

## Description

---

Connect touch operation and other modules, very suitable for projects that have already developed input modules based on keyboard, mouse and gamepad.

## Brief Introduction

---

- This plugin provides several basic touch interaction components, which can be easily connected to your own game module.
- Each component supports the conversion of touch operations into various other inputs, such as simulating spacebar input by pressing a button, or dragging a joystick to simulate gamepad joystick input. Based on this, the original input flow can be quickly connected without modifying the original input module.
- This plugin contains two sets of input cases, which are directly triggering the logic written in the button and connecting the original EnhancedInput input system based on keyboard, mouse and gamepad.
- This plugin provides a debugger, you can view the history of real input, input and virtual key input.

## Initial Setup

---

Project Setting:

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

## Demo Project

---

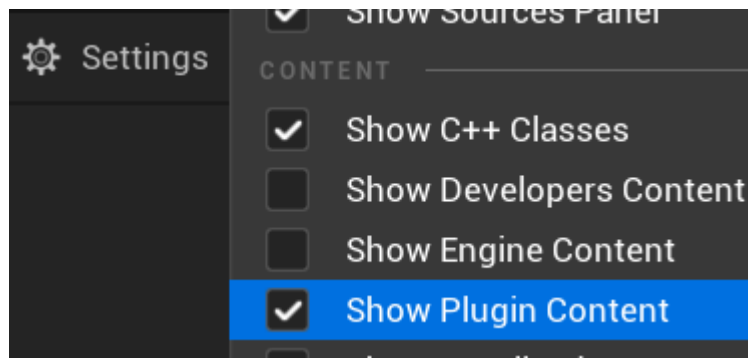
Toggle the bUseVirtualKey property of BP\_EJPlayerController to switch between two sets of configurations (whether or not to use virtual keys).

## Quickly Use

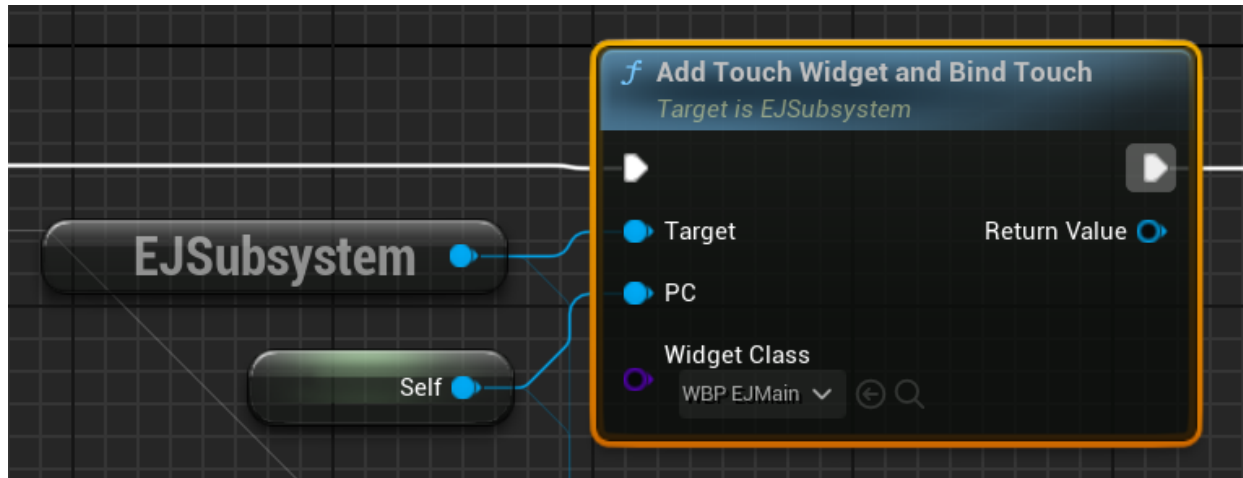
---

If your project has already developed input modules, try the following process to use this plugin quickly:

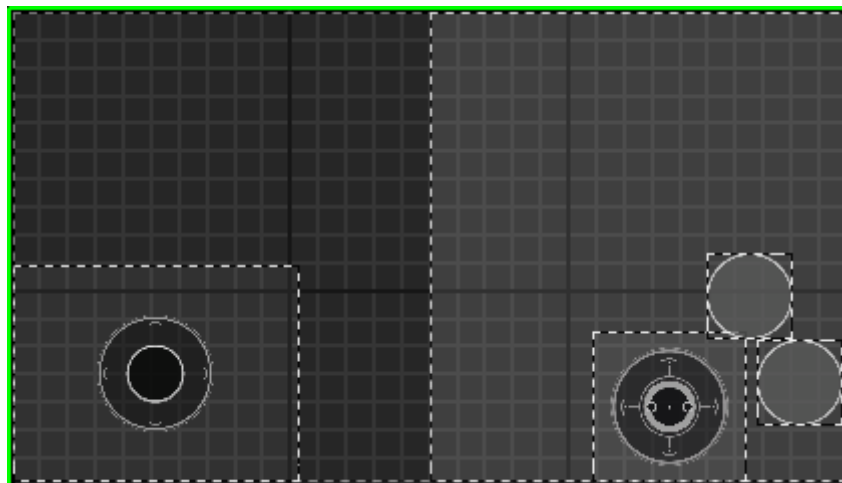
- Show Plugin Content



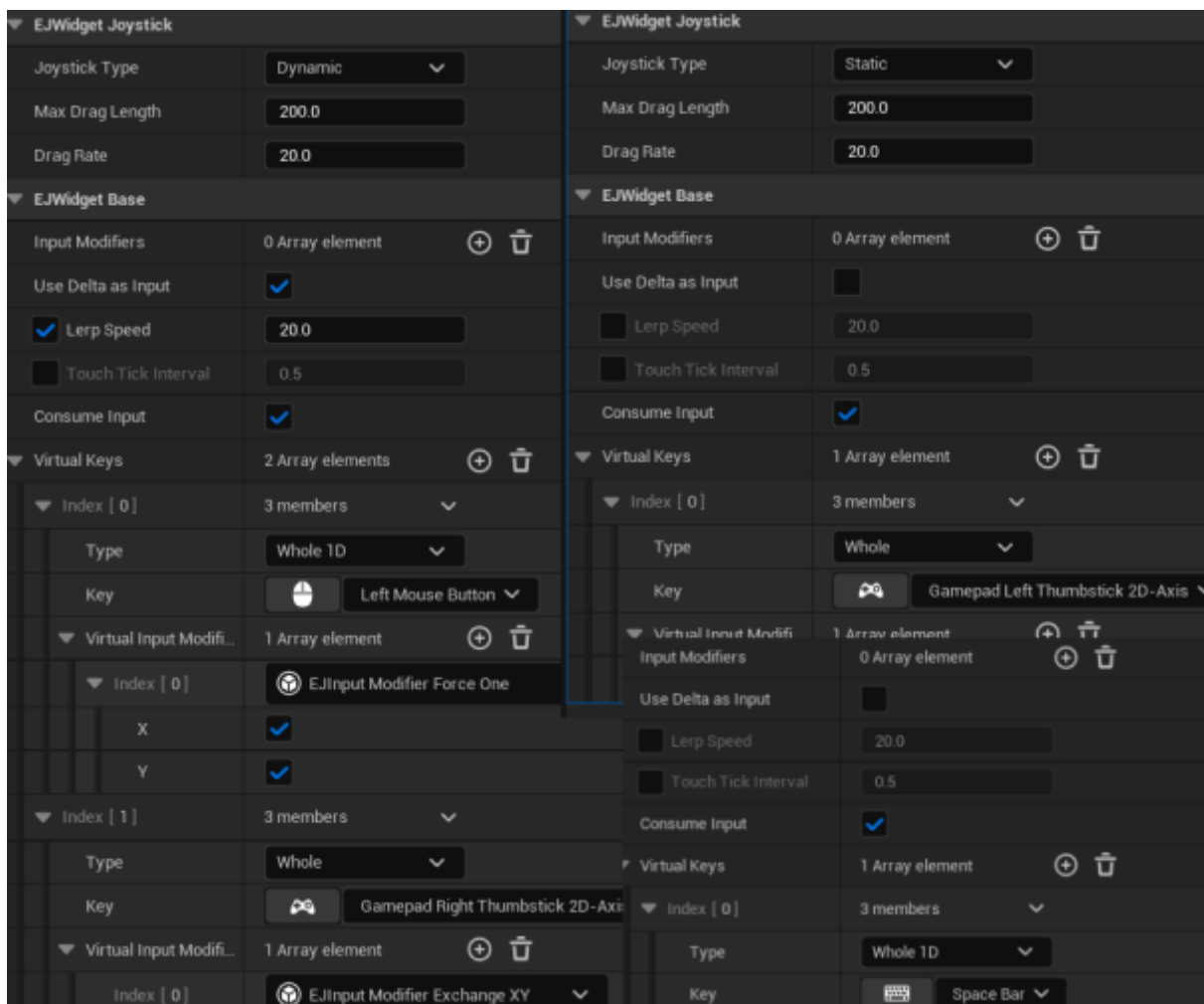
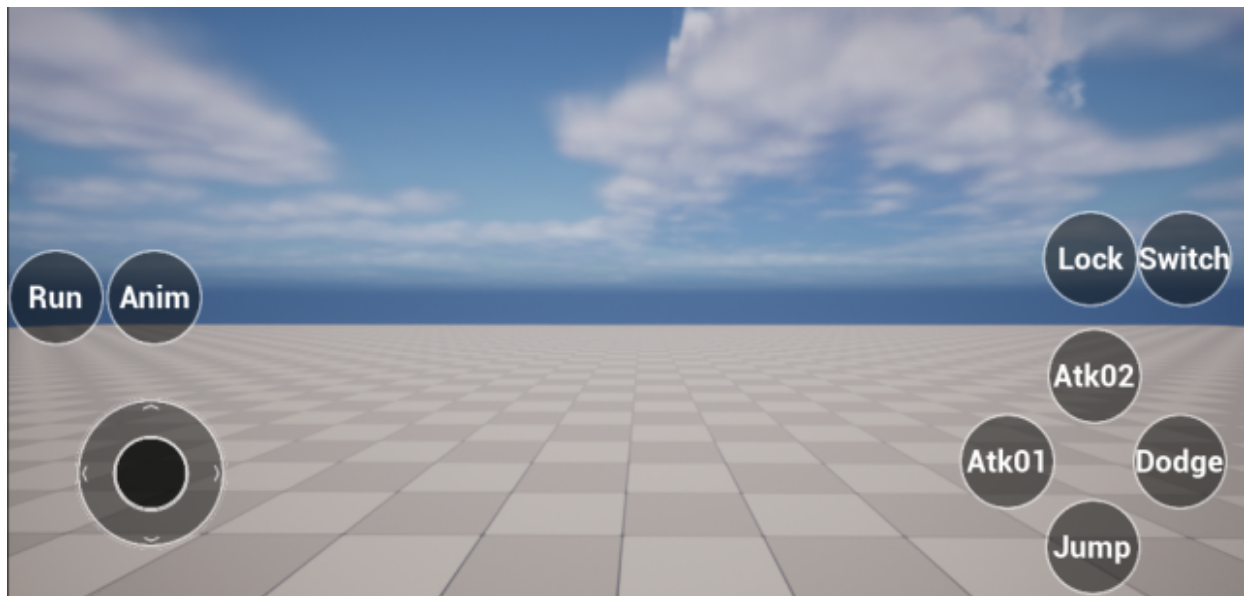
- Open your PlayerController blueprint, call UEJSubsystem::AddTouchWidgetAndBindTouch, and pass WBP\_EJMain as WidgetClass



- The default components in the WBP\_EJMain:
  - Static movement joystick, mapped to gamepad left thumbstick
  - Dynamic shoot joystick, mapped to mouse left button, gamepad right thumbstick
  - Button1, mapped to spacebar
  - Button2, mapped to left shift



- Add buttons and adjust virtual keys according to the needs of the your project:



## Interface, TouchInfo

If your project is not yet available and does not plan to be compatible with keyboard, mouse and gamepad, change the WidgetClass to WBP\_EJMain\_NonVirtual, and each component provides the following API for connecting other modules:

- **OnTouchBegin:** The moment the touch begin
- **OnTouchMove:** When the touch point moves
- **OnTouchEnd:** The moment the touch end
- **OnTouchTick:** During the activation of touch
- **OnViewportResize:** When the window size changes (all touch will be finished)

If you need the information of this touch process, you can get it from variable **TouchInfo**:

- **TouchState:** Current state of touch
- **CurRealInput:** Input value for the most original version (current)
- **TargetRealInput:** Input value for the most original version (target)
- **CurInputValue:** The final input value
- **StartLocation:** The start location based on **Widget::Center**
- **CurLocation:** The current location based on **Widget::Center**
- **PreLocation:** Location before the last move based on **Widget::Center**

## Input Process (Important)

---



1. The touch information enters the touch processing module in the form of **FVector**
2. Determine which component to activate according to the location
3. Convert to UI coordinates (X to the right, Y to the bottom, in pixels)
4. Convert to the offset vector based on the center of the component (**StartLocation**, **CurLocation**, **PreLocation** at this level)
5. Processed the vector by the component (for example, **MaxDragLength** and **DragRate**)
6. Set to **TargetRealInput** (if **Lerp** is enabled, **CurRealInput** and **TargetRealInput** will be different)
7. Use **CurRealInput** by default. If **bUseDeltaAsInput** is enabled, use the difference between the current **CurRealInput** and the last **CurRealInput**.
8. Convert to the left-handed coordinate system with X up and Y right.
9. Process input with **InputModifiers** (the final **CurInputValue**)
10. **CurInputValue** is the current input. Custom logic generally refers to this directly. **CurInputValue.X** > 0 when joystick is dragged upward and **CurInputValue.Y** > 0 when joystick is dragged to the right.

## Input Modifier

The InputModifiers mentioned above, similar to the UInputModifier in EnhancedInput, is used to modify input according to requirements. The following is a list:

- Normalize: Normalize the input
- ForceOne: Set a dimension to 1 (after processing, only consider whether it is active, regardless of length)
- Negate: Negate a dimension
- Multiply: Multiply a dimension by a number
- ExchangeXY: swap X and Y

## Virtual Key

▼ 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	▼	▼

Each component can trigger one or more virtual keys based on CurlInputValue. The general process is as follows:

- First get the desired data according to the defined type
  - Whole: Take the entire InputValue
  - Whole1D: Take InputValue.Size as X
  - UpDown: Take InputValue.X as X
  - Up: Take Max(0, InputValue.X) as X

- Down: Take  $\text{Min}(0, \text{InputValue.X})$  as X
- LeftRight: Take  $\text{InputValue.Y}$  as X
- Left: Take  $\text{Min}(0, \text{InputValue.Y})$  as X
- Right: Take  $\text{Max}(0, \text{InputValue.Y})$  as X
- Process input with VirtualInputModifiers
- Send the resulting value to the input module

## Function, Variable

---

### EJSubsystem

- AddTouchWidgetAndBindTouch: Initialize touch module, create widget, bind input
  - All widget that inherits `IEJWidgetItemInterface` within the UserWidget will be incorporated into touch management.
- GetWidgetCenter: Get the center of widget
- GetWidgetSize: Get the size of widget
- IsItemWidgetActive: Whether the widget is active
- IsWidgetTouchAtLocationByInitInfo: Whether the widget should be activated by the touch operation at this location
  - At present, this plugin use the bounding box of the rendered Geometry to make a simple judgment, and you can customize it if necessary.

### EJWidgetInterface

- OnTouchBegin: The moment the touch begin
- OnTouchMove: When the touch point moves
- OnTouchEnd: The moment the touch end
- OnTouchTick: During the activation of touch
- OnViewportResize: When the window size changes (all touch will be finished)

If you don't inherit `EJWidget_Base` and only implement `EJWidgetInterface`, you need to override the following functions:

Many functions are based on `EJWidget_Base` (such as `UseDeltaAsInput`, `Lerp`, `VirtualKey`, `InputModifiers`).

- SetTouchInfo / GetTouchInfo: Subclass store / read TouchInfo.
- WhetherConsumeInput: Whether to consume input.
- GetTouchTickInterval: The interval of TouchTick

## EJWidget\_Base

Function:

- IsTouchActive
- GetCurInputValue
- SubmitTargetRealInput: Submit the TargetRealInput processed by the component

Variables:

- InputModifiers
- bUseDeltaAsInput
- bEnableLerp
- LerpSpeed
- bEnableTouchTick
- TouchTickInterval
- bConsumeInput
- VirtualKeys
- TouchInfo

## EJWidget\_Pad

Variables:

- DragRate: How many pixels to drag to get 1 unit of input

## EJWidget\_Joystick

Function:

- GetInnerWidget: Get the inner widget (used to control the location within the code)
- GetOuterWidget: Get the outer widget (used to control the location within the code)

Variables:

- JoystickType: static or dynamic joystick, the difference lies in whether the starting position changes
- MaxDragLength: The maximum pixels of drag
- DragRate: How many pixels to drag to get 1 unit of input

## Debugger

---

## How to open:

- Tools - Debug - Enhanced Joystick Debugger

## Configuration:

- Editor Preferences - Plugins - Enhanced Joystick Debugger
  - MaxInfoItemNumber: Maximum number of information collected by a single component
  - MaxColumnNumber: Maximum number of columns displayed

## Didn't use virtual keys:



## Use virtual keys:

