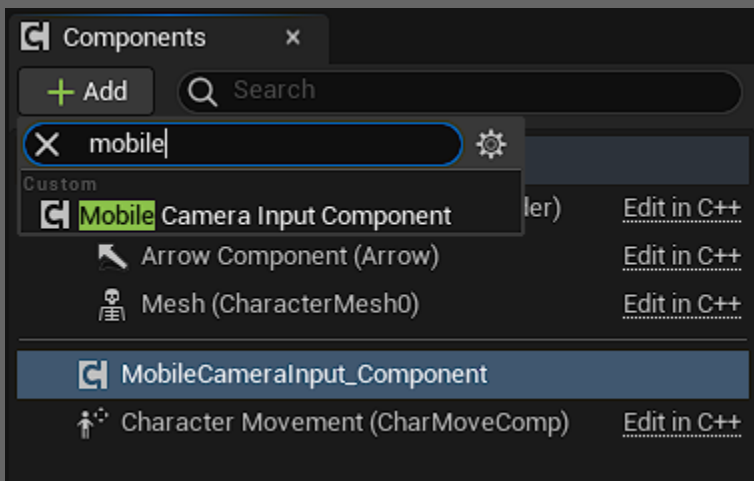


# Mobile Camera Enhanced Input

## Getting Started

The Mobile Camera Input component is a plug and play product that makes use of Enhanced Input's modularity to inject mobile third person camera controls into any player character.



To implement this product into your character, open your character blueprint and click the 'Add' button under the 'Components' tab and search for and select *Mobile Camera Input Component* in the 'Components' tab.

Your Character will now have third person mobile controls.  
For configuration see [here](#).

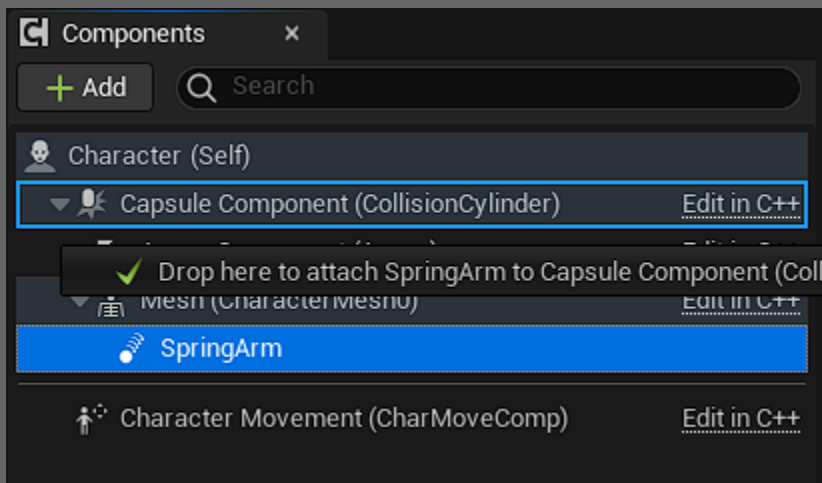
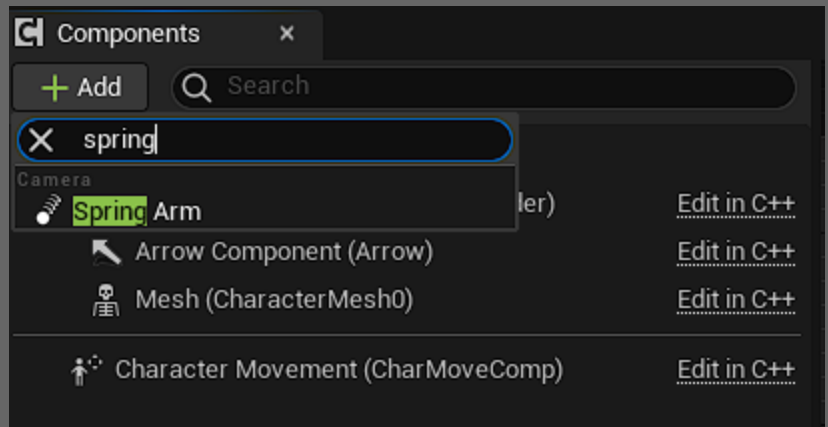
*As of 5.3.1, one extra step may be required if you want dynamic touch mappings.*

**Note:** The component needs a Spring Arm present in the character to function. **If no Spring Arm is present in your character, it will create one**, and will either create and attach a camera or attach an existing camera to it automatically. If you have multiple cameras, set your desired camera's component tag to the same thing as the 'Camera Tag' variable in the component.

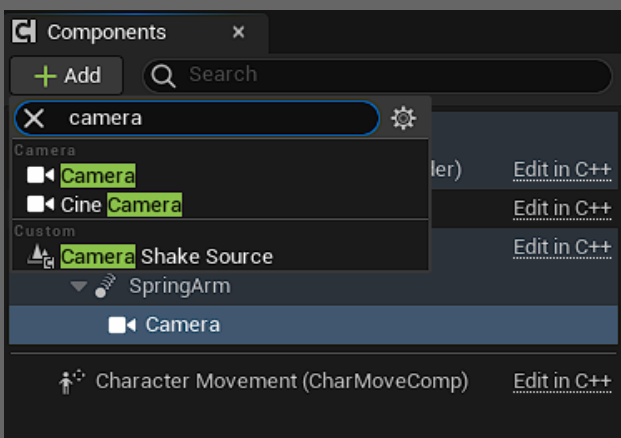
It is **not** necessary to create a spring arm or camera, however doing so may have some benefits related to more advanced configuration, such as when post-process effects are desired, or to modify camera collision in the spring arm.

# Adding a Spring Arm

In your character blueprint, click the 'Add' button under the components tab and search for 'Spring Arm'.



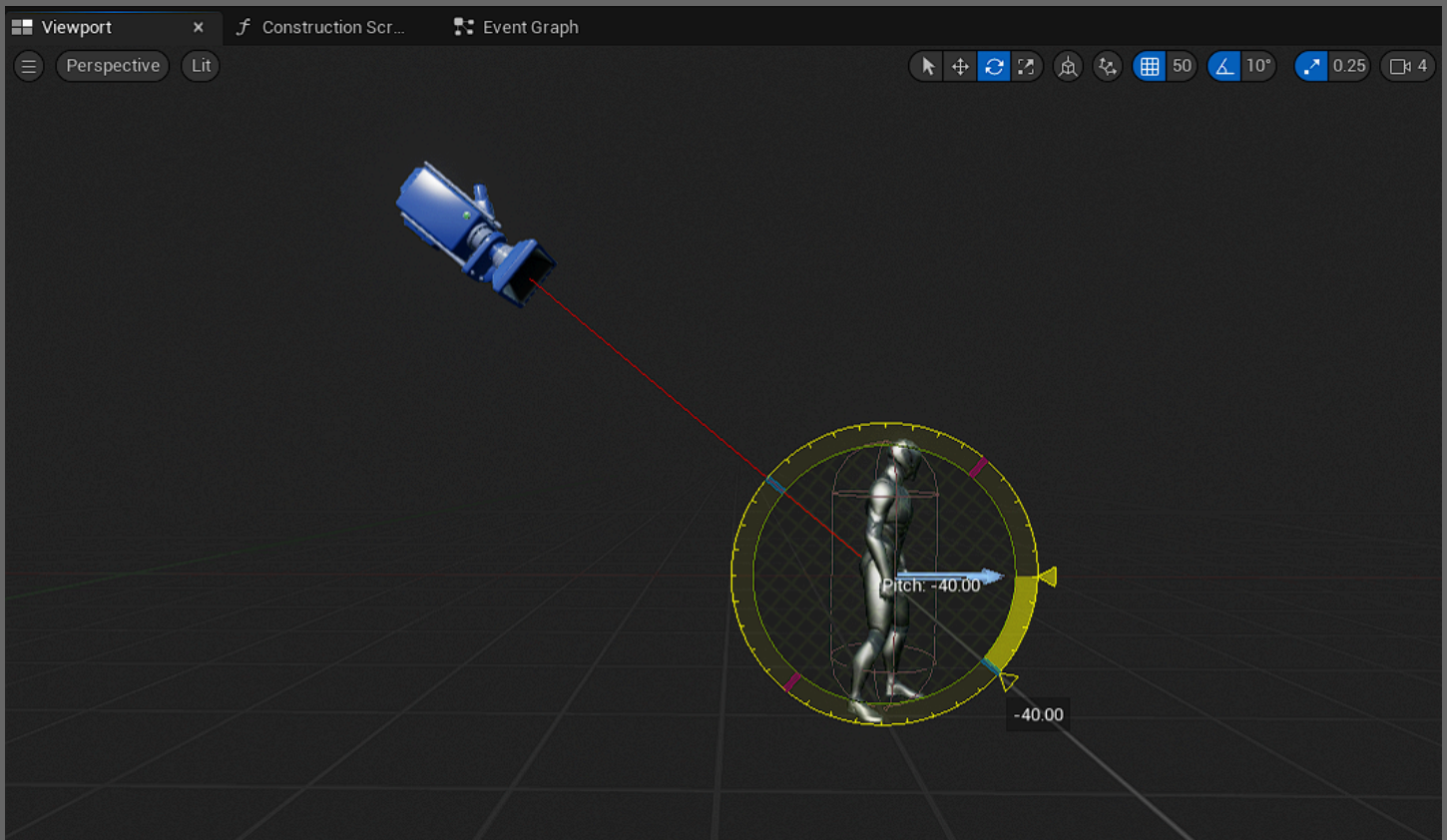
Ensure the Spring Arm is attached to the root Capsule Component.



Next add a Camera Component, or move an existing one to attach it to the Spring Arm.

Make sure the Camera's locations and rotations are all 0.

Select the Spring Arm and press E in the viewport to rotate pitch to desired starting rotation. Later you can define rotation limits so keep that in mind. The starting pitch will be clamped if it's not within the defined range.



# Features

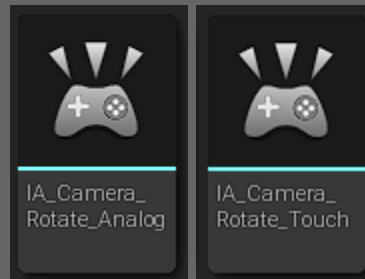
Core controls within in the Mobile Camera Input Component Include:

- [Rotate](#)
- [Pan](#)
- [Zoom](#)
- [Double Tap](#)



By selecting the *Mobile Camera Input Component* in your character blueprint you can access these settings to configure your camera input, displayed in the 'Details' tab.

Rotate and Pan inputs use separate Input Actions for analog and touch input for runtime efficiency and simplicity, as touch drag uses screen location and analog uses 0-1 input values.



# Rotation

Rotation is controlled by analog input, drag input, or a two finger touch rotation gesture.



- Keep Relative Rotation - Affixes camera rotation to character rotation. Not advised to be combined with Orbital Rotation.
- Orbital Rotation - Whether the controlled character or the offset location should act as the axis of rotation

▼ Rotation	
Keep Relative Rotation	<input type="checkbox"/>
Orbital Rotation	<input type="checkbox"/>
Additive Rotation	<input type="checkbox"/>
Use Control Rotation	<input type="checkbox"/>
Camera Rotation Lag Speed	<input type="text" value="1.0"/>
Rotation Y Rate Scale	<input type="text" value="2.0"/>
Rotation Z Rate Scale	<input type="text" value="2.0"/>
Min Camera Y	<input type="text" value="-89.0"/>
Max Camera Y	<input type="text" value="89.0"/>

- Additive Rotation - Whether rotation is continuous if held. Has no effect on virtual joystick controls.
- Camera Rotation Lag Speed - How quickly camera rotation will blend from current to updating character rotation
- Rotation Y/Z Rate Scale - Multipliers for input speed, use negative values to invert direction
- Min/Max Camera Y - Limits for camera pitch rotation

# Panning

Panning is controlled by either analog or touch drag input.



- Keep Relative Offset - Whether the pan offset should be fixed to the character's rotation.
- Additive Panning - Whether panning is continuous if held
- Pan Radius - Limit for pan travel in radius
- Pan Rate Scale - Input multiplier for pan speed, use negative value to invert
- Pan Time to Reset - Delay before returning to origin after completing a camera panning action. If 0, will not reset
- Pan Offset Interp Speed - Speed which the camera will blend when moving or panning
- Pan Offset - Starting relative camera offset

▼ Panning			
Keep Relative Offset		<input type="checkbox"/>	
Additive Panning		<input type="checkbox"/>	
Pan Radius		300.0	
Pan Rate Scale		1.0	
Pan Time to Reset		5.0	
Pan Offset Interp Speed		3.0	
▶ Pan Offset		0.0	0.0

By default, the pan input uses a [trigger](#) that requires a minimum specified distance traveled in finger location within a specified timeframe to be accepted.

# Zooming

Zooming is controlled by a two finger pinch gesture.






- Additive Zoom - Whether zoom is continuous if held
- Camera Min/Max Distance - zoom limits

Zoom	
Additive Zoom	<input type="checkbox"/>
Camera Min Distance	400.0
Camera Max Distance	1600.0
Zoom Rate Scale	1.0

- Zoom Rate Scale - Multiplier for zoom input, use negative value to invert

- Show Debug Locations - Draws locations of current and goal camera targets
- Touch Interface - Reference to a Touch Interface asset for digital joystick controls
- Input Config - Reference to a Player Mappable Input Config containing enhanced input configurations. **As of 5.3.1 this is now a Data Table Row.**

Default	
Show Debug Locations	<input checked="" type="checkbox"/>
Touch Interface	 DualVirtualJoysticks
Input Config	 PMIC_Mobile_Joystick
Double Tap Action	 IA_Camera_Reset
Touch 2 Double Tap Action	None

- Double Taps - References to Enhanced Input Actions to trigger when double tapping the screen. By default the touch 1 double tap triggers a camera reset. Double Tap delays are handled in the 'Double Tap' macro in the component.

# Conflicting Inputs

Due to limitations in input options on mobile devices, developers may have to manage conflicting inputs. For example the Pan action is mapped to Touch 1 and Rotate is mapped to Touch 2. If left as is, a rotate gesture will also inadvertently trigger a pan event. To resolve this I make use of Enhanced Input's ability to add and remove Input Mapping Contexts, and thus their actions, from player input.

Touch 1 and Touch 2 events are separated, each in their own input mapping context. Logic follows that if a user triggers a Touch 2 event, they are making a conscious decision to physically place 2 fingers on the screen and thus that is the highest priority action. So naturally all Touch 1 actions are disabled when a Touch 2 action is triggered by toggling 'Touch1' contexts.

Therefore it is advised to add any new Touch 1 inputs to IMC\_Camera\_1Touch.

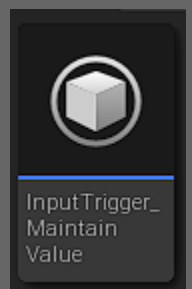
---

## Maintain Value Input Trigger

Another method of managing conflicting inputs, this product includes an input trigger named InputTrigger\_MaintainValue.

By default, the pan and touch movement inputs each use a trigger that remedies an input conflict between them. **It requires the input value be maintained ( with tolerance ) for a specified duration** to be accepted. Touch movement requires touch and hold in place, and panning requires the opposite - to exceed the tolerance before time runs out.

This allows touch movement to coexist with other touch actions, as they will both use Touch 1 input.





**NOTE:** In UE 5.1, there appears to be a bug in which this trigger only works properly if placed in the Input Action itself and NOT the Input Mapping Context. This appears to have been resolved in UE 5.2. If you experience issues with this trigger, try moving it to the input action. The trigger can be disabled by removing it from the Input Action assets 'IA\_Camera\_Pan' and 'IA\_Set\_Destination\_Touch' or any duplicates.

Triggers		1 Array elements		
Index [ 0 ]	Input Trigger Maintain Value			
Tolerance	50.0			
Trigger Type	Blocker			
Starting Point	0.0	0.0	0.0	
Last Update	0.0			
Conclusion	None			
Affected by Time Dilation	<input type="checkbox"/>			
Actuation Threshold	0.25			

The tolerance in deviance from the initial value and the required hold time can both be modified.

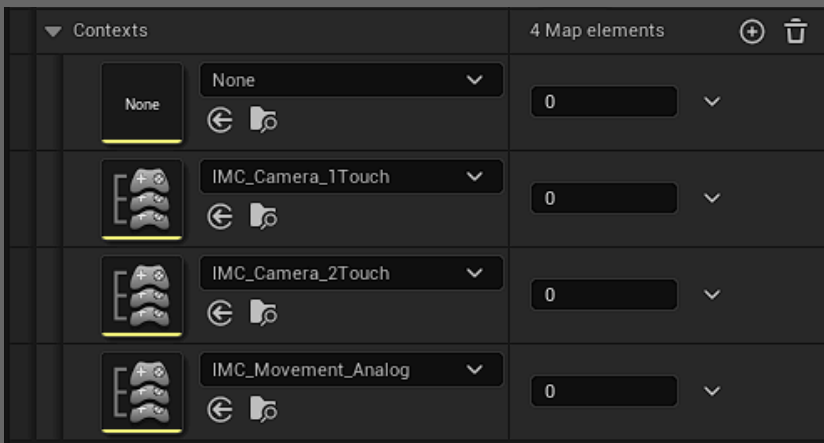
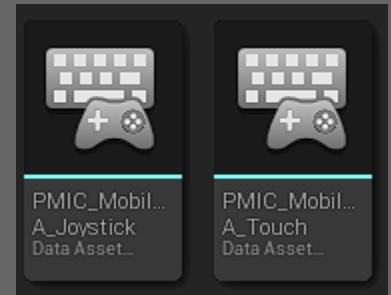
- Tolerance defines maximum permitted deviance from initial value before the trigger fails or succeeds and locks the input to that conclusion.
- Actuation Threshold defines required hold time

**'Starting Point', 'Last Update' and 'Conclusion' are internal usage and should not be modified.**

# Combining Existing Inputs

See [Enhanced Input in Unreal Engine 5.3](#) for how to configure Enhanced Input in general.

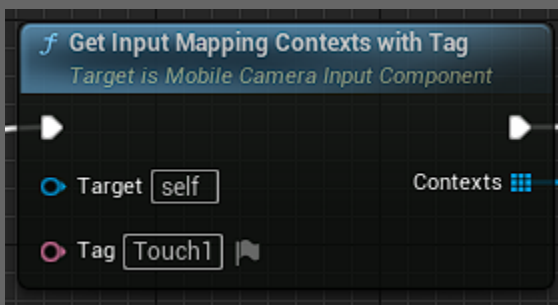
Any Player Mappable Input Config can be initiated by the component. Two config types are included, one for touch and one for analog joystick input, with variations A and B.



To combine with your existing player input, simply edit or copy one of the aforementioned configs and add your Input Mapping Context(s) to the 'Contexts' array.

Ensure the config is selected in the component's '[Input Config](#)' variable.

***Epic Games has deprecated PMICs, as of 5.3.1 the PMICs have been replaced by a data table DT\_InputConfigs, in which each row acts as a single config.***



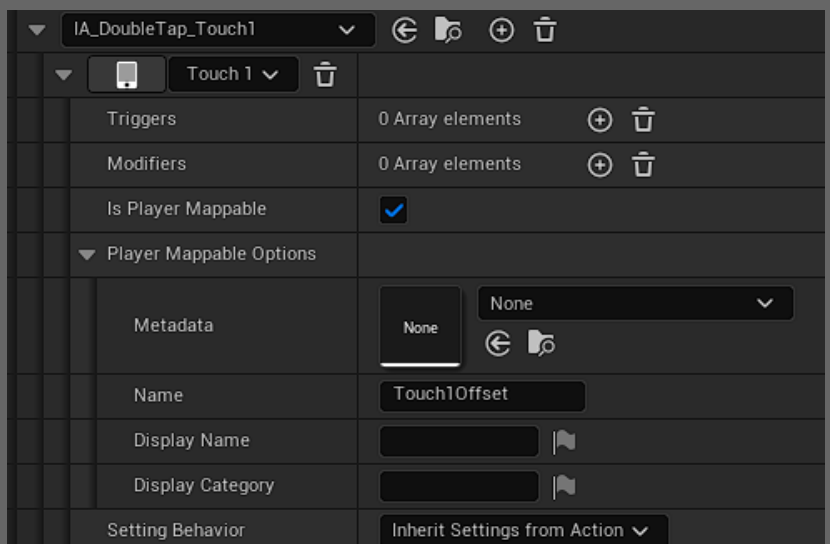
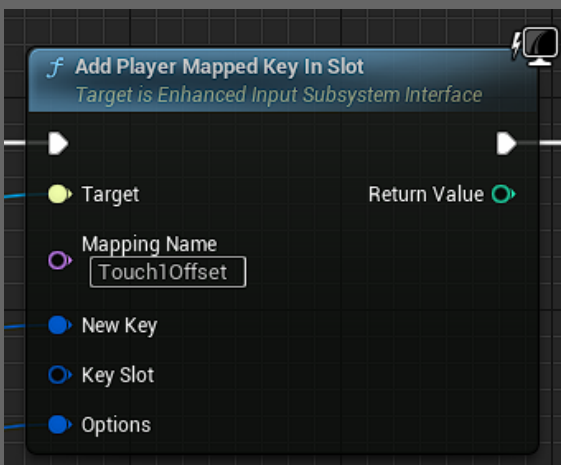
This component uses Input Mapping Context Descriptions as tags for remapping. Here any context with 'Touch1' tag will be deactivated temporarily to avoid input conflicts. This function may be useful for further expansion. It will return any contexts from the config with matching descriptions.

# Touch Offset

Digital joysticks don't trigger Touch events but they still count as a finger on the screen, and the challenge is they don't register any touch index state as being pressed, at least as of UE 5.3.1, so they will 'silently' block any Touch actions that collide. It's not possible to prevent them from consuming touch events but it is possible to dynamically increase the finger mappings of following inputs, so touch 2 events temporarily become touch 3 events, etc. Touch Offset is a feature that dynamically adjusts the touch index mappings of actions to compensate for certain actions consuming their input. The result is, as an example, the ability to trigger Touch 1 events while using a digital joystick, or two, or while using touch movement. Running and tapping ( picking up items or strafing and attacking ) without canceling either action is now possible.

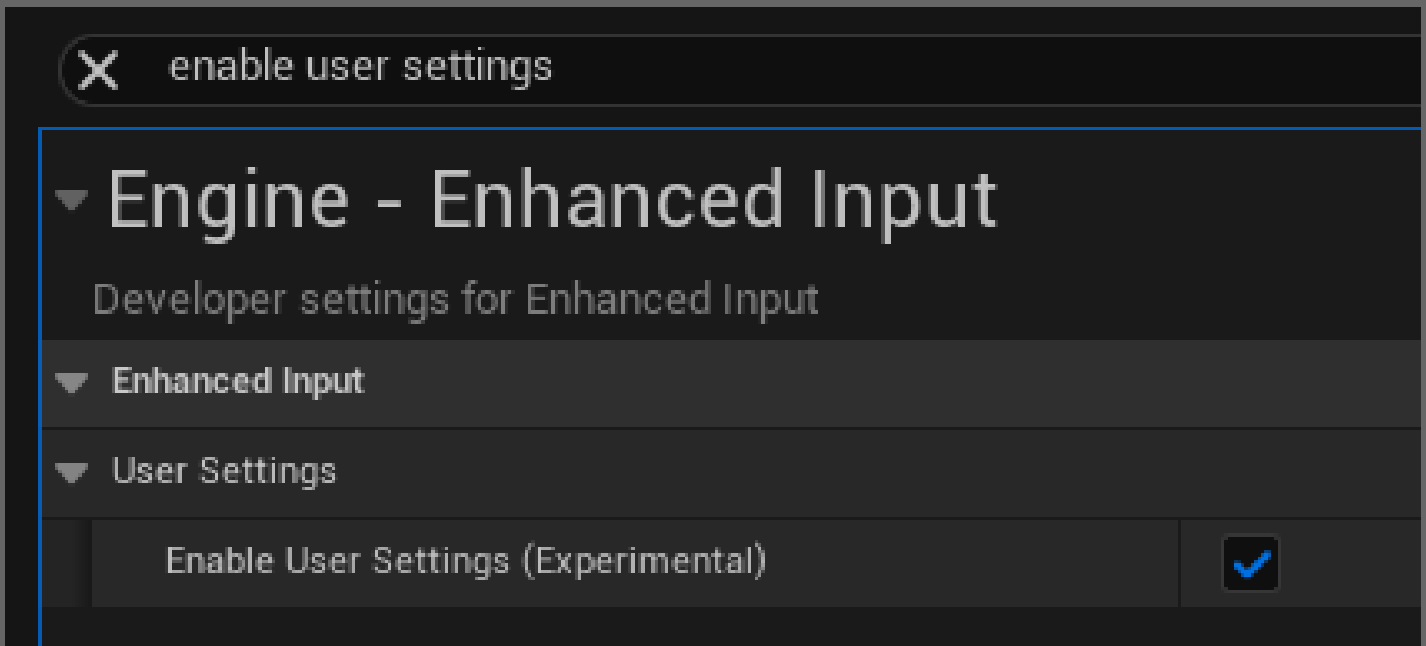
This is not necessary for physical gamepad joysticks. This is a mobile product but if you want to use a physical gamepad anyway, or otherwise disable this feature, just remove the IA\_TouchOffset action from any input mapping contexts.

This works via Player Mappable Options ( now Player Mappable Key Settings ) in the Mapping Contexts. Name is used as a tag and set by default to 'Touch1Offset' for Touch 1 actions or 'Touch2Offset' for Touch 2 actions. The tags to search are set by the 'Dynamic Mapping Names' variable in the mobile camera input component.



### 5.3.1

“Add Player Mapped Key in Slot” node pictured above has been completely deprecated, now replaced by “Map Player Key”. Touch Offset now relies on an experimental alternative system, ‘Enhanced Input User Settings’, this is not enabled by default, so it requires checking this box in project settings - “Enable User Settings”.



## Feedback and Questions

If any questions remain please do not hesitate to ask on the [store page](#).  
Additionally, feel free to make feature requests and especially bug reports.

Thank you!