

描述

连接 Touch 操作和游戏模块，非常适合已经开发了输入模块（键鼠和手柄）的项目

简介

- 该插件提供了几个基础的 Touch 交互组件，可以非常便捷的接入自己的游戏模块。
- 每个组件都支持将 Touch 操作，转变为各种其它输入，例如通过按下按钮来模拟空格键输入，或者拖拽摇杆来模拟手柄摇杆移动。基于此，可以快速接入原有的输入流程，而不用修改原来的输入模块。
- 插件内包含两套输入案例，分别是直接触发按钮内编写的逻辑，以及接入原有的键鼠、手柄操作的 EnhancedInput 输入系统。
- 附带调试功能，可以查看历史真实输入、处理后输入、虚拟键输入。

初始设置

Project Setting:

- UseMouseForTouch: True
- DefaultViewportMouseLockMode: Do Not Lock
- DefaultTouchInterface: None

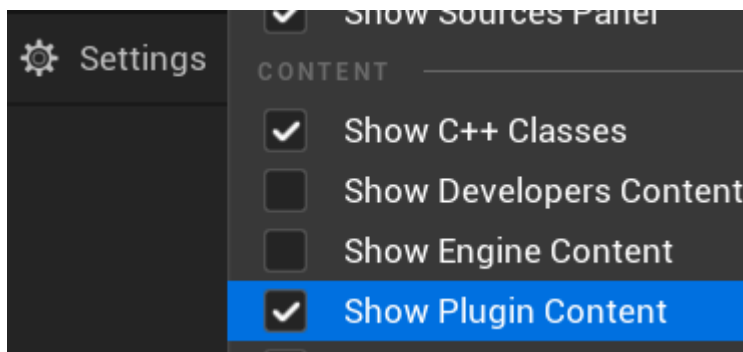
示例项目

切换 BP_EJPlayerController 的 bUseVirtualKey 属性，就可以切换两套配置（是否使用虚拟键）

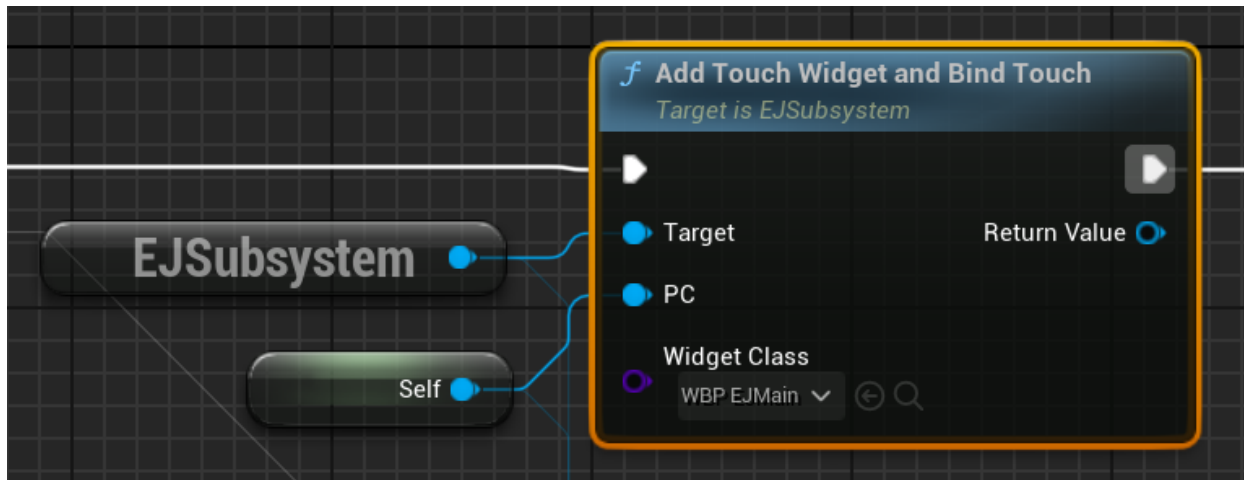
快速接入

如果你的项目已经可以用键盘、鼠标操控，那么可以将该插件快速接入你的项目，具体流程如下：

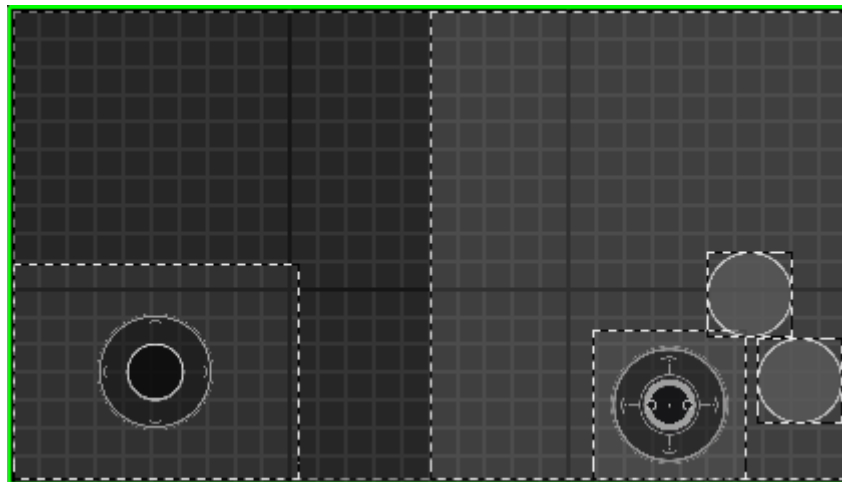
- 显示项目 PluginContent



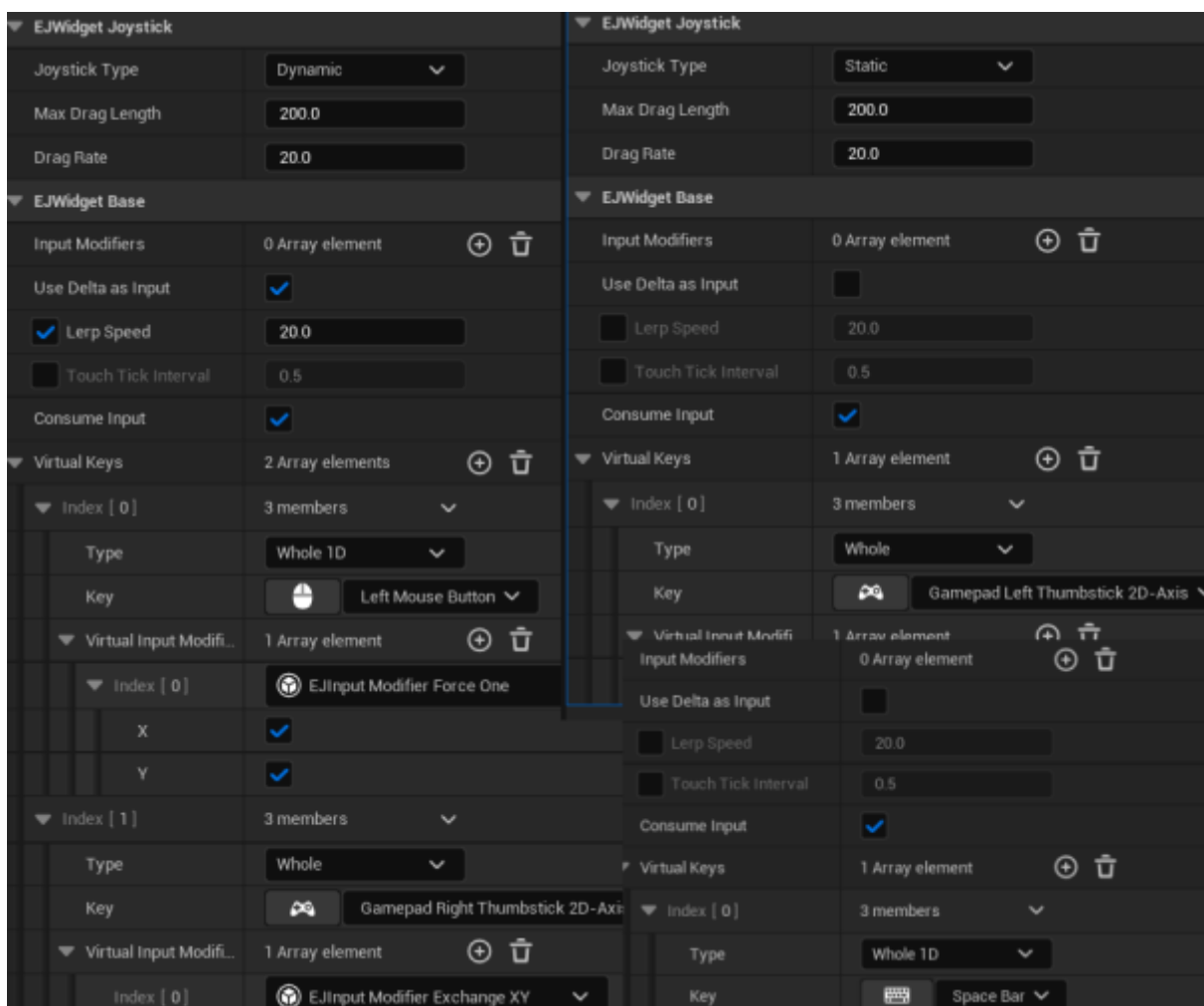
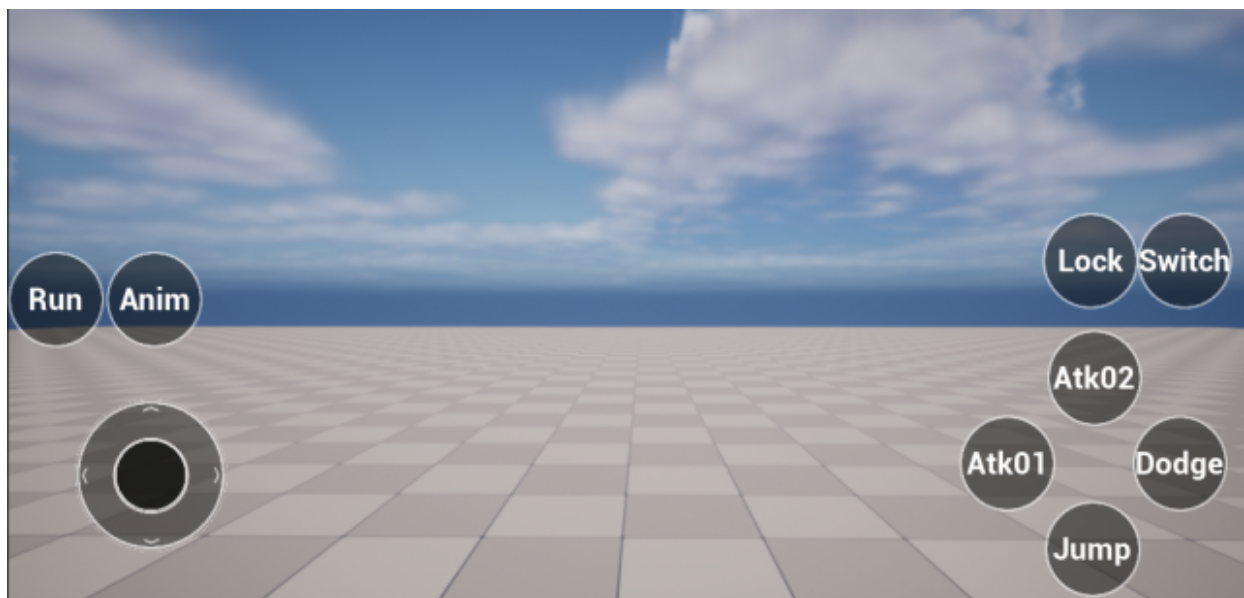
- 打开 PlayerController 蓝图，调用 UEJSubsystem::AddTouchWidgetAndBindTouch，传入 WBP_EJMain



- 默认的几个组件如下：
 - 静态移动摇杆，映射到手柄左摇杆
 - 动态射击摇杆，映射到鼠标左键、手柄右摇杆
 - 按钮1，映射到空格键
 - 按钮2，映射到 LeftShift 键



- 根据项目需要增加按钮，调整触发的虚拟键：



接口和 Touch 信息

如果你的项目还没有且之后不打算兼容键盘和鼠标，则将绑定的 Widget 改为 WBP_EJMain_NonVirtual，每个组件都提供了以下 API，用于接入其它模块：

- OnTouchBegin：开始 Touch 的瞬间

- OnTouchMove: Touch 点发生移动的时候
- OnTouchEnd: Touch 结束的瞬间
- OnTouchTick: Touch 激活期间
- OnViewportResize: 窗口大小改变时 (会主动结束所有的 Touch)

如果你需要获取本次 Touch 过程的信息，可以通过成员变量 TouchInfo 来获取：

- TouchState: 当前状态
- CurRealInput: Widget上得出的最原始版本的输入信息 (当前)
- TargetRealInput: Widget上得出的最原始版本的输入信息 (目标)
- CurInputValue: 计算后得出最终输入值
- StartLocation: 开始位置 (基于Widget::Center)
- CurLocation: 当前位置 (基于Widget::Center)
- PreLocation: 最近一次移动之前的位置 (基于Widget::Center)

输入流程说明 (重要)

1. Touch 信息以 FVector 的形式进入到 Touch 处理模块
2. 根据位置判断激活哪个组件
3. 转为UI坐标 (X朝右, Y朝下, 以像素为单位)
4. 转为以组件中心为原点的偏移向量 (StartLocation、CurLocation、PreLocation 在这一层级)
5. 组件内部处理该向量 (例如摇杆的最大拖拽长度、拖拽比率)
6. 记录到 TargetRealInput (如果开启 Lerp, 则 CurRealInput、TargetRealInput 会不同)
7. 默认选用 CurRealInput, 如果开启 bUseDeltaAsInput, 则选用当前 CurRealInput 和上一次 CurRealInput 的差值
8. 将输入转到 X 朝上, Y 朝右的左手坐标系
9. 使用 InputModifiers 处理向量 (CurInputValue)
10. CurInputValue 就是当前的输入量, 自定义逻辑一般直接参考这个即可, 摇杆往上拖拽时 $\text{CurInputValue.X} > 0$, 摇杆往右拖拽时 $\text{CurInputValue.Y} > 0$





输入修正

上面提到的 InputModifiers, 类似于 EnhancedInput 内的 UInputModifier, 用于根据需求来修改输入, 下面是列举:

- Normalize: 单位化向量
- ForceOne: 将某个维度置为1 (处理完后相对于仅考虑是否激活, 而不在意长度)
- Negate: 将某个维度取反

- Multiply: 每个维度乘上一个数
- ExchangeXY: 交换 X 和 Y

虚拟键

▼ Virtual Keys		2 Array elements	⊕	🗑
▼ Index [0]		3 members	▼	
Type		Whole 1D ▼		
Key		 Left Mouse Button ▼		
▼ Virtual Input Modifi...		1 Array element	⊕	🗑
▼ Index [0]		 EJInput Modifier Force One ▼ ▼		
X		<input checked="" type="checkbox"/>		
Y		<input checked="" type="checkbox"/>		
▼ Index [1]		3 members	▼	
Type		Whole ▼		
Key		 Gamepad Right Thumbstick 2D-Axis ▼		
▼ Virtual Input Modifi...		1 Array element	⊕	🗑
Index [0]		 EJInput Modifier Exchange XY ▼ ▼		

各个组件支持在 CurlInputValue 的基础上，触发配置的基础键，大致流程如下：

- 先根据定义的类型，抓取想要的数数据
 - Whole: 取出整个 InputValue
 - Whole1D: 取 InputValue.Size 作为 X
 - UpDown: 取 InputValue.X 作为 X
 - Up: 取 $\text{Max}(0, \text{InputValue.X})$ 作为 X
 - Down: 取 $\text{Min}(0, \text{InputValue.X})$ 作为 X
 - LeftRight: 取 InputValue.Y 作为 X
 - Left: 取 $\text{Min}(0, \text{InputValue.Y})$ 作为 X
 - Right: 取 $\text{Max}(0, \text{InputValue.Y})$ 作为 X
- 使用 VirtualInputModifiers 处理向量
- 最后将得出的值发送到输入模块

函数和变量说明

EJSubsystem

- AddTouchWidgetAndBindTouch: 初始化 Touch 模块, 创建 Widget, 绑定输入
 - 传入 UserWidget 内部所有继承了 `IEJWidgetItemInterface` 的 Widget, 都会被纳入到 Touch 管理
- GetWidgetCenter: 获取 Widget 中心位置
- GetWidgetSize: 获取 Widget 的大小
- IsItemWidgetActive: Widget 是否处于激活状态
- IsWidgetTouchAtLocationByInitInfo: Widget 是否应该被该位置的 Touch 操作激活
 - 目前使用的是渲染的 Geometry 的包围盒进行简单的判断, 如果有这方面的需要可以自行定制

EJWidgetInterface

- OnTouchBegin: 开始 Touch 的瞬间
- OnTouchMove: Touch 点发生移动的时候
- OnTouchEnd: Touch 结束的瞬间
- OnTouchTick: Touch 激活期间
- OnViewportResize: 窗口大小改变时 (会主动结束所有的 Touch)

如果不继承 EJWidget_Base, 而是只实现 EJWidgetInterface, 那么你需要重写以下函数:

很多功能都是基于 EJWidget_Base 完成的 (例如 UseDeltaAsInput、Lerp、VirtualKey、InputModifiers)

- SetTouchInfo / GetTouchInfo: 子类存储 / 读取 TouchInfo
- WhetherConsumeInput: 是否消耗输入
- GetTouchTickInterval: 自定义 Tick 的间隔

EJWidget_Base

函数:

- IsTouchActive: 是否激活
- GetCurlInputValue: 获取当前输入
- SubmitTargetRealInput: 提交组件处理后的 TargetRealInput

变量:

- InputModifiers: 处理输入值

- bUseDeltaAsInput: 是否将差量作为输入
- bEnableLerp: 是否开启 Lerp
- LerpSpeed: 输入变化的Lerp速度
- bEnableTouchTick: 是否开启 TouchTick
- TouchTickInterval: TouchTick 间隔 (TouchActive 期间)
- bConsumeInput: 是否消耗输入
- VirtualKeys: 触发的虚拟键
- TouchInfo: 本次 Touch 过程中的信息

EJWidget_Pad

变量:

- DragRate: 拖拽比率 (拖拽多少像素获得 1 单位的输入)

EJWidget_Joystick

函数:

- GetInnerWidget: 获取内部的 Widget (用于代码内控制位置)
- GetOuterWidget: 获取外部的 Widget (用于代码内控制位置)

变量:

- JoystickType: 摇杆类型, 分为静态摇杆和动态摇杆, 区别在于起始位置是否变化
- MaxDragLength: 限制拖拽最大长度
- DragRate: 拖拽比率 (拖拽多少像素获得 1 单位的输入)

调试器

位置:

- Tools - Debug - Enhanced Joystick Debugger

配置位置

- Editor Preferences - Plugins - Enhanced Joystick Debugger
 - MaxInfoItemNumber: 最大收集信息条数
 - MaxColumnNumber: 显示的最大列数

不使用虚拟键:



使用虚拟键：

