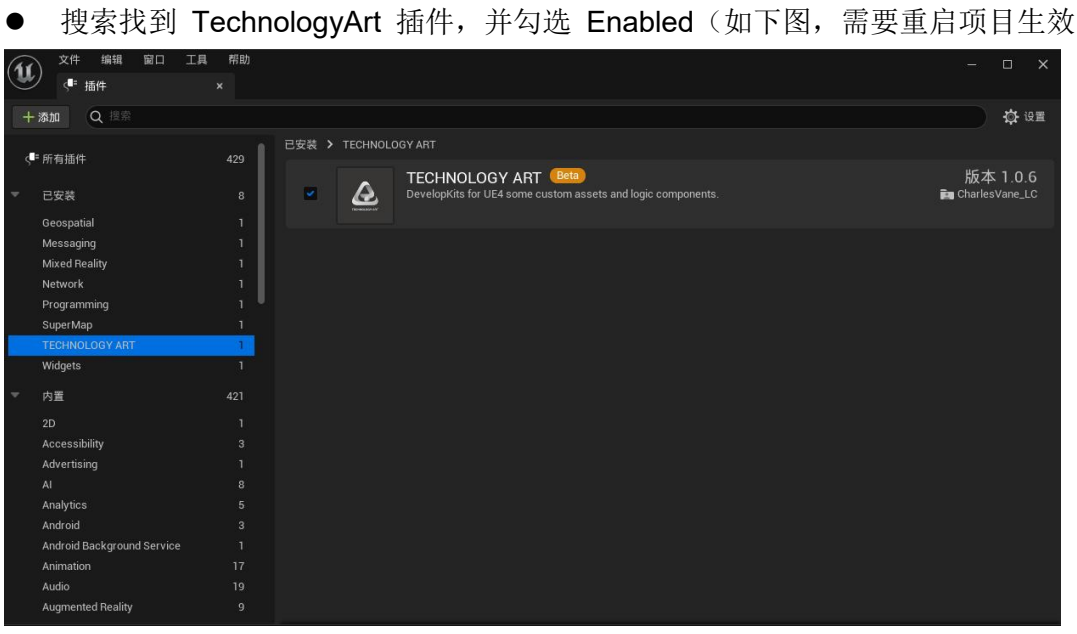
- 将插件文件放置到引擎根目录下的 Plugins/Marketplace 文件夹中（如下图所示）



- 在引擎界面右上角“设置”中找到“插件”项



- 搜索找到 TechnologyArt 插件，并勾选 Enabled（如下图，需要重启项目生效）



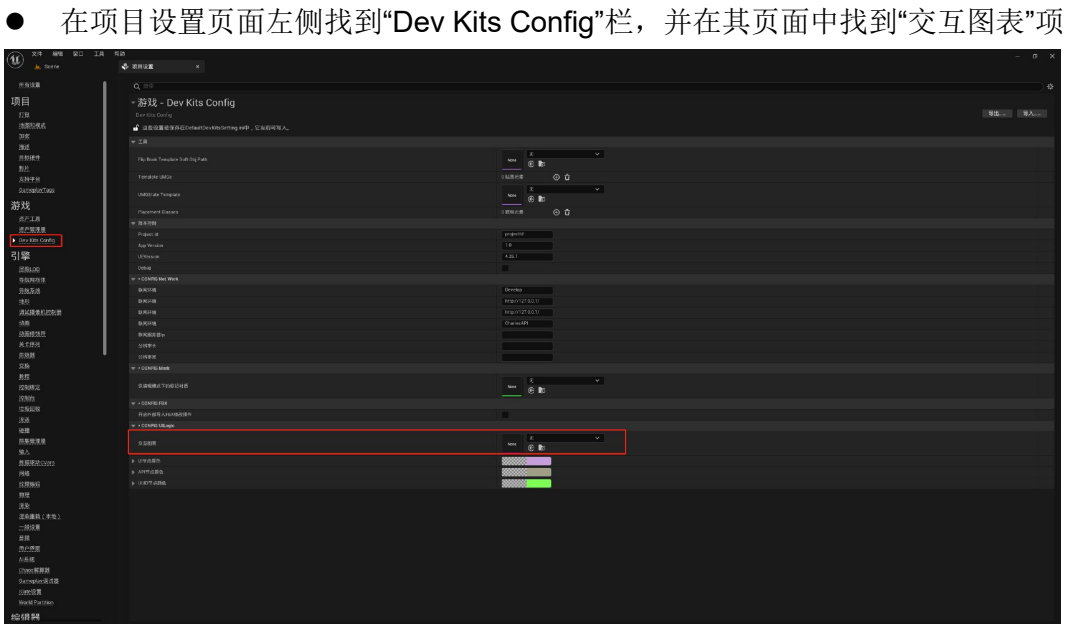
- 重启后在上方工具栏找到插件图标，点击进入插件账号验证界面

- 使用授权的账号密码进行登录成功后即可根据账号权限使用资产

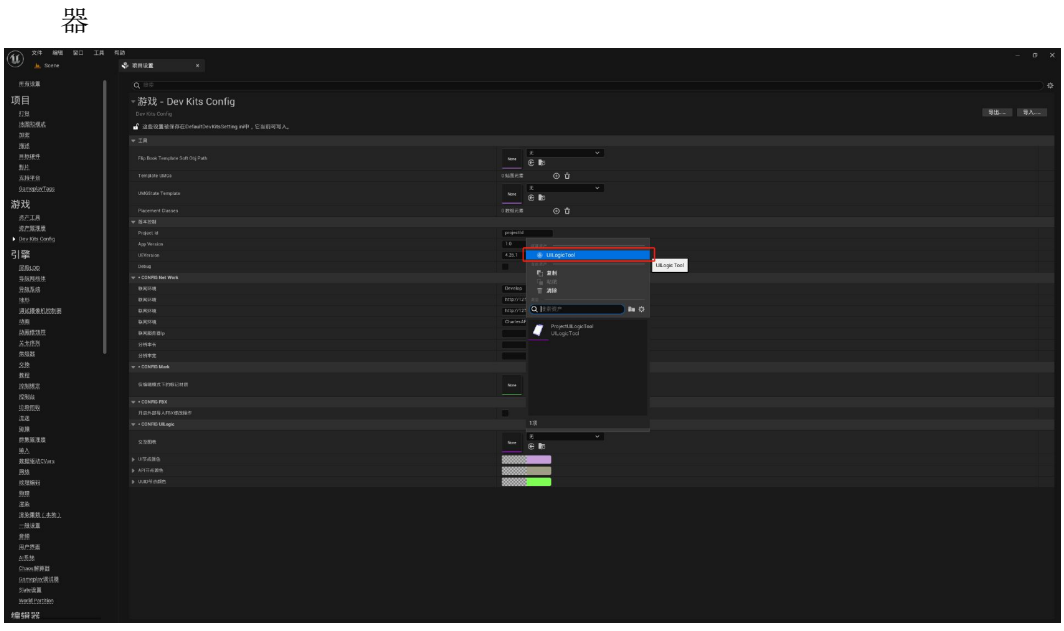
● 为项目配置 UI 管理器（必要）

- 在引擎界面上方菜单“设置”中找到“项目设置”项

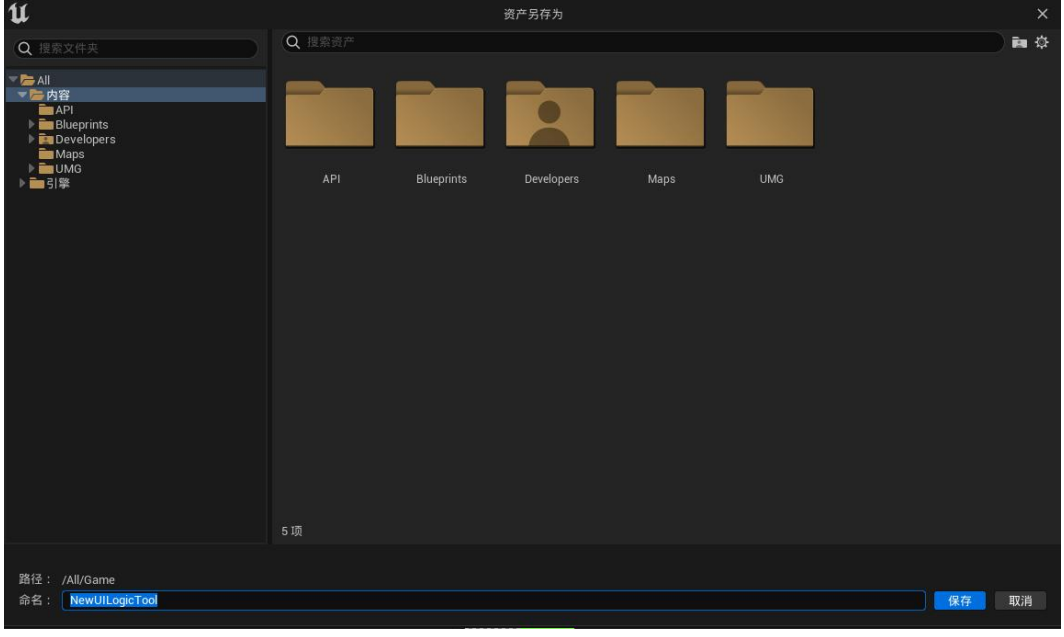
- 在项目设置页面左侧找到“Dev Kits Config”栏，并在其页面中找到“交互图表”项



- 点击交互图标右侧下拉箭头，在弹出菜单中点击“UILogicTool”新建空白 UI 管理器

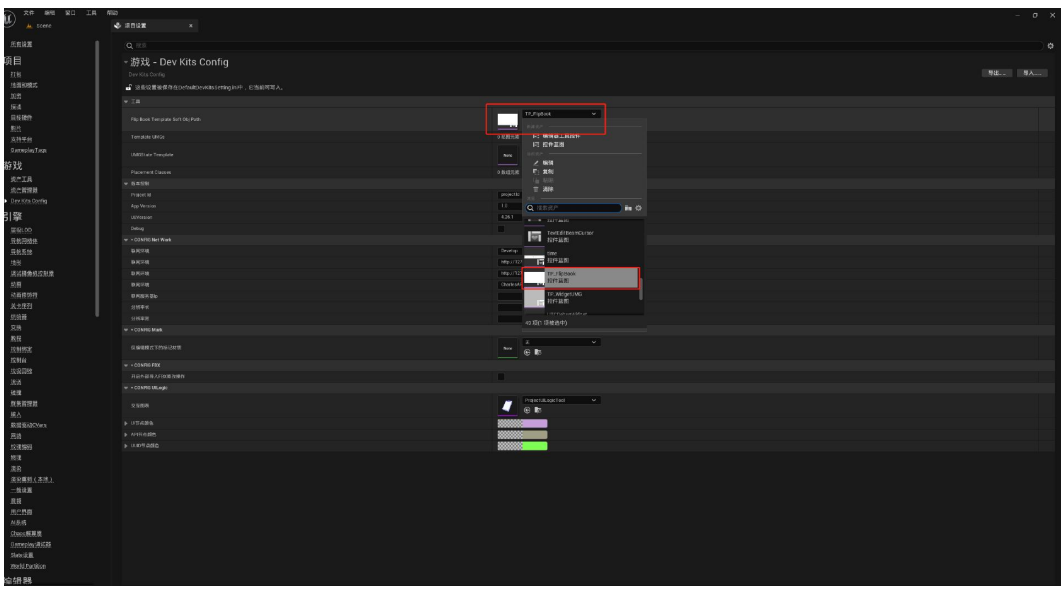


- 在弹出的资产保存窗口中，自定义一个保存路径及名称



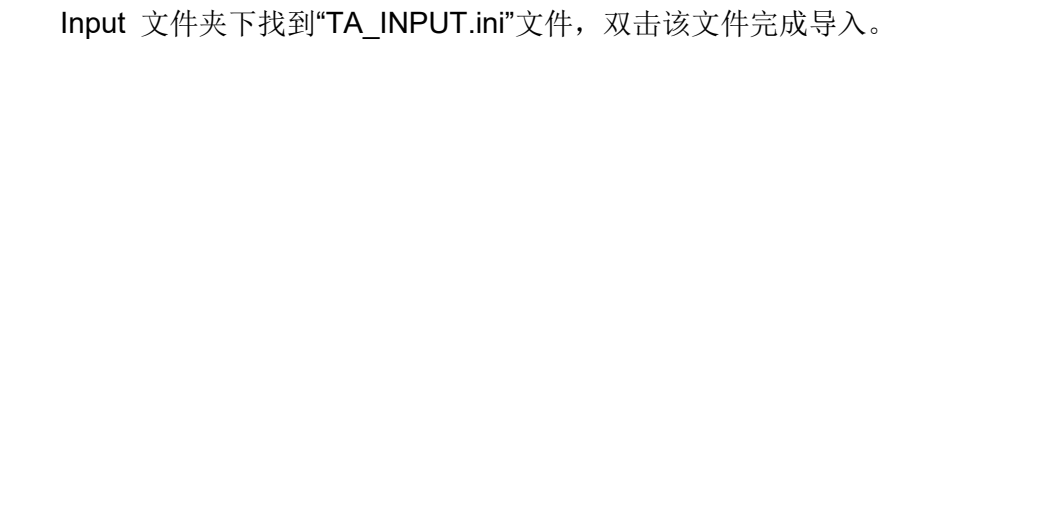
- 完成新建 UI 管理器及配置

- 配置序列模板



● 为项目配置 GameMode（必要）

- 在项目设置页面左侧找到“地图和模式”栏，点击后在右侧找到默认游戏模式项

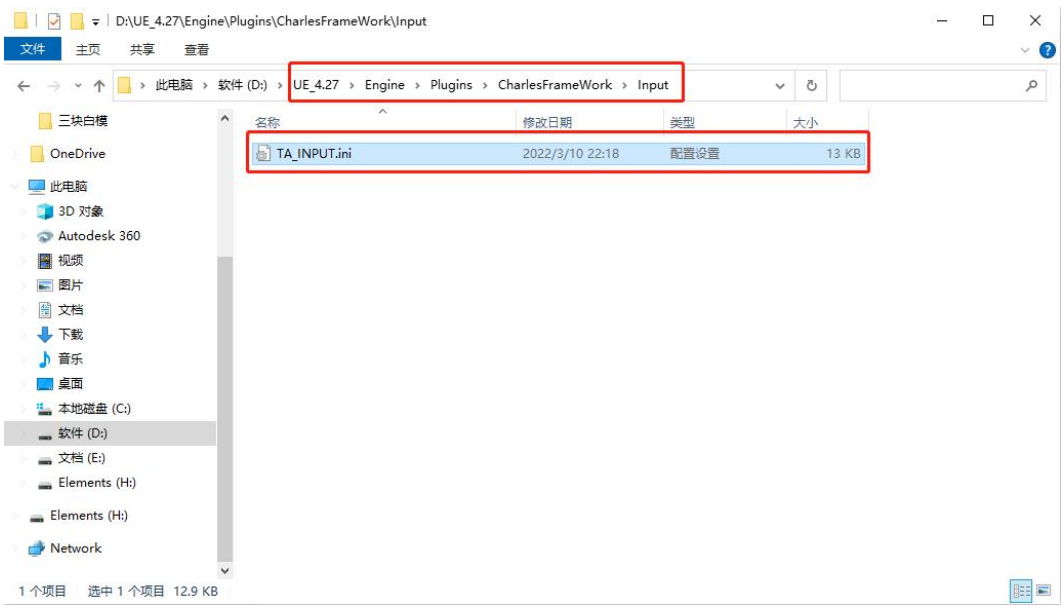


- 点击下拉箭头并搜索"Dev"找到“BP_DevKitMode"，点击设置为默认游戏模式

● 为项目配置输入映射（必要）

- 在项目设置页面左侧找到“输入”，点击该项，并在该项页面右上角找到导入按钮。

- 点击导入按钮，在弹出窗口中，找到插件安装路径，并在 CharlesFrameWork-Input 文件夹下找到“TA_INPUT.ini”文件，双击该文件完成导入。



TechnologyArt-插件预设类

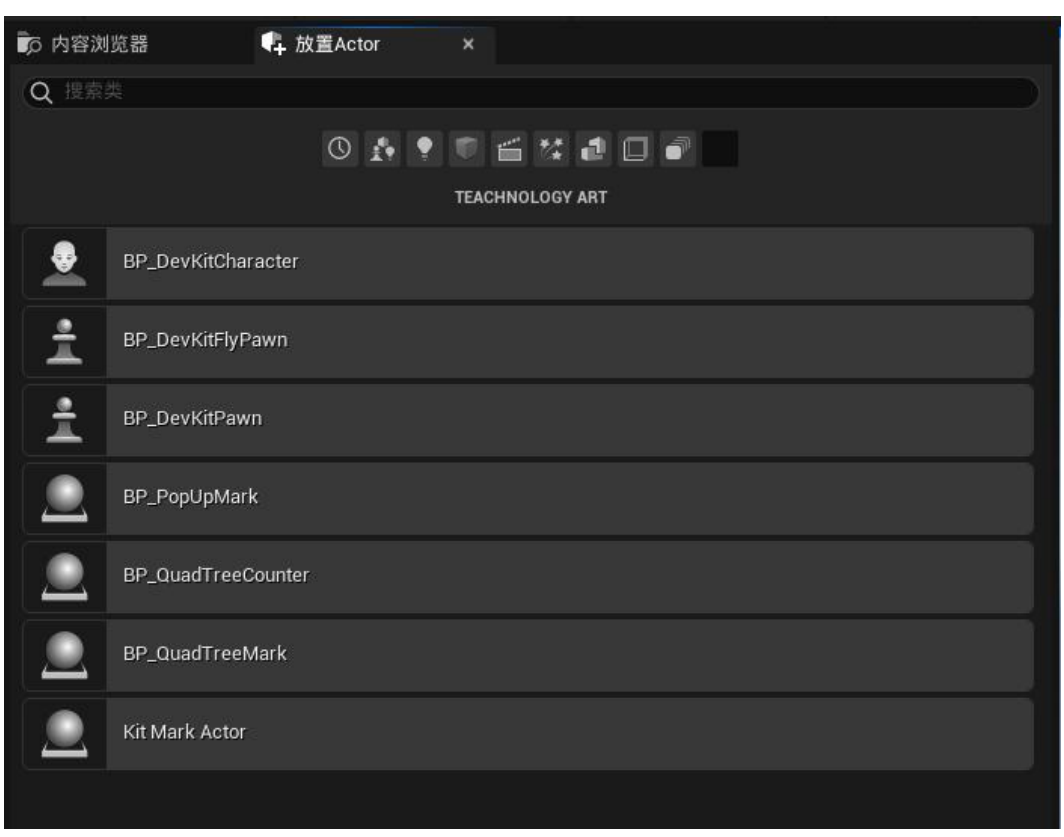
● 基本概述

插件提供预先设计好的功能类，帮助用户快速实现常用基础功能，可在多个项目中快速复用。预设类与 API 高度联动，通过 API 可完全控制预设类。当用户有自定义需求时，仍可继承或覆写预设类。

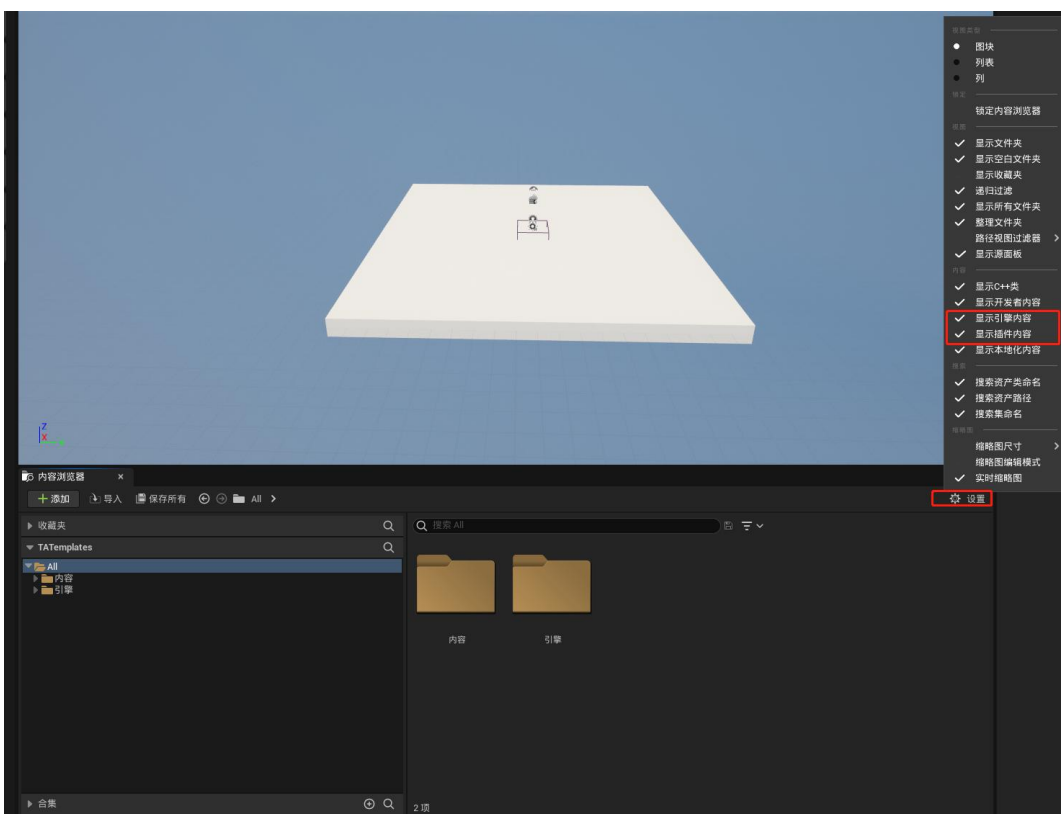
注意：如需对预设类进行功能拓展，建议将插件中预设类的原始文件复制，并将复制文件移动到个人文件夹中，再对复制的类进行功能拓展。

● 预设角色类

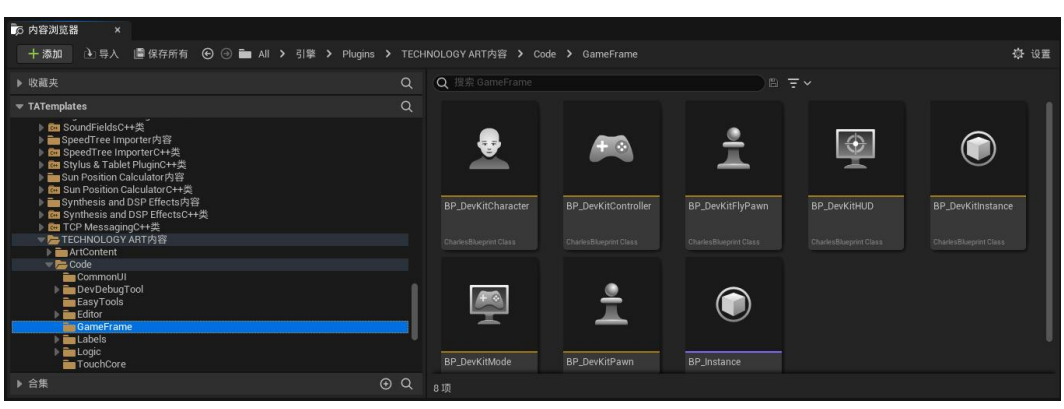
- 在左侧“放置 Actor”面板中可以快速找到插件预设的 actor 类，拖拽到场景中即可使用



- 如需对预设类进行自定义修改，可在内容浏览器右下角视图选项中勾选“显示插件内容”。



- 在左侧文件路径中找到“GameFrame”文件夹，文件夹中可找到三个预设角色类，可复制到个人文件夹中进行自定义修改。

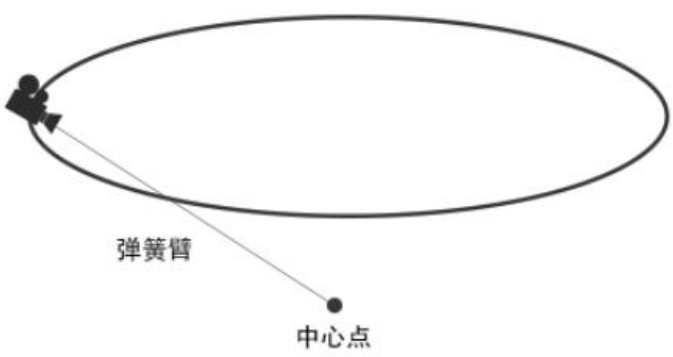


● 鸟瞰角色

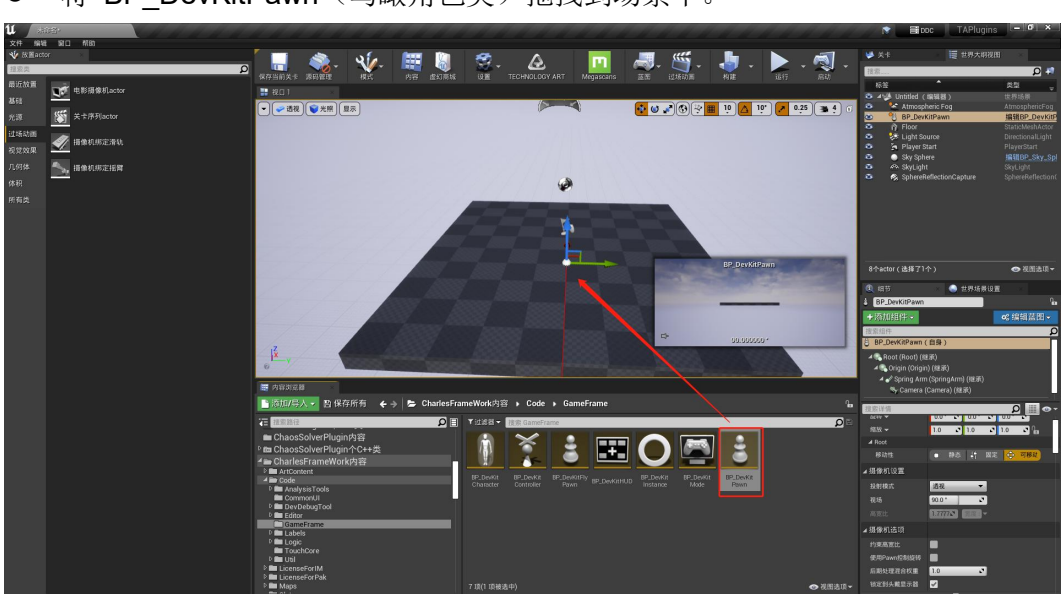
输入按键	执行操作
鼠标左键	控制相机围绕中心点的角度操作
鼠标右键	控制相机中心点平移
鼠标滚轮上滚/下滚	控制弹簧臂伸缩
键盘 WASD	控制相机中心点平移
移动端触摸点击	控制相机围绕中心点的角度操作
移动端两指缩放操作	控制弹簧臂伸缩
鼠标中键	控制中心点垂直移动

- 以上按键所对应的执行操作可通过 pawn 的参数面板进行配置

- 鸟瞰角色是一种围绕中心点旋转的相机，相机与中心点的距离由弹簧臂（Spring Arm）控制，中心点可进行位移交互。

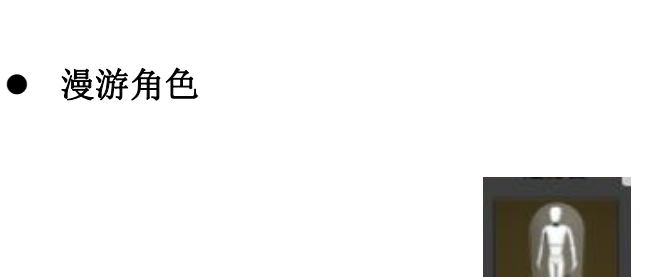


- 将 BP_DevKitPawn（鸟瞰角色类）拖拽到场景中。



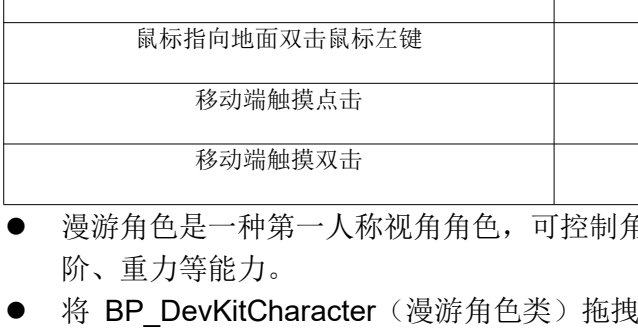
- 在鸟瞰角色右侧细节面板中，可设置继承自 Camera 的基本参数，如视场（FOV）、后期等。

- 下拉细节面板，在 MoveCurve 插槽中，可自定义或修改相机执行移动 API 时的运动曲线。



- 在 Pawn Property Controller Switch 中可选择是否锁定旋转、移动、缩放操作。

- 在 Pawn Property LMB 中可设置相机偏航角（ZRotator）、俯仰角（YRotator）的限制范围，还可以设置旋转的速度。



- 在 Pawn Property RMB 中可设置鼠标控制角色平移的范围、平移速度、中心点距离与平移速度的变化范围。

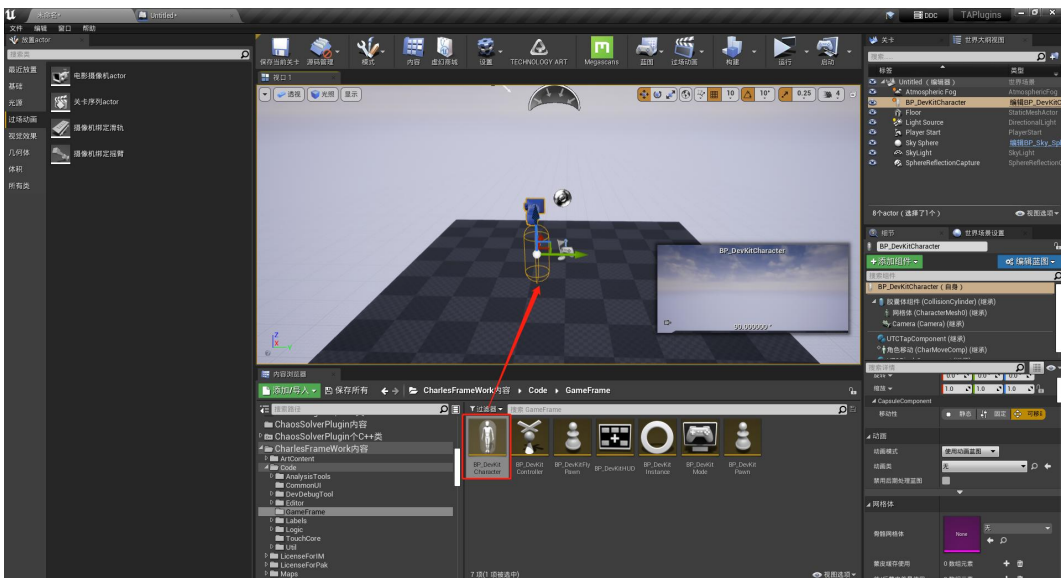
- 在 Pawn Property Initialize 中可设置角色初始弹簧臂长度、相机初始角度、角色 Key 值（对应 API 中填写的 PawnID）、设置该角色是否为程序运行时默认使用角色。

- 在 Pawn Property Zoom 中可设置弹簧臂长度在设置范围内的伸缩速度变化。

- **漫游角色**

输入按键	执行操作
鼠标左键	控制漫游角色前进与前进方向
鼠标右键	仅控制漫游角色视角
键盘 WASD	控制漫游角色前进后退左右平移
鼠标指向地面双击鼠标左键	角色自动寻路走向鼠标指向位置
移动端触摸点击	控制漫游角色前进与前进方向
移动端触摸双击	角色自动寻路走向鼠标指向位置

- 漫游角色是一种第一人称视角角色，可控制角色在水平面上的移动，拥有登上台阶、重力等能力。
- 将 BP_DevKitCharacter（漫游角色类）拖拽到场景中。



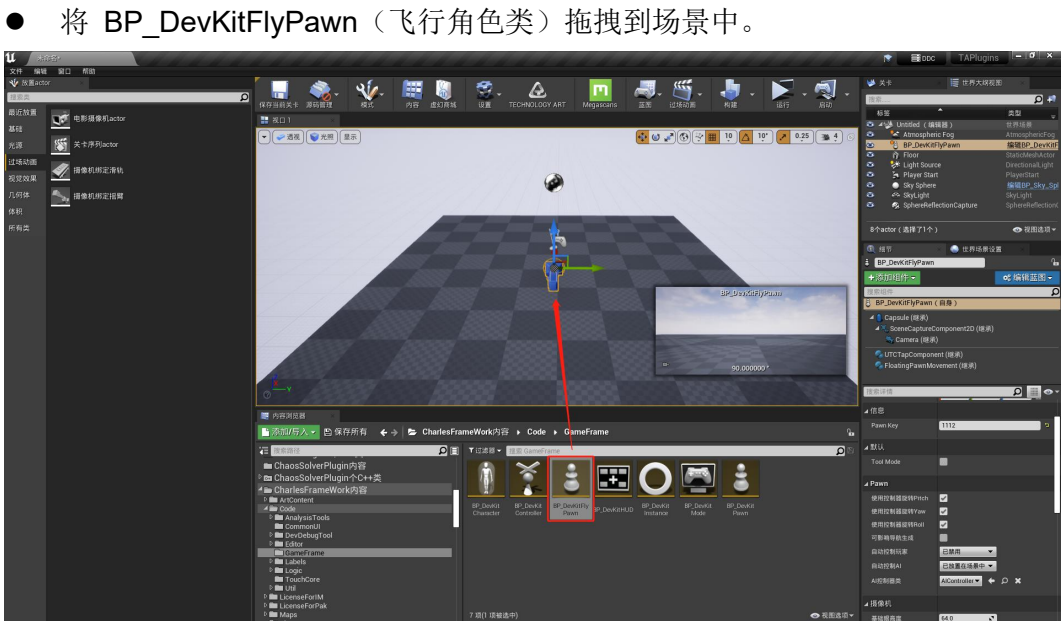
- 在右侧细节菜单中，可设施视角旋转速度、角色 Key 值（对应 API 中的 PawnID）、设置该角色是否为程序运行时默认使用角色。

- 在角色移动：行走一栏中，可以设置角色最高能登上的台阶高度、最大能移动的斜面角度、最大行走速度等。

- 飞行角色

输入按键	执行操作
鼠标左键	控制飞行角色视角
鼠标右键	仅控飞行角色前后左右平移，仅在鼠标指向物体时可用
鼠标滚轮上滚/下滚	控制飞行角色与鼠标指向物体的距离缩短与拉远
键盘 WASD	控制飞行角色朝当前方向前后左右移动
键盘 QE	控制飞行角色垂直上下移动

- 飞行角色是一种第一人称视角角色，可控制角色自由飞行。
- 将 BP_DevKitFlyPawn（飞行角色类）拖拽到场景中。

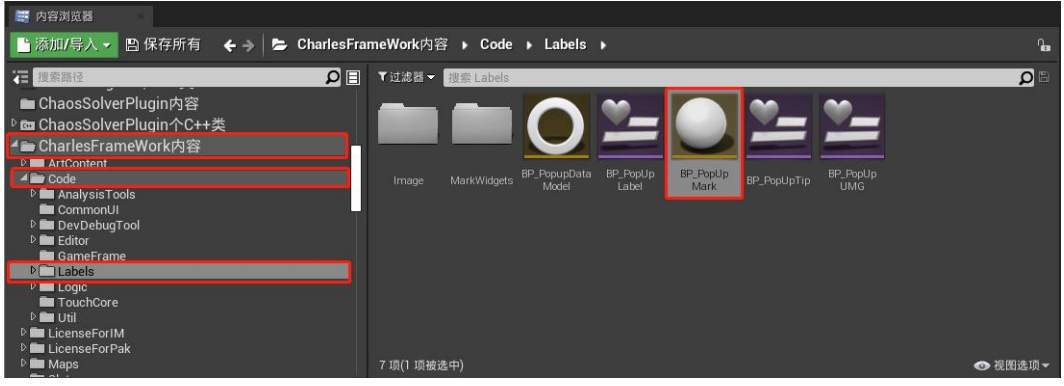


- 右侧细节面板可填写 Pawn Key（对应 API 中的 PawnID）

预设标签类

弹窗标签

- 弹窗标签预设类提供在场景中显示/隐藏标签及点击标签弹窗的能力，提供样式、大小等 UI 接口。同时该预设类下可挂载任意自定义组件，如模型的显示隐藏、粒子的激活与停止、自定义场景 UI 的显示隐藏等皆可以挂载在该类下，通过 API（ShowMarkAPI）对其控制。
- 在插件路径中找到“BP_PopUpMark"。



- 将“BP_PopUpMark"拖拽到场景中。

- 右侧细节面板中“TACONFIG”一栏中，可设置标签样式、标签渲染枢轴、标签大小缩放、是否需要点击弹窗。

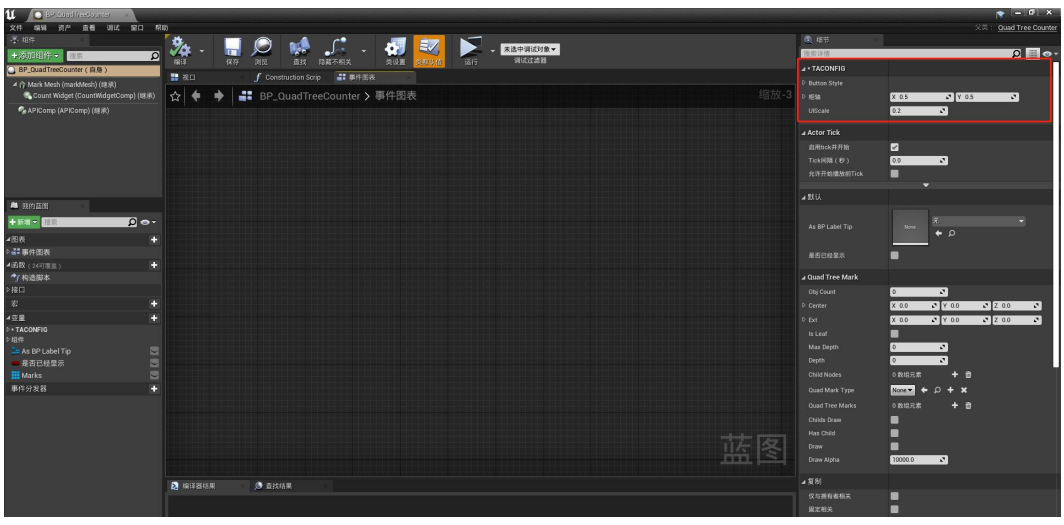
- 在 Actor 栏中的标签插槽，可添加 Tag Name，该名称可用于 ShowMarkAPI 中的调用（对应 API 中的 Mark Name）。同一 Actor 可添加多个 Tag Name，可对应不同 API 调用同个标签显示隐藏。

- **密集撒点聚合标签**

- 密集撒点聚合，是智慧系应用场景中常用的标签功能之一，使大量标签聚集时可聚合并显示堆叠数量。

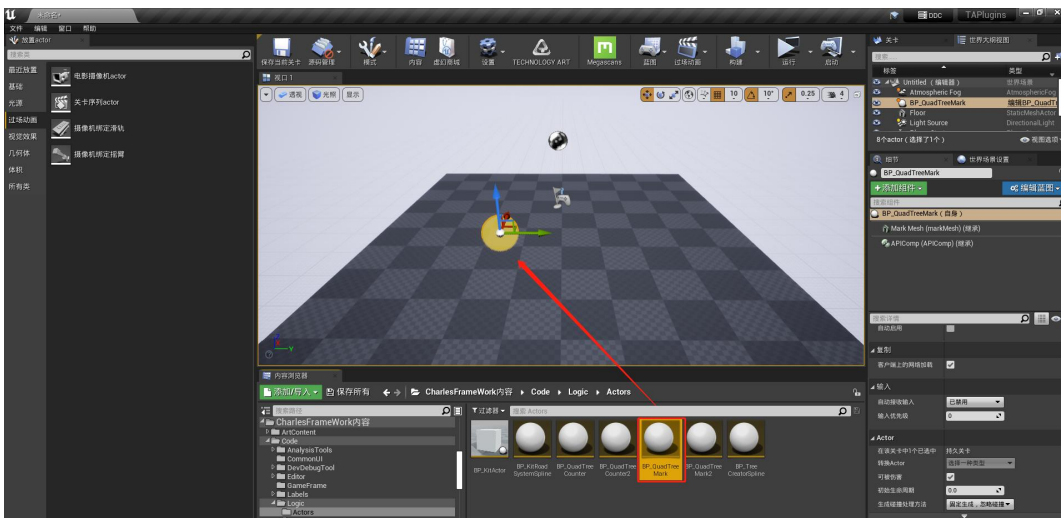
- 在插件目录中找到 Actor 文件夹。

- BP_QuadTreeCounter 代表标签样式与类型，可双击进入蓝图界面进行样式、参数设置。也可在左侧组件栏自定义所需挂载的组件。该预设类仅定义撒点标签样式，无须拖拽到场景中摆放。



注意：添加新的组件后，必须将其拖拽到 Count Widget 下。

- BP_QuadTreeMark 代表标签撒点位置，可手动拖拽到场景中摆放，也可自行拓展动态生成逻辑。



- 撒点类型与聚合参数均可在“Quad Tree Mark API”中进行调整（详见 API 说明）

- **聚合标签开发拓展-如何获取未聚合时的每个独立标签对象**

- 在 QuadTreeCounter 中，拥有以下参数

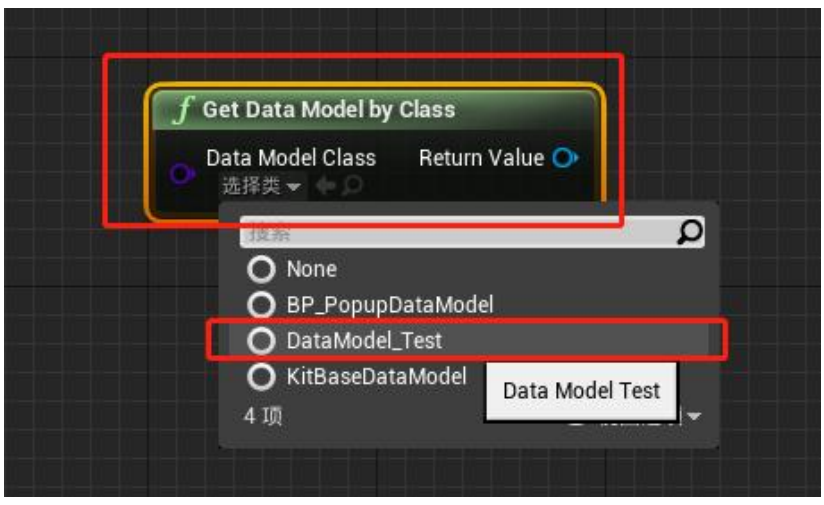
- 其中 Obj Count 代表当前标签的聚合数量，即 Obj Count=1 时，则标签未聚合
- 此时获取 Quad Tree Marks 数组中的第 0 个元素即可获取未聚合的单个标签对象

- **其他预设**

- **DataModel**

- DataModel 是一个全局可调用的“数据仓库”。
- 在蓝图创建窗口中，选择“KitBaseDataModel”做为父类并创建蓝图。

- 创建后即可在全局蓝图中调用该 DataModel。



TechnologyArt-插件其他拓展功能

● 基本概述

插件拓展了众多快捷功能，包含美术、程序、UI 等多个方面，了解并使用这些功能可提高制作效率。

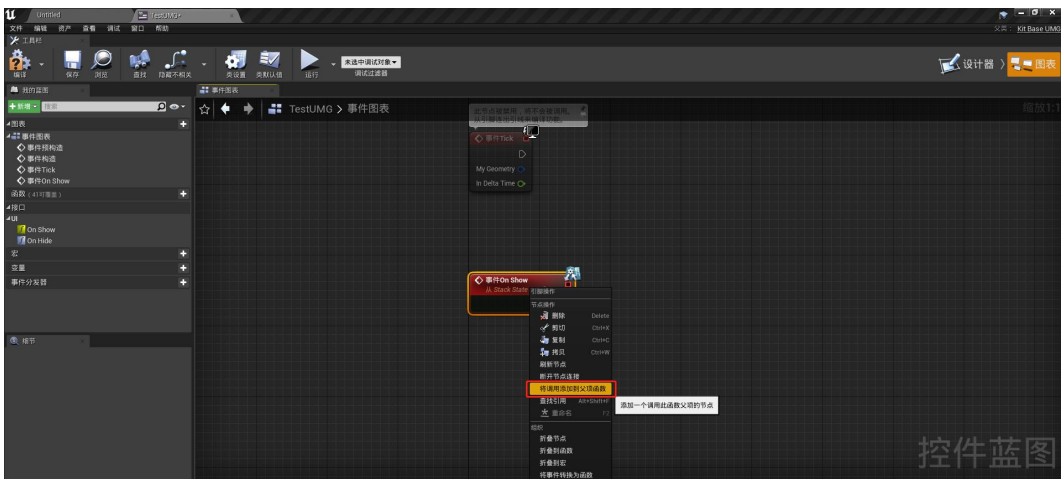
● UI 控件功能拓展

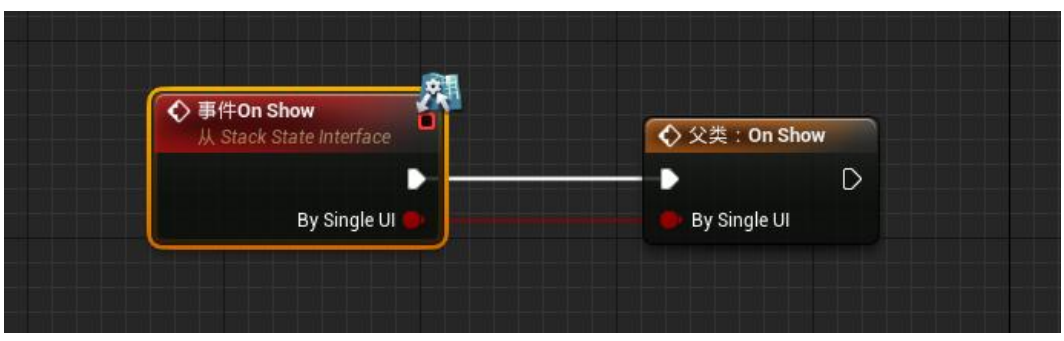
● 控件事件

- 当控件父类为 Kit Base UMG 时，控件蓝图将拥有显示隐藏接口

- On Show：该事件将在该控件显示时执行
- On Hide：该事件将在该控件隐藏时执行
- 右键点击 On Show/Hide 的实现事件将可覆写或拓展该控件显示隐藏时执行的逻辑。不实现事件时将默认继承父类中的显示隐藏方法。

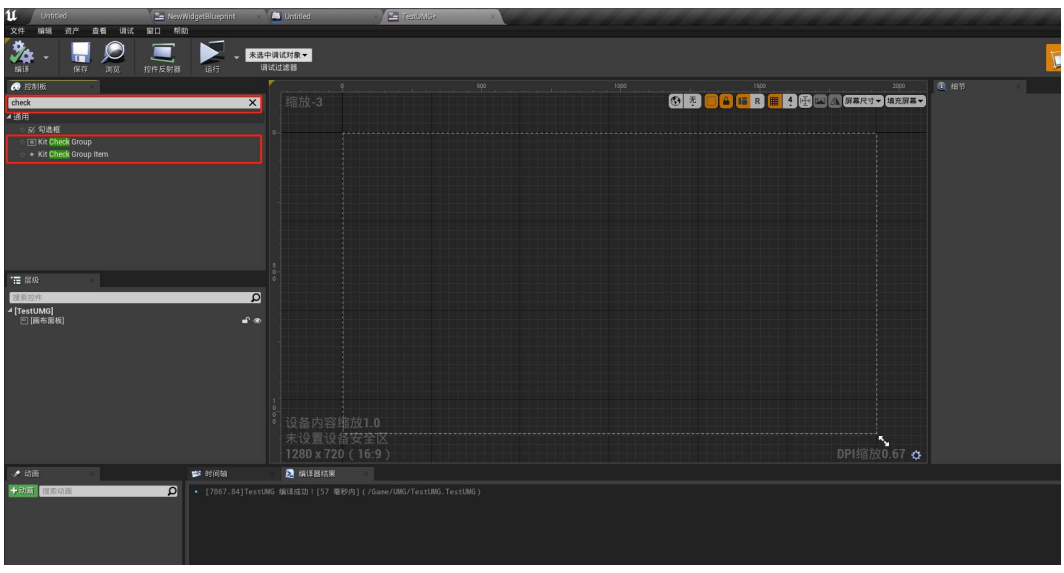
- 如需继承父类中的显示隐藏控制（由 API 控制），需右键事件点击“将调用添加到父项函数”，将该节点与事件连接。相反，如需覆写父类中的 Show/Hide 方法，则无需调用父项函数。



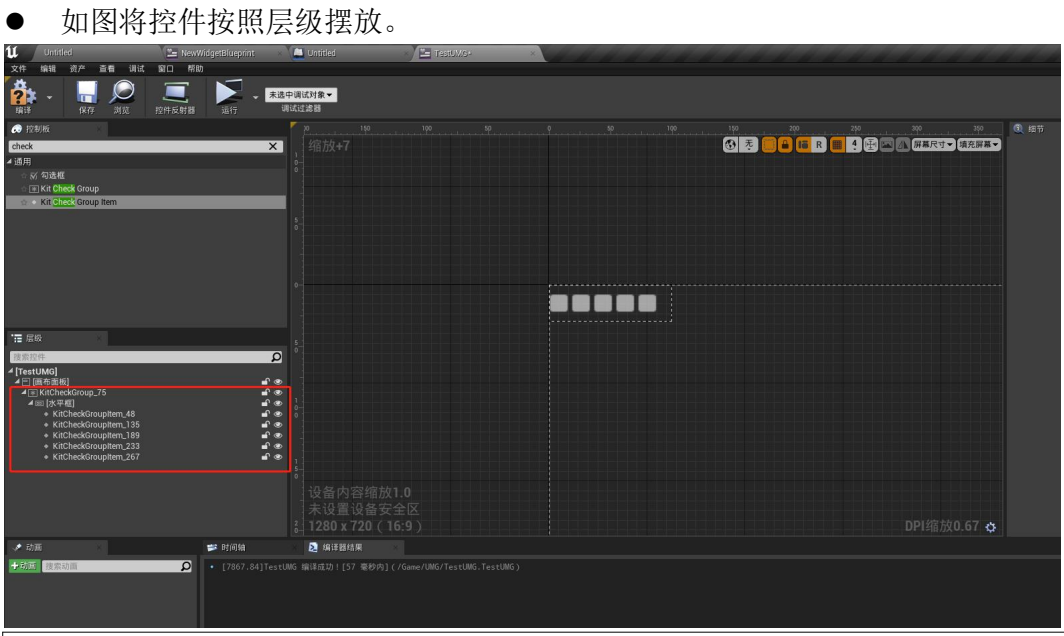


● 勾选组（Check Group）

- 勾选组是一组勾选按钮状态互斥的 UI 控件，并使勾选后的按钮无法第二次点击取消选中，仅可通过点击同组其他勾选按钮更换选中按钮，有效防止在当前状态时重复调用当前状态事件。勾选组拥有选中、未选中的事件回调，极大的方便 UI 制作。
- 在 UMG 控件中，搜索“Check”。

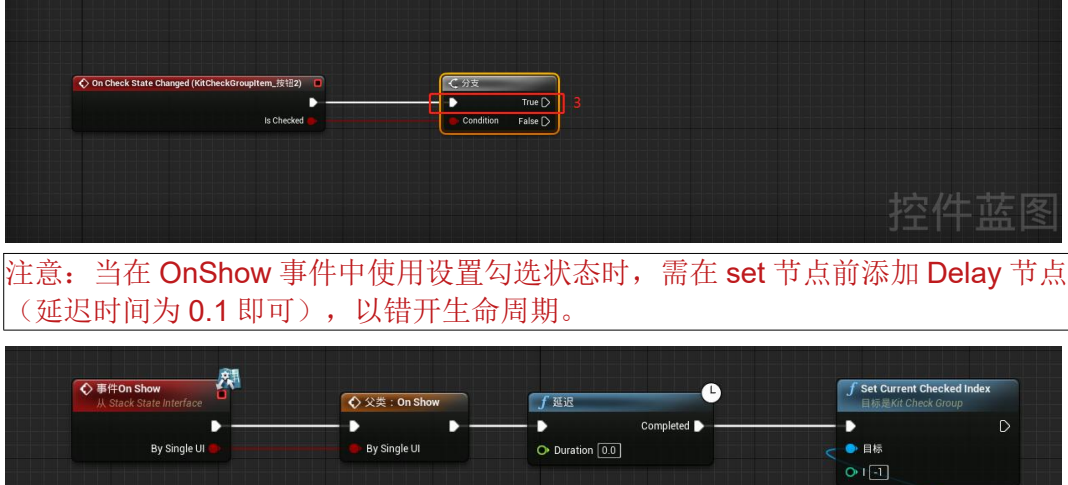


- 如图将控件按照层级摆放。



注意：使用 KitCheckGroup 必须在 Check Item 之上包裹一个面板控件，不限与水平框面板，画布面板、垂直框面板等均可。

- CheckGroup 仍有常规参数设置，其中勾选 Group Allow Switch 使勾选按钮可重复点击以取消选中状态。



- Default Check index 可设置该勾选组显示时默认选中的 check 按钮序号，“-1”为不进行默认勾选，“0”为默认选中勾选组中的第一个按钮，以此类推。

- 在控件蓝图中，CheckGroup 拥有以下封装函数：

Get Current Checked：返回当前选中按钮

Get Current Checked Index：返回当前选中按钮序号

Set Check Next：设置选中状态为当前选中按钮的下一个，如已为最后一个，不做操作

Set Check Prev：设置选中状态为当前选中按钮的上一个，如已为第一个，不做操作

Set Current Checked：设置当前选中按钮

Set Current Checked Index：设置当前选中按钮序号

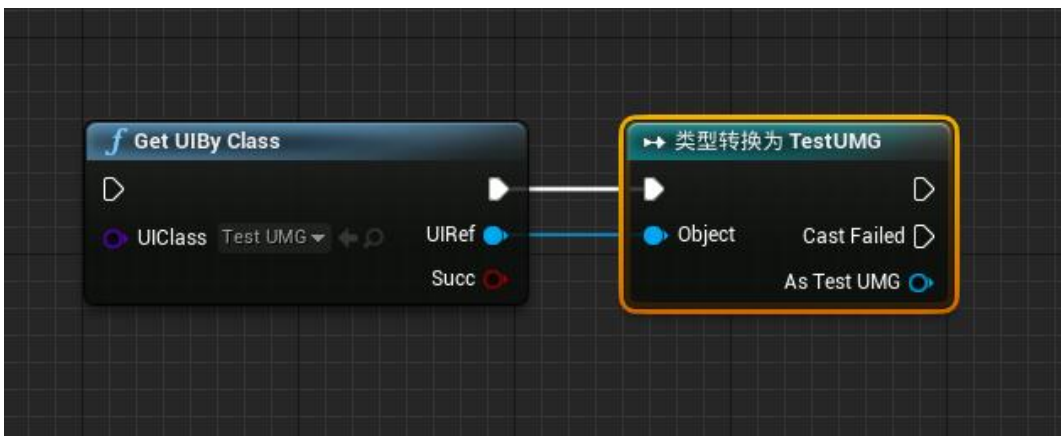
注意：CheckGroup 中，按钮 Index 从 0 为起始 Index。

- 在选中按钮 1-选中按钮 2 的操作流程中，勾选按钮按下及弹起事件顺序如下，某些情况下需特别注意事件执行顺序。

注意：当在 OnShow 事件中使用设置勾选状态时，需在 set 节点前添加 Delay 节点（延迟时间为 0.1 即可），以错开生命周期。

- **控件蓝图通信**

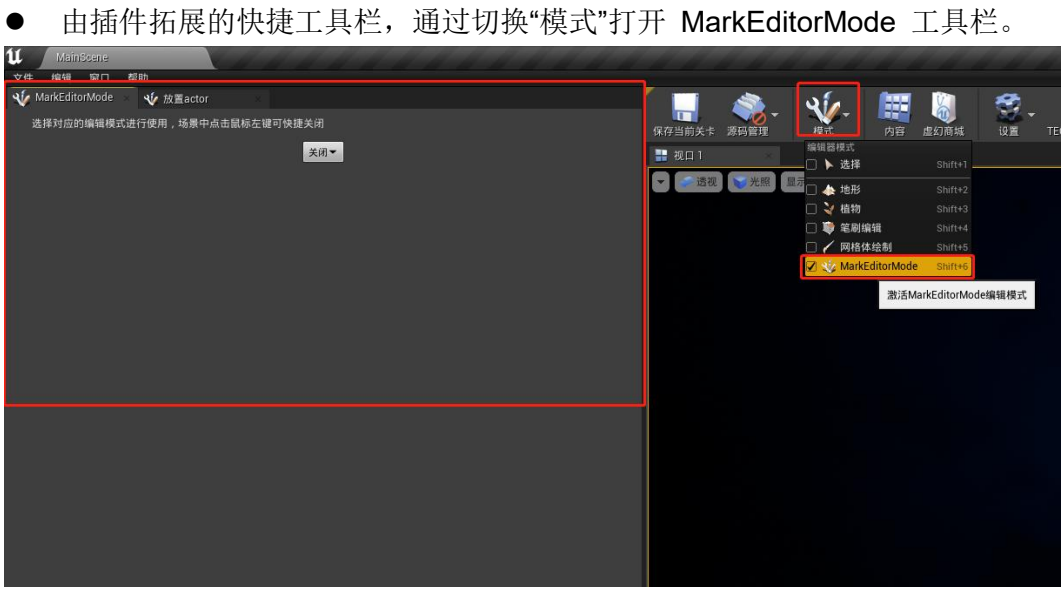
- 在全局蓝图中可使用 Get UIBy Class 进行控件蓝图间通信，控件蓝图父类必须为 Kit Base UMG。



编辑器功能拓展

MarkEditorMode

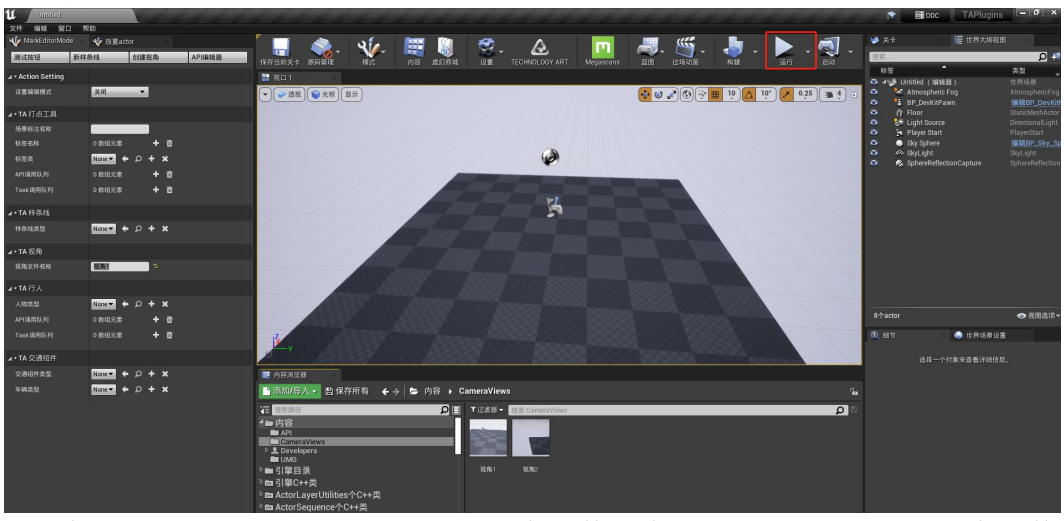
- 由插件拓展的快捷工具栏，通过切换“模式”打开 MarkEditorMode 工具栏。



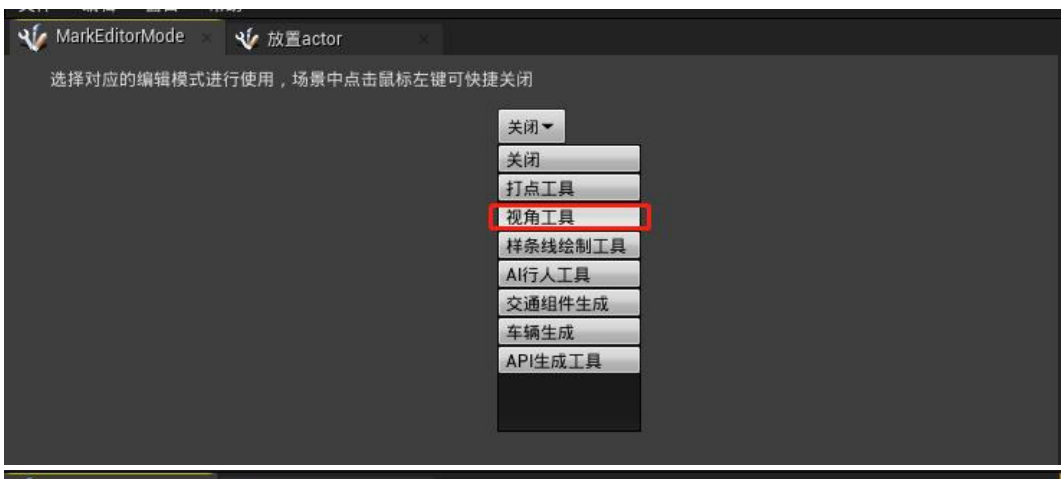
注意：在 MarkEditorMode 模式下时，编辑器交互被工具交互代替，这将无法使用选择模式下的功能，如选择物体等。注意不使用工具栏时将模式切换回选择模式

- **创建与使用视角文件**

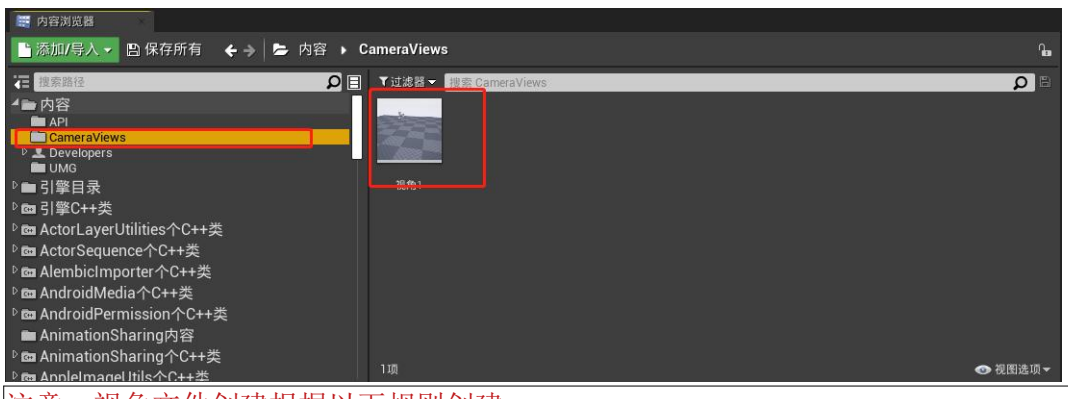
- 插件工具栏提供快速创建视角文件功能，调整镜头时，无须手动记录坐标、弹簧臂长度、当前角色等参数，创建视角文件后，通过调用 API 指定视角文件，快速完成镜头设置。
- 在使用 Dev Kit pawn（鸟瞰角色）、Dev Kit Fly Pawn（飞行角色）、Dev Kit character（漫游角色）的前提下，点击上方菜单“运行”按钮，运行程序。（如未使用插件所带的三种角色之一，则无法在运行时记录视角文件）



- 在 MarkEditorMode 工具栏下方“视角文件名称”一栏填写需要创建的视角文件名称，填写名称后点击“创建视角”。



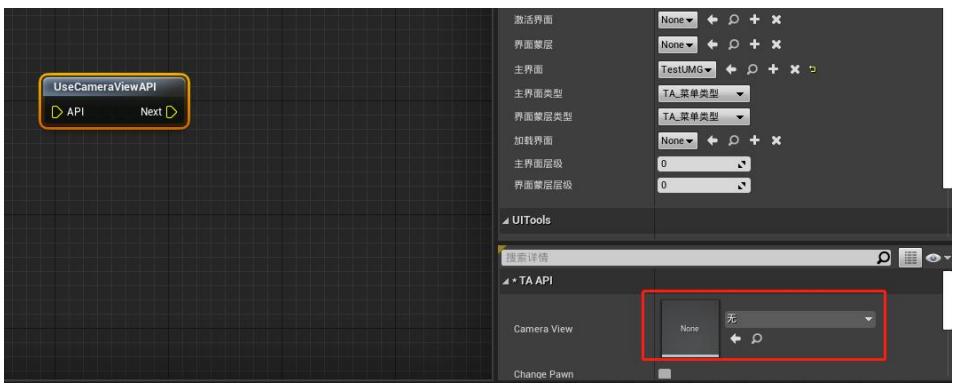
- 视角文件将被创建在内容下 CameraViews 文件夹内

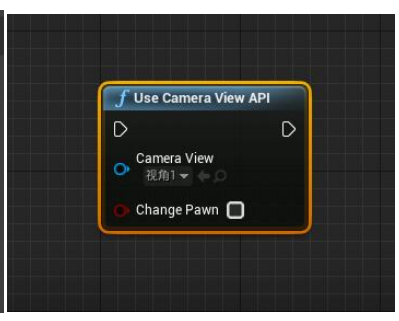


注意：视角文件创建根据以下规则创建。

运行时记录：记录当前 DevKitPawn 的坐标信息、伸缩臂信息、PawnID、过渡时间
未运行时编辑器视口下记录：记录当前视口坐标信息，伸缩臂长度将由当前视口角度射线击中的碰撞为准。（当视口中无有效碰撞物体时，记录的坐标可能会在远处，建议在朝向物体时再记录视角文件。

- 视角文件将可被使用在所有 API 插槽或 Use Camera View API 的 Camera View 插槽中。





- 打点工具

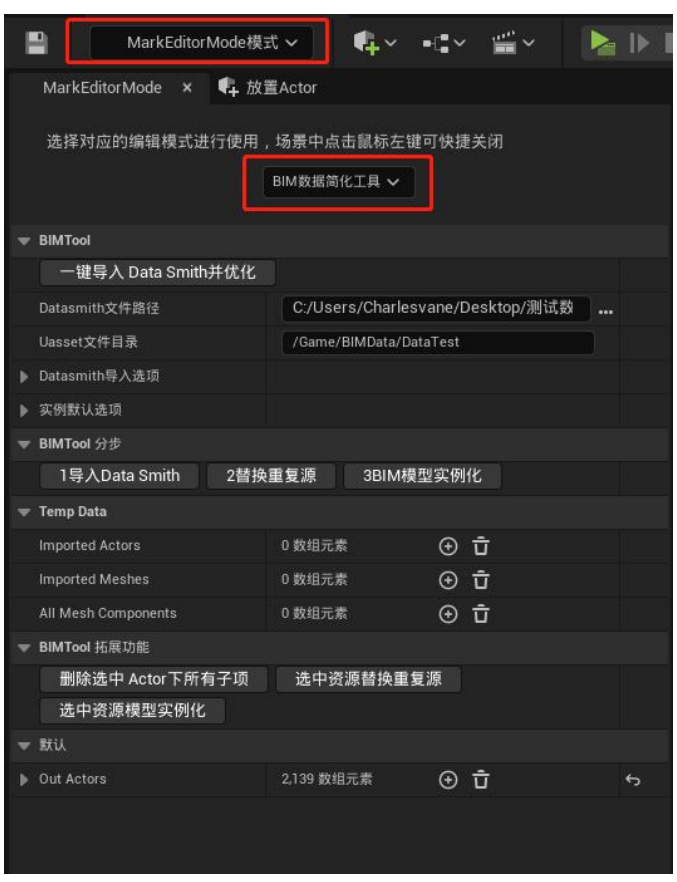
- 插件工具栏提供快速打点工具，指定标注名称、标签名称、标签类后，可使用左键在场景中点击，在点击位置将生成一个指定标签类的 actor。并将标注名称作为 actor 名称，标签名称作为 actor tag（便于使用 show mark api）。

- 标签类将可以指定所有继承自“KitMarkActor”类的蓝图类，故该功能不仅限与标签打点，可根据开发需要灵活使用。

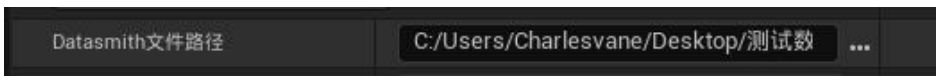
- **BIM 数据简化工具**

- 该工具通过检测相同模型并替换为同一资源，并从简化后的资源进行实例化，原多个相同模型资源将合并到同一实例中，同时将保留原模型中的元数据（metadata），此工具将可大幅优化 BIM 机电设备性能，同时实现构件点击获取元数据等功能。模型进行实例化后，应配合相关实例 API、绑定实例监听事件、实例 UUID 等使用。

- 在“MarkEditorMode”中找到“BIM 数据简化工具”



- 在“Datasmith 文件路径”中填入需要导入并优化的 Datasmith 文件，右侧省略号可在文件夹中选取



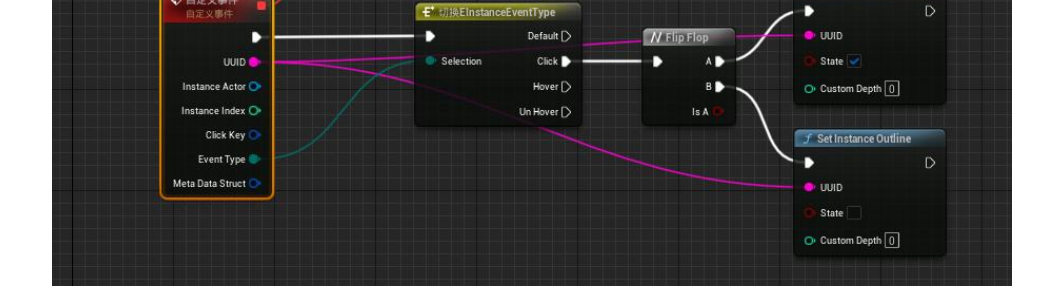
- 在“Uasset 文件目录中”中填入 datasmith 导入后文件生成文件夹，如填入文件路径不存在，则会自动生成

- 在“Datasmith 导入选项”与“实例默认选项”中可设置文件导入设置与实例生成设置，其中实例选项中可设置实例 Actor 最小渲染距离与最大渲染距离。

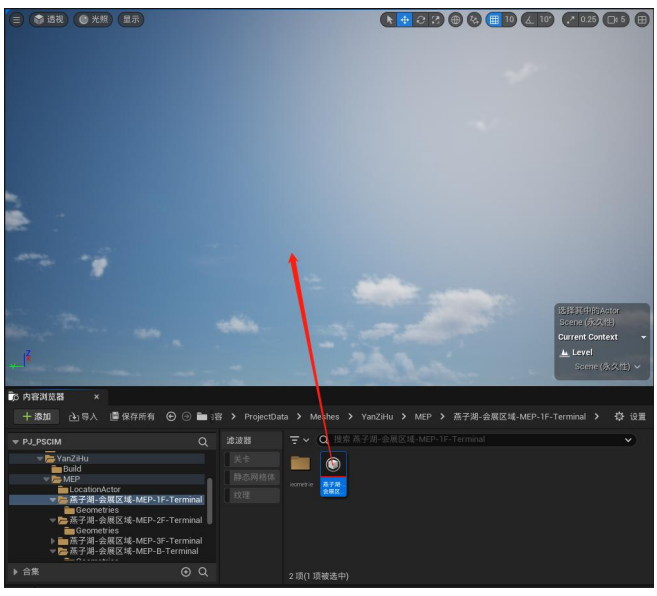
- 完成以上设置后，点击“一键导入 Datasmith 并优化”按钮，工具将自动对 datasmith 文件进行优化并在场景中生成实例模型。

- 以上为全自动化流程，如有多个 datasmith 文件需进行共同优化，可使用工具下方“BIMTool 拓展功能”进行手动优化。

- 可在资产库中多选文件夹，并进行类型过滤，选中需要进行优化的模型资产。



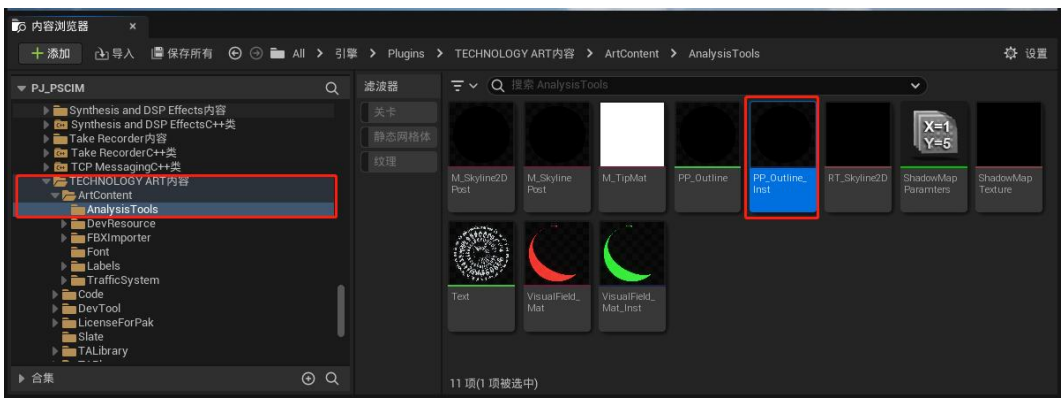
- 需先使用“选中资源替换重复源”对资产进行优化
- 再使用“选中资源模型实例化”对选中资产进行实例化，实例化前需确保 datasmith 模型处于场景中，如不在场景中，将导致实例化失败，可如下图将 datasmtih 场景文件拖入场景中。



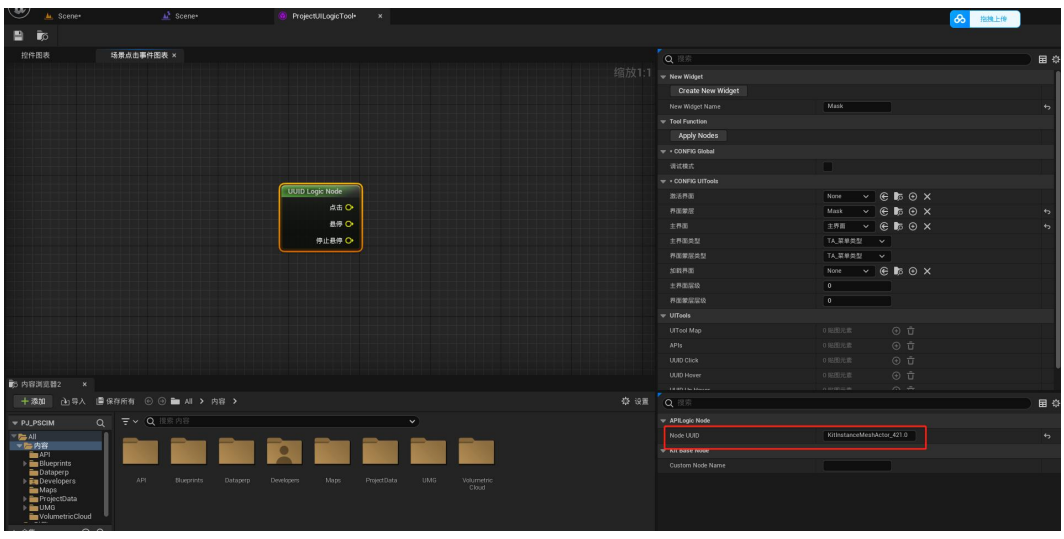
- 完成实例化生成后，再使用“删除选中 Actor 下所有子项”对 datasmtih 原模型进行快速删除。
- 完成 BIM 数据优化后，可通过“On Click Instance Listenner”对鼠标与实例模型之间的交互进行监听。

- 其中"UUID"可获取实例 Actor 内每个索引模型的 UUID，“Instance Actor”可获取实例 Actor，“Instance Index”可获取当前与鼠标交互的模型在实例 Actor 中的索引，“Event Type”可对鼠标交互事件进行区分，"Meta Data Struct”可获取与鼠标交互的模型元数据、世界坐标。
- 由于实例模型的特殊性，对选中模型的描边无法使用“Set component outline”API，需使用“Set instance outline”对选中实例索引进行描边。实例描边已对 Nanite 模型进行兼容，可以 Nanite 模型进行描边。

- 描边颜色可在插件资产中搜索“PP_Outline_inst"材质进行设置。

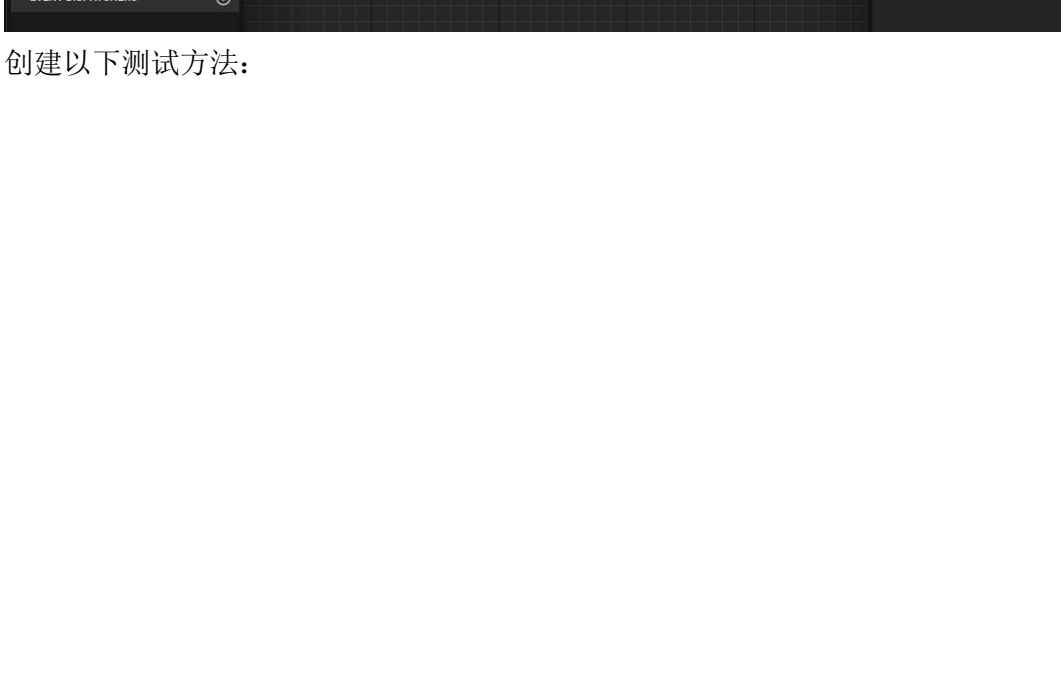


- 实例 UUID 同样可在管理器中使用，在场景中可通过点击实例 Actor 后右键该实例 Actor 中的
  单个模型，快速复制该模型的 UUID。

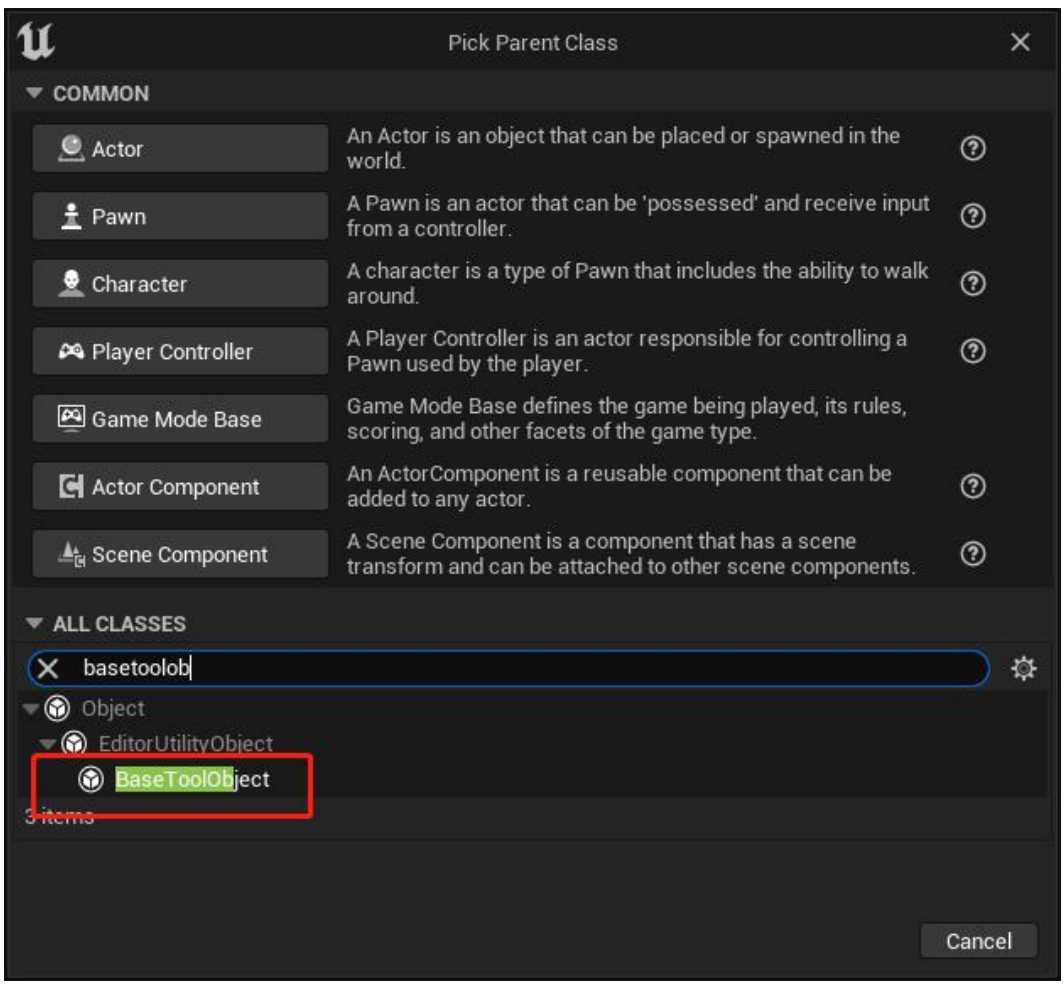


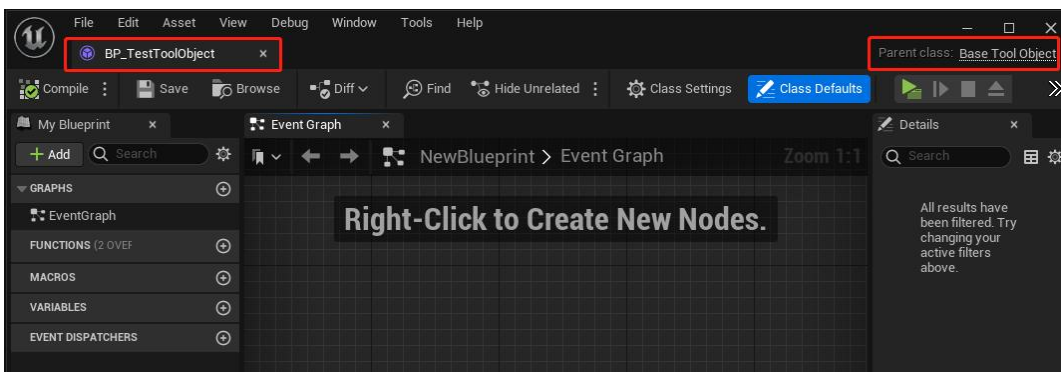
- **自定义编辑器功能拓展**

类似 BIM 数据简化工具的做法：

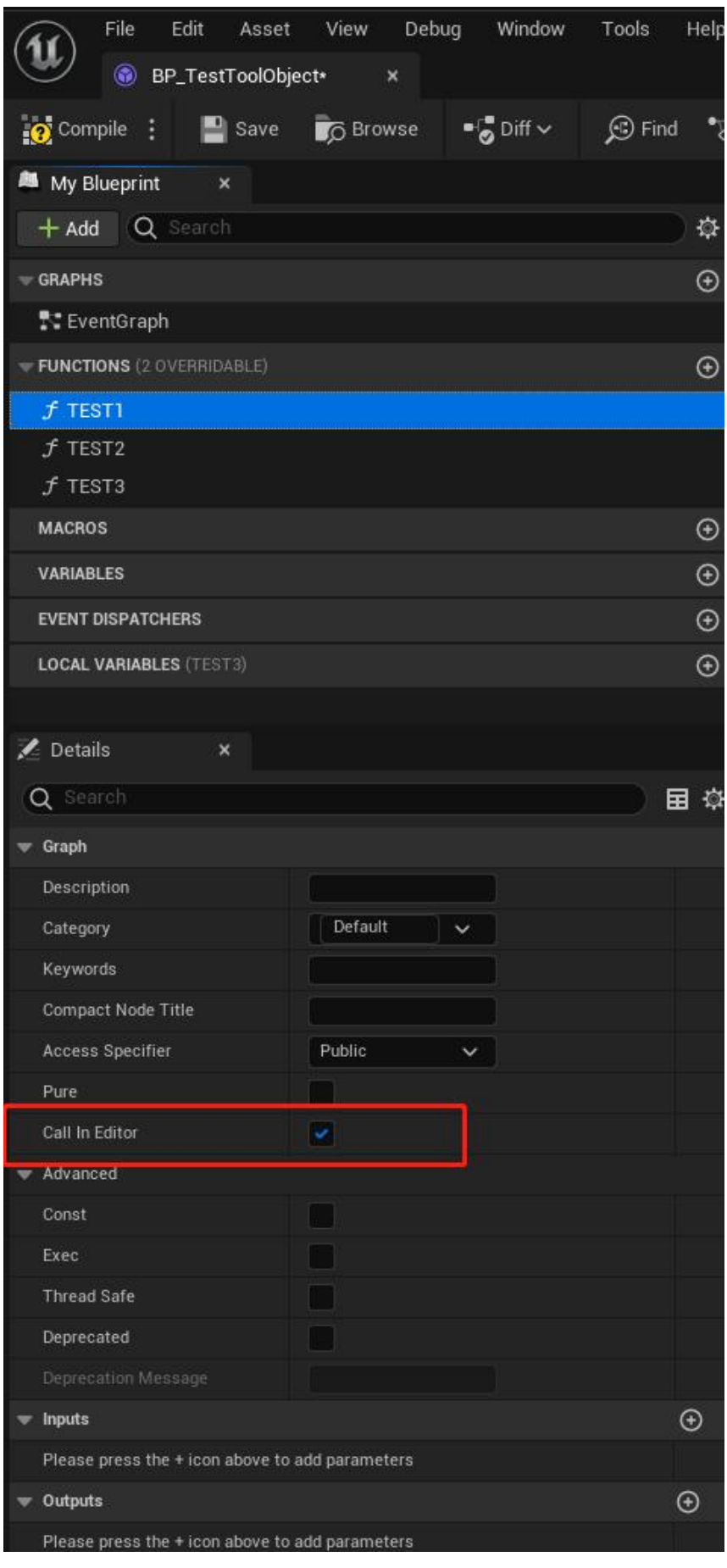


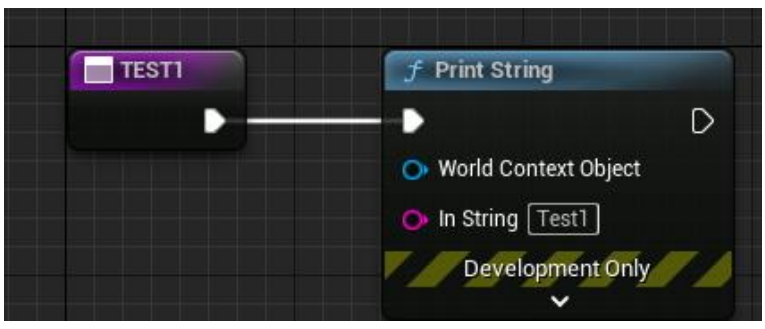
创建该类型蓝图 BaseToolObject：



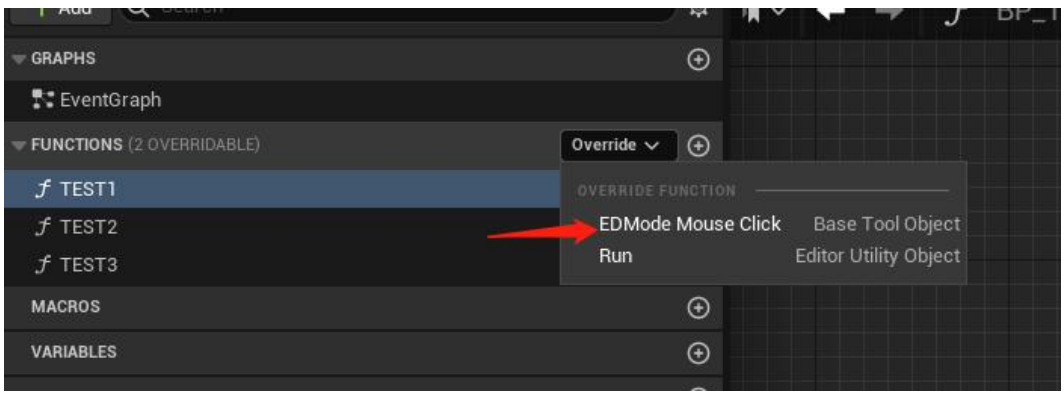


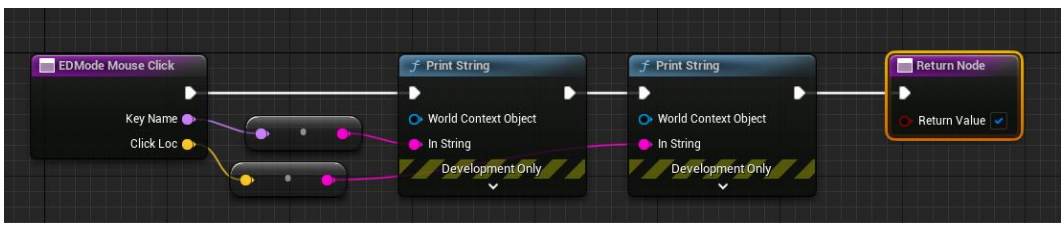
创建以下测试方法：



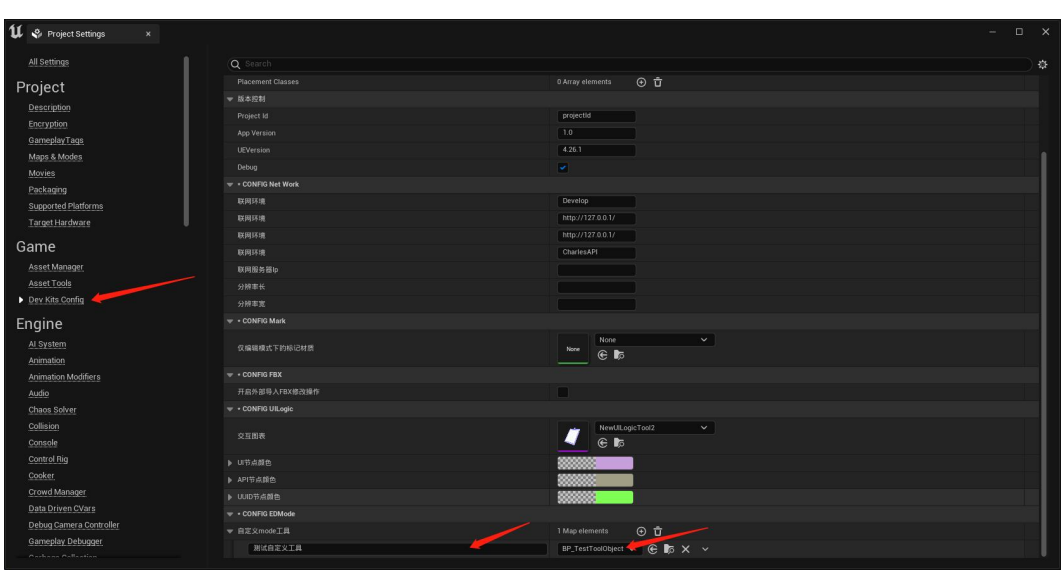


其中有一点比较重要的是重写这个鼠标点击事件 （如果需要的话）

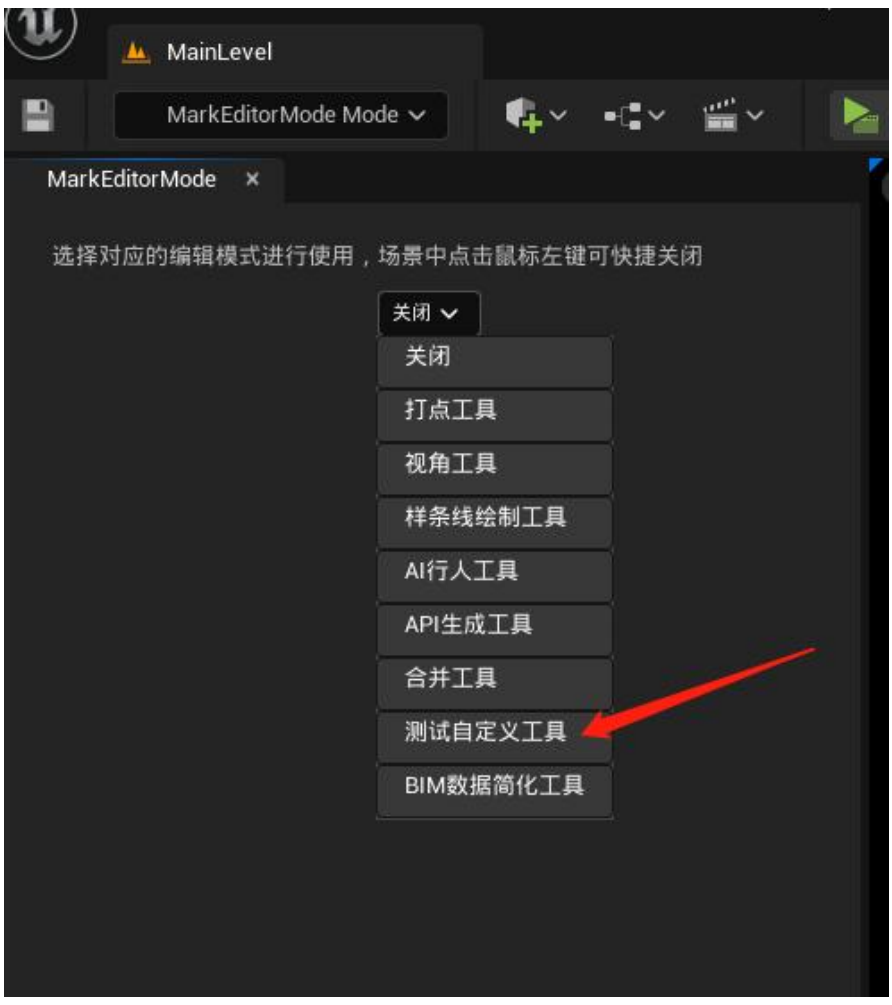




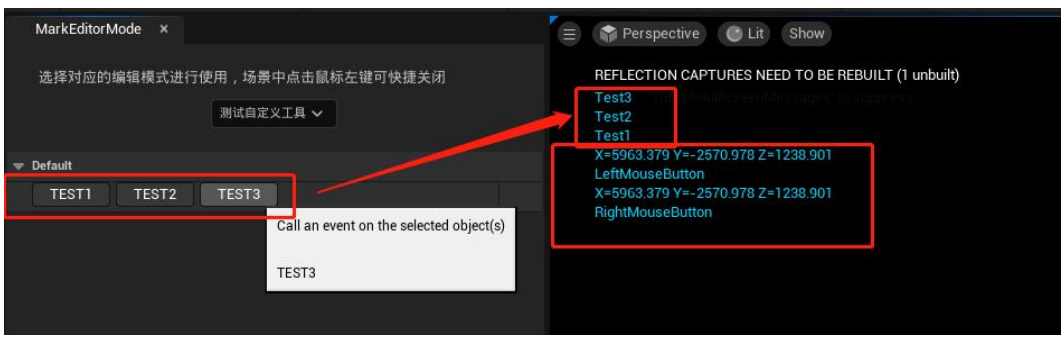
其中 KeyName 是：捕获点击事件是左右键的区分 分别是 ：
左键/右键 || LeftMouseButton/RightMouseButton
ClickLoc 是点击的点在场景中的坐标
reture 的 bool 值 控制的是是否阻止继续向下传递调用：true 时则只执行自定义的部分
false 时会调用原先的左键选择功能和右键关闭工具功能和右键菜单弹出功能。
开发完成后在 config 中进行配置：



然后需要重启编辑器，可以看到自定义工具被插入到工具选项中了：



开启工具测试，可以看到调用编辑器方法成功了，并且鼠标事件也顺利打印：



接下来就可以随时开始进行编辑器工具的开发拓展了，直接使用蓝图即可无需 C++

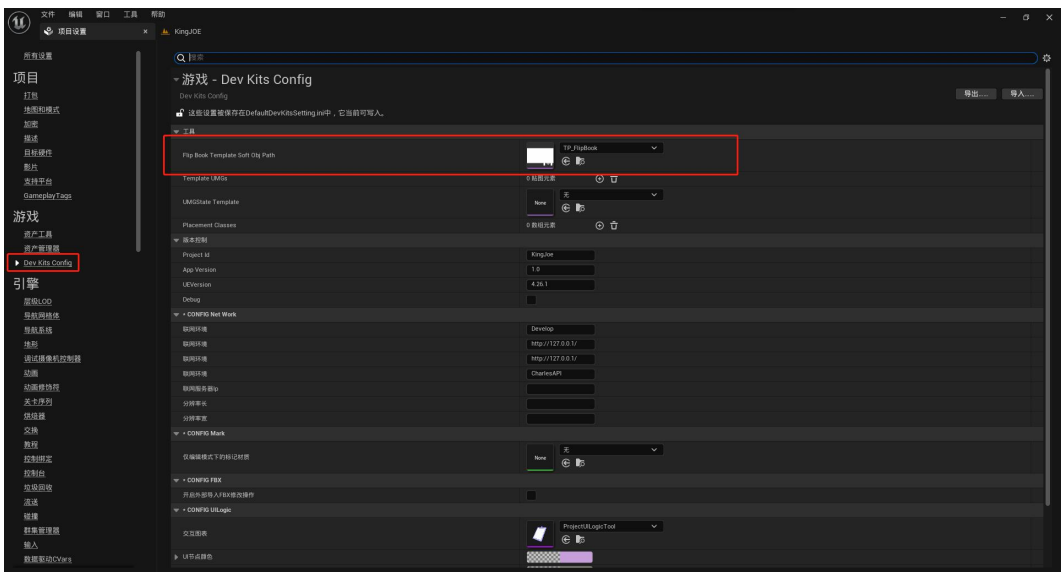
● 图片序列功能拓展

● 拓展功能概述

播放图片序列文件时，通常需要在后半段进行循环展示播放，其中不循环的前半段作为图片序列的入场动画，插件将针对该需求进行拓展。

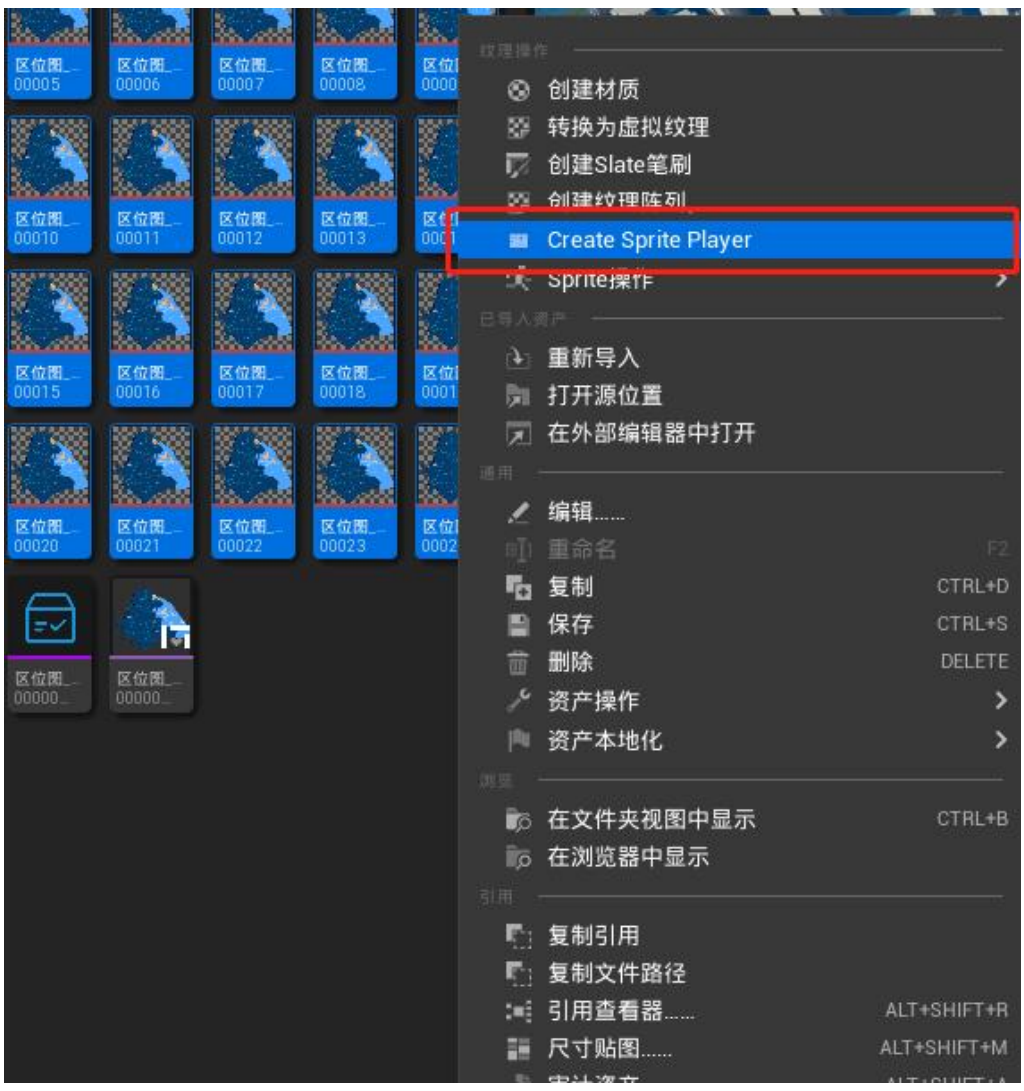
● 图片序列功能使用

- 在项目设置中添加序列 UMG 模板，使用插件自带模板"TP_FlipBook"

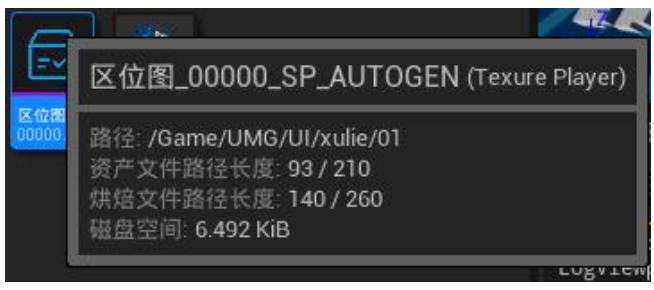


- 导入图片序列文件，并将纹理组改为 UI

- 选择需要创建为序列的图片文件
- 在右键菜单中找到“Create Sprite Player



- 创建文件完成后可以双击进入文件进行循环帧、帧率等设置



- ”Is Autoplay“可设置是否自动播放
  ”Is Loop“可设置是否使用循环
  ”Loop Start Key Frame“可设置开始循环帧
  ”Play Rate“可设置播放速度
  ”Start Key Frame“可设置播放开始帧
  ”Frames Per Second“可设置每秒播放帧数

- 右键 Texture Player 文件，点击”Create Texture UMG Player“即可创建序列控件 UMG

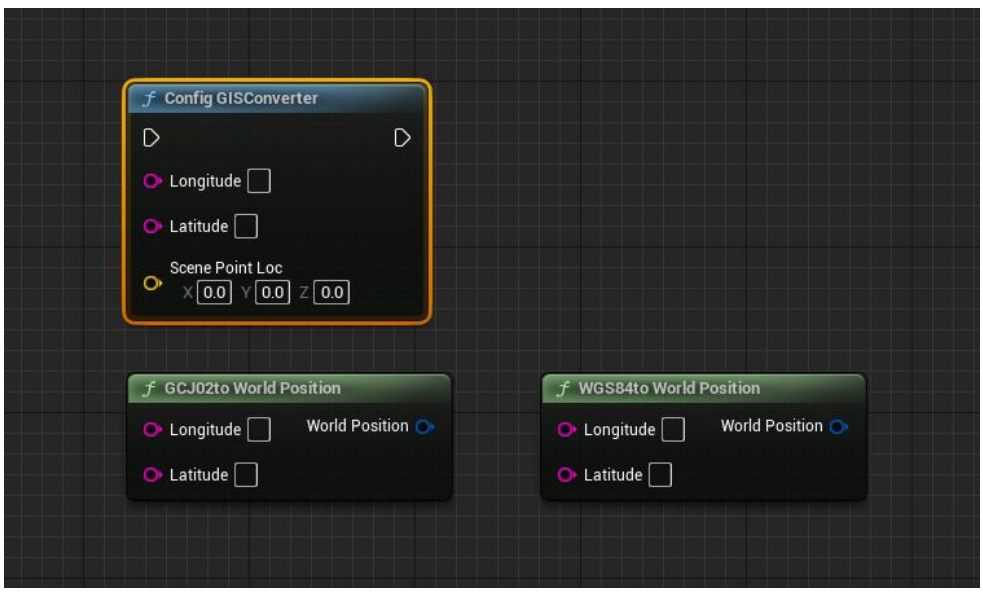
- 该 UMG 可以在其他控件 UMG 中使用或蓝中加载

- 创建的序列 UMG 拥有”Play“函数，可以在非自动播放的情况下手动播放序列

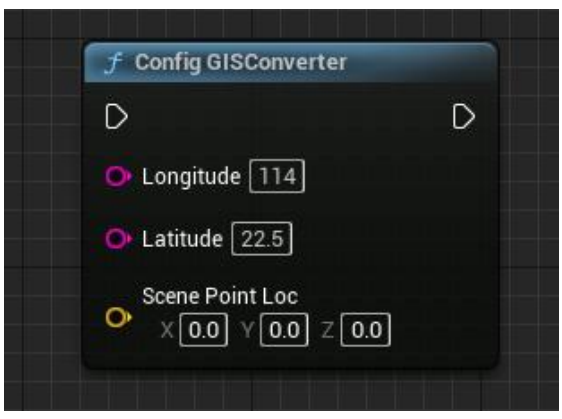
- 其他功能拓展

- 真实坐标系转换到 UE 坐标系

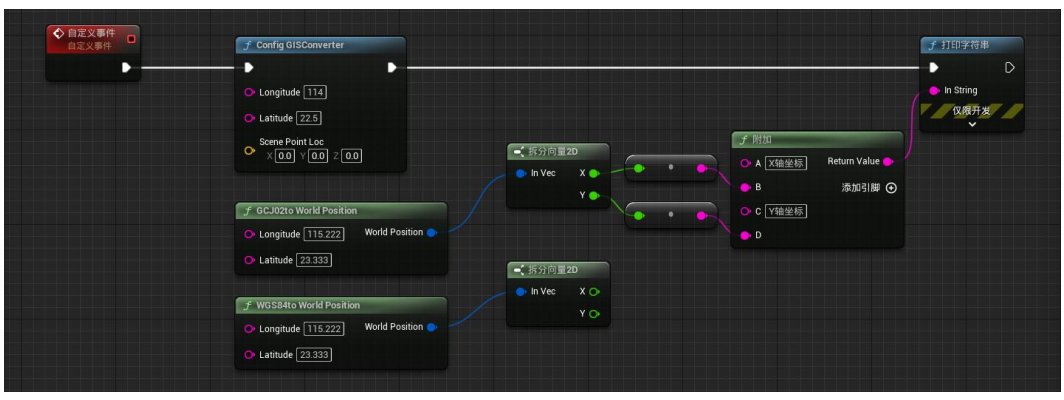
- 插件支持快速转换 WGS84 坐标系、GCJ02 坐标系（火星坐标系）转换到 UE 世界坐标系



- 先通过“Config GISConverter”节点对 UE 场景参考点进行标定，如在 UE 场景中坐标（0，0，0）所对应真实世界中的（114°E，22.5°N）。如为西经/南纬，值为负。



- 根据需求使用 GCJ02 或 WGS84 的转换节点获得 UE 世界坐标



TechnologyArt-UILogicTool

- 基本概述

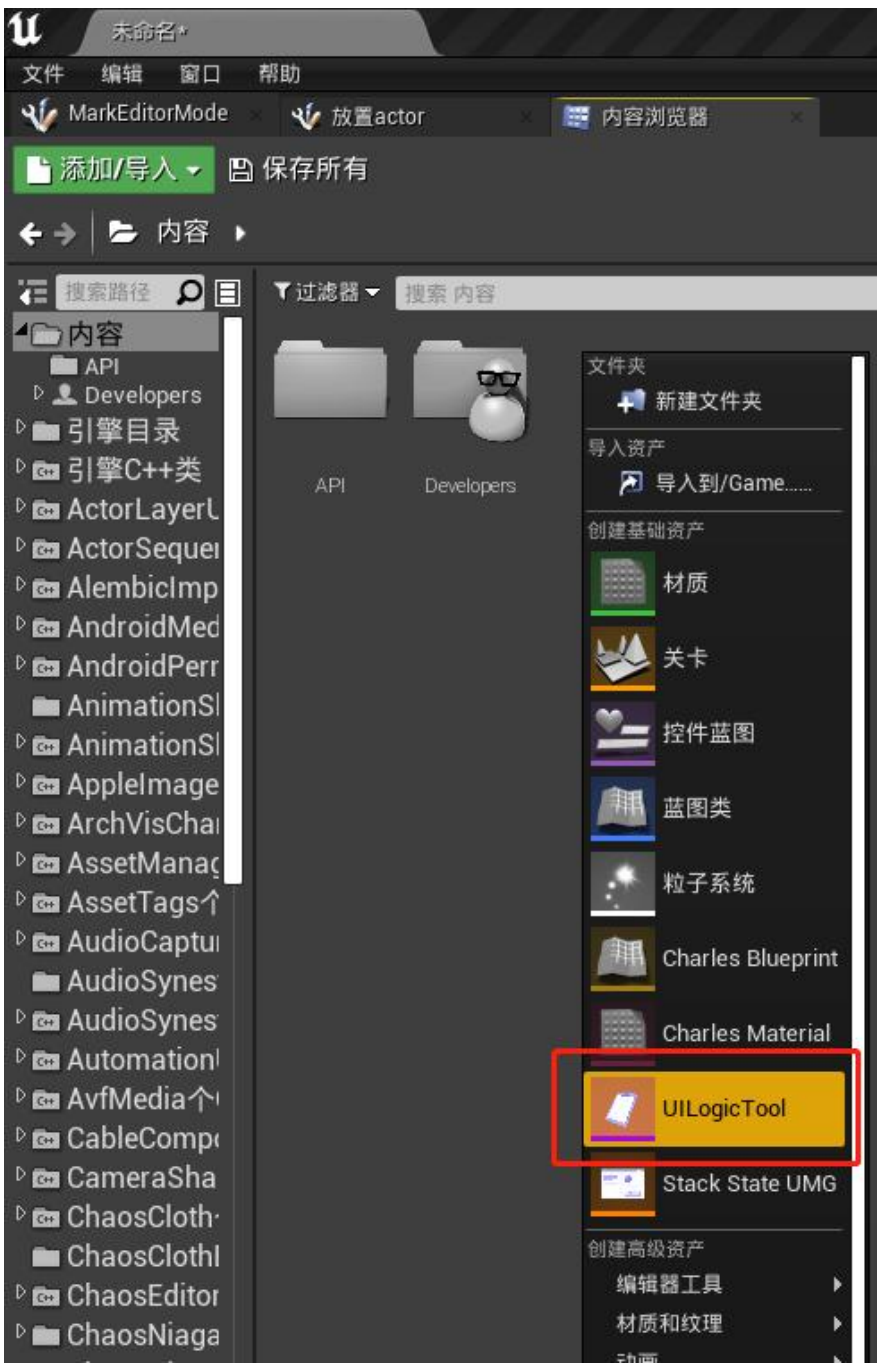
UILogicTool（以下简称 UI 管理器），以模块链接的方式快速创建各个 UMG（UI 界面）之间的交互逻辑，可使用户以无代码的方式快速完成交互设计。

- UI 管理器创建与使用

- 创建 UI 管理器并为项目配置

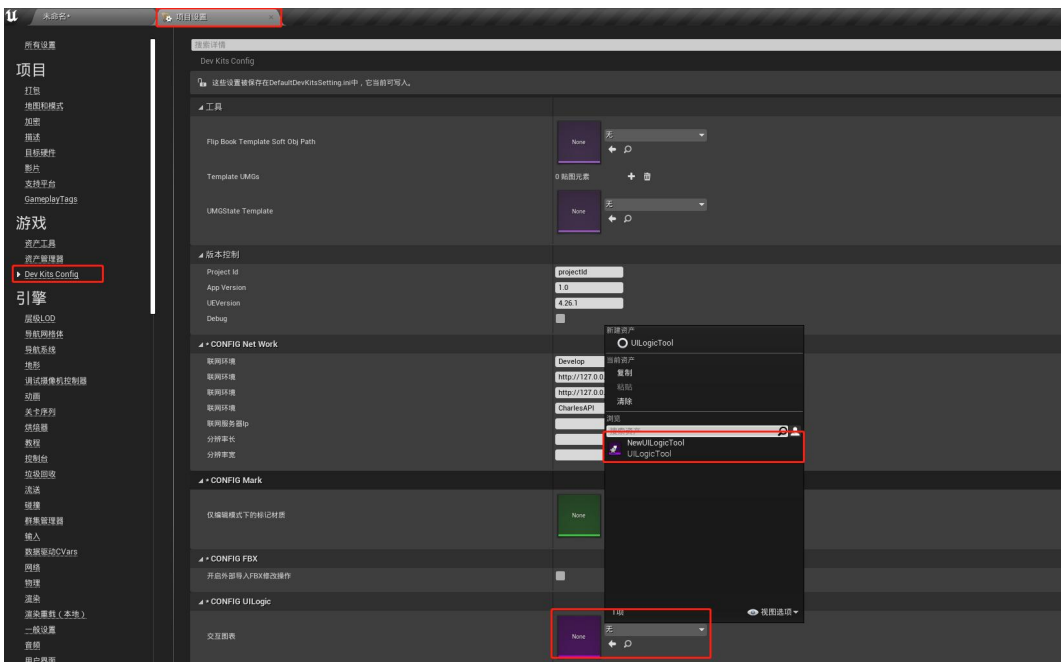
注意：根据首节 License 文档中的配置步骤操作后，项目使用的 UI 管理器将在创建时的指定路径生成，以下为未创建时情况。

- 在资产管理器中右键空白处，在弹出菜单中点击“UILogicTool”项



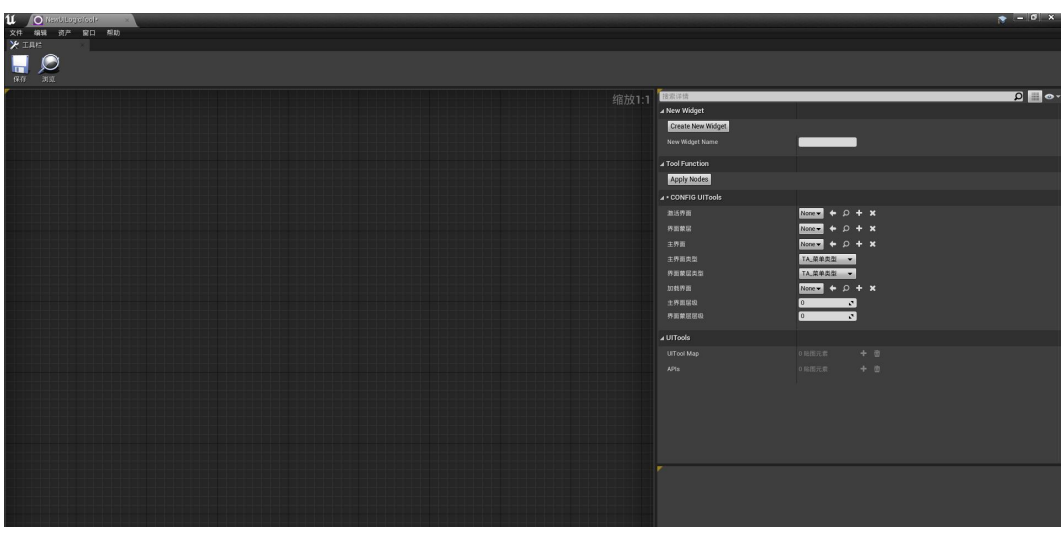
- 完成创建

- 在项目设置-Dev Kits Config-交互图表 中可指定新创建的 UI 管理器



● 使用 UI 管理器

- 双击资产打开 UI 管理器

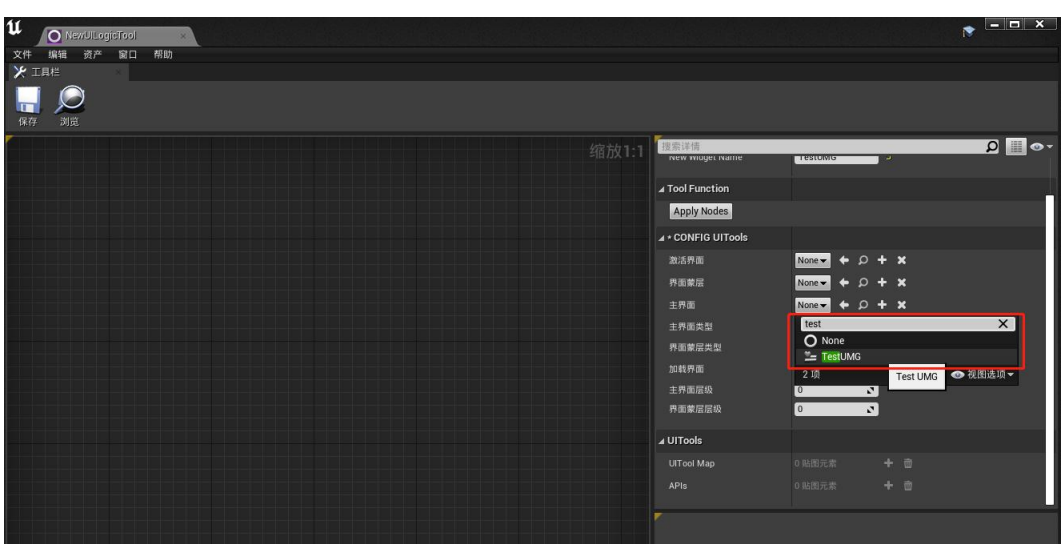


- 为接下来需要创建的 UMG 控件填写名称，点击 Create New Widget 创建一个 UI 管理器可控的 UMG 控件资产

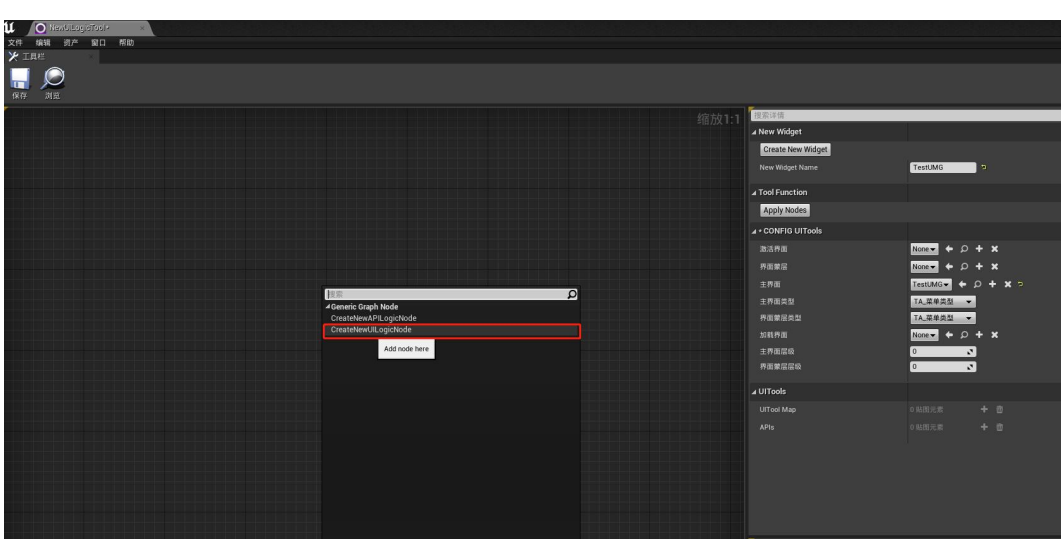
注意：不建议直接使用 UE4 原生用户控件类，如仍需要使用原生用户控件类需知悉以下可能出现的问题。
1.UI 管理器无法管理该类 UMG
2.部分 API 无法控制该类 UMG

- 新的 UMG 控件将创建在内容资产下的 UMG 文件夹，点击保存所有保存该资产

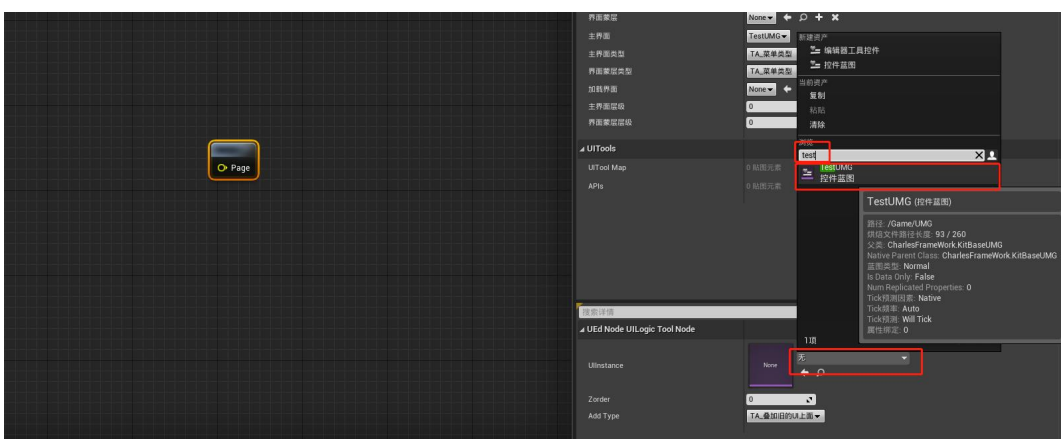
- 回到 UI 管理器界面中，在主菜单项下拉菜单中搜索新创建的 UMG 控件，并选择指定



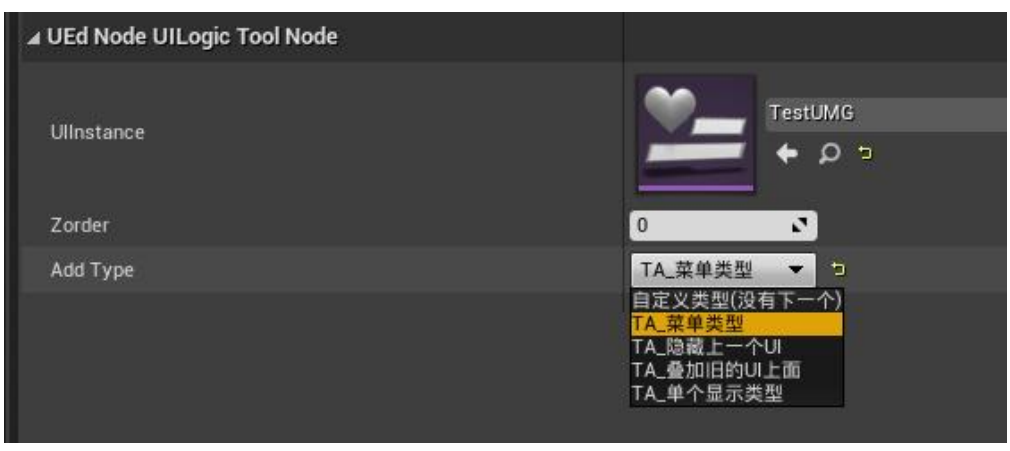
- 指定的 UMG 将作为该项目启动时默认加载界面（入口界面），其他界面交互皆基于该界面
- 右键左侧空白处，在弹出菜单中选择“CreateNewUILogicNode”项，创建一个 UI 管理节点。



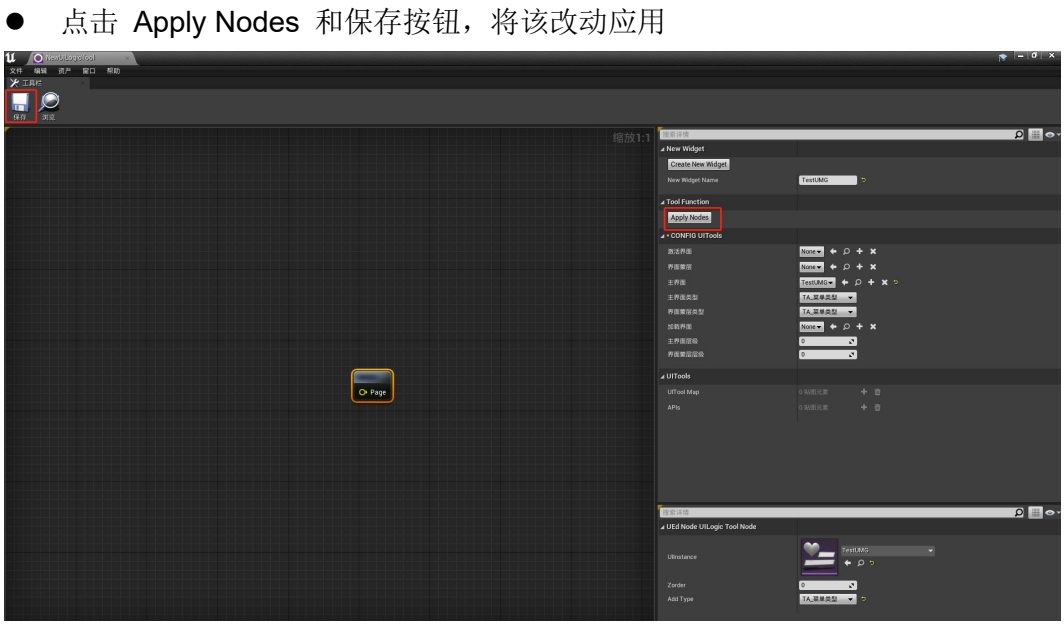
- 在右侧面板 UIInstance 栏下拉菜单中搜索指定的主菜单，并选择指定



- 将 Add Type 指定为“TA_菜单类型”，该类型将会使主菜单界面保持显示。

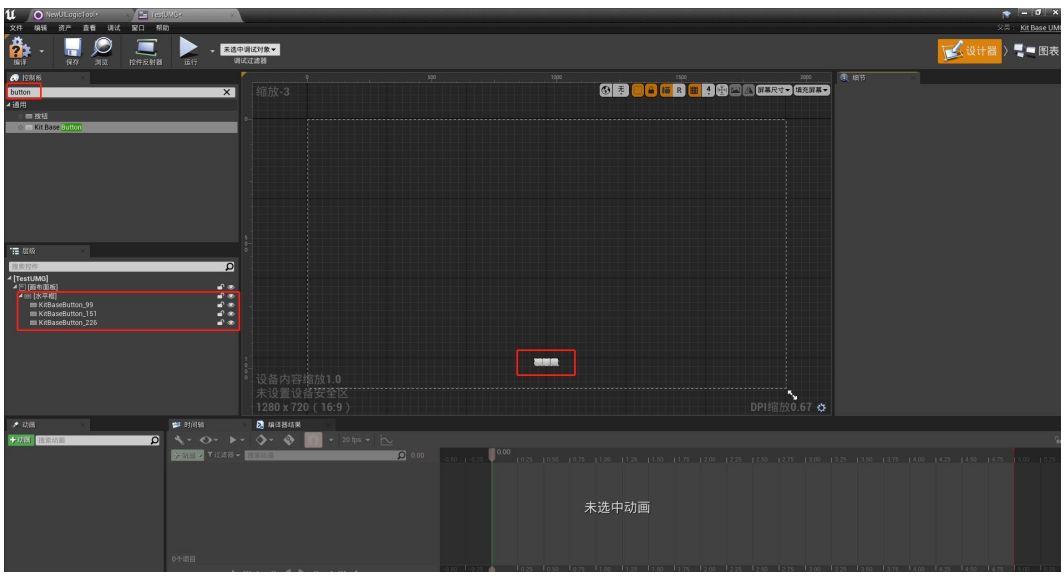


- 点击 Apply Nodes 和保存按钮，将该改动应用



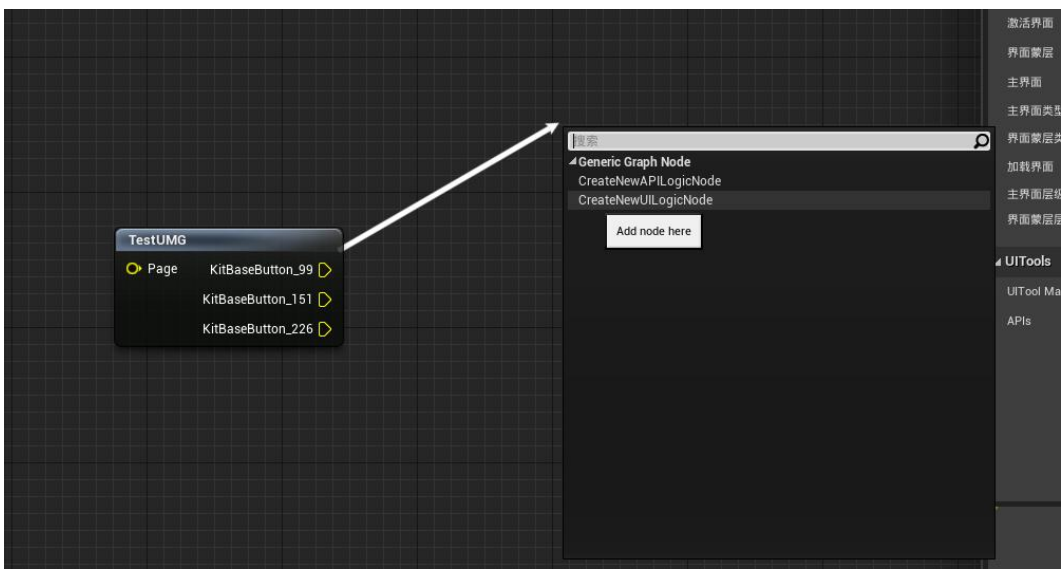
注意：在 UI 管理器中，完成改动后都应该点击 Apply Nodes 与保存按钮，以避免相关改动未应用

- 双击节点进入主菜单控件界面，为该界面添加数个 Kit Base Button 或 Kit Check Group Item，添加完成后点击左上角“编译”并“保存”。



- 回到 UI 管理器中，右键节点并选择“刷新节点”。

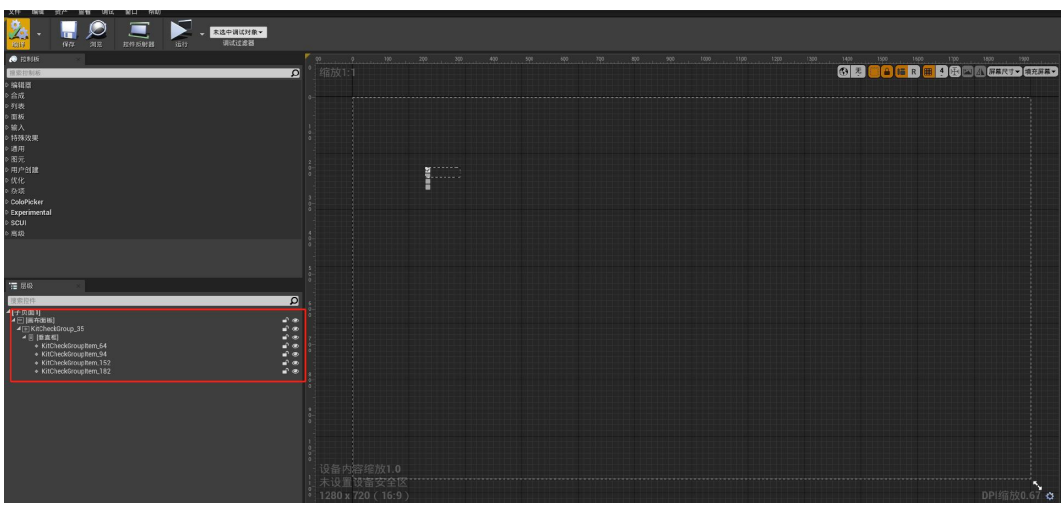
- 此时 UI 节点将显示控件内按钮的输出接口，从接口中拉出箭头并创建下一步节点，由此完成交互逻辑。



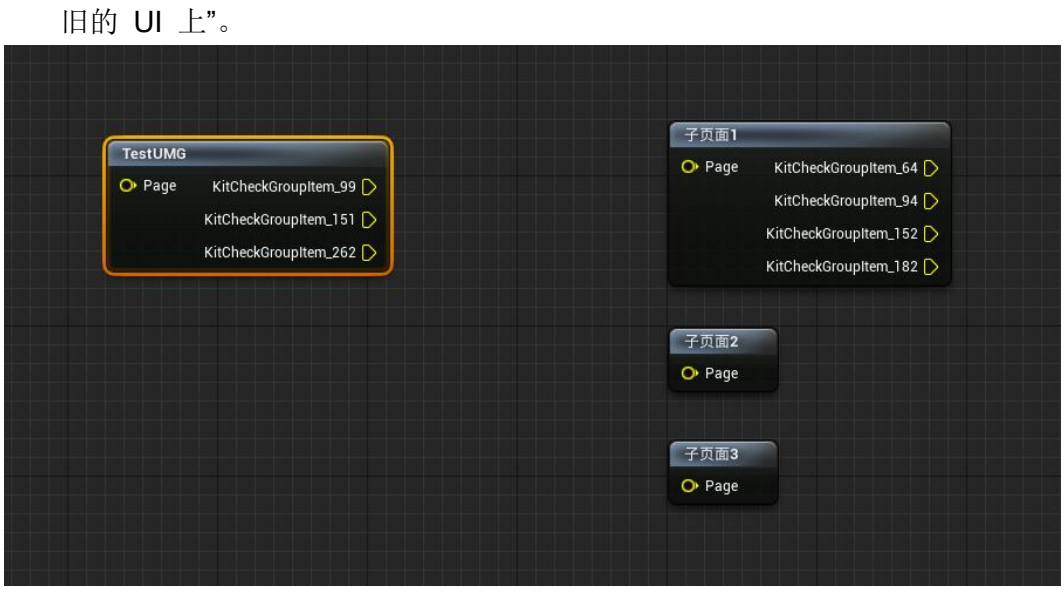
注意：当前版本中，UI 管理器的 UI 节点仅识别 Kit Base Button、Kit Check Group Item 两种类型组件。

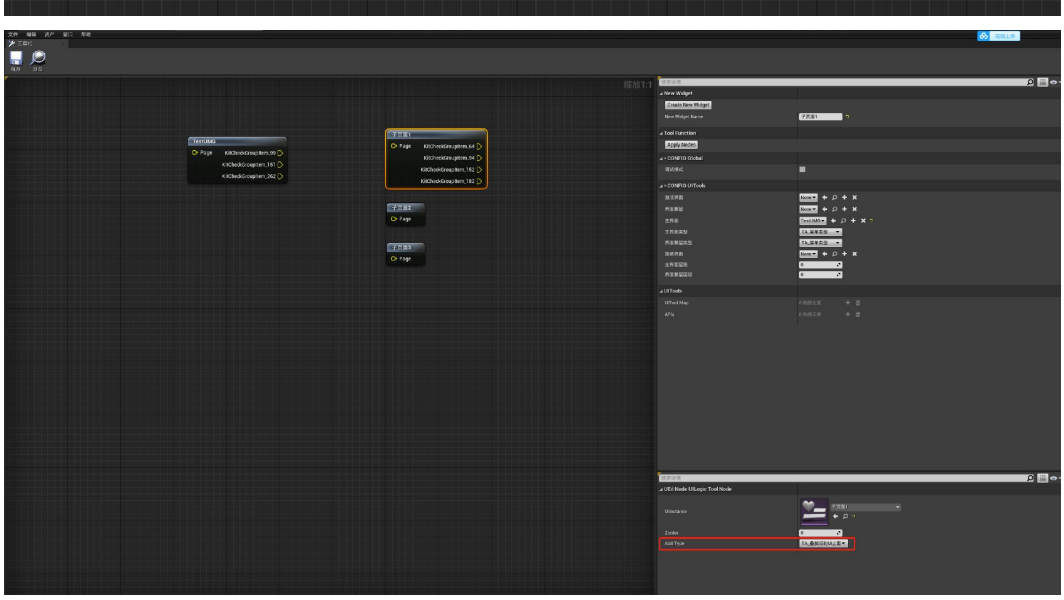
- **多级 UI 嵌套**

- 当有多层级子菜单嵌套时，应当尽量遵守“切换时回退上一级”的准测，下列步骤将演示如何遵守该准测。当熟悉 UI 管理器逻辑后，可根据开发需求自行决定是否需要回退，并灵活设置 Add Type。
- 在 UI 管理器中创建 3 个子页面 UMG,为其中任意一个子页面控件添加任意数量按钮。

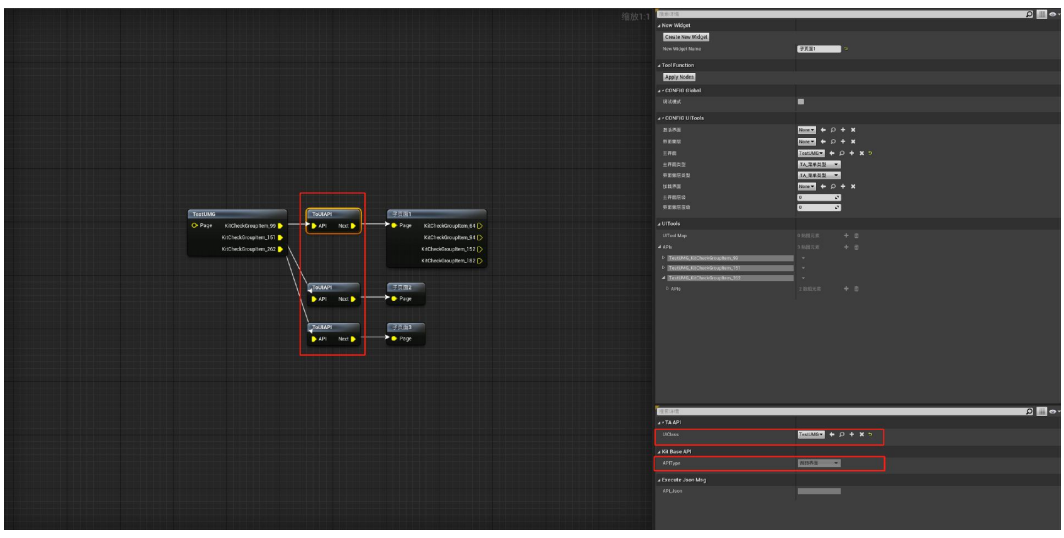


- 在 UI 管理器中添加这 3 个子页面（创建节点后，右键刷新可获取到该 UMG 内的按钮），并在 3 个子页面节点右侧面板中，设置 Add Type 为“TA_叠加在旧的 UI 上”。

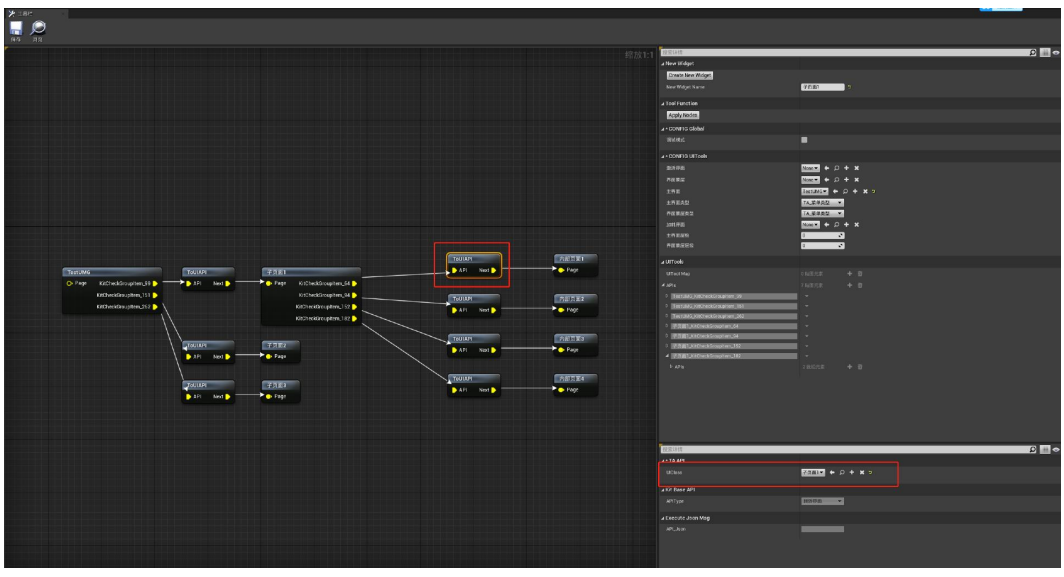




- 由于 3 个子页面均使用叠加类型，此处应在主菜单与子页面之间添加“跳转界面”类型 API，并选择跳回“TestUMG”。此处如不遵守“切换时回退上一级”的原则，3 个子页面将按照点击顺序叠加在主菜单之上。

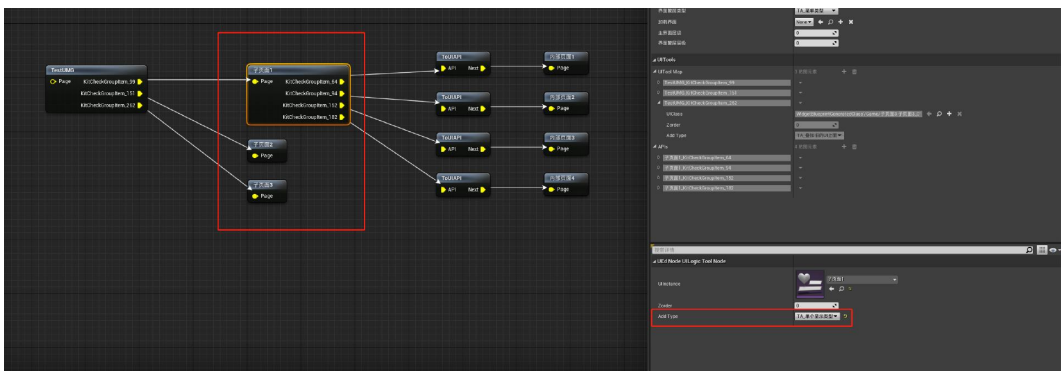


- 依次类推继续增加下一级内部页面，依然将所有内部页面设置为叠加类型，此时内部页面的上一级为“子页面 1”，“跳转界面”应跳转至“子页面 1”

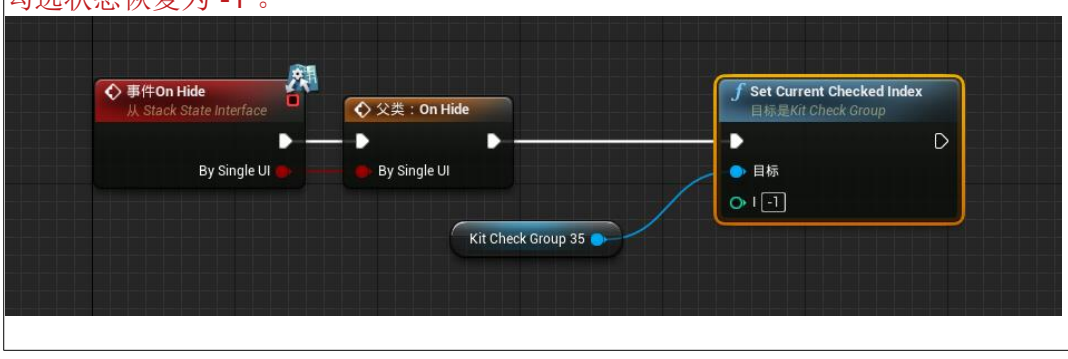


- **灵活拓展**

- 某些开发情况下，可灵活使用 Add Type，如制作一级子页面时，可不进行回退，仅将所有子页面设置为“单个显示”类型即可实现相同的功能逻辑。



注意：当未遵循“切换时回退上一级”时，默认 check 的勾选组将会保持当前勾选状态，故当离开有默认勾选功能的页面时，应当在该页面的 Onhide 中手动将勾选组的勾选状态恢复为“-1”。



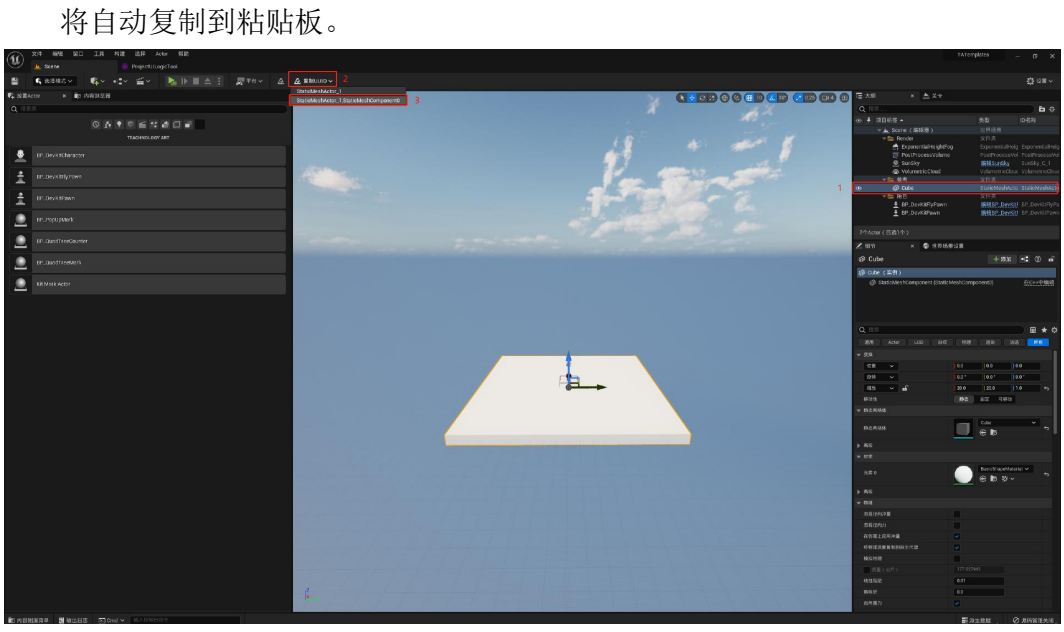
- **场景点击事件图表的使用**

- **基本概述**

在 UILogic 中，除了可以对 UI 控件交互进行管理外，还支持对场景模型、POI 的交互进行管理。在“场景点击事件图表”中，可以通过使用 UUID 对指定场景模型、POI 绑定 API 事件，同时支持绑定到点击、悬停、未悬停事件。

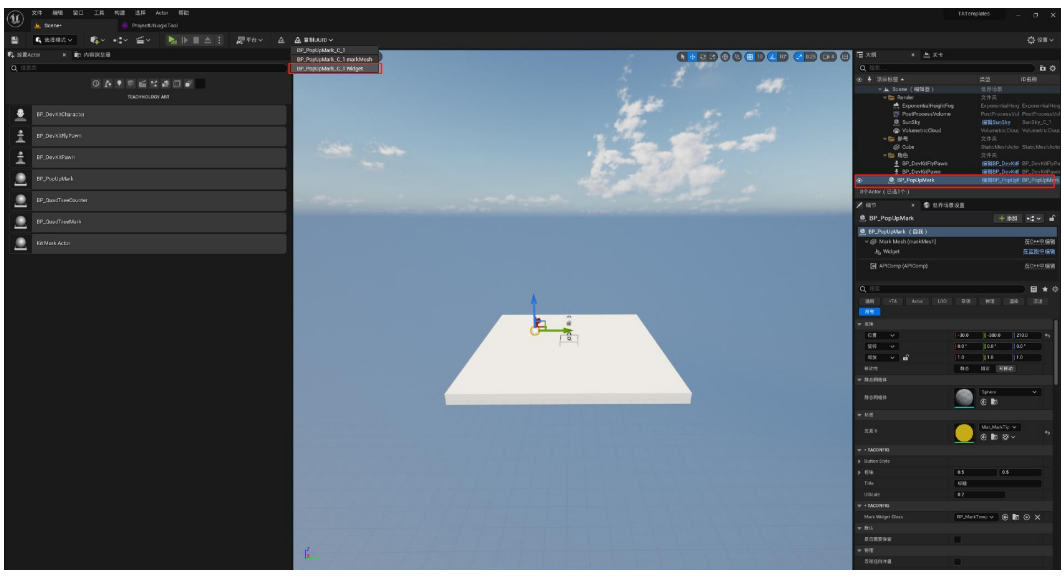
- **获取模型 UUID**

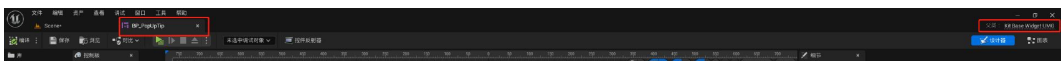
- 点击场景中任意 Actor，点击上方菜单“复制 UUID”项，在下拉菜单中，将自动识别该 Actor 的 Actor ID 及所有模型（static mesh）组件 ID。点选所需的 ID，将自动复制到粘贴板。



- **获取 Widget UUID**

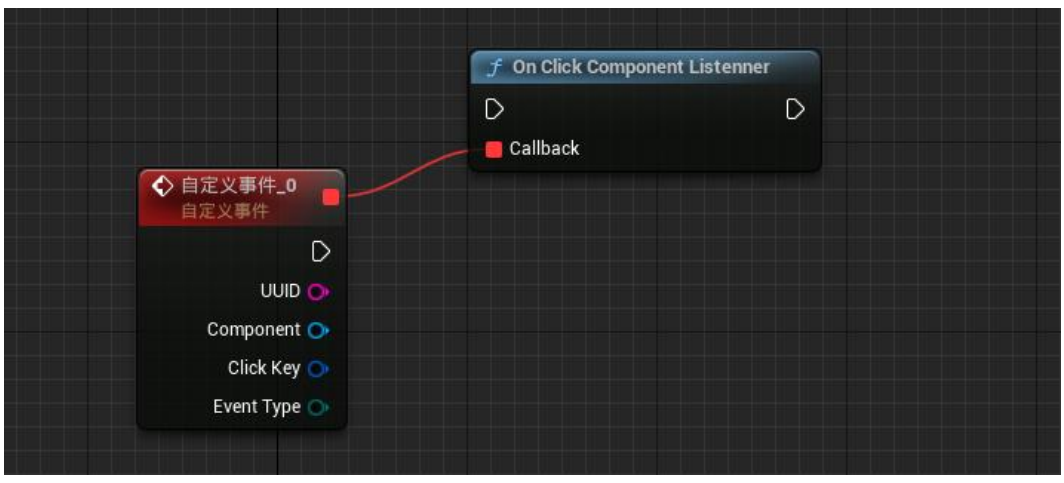
- 点击场景中任意 Actor，点击上方菜单“复制 UUID”项，在下拉菜单中，将自动识别该 Actor 的 Actor ID 及所有父类继承自“kit base widget UMG”的 Widget 组件 ID。点选所需的 ID，将自动复制到粘贴板。



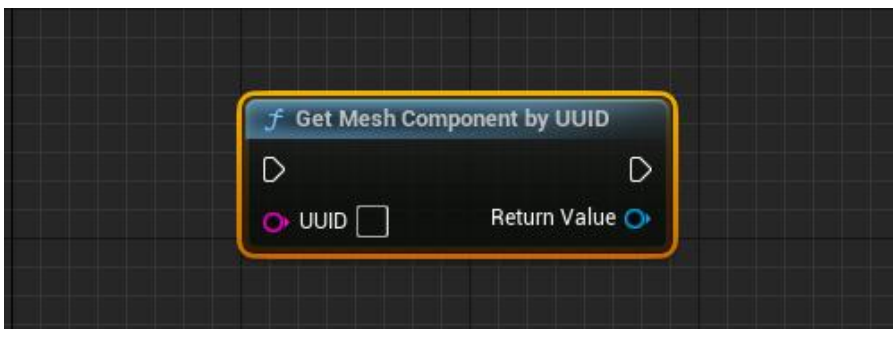


- **UUID 在蓝图中的应用**

- 该事件用于监听鼠标检测所击中的组件

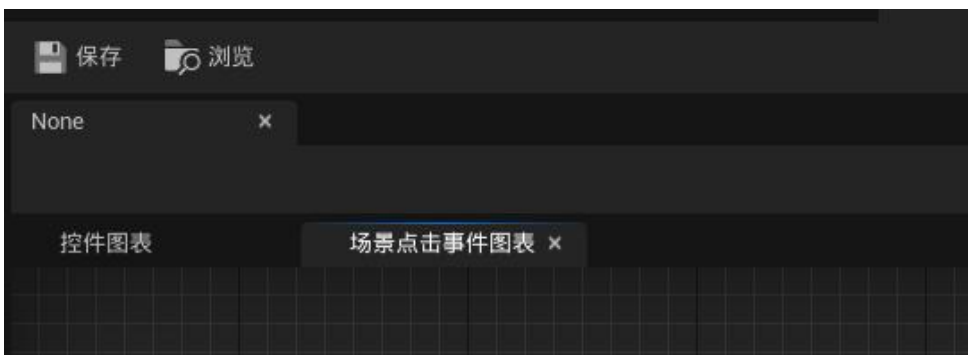


- 可通过 UUID 获取对应 Static Mesh 组件

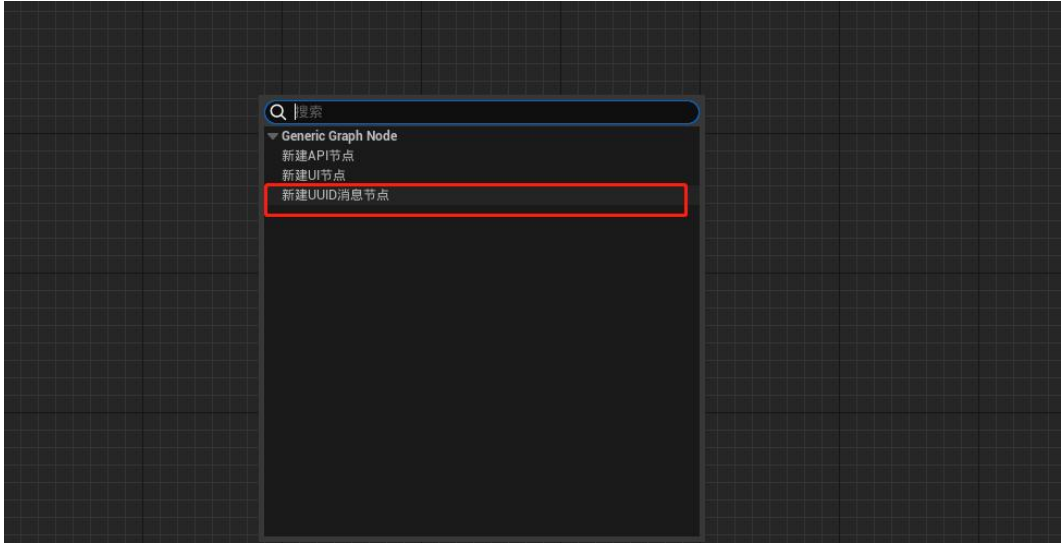


- **在图表中新建 UUID 消息节点**

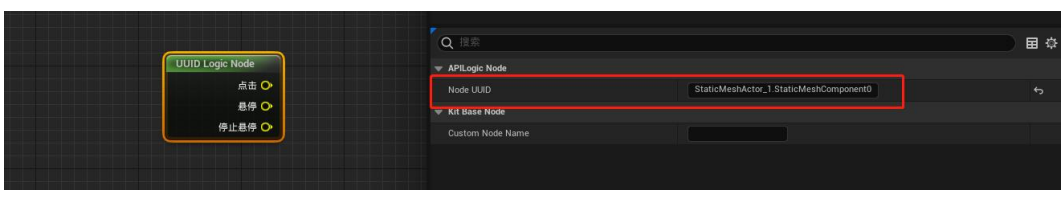
- 打开 UILogic，在选项卡中找到“场景点击事件图表”。



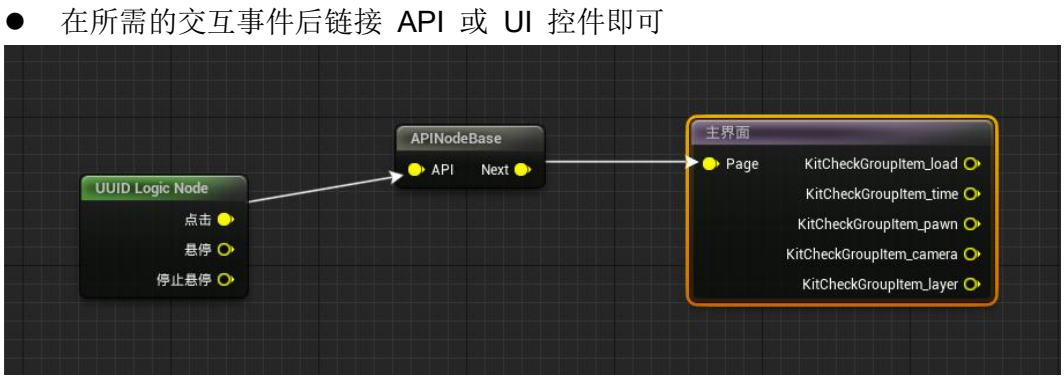
- 在图表中右键菜单中点击“新建 UUID 消息节点”



- UUID 消息节点的参数中，将会自动粘贴最后一个复制的 UUID



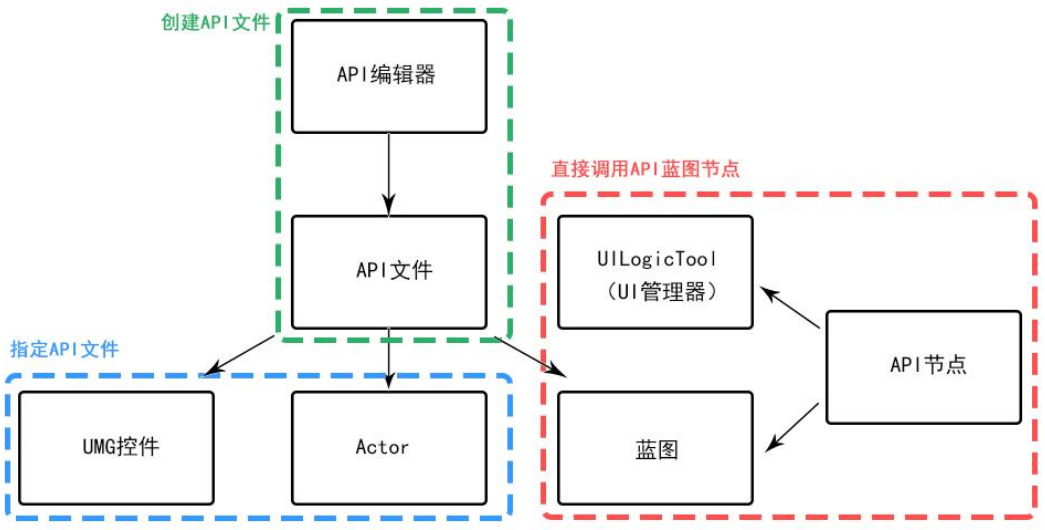
- 在所需的交互事件后链接 API 或 UI 控件即可



TechnologyArtAPIs

● 基本概述

基于常用基础功能进行封装，可在全局及插件拓展的管理器中快速调用，该类封装功能称之为“API”(以下简称 API)。API 拥有基本参数设置，也可对其封装功能进行二次拓展。

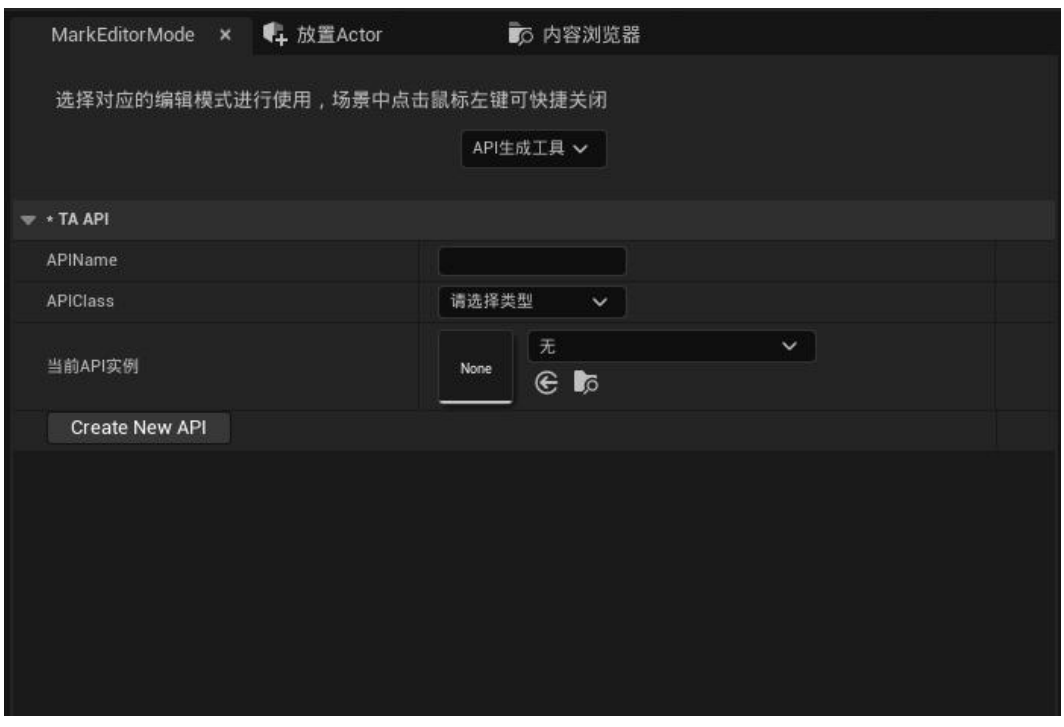


（API 结构图）

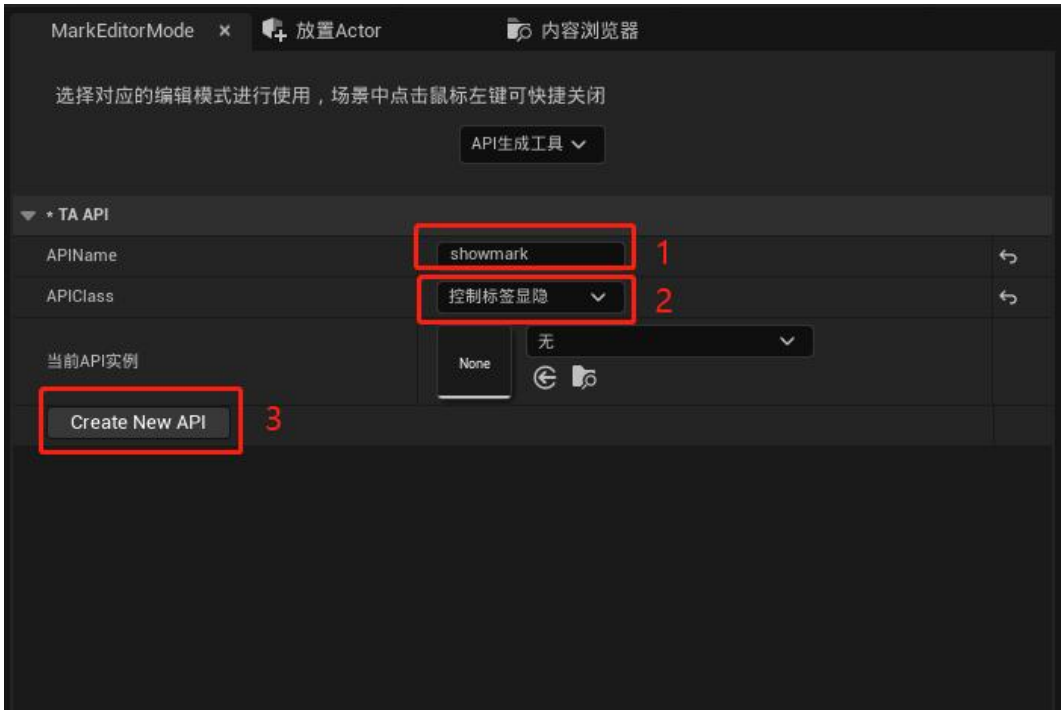
● API 创建与使用

● API 编辑器创建 API 文件

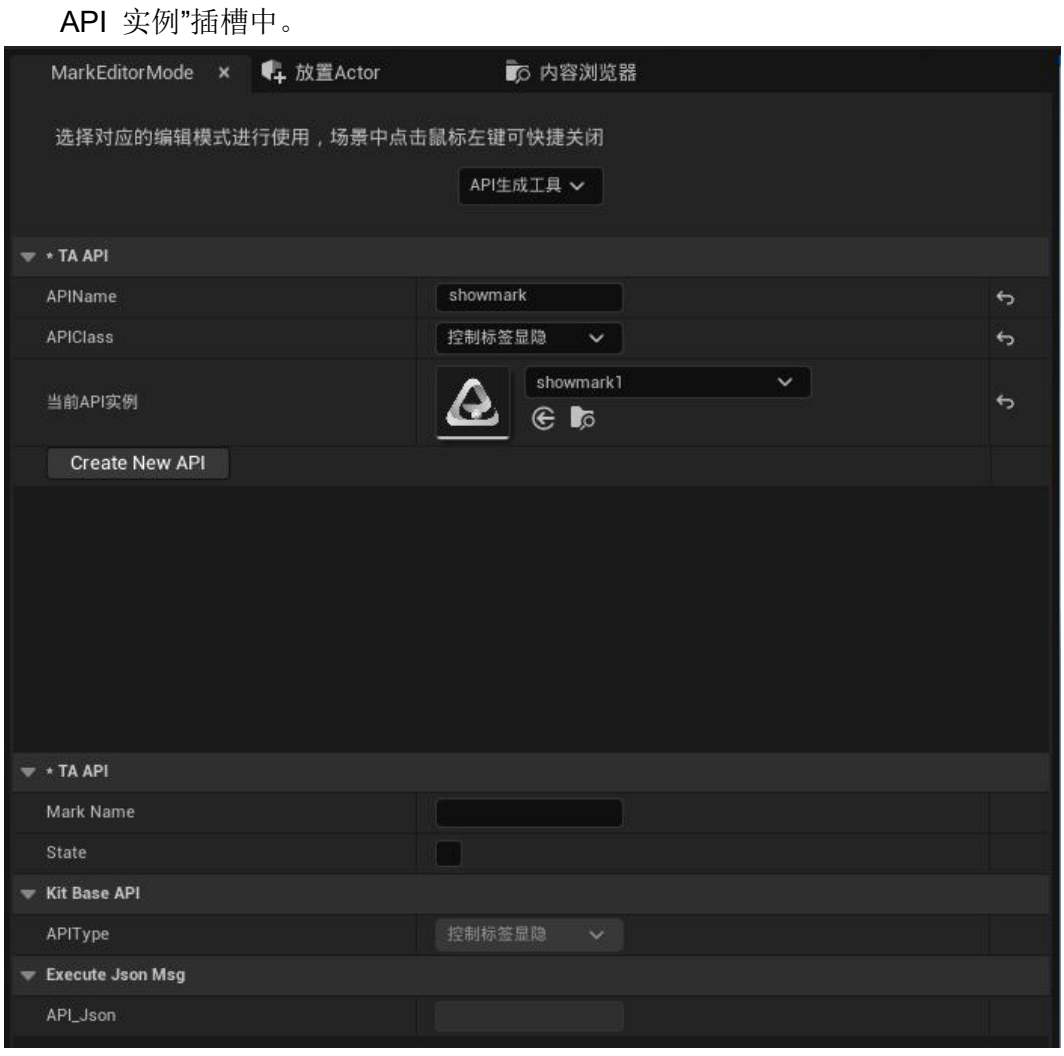
- 点击上方菜单“模式”的下拉箭头，勾选“MakeEditorMode”项，在弹出的面板中点击“API 生成工具”按钮开启 API 编辑器



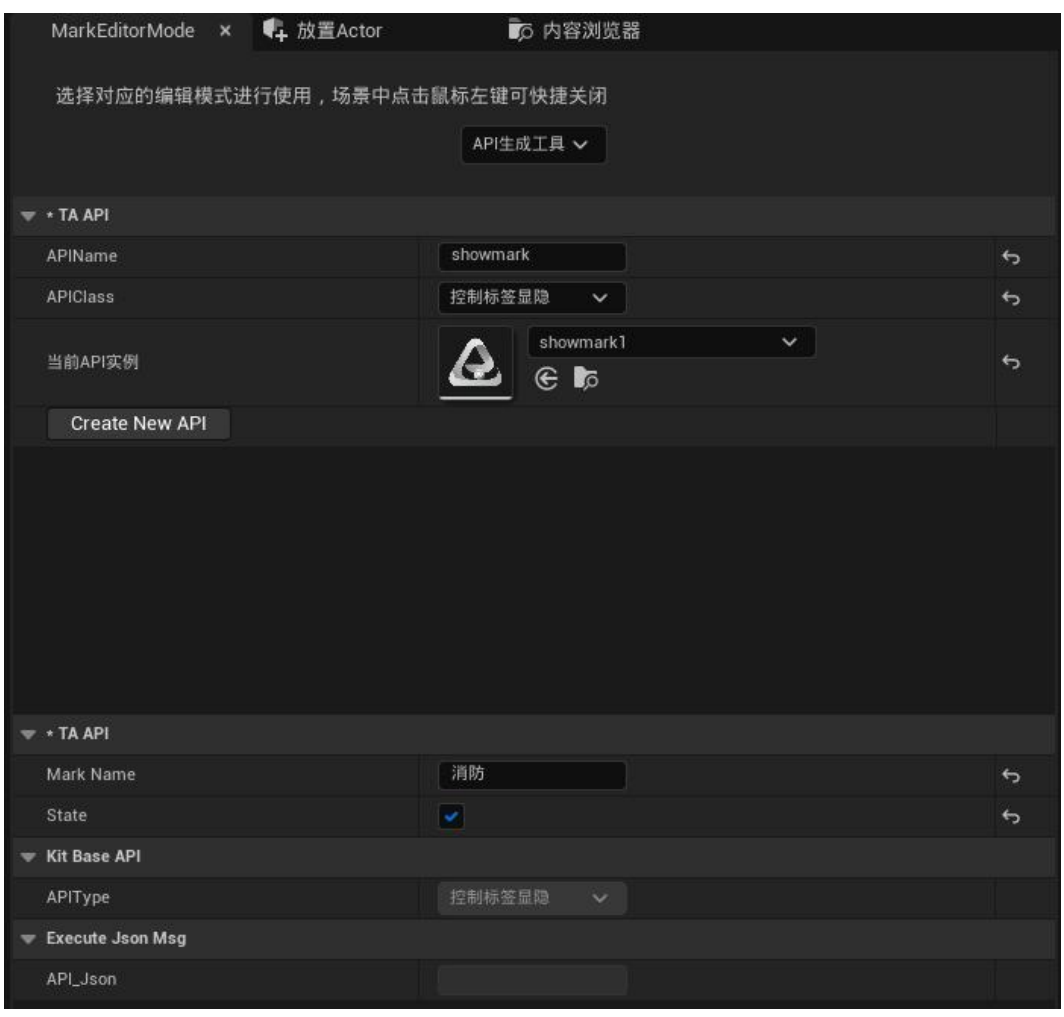
- 为创建的 API 填写一个名称，点击 APIClass 项的下拉箭头为该 API 选择一个需要的类型，完成选择后点击“Create New API”按钮



- 此时在文件夹中将会自动创建一个新的 API 文件，并自动将该 API 填入“当前 API 实例”插槽中。



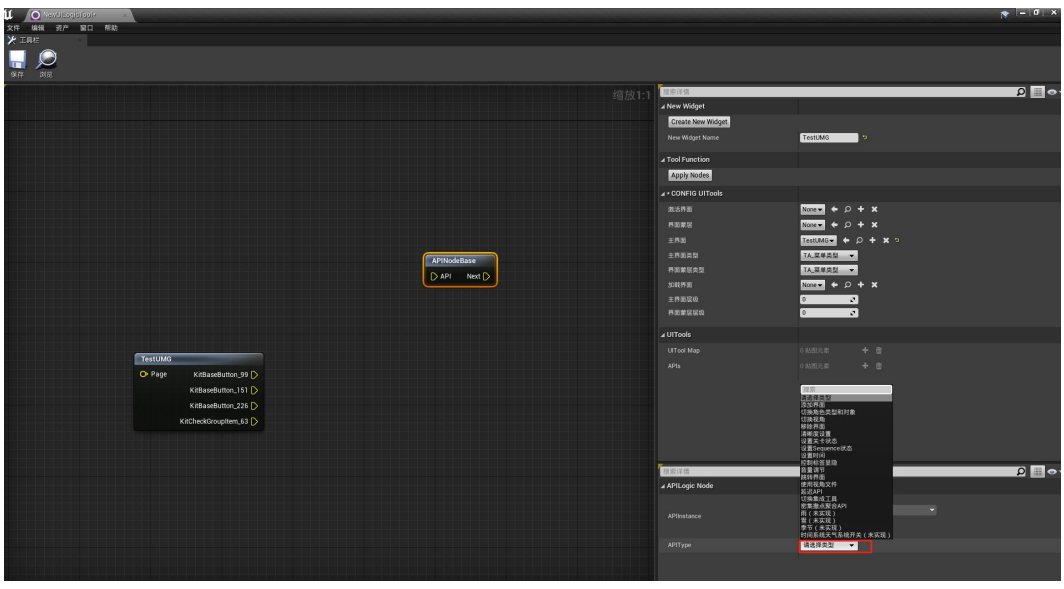
- 此时在 API 编辑器中出现该 API 可修改参数，按需调整即可完成 API 创建



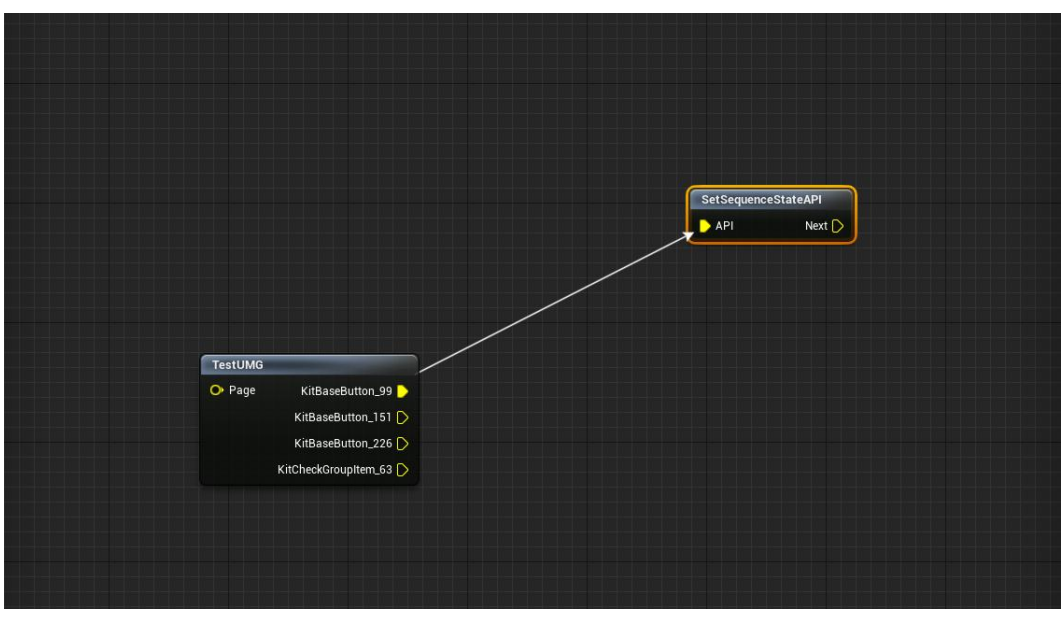
● 在 UI 管理器中创建 API 文件

- 点击进入项目使用的 UI 管理器中，在空白处单击右键，在弹出菜单中选择“CreateNewAPILogicNode”，创建一个空白的 API 调用节点。

- 在右侧面板中选择一个需要的 APIType，即可创建一个新的 API 文件。



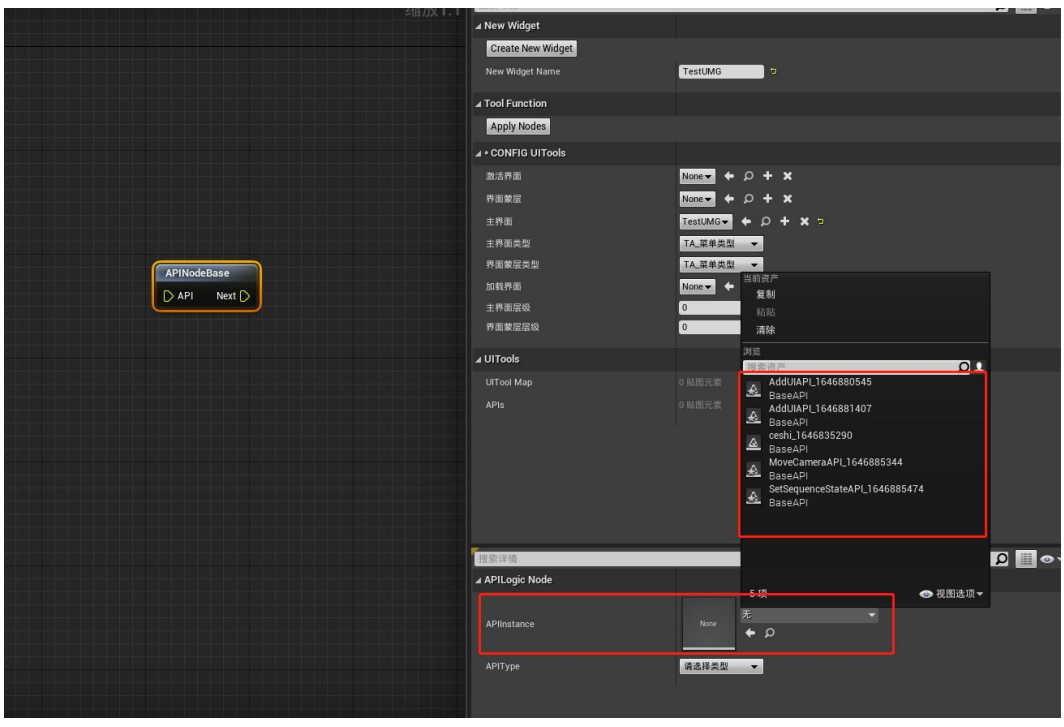
- 通过与 UI 节点的按钮输出链接，可使程序按照链接顺序执行 API 或加载新的界面



- **在 UI 管理器中使用 API**

- 点击进入项目使用的 UI 管理器中，在空白处单击右键，在弹出菜单中选择“CreateNewAPILogicNode”，创建一个空白的 API 调用节点。

- 在右侧面板“API Instance”插槽中指定需要使用的 API 文件。

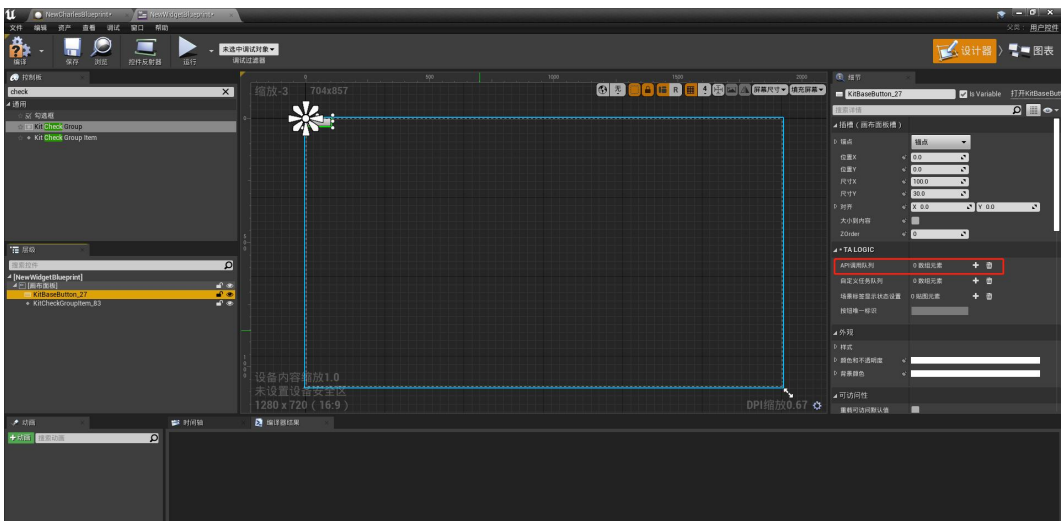


- 或选择 API Type 类型进行新建 API

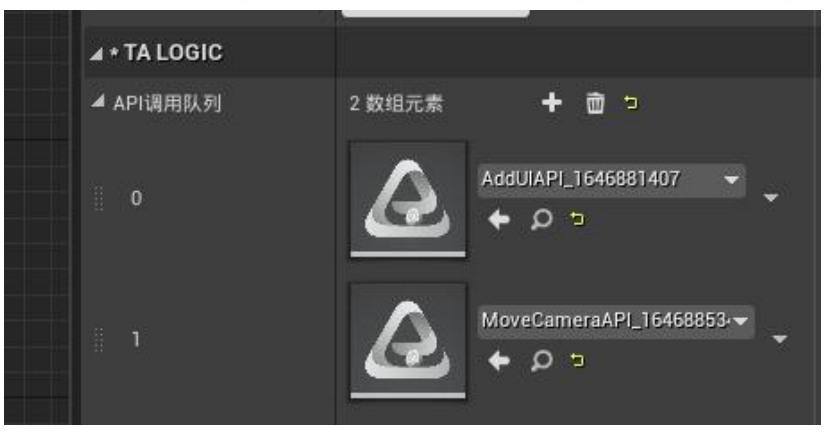
- **在 UMG 控件内使用 API**

- 在控件界面中，添加一个 Kit Base Button 和 Kit Check Group Item

- 点击 Kit Base Button 或 Kit Check Group Item，在右侧细节面板中找到“API 调用队列”项，点击右侧“+”按键，向队列添加一个 API 文件插槽，如需调用多个 API 文件，则添加多个 API 文件插槽。



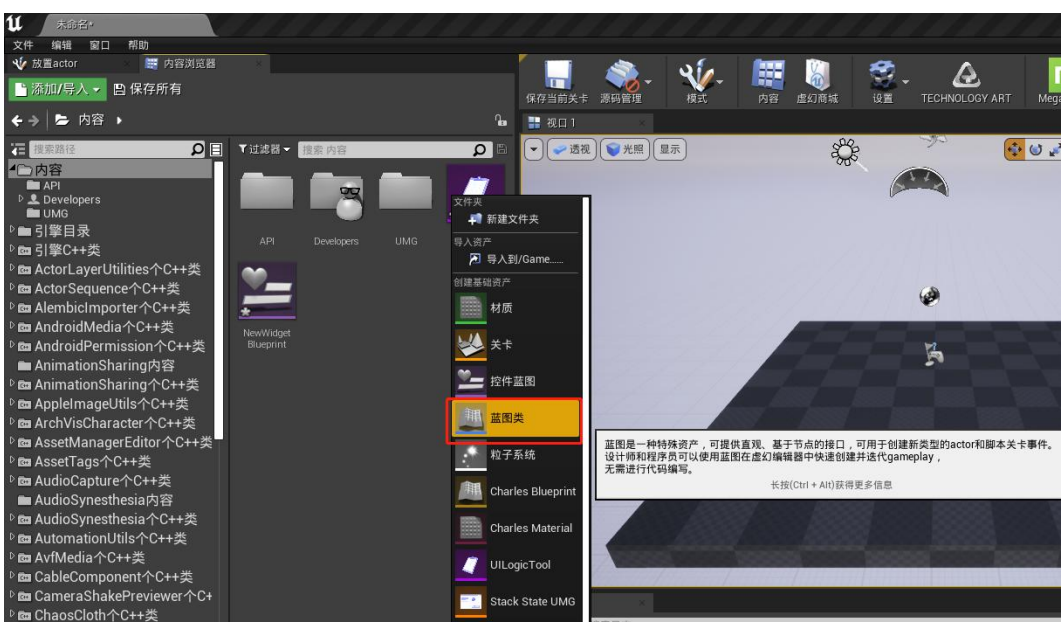
- 指定 API 文件后，在程序运行中点击该按钮将由上至下执行 API 逻辑。



- 同样在按钮右侧细节面板中，可快速调用 ShowMarkAPI，无需创建新的 API 文件。点击“+”按键添加一个 Mark 名称（Actor Tag）及状态插槽。在程序运行中点击该按钮将执行该 ShowMarkAPI。

● 在插件拓展的 Actor 中使用 API

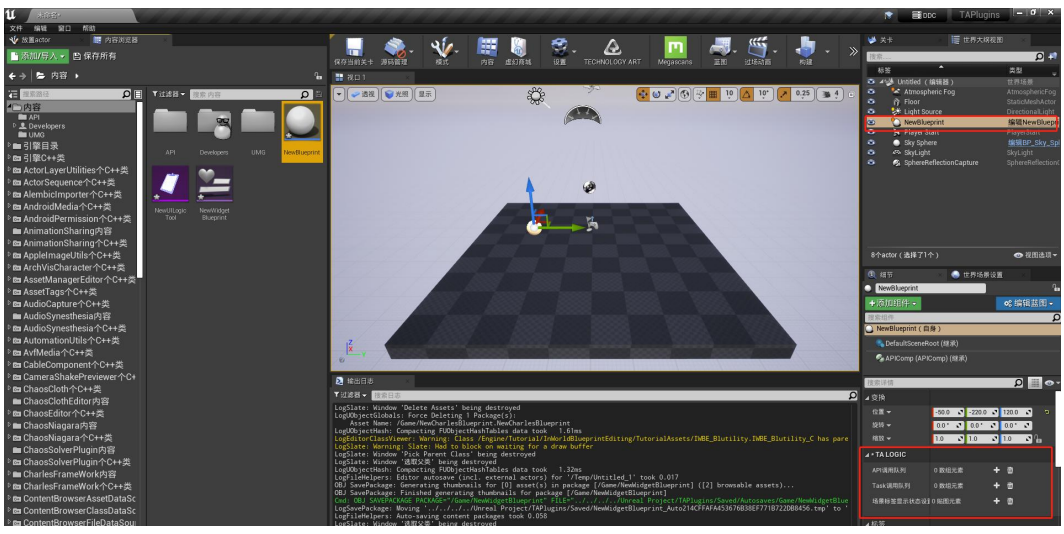
- 在资产管理器右键菜单中选择创建蓝图类。



- 在弹出窗口中搜索“KitBaseActor”，并选择创建该类

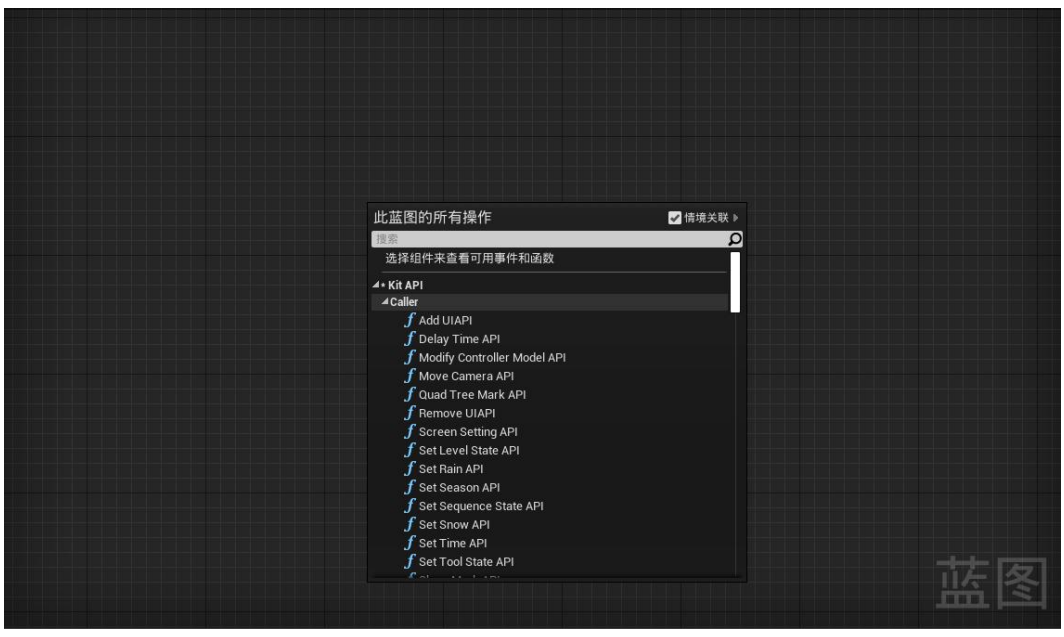
- 将创建的 Actor 拖拽到场景中

- 在世界大纲中选中该 Actor，将在下方细节面板看到与控件界面中相同的 API 插槽设置。插槽中指定 API 文件后，程序将在点击到该 Actor 下挂载的任意碰撞后执行。



- 在蓝图中使用 **API**

- 在全局所有蓝图中均可在右键菜单中找到 Kit API 下的 Caller 项，可直接调用所有 API

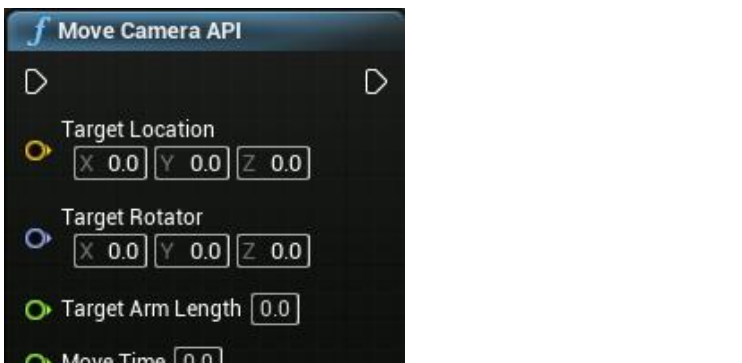


- 在全局所有蓝图中均可在右键菜单中找到以下函数

Execute API：执行单个 API 文件

Execute API：执行 API 队列

Execute Tasks：执行自定义任务队列



- **API** 功能与参数说明

- 角色控制类

- 切换角色类型和对象 ModifyControllerModelAPI

API 说明：切换角色类型，并且可以根据 Key 值来指定相应的角色对象，设置 Time 值以改变相机运动速度。

参数名	参数描述	参数类型
Controller Model	填写需要切换的角色类型	鸟瞰角色
		飞行角色
		漫游角色
Location Name	填写对应角色上的 Key 值，如为空则切换到场景中最后生成的对应类型的角色上	填写角色 KeyID
Time	填写切换到该角色时，镜头所运动的时间，如为 0，则角色瞬间切换	填写数值，单位为秒

前端调用代码示例

```
function ModifyControllerModel(ControllerModel, LocationName,Time) {
    let obj = {
        action: "ModifyControllerModel",
        requestId: 10, //开发者自行分配
        params: {
            controllerModel: ControllerModel,
            locationName: LocationName,
            time:Time
        }
    }
    ue4("ExecuteAPIs", obj,TAAPICallbackCommon);
}
```

- 切换视角 MoveCameraAPI

API 说明：根据指定的 PawnID 移动角色。

参数名	参数描述	参数类型
Target Location	目标视角位置	填写坐标
Target Rotator	目标视角旋转量	填写旋转度数
Target Arm Length	目标弹簧臂长度（切换到非鸟瞰角色是忽略此参数）	填写长度值
Move Time	填写切换到该角色时，镜头所运动的时间，如为 0，则角色瞬间切换	填写数值，单位为秒

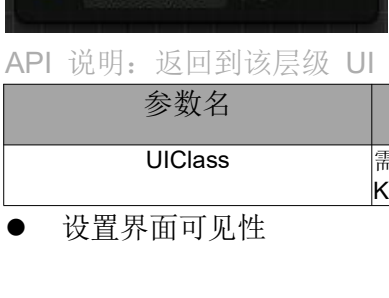
前端调用代码示例

```
function MoveCamera(TargetLocation, TargetRotator, TargetArmLenght, MoveTime,PawnId) {
    let obj = {
        action: "MoveCamera",
        requestId: 10, //开发者自行分配
        params: {
            targetLocation: TargetLocation,
            targetRotator: TargetRotator,
            targetArmLenght: TargetArmLenght,
            moveTime: MoveTime,
            pawnId:PawnId
        }
    }
    ue4("ExecuteAPIs", obj,TAAPICallbackCommon);
}
```

前端调用代码示例

```
function GetCameraView() {
    let obj = {
        action: "GetCameraView",
        requestId: 10, //开发者自行分配
        params: {
           //无
        }
    }
    ue4("ExecuteAPIs", obj,TAAPICallbackCommon);
}
```

- 使用视角文件 UseCameraViewAPI



API 说明：根据指定的视角文件切换视角

参数名	参数描述	参数类型
Camera View	使用视角文件	指定视角文件
Change Pawn	是否切换到创建视角文件时对应的 Pawn，否时将当前角色移动到视角文件对应的坐标上	是/否

- **界面控制类**

- 添加界面 AddUIAPI

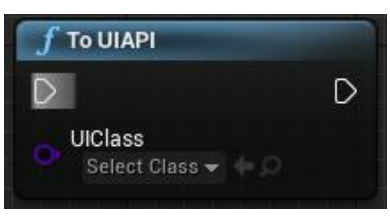
API 说明：根据指定的 UI 类型以及层级根据某种方式添加 UI 到视口

参数名	参数描述	参数类型
UIClass	需要添加的 UMG 类（需继承自 Kit Base UMG)	指定 UMG 控件文件
Add Type	自定义控件 Show/Hide	自定义类型
	特殊类型，不接受之后添加的 UI 的隐藏（包括单个显示类型），只能通过 RemoveUI 移除	**TA_菜单类型**
	添加后隐藏上一个 UI	**TA_隐藏上一个 UI**
	添加后叠加在旧的 UI 上面	**TA_叠加旧的 UI 上面**
	添加后隐藏除该 UI 的所有 UI	**TA_单个显示类型**
Zorder	UI 层级	填写层级数，数值越大，显示优先级越高

- 移除界面 RemoveUIAPI

API 说明：移除当前视口的 UI，并回到上层

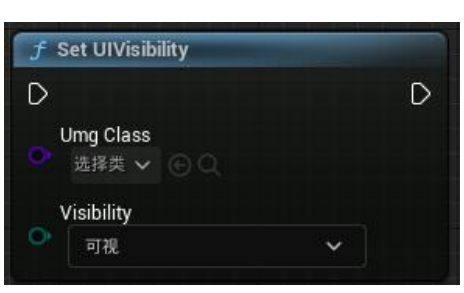
- 跳转界面 ToUIAPI



API 说明：返回到该层级 UI（需要在已创建当前类型 UI 的情况下生效）

参数名	参数描述	参数类型
UIClass	需要添加的 UMG 类（需继承自 Kit Base UMG)	指定 UMG 控件文件

- 设置界面可见性



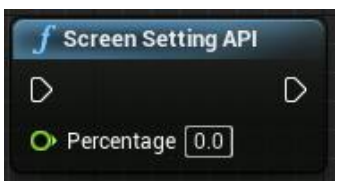
API 说明：设置指定控件可见性，如隐藏指定控件，需在显示时手动设置可见性，否则将保持可见性为隐藏

参数名	参数描述	参数类型

UMG Class	目标 UMG	填写继承自 kit base umg 的控件类
Visibility	可见性	可见类型

- **参数设置类**

- 清晰度设置 ScreenSettingAPI



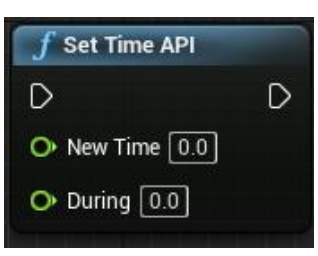
API 说明：设置清晰度（屏幕百分比）

参数名	参数描述	参数类型
Percentage	清晰度百分比，默认值为 100	填写数值

前端调用代码示例

```
function ScreenSetting(Percentage) {
    let obj = {
        action: "ScreenSetting",
        requestId: 10, //开发者自行分配
        params: {
            percentage: Percentage
        }
    }
    ue4("ExecuteAPIs", obj,TAAPICallbackCommon);
}
```

- 设置时间 SetTimeAPI



API 说明：基于 SunSky 的时间设置，需要场景中使用 SunSky

参数名	参数描述	参数类型
New Time	目标时间	填写数值（0-24）
During	过渡时间	填写数值，单位为秒

前端调用代码示例

```
function SetTime(NewTime, During) {
    let obj = {
        action: "SetTime",
        requestId: 10, //开发者自行分配
        params: {
            newTime: NewTime,
            during: During
        }
    }
    ue4("ExecuteAPIs", obj,TAAPICallbackCommon);
}
```

- 音量调节 SoundSettingAPI

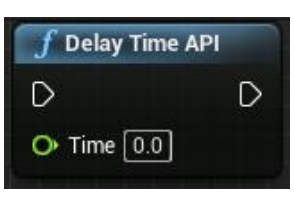
API 说明：设置音量

参数名	参数描述	参数类型
Volume	目标音量，默认值为 1	填写数值

前端调用代码示例

```
function SoundSetting(Volume) {
    let obj = {
        action: "SoundSetting",
        requestId: 10, //开发者自行分配
        params: {
            volume: Volume
        }
    }
    ue4("ExecuteAPIs", obj,TAAPICallbackCommon);
}
```

- 延迟 API DelayTimeAPI



API 说明：延迟，与其他的 API 配合使用，可以隔开两个 API 的执行时间或者延迟执行某个 API

参数名	参数描述	参数类型
Time	延迟时间	填写数值，单位为秒

- **场景控制类**

- 设置关卡状态 SetLevelStateAPI

API 说明：设置关卡状态，设置可见性和加载卸载，加载状态下 Visiable 才生效。

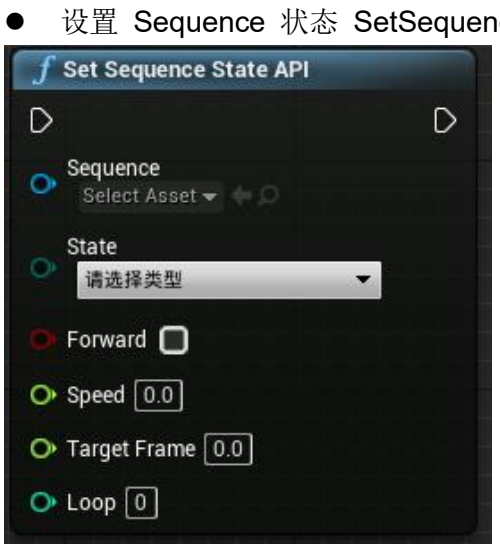
参数名	参数描述	参数类型
Level Names	关卡名称，可填写多个关卡名称，使用";"（英文分号）隔开	填写关卡名称
Visiable	可见状态，需要先 Load 关卡	可见/不可见

Load	加载状态	加载/卸载

前端调用代码示例

```
function SetLevelState(LevelNames, Visiable, Load) {
    let obj = {
        action: "SetLevelState",
        requestId: 10, //开发者自行分配
        params: {
            levelNames: LevelNames,
            visiable: Visiable,
            load: Load
        }
    }
    ue4("ExecuteAPIs", obj,TAAPICallbackCommon);
}
```

- 设置 Sequence 状态 SetSequenceStateAPI



API 说明：设置 Sequence 状态。

参数名	参数描述	参数类型
Sequence	指定需要控制的 Sequence 文件	指定 Sequence 文件
State	Sequence 状态操作	播放、暂停、停止、播放到目标帧数
Forward	播放方向	加载/卸载
Speed	播放速度，默认值为 1	填写速度
Target Frame	当 State 选择为播放到目标帧数时，在此填写目标帧数	填写目标帧数
Loop	循环播放次数，0 为无限循环	填写次数

- 控制标签显隐 ShowMarkAPI

API 说明：设置场景中标签的显示隐藏

参数名	参数描述	参数类型
Mark Name	填写 Actor 下 Tag Name 以控制所有拥有相同 Name 的 Actor，可填写多个 Name，使用“;"（英文分号）隔开	填写 Tag Name
State	标签显示状态	显示/隐藏

前端调用代码示例

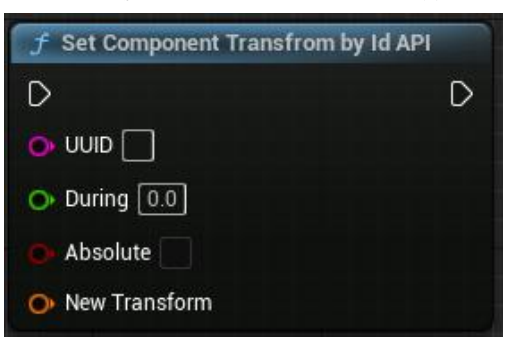
```
function ShowMark(MarkName, State) {
    let obj = {
        action: "ShowMark",
        userId: sessionStorage.getItem("userId"),
        params: {
            markName: MarkName,
            state: State
        }
    }
    ue4("ExecuteAPIs",obj,FunctionCallbackCommon);
    ue4("ExecuteAPIs", obj,TAAPICallbackCommon);
}
```

- 设置模型（Static Mesh）组件资源

API 说明：通过指定 UUID 切换该模型组件引用的资产

参数名	参数描述	参数类型
UUID	场景中 Actor、static mesh 组件、widget 组件的 ID	填写 Static Mesh 组件的 ID
Mesh	指定切换的模型资产	模型资产

- 设置模型（Static Mesh）组件变换



API 说明：通过指定 UUID 变换该组件的位置、旋转、缩放，可控制过渡时间、相对坐标、世界坐标

参数名	参数描述	参数类型
UUID	场景中 Actor、static mesh 组件、widget 组件的 ID	填写 Static Mesh 组件的 ID
During	过渡时间	填写数值，单位为秒
Absolute	变换坐标系	绝对坐标/相对坐标
New Transform	目标变换参数值	变换的目标参数

前端调用代码示例

```
function SetComponentTransfromById(UUID,During,Absolute,NewTransform) {
    let obj = {
```

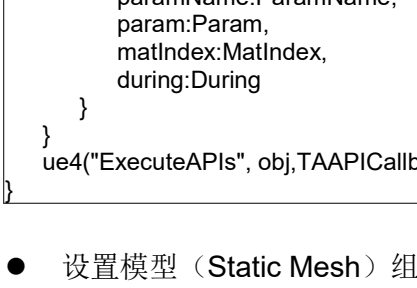
```
        action: "SetComponentTransfromById",
        requestId: 10, //开发者自行分配
        params: {
            UUID: UUID,
            during:During,
            absolute:Absolute,
            newTransform:NewTransform,
        }
    }
    ue4("ExecuteAPIs", obj,TAAPICallbackCommon);
}
```

- 设置模型（Static Mesh）组件材质

API 说明：通过指定 UUID 变换该组件的材质

参数名	参数描述	参数类型
UUID	场景中 Actor、static mesh 组件、widget 组件的 ID	填写 Static Mesh 组件的 ID
Mat	指定切换材质资产	指定材质资产
Mat Index	指定材质 ID	设置 ID 为-1 时，将为所有材质 ID 切换指定材质

- 设置模型（Static Mesh）组件可见性



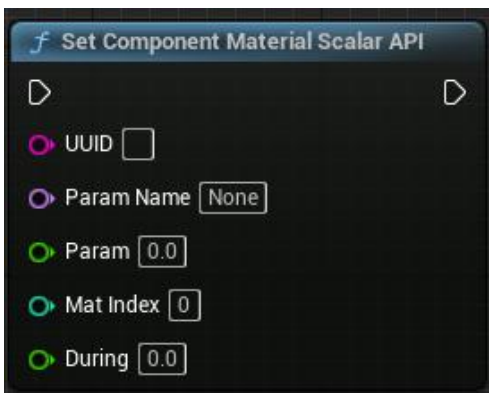
API 说明：通过指定 UUID 切换该组件的可见性

参数名	参数描述	参数类型
UUID	场景中 Actor、static mesh 组件、widget 组件的 ID	填写 Static Mesh 组件的 ID
Visable	设置可见性	可见/不可见

前端调用代码示例

```
function SetComponentVisible(UUID,Visible) {
    let obj = {
        action: "SetComponentVisible",
        requestId: 10, //开发者自行分配
        params: {
            UUID: UUID,
            visible:Visible
        }
    }
    ue4("ExecuteAPIs", obj,TAAPICallbackCommon);
}
```

- 设置模型（Static Mesh）组件材质标量参数



API 说明：通过指定 UUID 切换该组件指定材质 ID 的材质标量参数

参数名	参数描述	参数类型
UUID	场景中 Actor、static mesh 组件、widget 组件的 ID	填写 Static Mesh 组件的 ID
Param Name	指定参数名称	填写材质中参数化标量的参数名
Param	标量参数目标值	填写材质中参数化标量的目标参数
Mat Index	目标材质的材质 ID	填写目标材质在模型组件中的材质 ID
During	参数变化的过渡时间	填写数值，单位为秒

前端调用代码示例

```
function SetComponentMaterialScalar(UUID,ParamName,Param,MatIndex,During) {
    let obj = {
        action: "SetComponentMaterialScalar",
        requestId: 10, //开发者自行分配
        params: {
            UUID: UUID,
            paramName:ParamName,
            param:Param,
            matIndex:MatIndex,
            during:During
        }
    }
    ue4("ExecuteAPIs", obj,TAAPICallbackCommon);
}
```

- 设置模型（Static Mesh）组件材质向量参数

API 说明：通过指定 UUID 切换该组件指定材质 ID 的材质向量参数

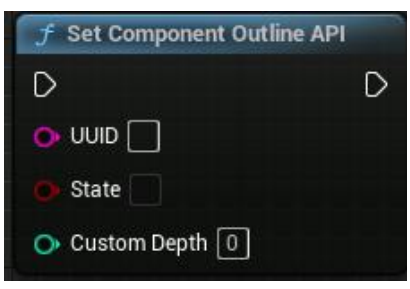
参数名	参数描述	参数类型
UUID	场景中 Actor、static mesh 组件、widget 组件的 ID	填写 Static Mesh 组件的 ID
Param Name	指定参数名称	填写材质中参数化向量的参数名
Param	标量参数目标值	填写材质中参数化向量的目标参数
Mat Index	目标材质的材质 ID	填写目标材质在模型组件中的材质 ID

During	参数变化的过渡时间	填写数值，单位为秒

前端调用代码示例

```
function SetComponentMaterialColor(UUID,ParamName,Param,MatIndex,During) {
    let obj = {
        action: "SetComponentMaterialColor",
        requestId: 10, //开发者自行分配
        params: {
            UUID: UUID,
            paramName:ParamName,
            param:Param,
            matIndex:MatIndex,
            during:During
        }
    }
    ue4("ExecuteAPIs", obj,TAAPICallbackCommon);
}
```

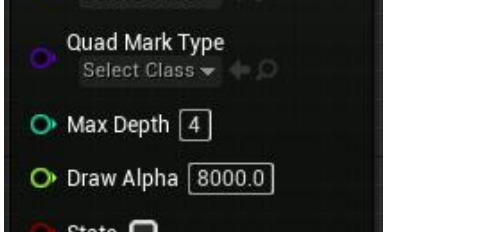
- 设置模型（Static Mesh）组件外描边



API 说明：通过指定 UUID 开启/关闭模型组件的外描边，可设置自定义深度通道

参数名	参数描述	参数类型
UUID	场景中 Actor、static mesh 组件、widget 组件的 ID	填写 Static Mesh 组件的 ID
State	开启/关闭外描边	开启/关闭
Custom Depth	指定自定义深度通道（需在项目设置中开启自定义模板）	当通道值为-1 时，则采用模型组件上设置的自定义深度通道值

- 关闭所有模型（Static Mesh）组件外描边



API 说明：关闭所有模型组件外描边

- **工具控制类**

- 切换集成工具 SetToolStateAPI

API 说明：设置集成工具状态

参数名	参数描述	参数类型
ToolType	选择需要开启的工具类型	选择需要开的工具

前端调用代码示例

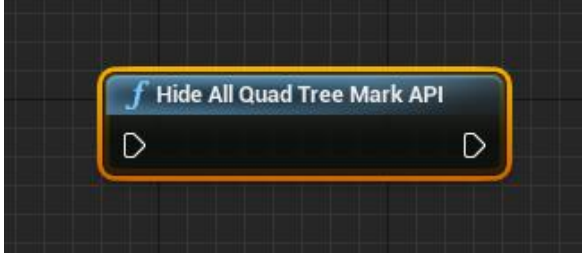
```
function SetToolState(ToolType) {
    let obj = {
        action: "SetToolState",
        requestId: 10, //开发者自行分配
        params: {
            toolType: ToolType
        }
    }
    ue4("ExecuteAPIs", obj,TAAPICallbackCommon);
}
```

- 密集撒点聚合 API QuadTreeMarkAPI

API 说明：设置撒点密集聚合标签的开关

参数名	参数描述	参数类型
Show Debug	是否绘制调试网格	是/否
Quad Counter Type	指定聚合标签的样式类型，默认会挂载一个 Widget 显示数量，界面和模型样式可自定义拓展，详见文档-插件预设类	指定 Counter 类
Quad Mark Type	指定聚合标签的撒点类，用于定位标签位置，可自行定义聚合标签生成方式，详见文档-插件预设类	指定 Mark 类
Max Depth	深度，细分程度，越大分越细	填写深度数值
Draw Alpha	每层的划分步长，越大越密集	填写步长数值
State	当前指定聚合标签类显示状态	显示/隐藏

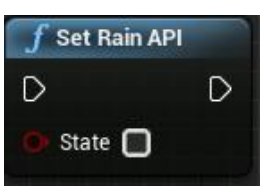
- 隐藏所有密集撒点聚合 API Hide All QuadTreeMarkAPI



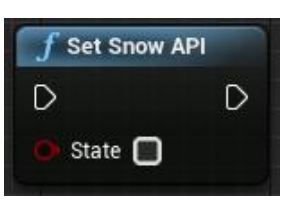
> API 说明：隐藏所有撒点密集聚合标签

- **未实现**

- 雨（未实现） SetRainAPI

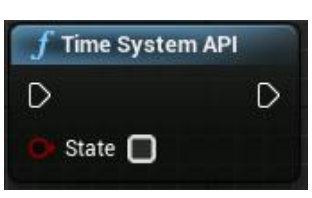


- 雪（未实现） SetSnowAPI



- 季节（未实现） SetSeasonAPI

- 时间系统天气系统开关（未实现） TimeSystemAPI

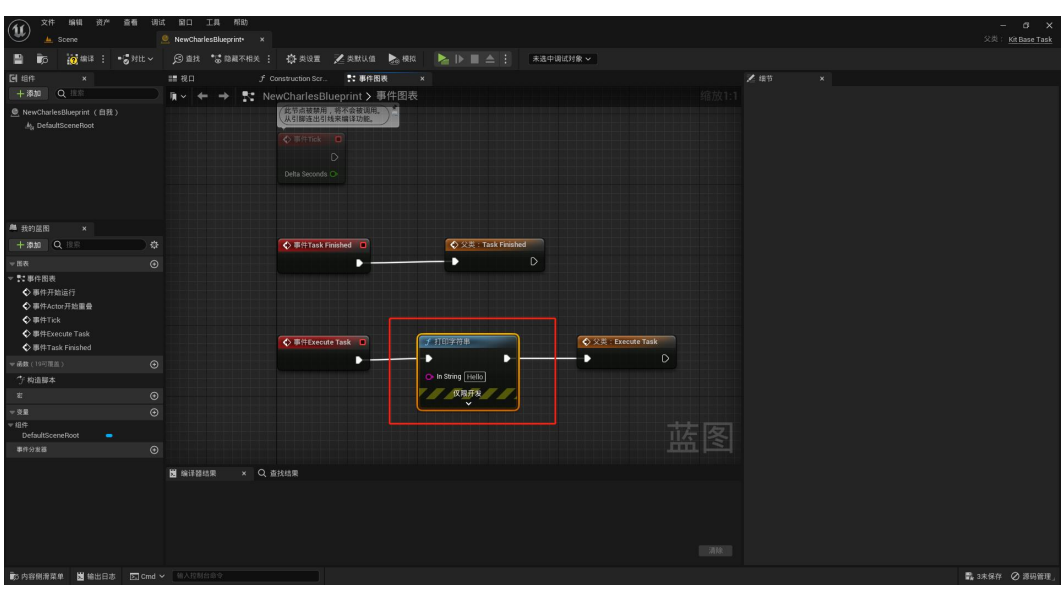


● 如何在管理器中调用自定义 API、函数

- **自定义任务队列（Task）**

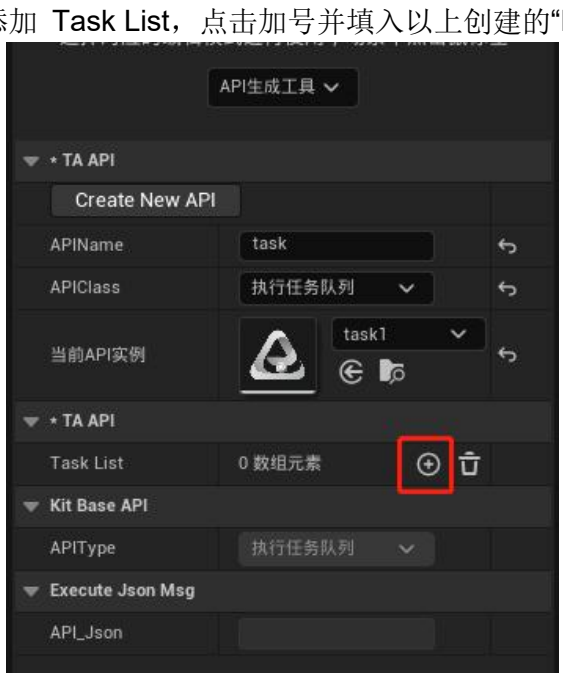
- 在管理器中，预置 API 并不足以完成所有复杂逻辑，故引入自定义任务队列，自定义队列可快速在管理器、UMG、Actor 中使用。
- 创建“KitBaseTask”

- TaskActor 中由“Task Finished”、“Execute Task”事件组成，在 Execute Task 事件后，可自定义逻辑。



- 在 API 生成工具中或在管理器中创建“执行任务队列”API。

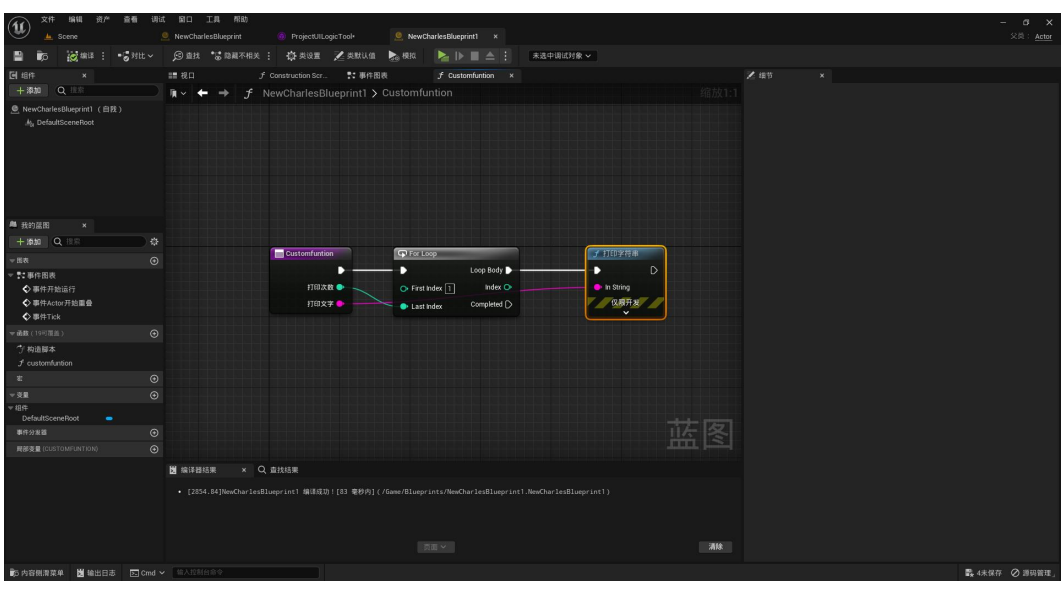
- 在 API 参数中添加 Task List，点击加号并填入以上创建的“KitBaseTask”。



- 此处 Task List 可填入多个 KitBaseTask 或同一个 KitBaseTask 填入多次，Task 事件将按索引次序执行。
- API 文件可快速在管理器、UMG、场景 Actor 中使用。

- **调用自定义函数与传参**

- 创建“Actor”，创建自定义函数。



- 在 API 生成工具中或在管理器中创建“指定 UUID 调用 Actor 自定义方法”API。

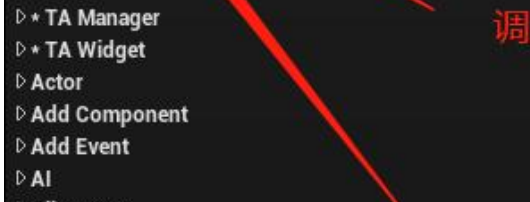
- 将自定义 Actor 拖入场景中并复制其 UUID 填入 API 参数中，“Function Name”为所需调用的函数，“Args”中添加需传入的参数，其索引对应自定义函数中的参数索引。

- API 文件可快速在管理器、UMG、场景 Actor 中使用。

TechnologyArt-Implementer

基本概述

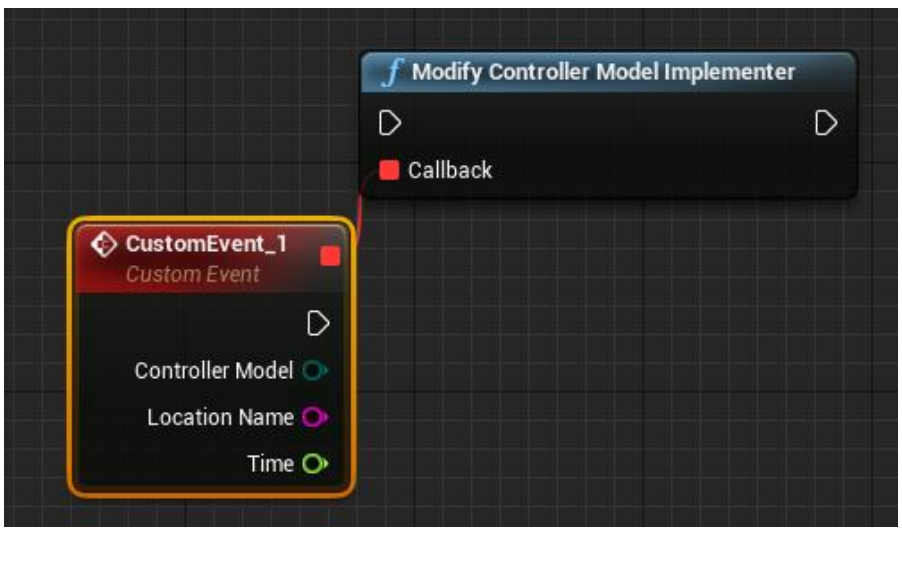
每个 API 带有一个实现器，方便二次开发使用，比如切换角色 API 的实现器



使用方式

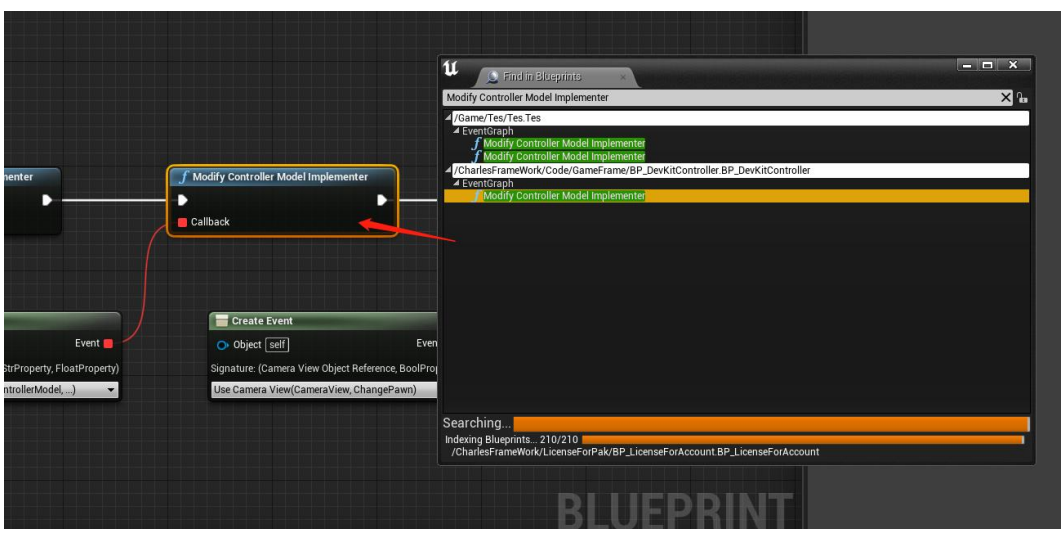
叠加实现

可以利用，绑定自定义事件来增加 API 的实现，会在原有基础上进行叠加：



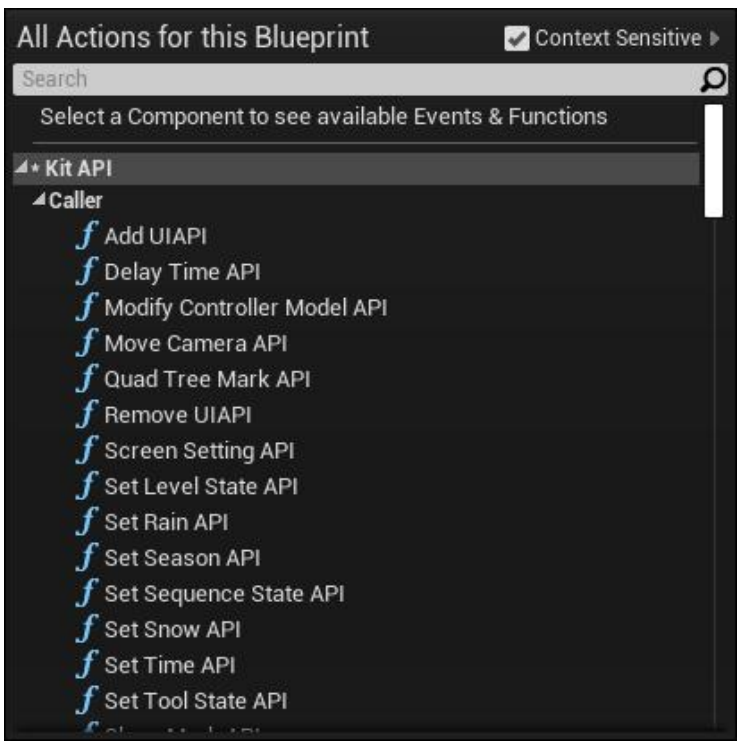
重写（覆盖）实现

如果需要重写原先的 API 实现 可以全局搜索该实现器，即可跳转到原先的实现

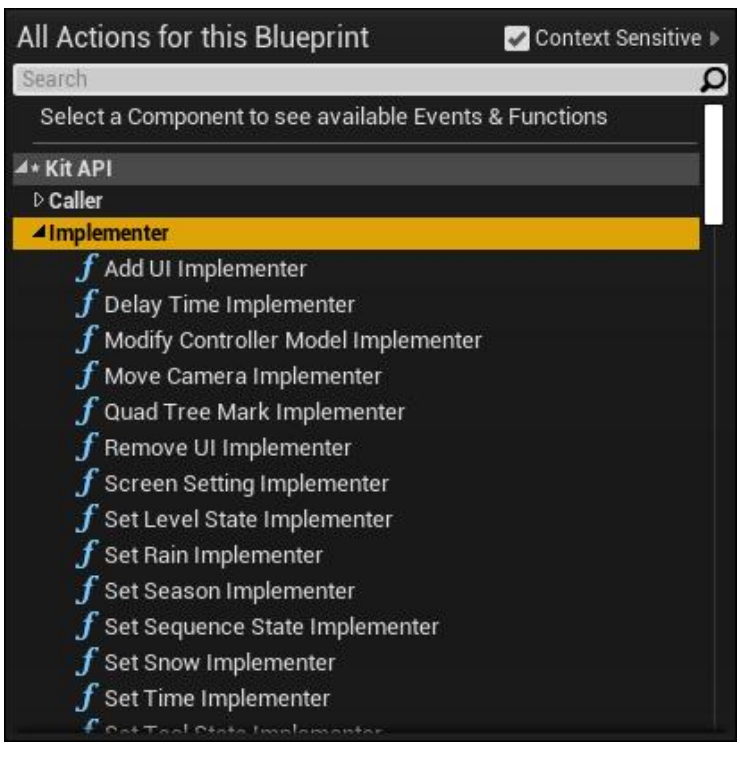


在蓝图列表中可以轻松获取到 API 的调用器和实现器，他们是一一对应的：

调用器：



实现器：

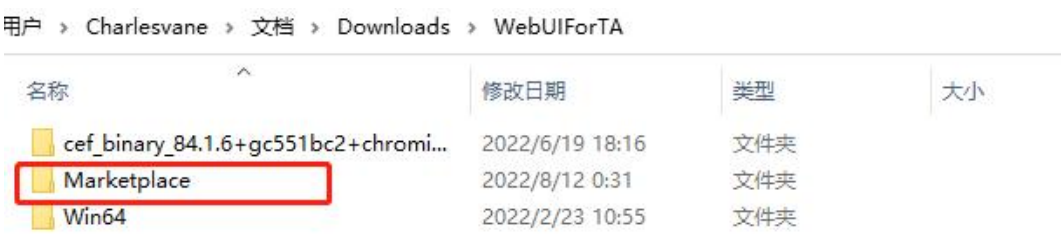


TechnologyArt-WebUI 拓展

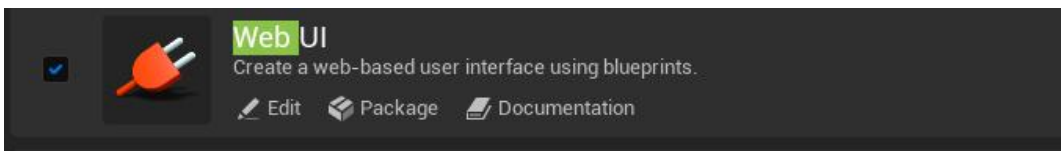
- 下载页 https://www.tartcloud.cn/download
- 支持插件版本 V1.0.6 以上的版本

V1.0.6↑ - 插件WebUI拓展	5.0.X	

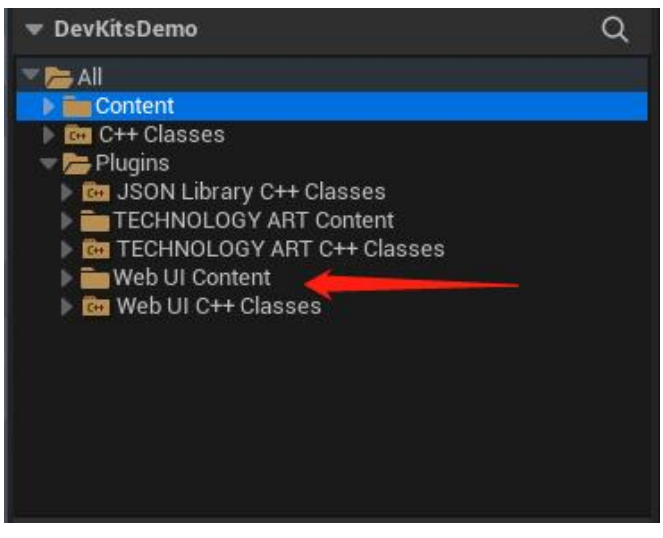
- 解压完成后有以下内容
- 将 Marketplace 拷贝到 UE 引擎根目录的对应文件夹下即可按照成功



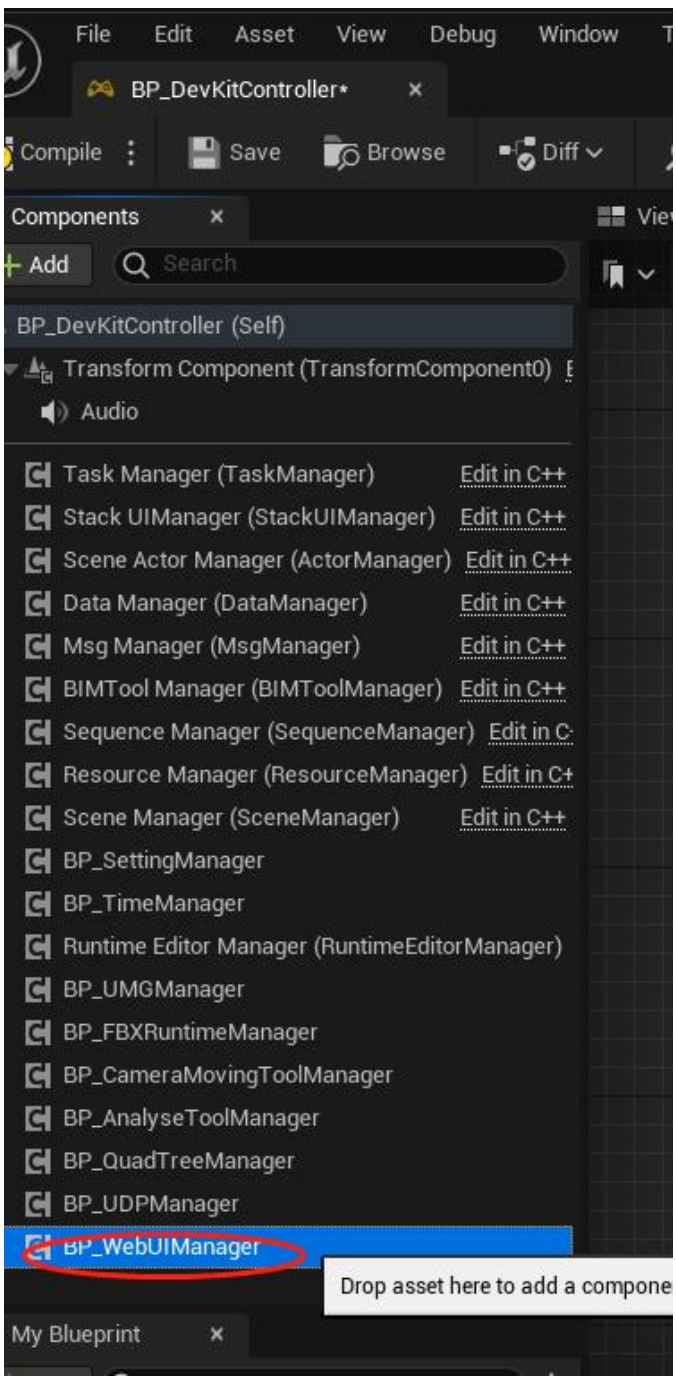
- 打开引擎打开对应的插件

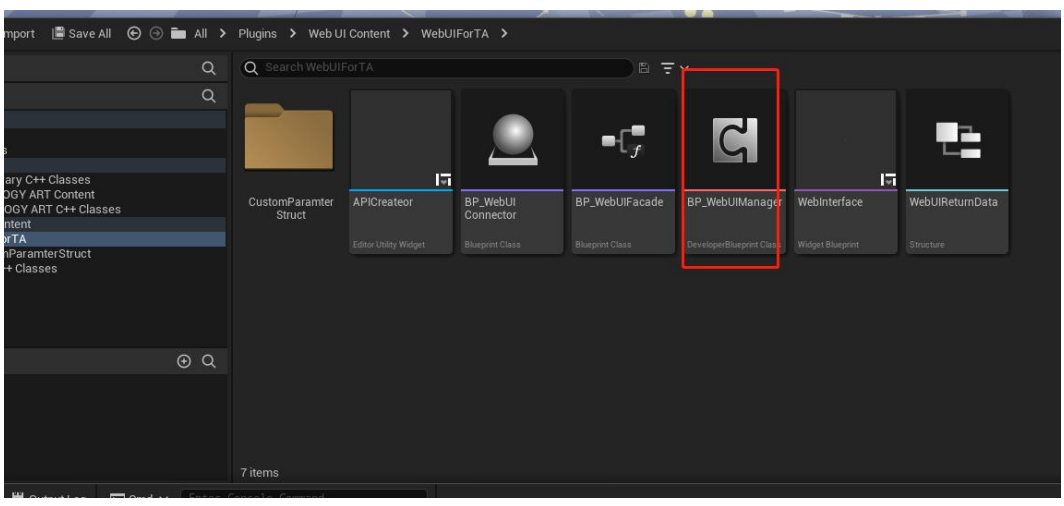


- 拓展内容在插件文件夹下



- 将里面的 WebUIManager 挂在到 Controller 上





webCode 参考以下附件：

sample.html

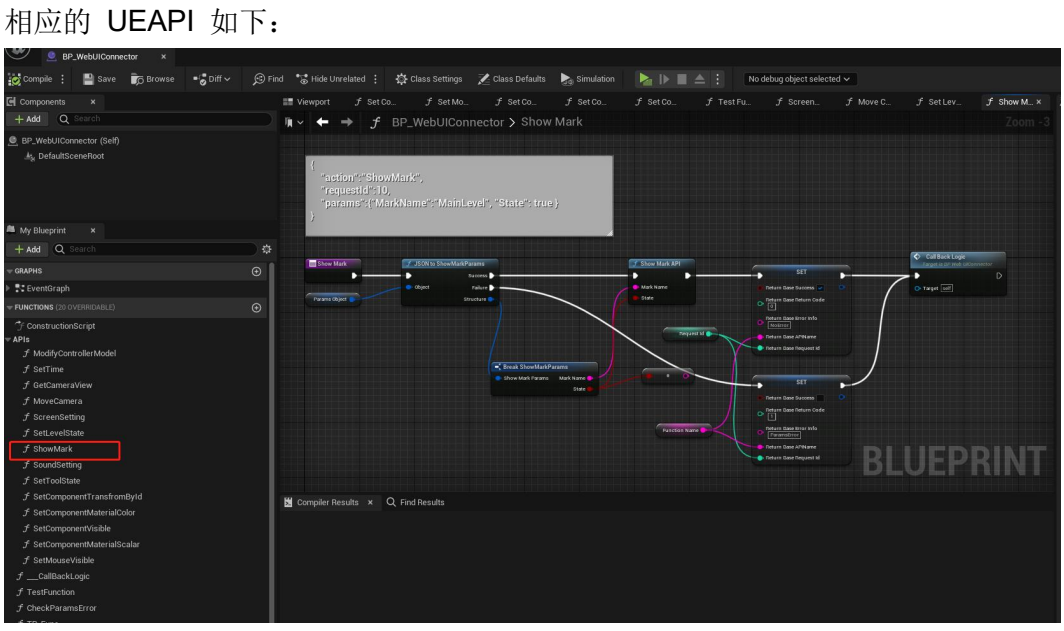
以下用一个示例方法来演示　只是设置

```
$("#ShowMark").click(function(e)
{
    console.log("call API");
    var APIStruct=new Object();
    APIStruct.Action="ShowMark";
    APIStruct.RequestId=10;
    var APIParams=new Object();
    APIParams.MarkName="common";
    APIParams.state=true;
    APIStruct.Params=APIParams;
    // executed in blueprints
    ue4("ExecuteAPIs", APIStruct,TAAPICallbackCommon);


});
```

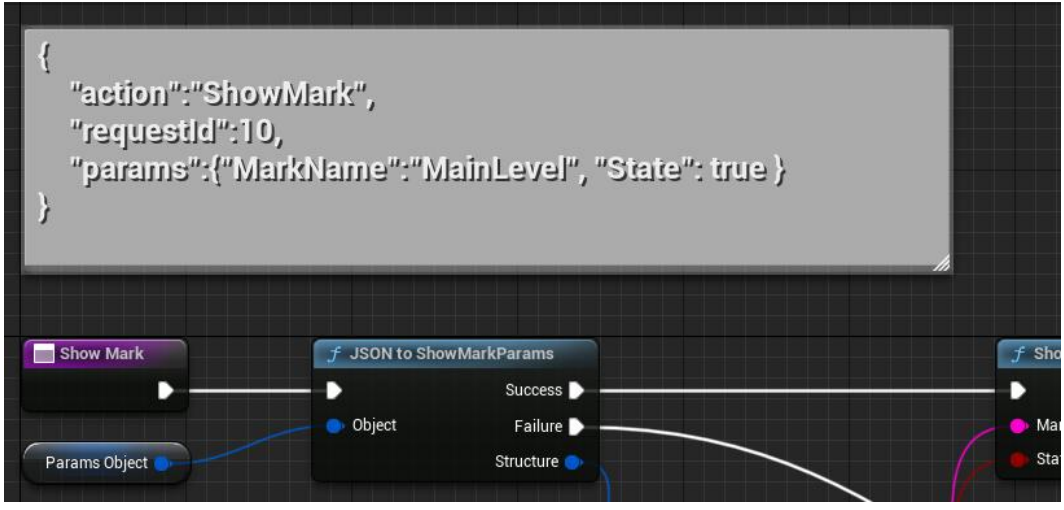
```
//通用接口:

//标签状态设置
/*
markName (string) : 标签名称
state (bool) : 是否显示
*/
function ShowMark(MarkName, State) {
    let obj = {
        action: "ShowMark",
        userId: sessionStorage.getItem("userId"),
        params: {
            markName: MarkName,
            state: State,
        }
    }
    ue4("ExecuteAPIs",obj,FunctionCallbackCommon);
    console.log(obj);
}
```

只要通过对应的 Action 和 params 就可以调用 api：

相应的 UEAPI 如下：



只要在 UIConnect 上面添加新的方法就可以用一样的方式使用 js 来调用：



/**** 接口发送格式： 使用通用结构

Action (API 名称) Params：根据 API 参数类型类型添加到 JsonObject

例如调用 ShowMarkAPI

{"Action":"ShowMarkAPI","Params":{"MarkName":"common","state":false}}

TAAPICallbackCommon 为统一回调方法放回消息格式如下：

{success: true, returnCode: 0, errorInfo: "", APIName: "ShowMarkAPI", requestId: 0, params: Object}

success：标识请求状态是否成功

returnCode：提示标识 0 为成功 1 为失败 2 为警告 并且可自定义消息

errorInfo：请求失败原因比如参数错误

aPIName:api 名称 （Action）

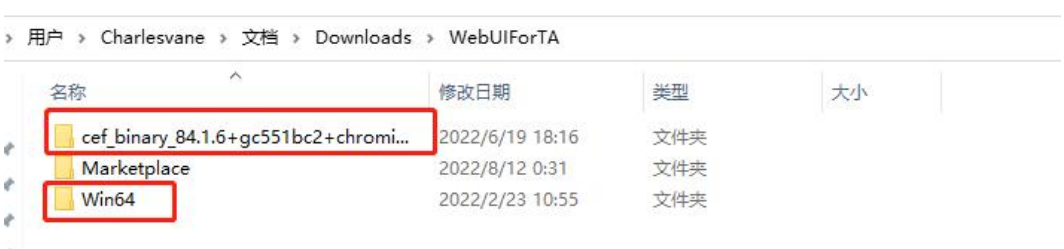
requestId：用来区分是哪个请求的回调

params：返回的结果，如果接口有返回值则通过这个对象来传输

*/

TechnologyArt-CEF 内核替换

上个章节下载的文件夹内部有另外两个文件夹就是 CEF 内核的编译产物 只有更新了内核才能播放视频和使用高版本的前端组件。



分别替换以下文件夹即可：