

# Force Feedback

Using the vibration functionality of mobile devices and controllers to convey a force occurring in the game to the player.



Force Feedback is the vibration of a device often used in games to convey a force occurring in the game to the player. This feature is known as "rumble" or controller vibration for gamepads or controllers. For example, you can use force feedback to simulate the shockwave when an explosion occurs in the game. This gives an additional dimension to a player's immersion.

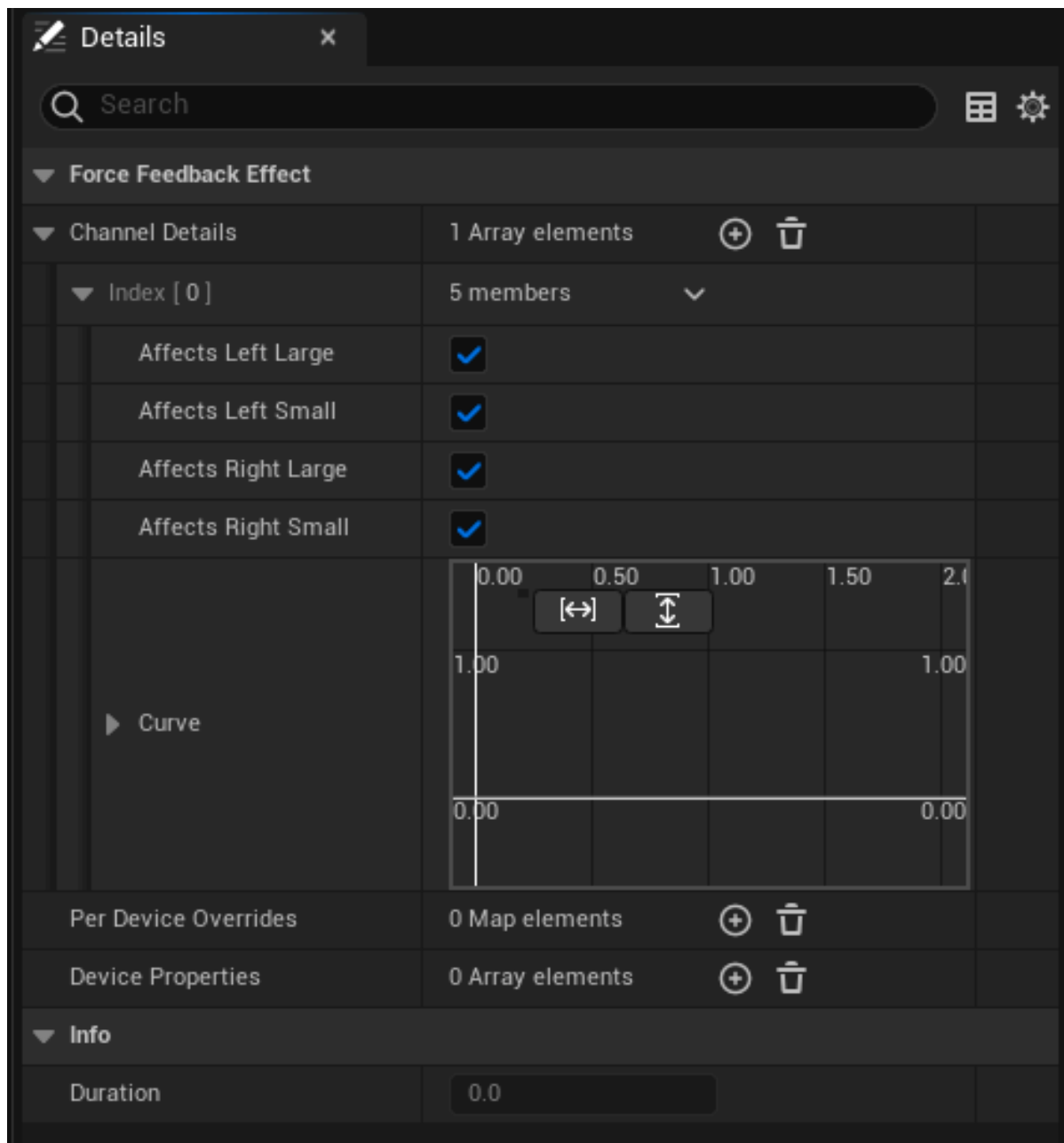
Force Feedback is supported by many common platforms such iOS, Android, and input controllers for consoles.



Support for certain feedback features are platform dependent. See your platform development documentation for full details.

## Anatomy of a Force Feedback Effect

A **Force Feedback Effect** asset contains properties used to define a specific force feedback effect. This enables designers to customize the force feedback to many different situations.



## Channel Details

Force Feedback Effects can have multiple channels. Each **channel** can play a different effect. For example, one channel could play a large, long vibration on the right side of the controller, while a second channel could play small, short bursts on the left side.

Each channel has the following properties that control how the channel effect is played:

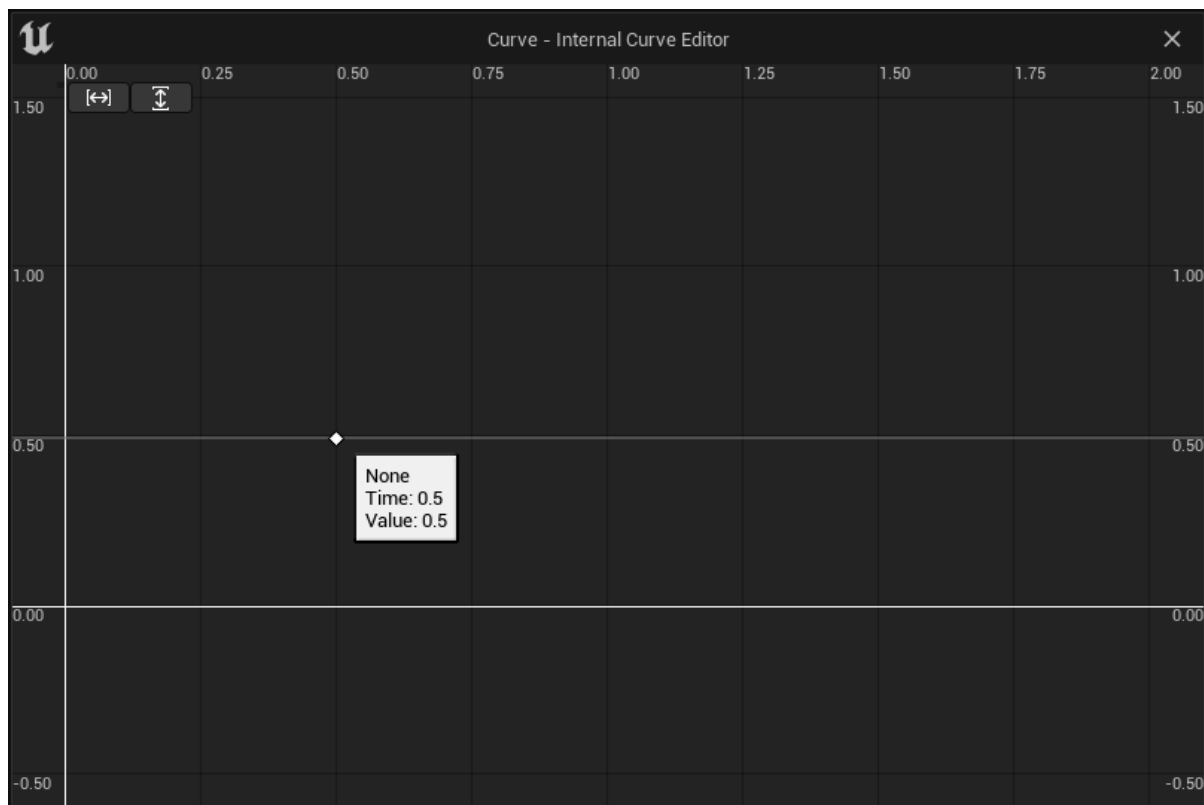
Channel Name	Description
<b>Affects Left Large</b>	If true, the left large motor will be used to play the effect.

Channel Name	Description
<b>Affects Left Small</b>	If true, the left small motor will be used to play the effect.
<b>Affects Right Large</b>	If true, the right large motor will be used to play the effect.
<b>Affects Right Small</b>	If true, the right small motor will be used to play the effect.
<b>Curve</b>	Controls the intensity of the effect over time. This defines the pattern of the vibration. Values above 0.0 cause vibration, while values below 0.0 do not cause vibration.

## Force Feedback Curve

The pattern of the effect for each channel is controlled by a Curve. You can add a key to the curve from the Internal Curve Editor by

- Using right-click and selecting add key.
- Open the internal curve editor by double-clicking on the curve's graph.





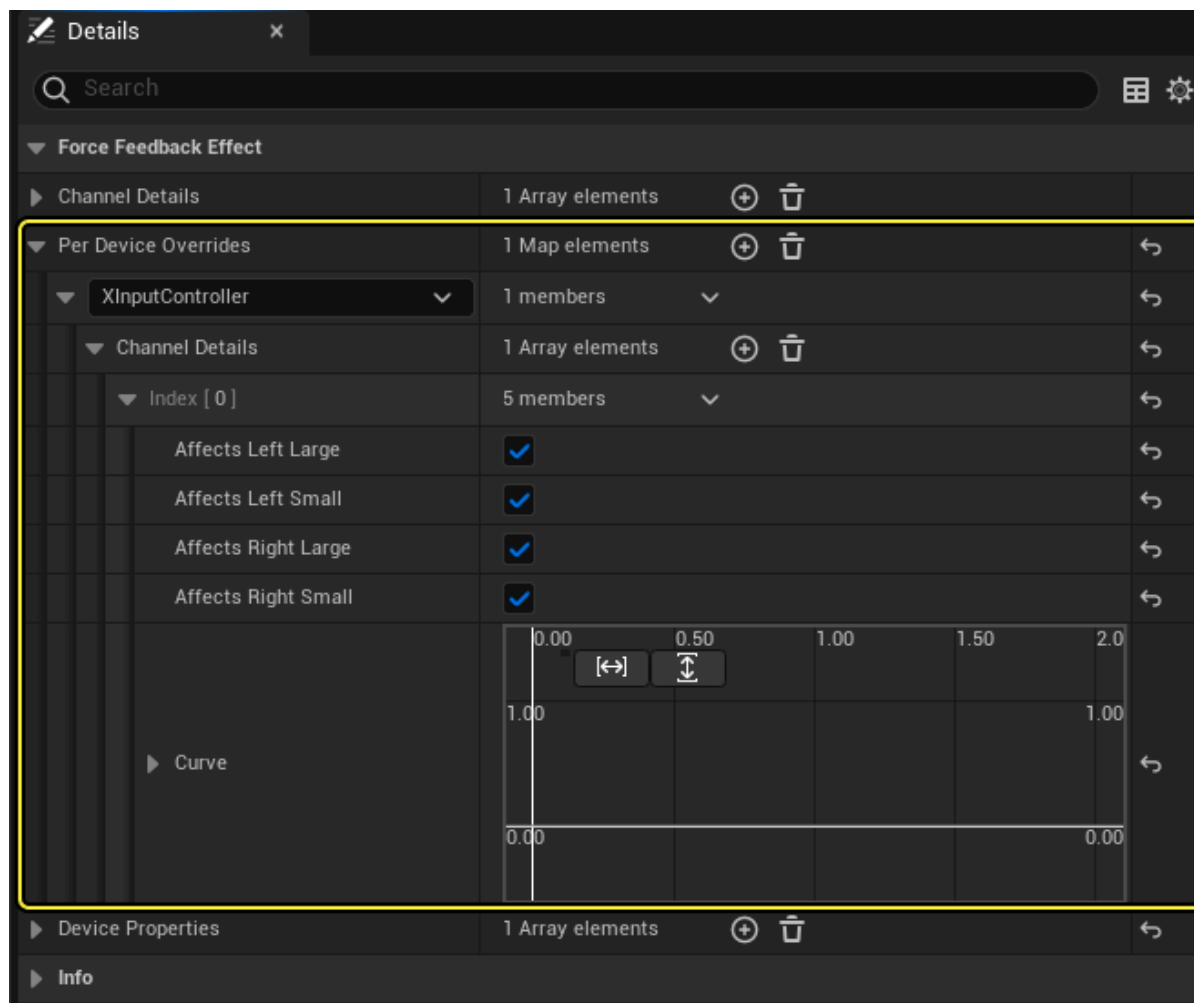
For information on Curves, Keys, creating external Curve Assets, and using the Curve Editor, see the [Curve Editor](#) and [Keys and Curves](#) pages.

## Per Device Overrides

When working with Force Feedback Assets in Unreal Engine, each platform has a different implementation of how its vibration motors or feedback systems function. Force Feedback assets use **Per Device Overrides** to support multiple platforms.

Per device overrides are an abstraction layer that provides you with a method to set different feedback settings for each platform. For example, you can apply a strong vibration to an Xbox controller, but a more detailed and nuanced vibration to a PlayStation controller.

To modify these settings, click your Force Feedback Effect, then navigate to **Details > Per Device Overrides**.



# Device Properties

**Device Properties** represent the different physical properties of an input device, such as its light color display or haptic trigger resistance.

Device Property Type

Description

Audio Based Vibration

Play a sound to an input device's speaker. On platforms that support it, this sound is played in the form of a vibration where the left and right audio channels represent the left and right side of the controller.

i

This feature is experimental, and only works for the PS5 DualSense Controller. To use this feature you will need to add the following to your configuration.

1	[SonyController]	
2	VibrationMode=Advanced	

Copy full snippet

Device Color (Curve)

Change the color of an input device's light over time with a curve.




i

This property has platform-specific implementations that may behave differently for each platform.

Device Color (Static)

Set the color of an Input Device to a static color. This will NOT reset the device color when the property is done evaluating. You can think of this as a one shot effect, that once enabled the device property color is set.

i

Device Property Type	Description
	<p>This property has platform-specific implementations that may behave differently for each platform.</p>
<b>Trigger Feedback</b>	<p>Set simple Trigger Feedback.</p> <p> This property has platform-specific implementations that may behave differently for each platform.</p>
<b>Trigger Resistance</b>	<p>Provide linear resistance to a trigger while it is pressed between a start and an end value.</p> <p> This property has platform-specific implementations that may behave differently for each platform.</p>
<b>Trigger Vibration</b>	<p>Set Trigger Vibration.</p> <p> This property has platform-specific implementations that may behave differently for each platform.</p>

See [Device Properties](#) for additional documentation.

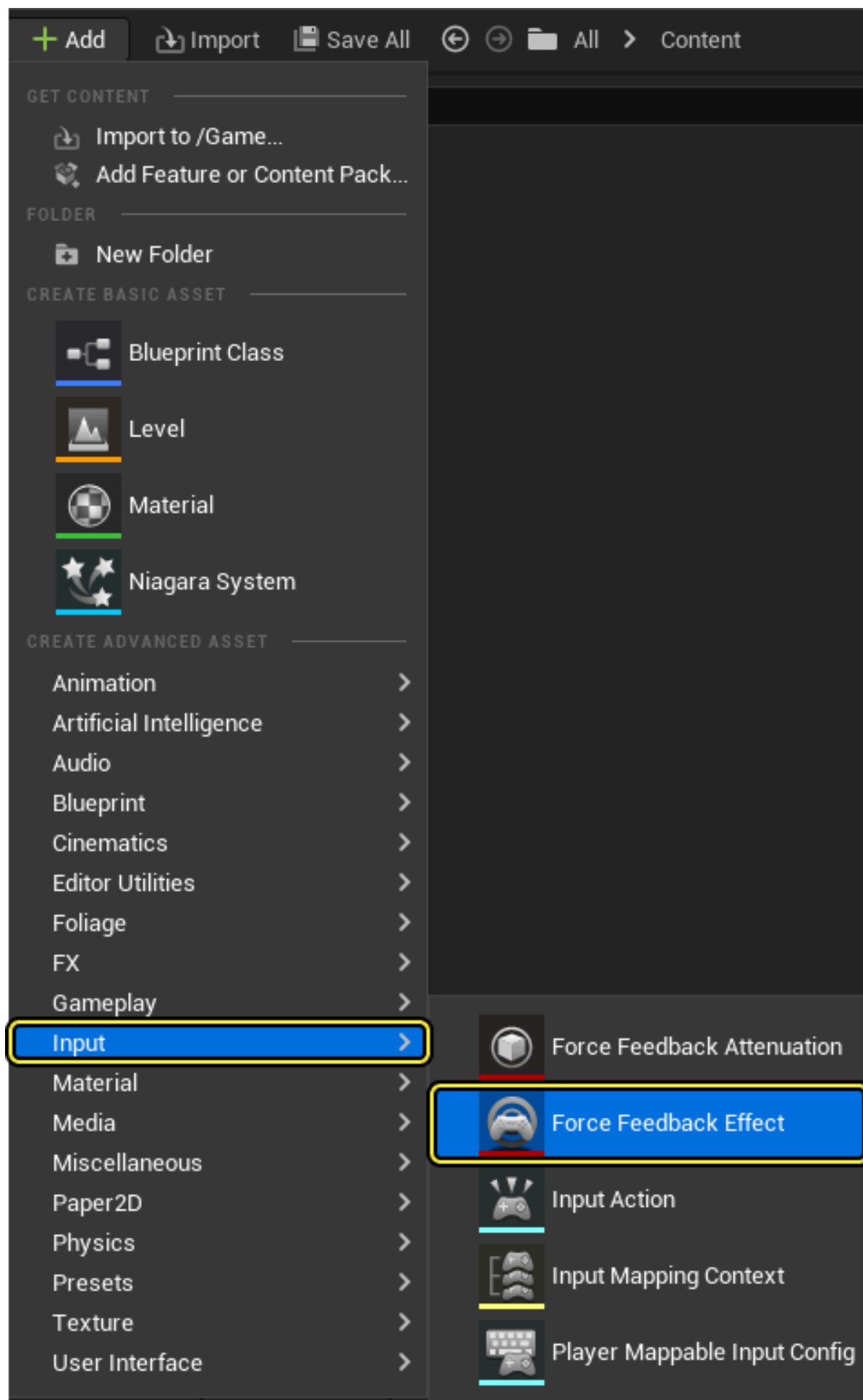
## Duration

The duration of the force feedback effect is calculated automatically based on the position of the last key in the curves for all channels. For example, if there are 3 channels and the last key in each set to a value of `1.25`, `1.5`, and `1.75`, then the duration for the overall effect is `1.75`

# Create a Force Feedback Effect Asset

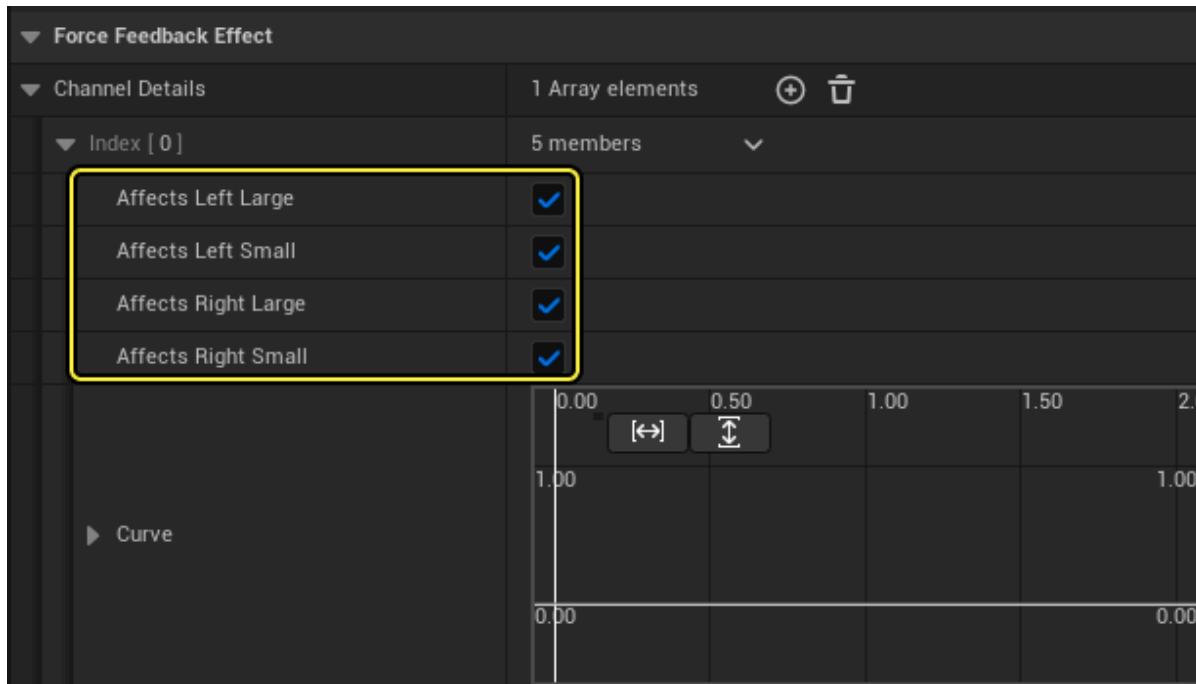
Force Feedback Effect assets are created using the **Content Browser**:

1. In the **Content Browser**, click **Add** and choose **Input > Force Feedback Effect**. Open the asset you have just created.

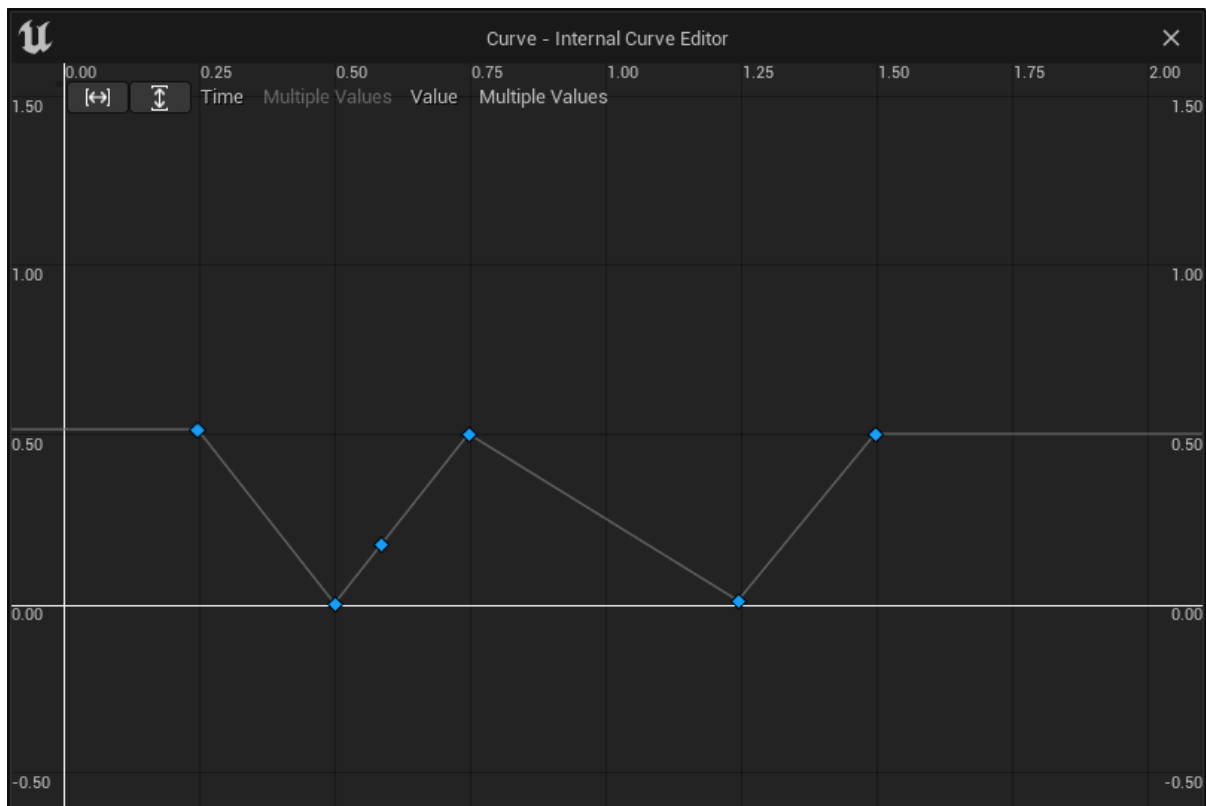




2. By default, the asset has one channel but you can add more. For each channel, select the combination of the four outputs that you want the channel to affect.



3. Hold **Shift** and click the **Left Mouse Button** on the curve to add one or more keys.



4. Manipulate the keys by entering values directly or dragging them in the curve editor.

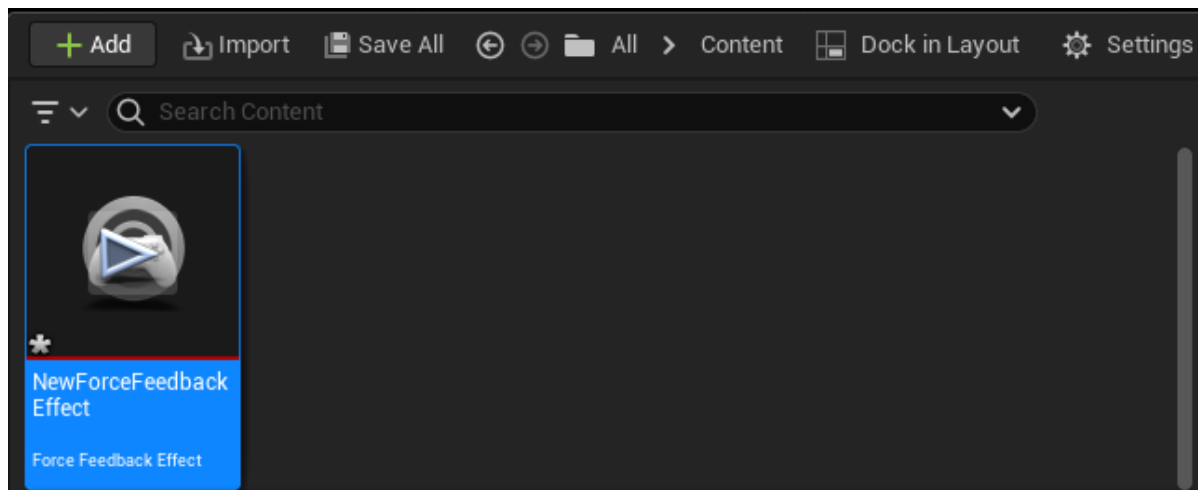


To adjust the curves between the keys, right-click on the curve segment to change its curve function, then adjust the tangent lines.

# Playing Force Feedback

## Preview in Editor

You can preview your Force Feedback Effect in the editor by clicking the "play" button that appears in the middle of the Force Feedback Effect's icon when you mouse over it.

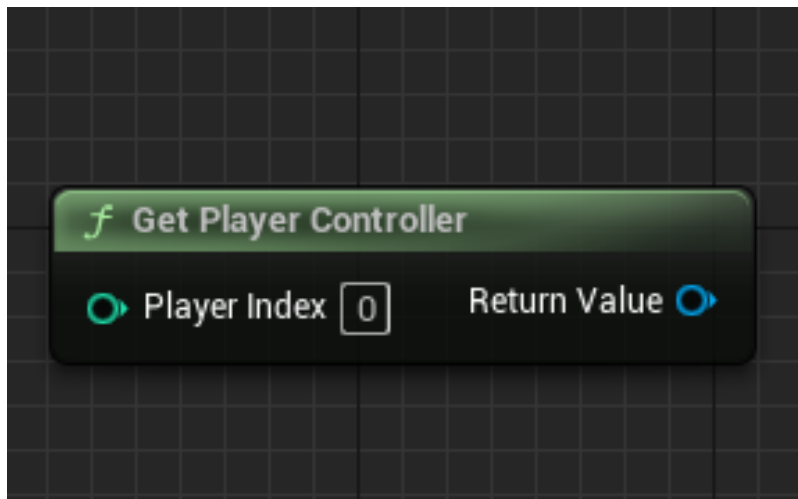


## Directly to a Player

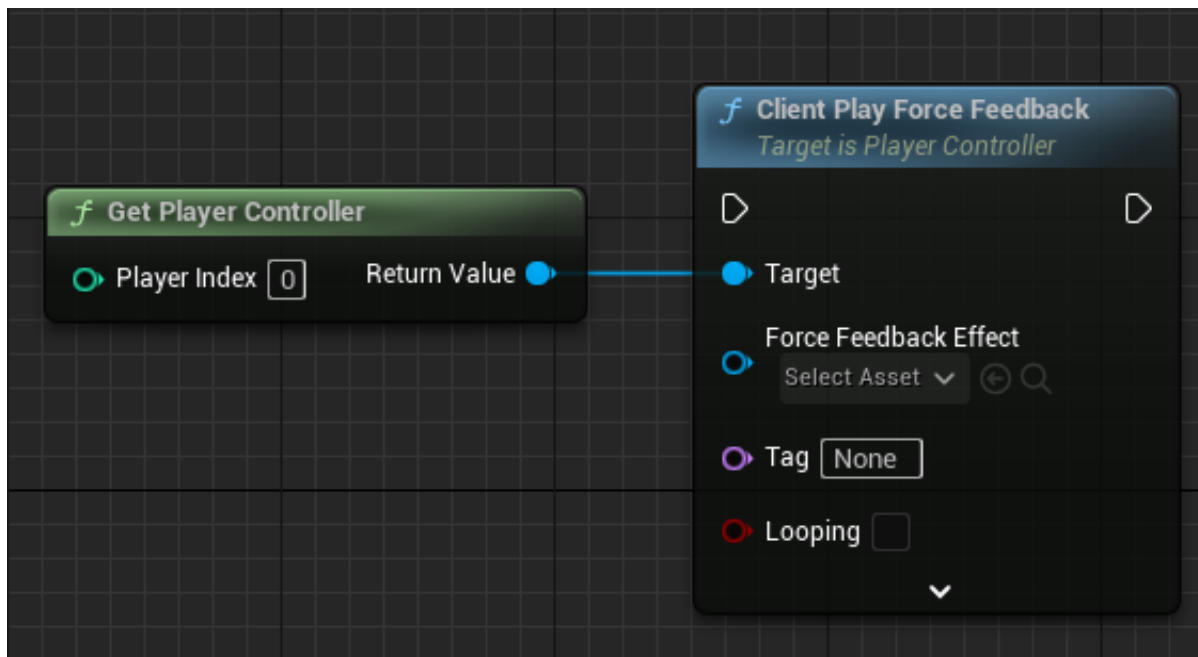
Force Feedback is implemented in the base `PlayerController` class. You will need access to the local Player Controller in order to play Force Feedback on the target device or controller.


## Playing Force Feedback in Blueprints

1. Get a reference to your Player Controller, either with a **Get Player Controller** node or a saved reference.

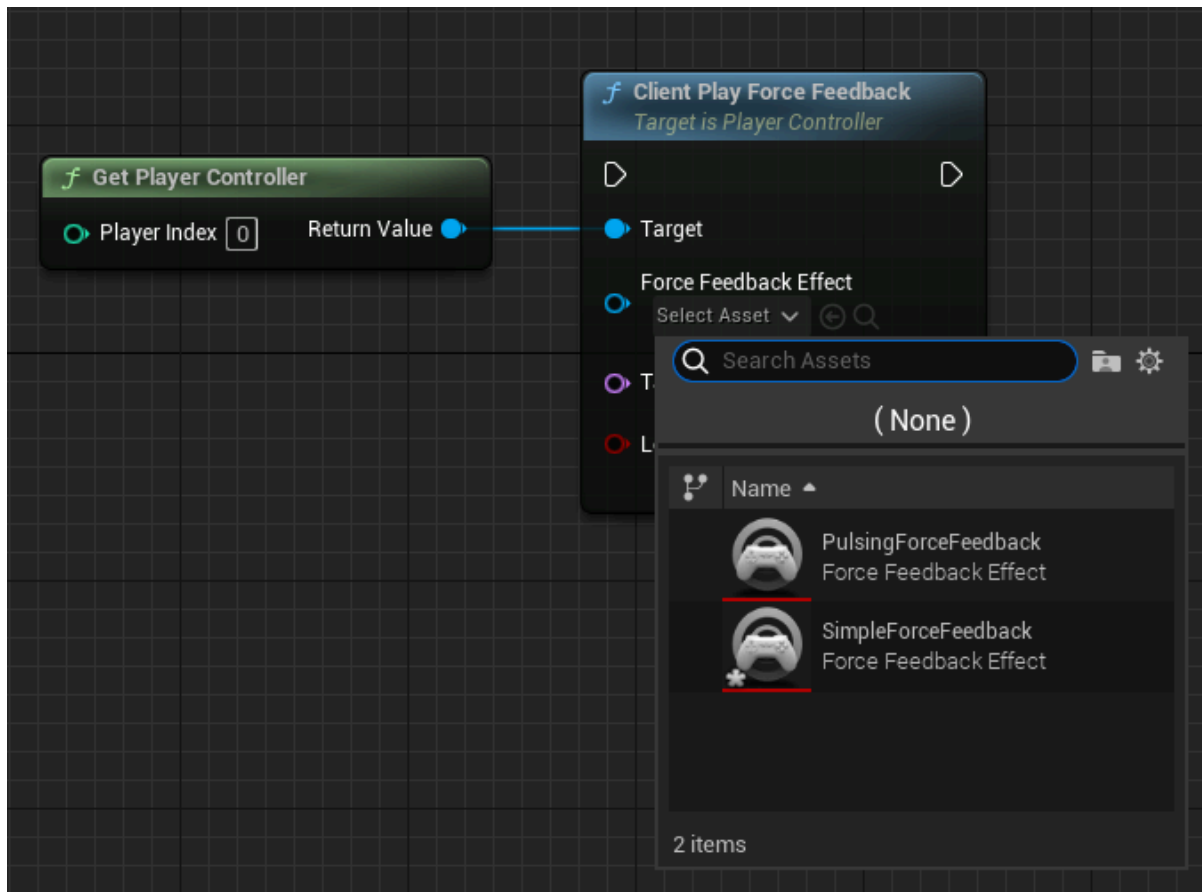


2. Drag off of the output pin of the reference, then type `Play Force Feedback` into the context menu and select **Client Play Force Feedback**.

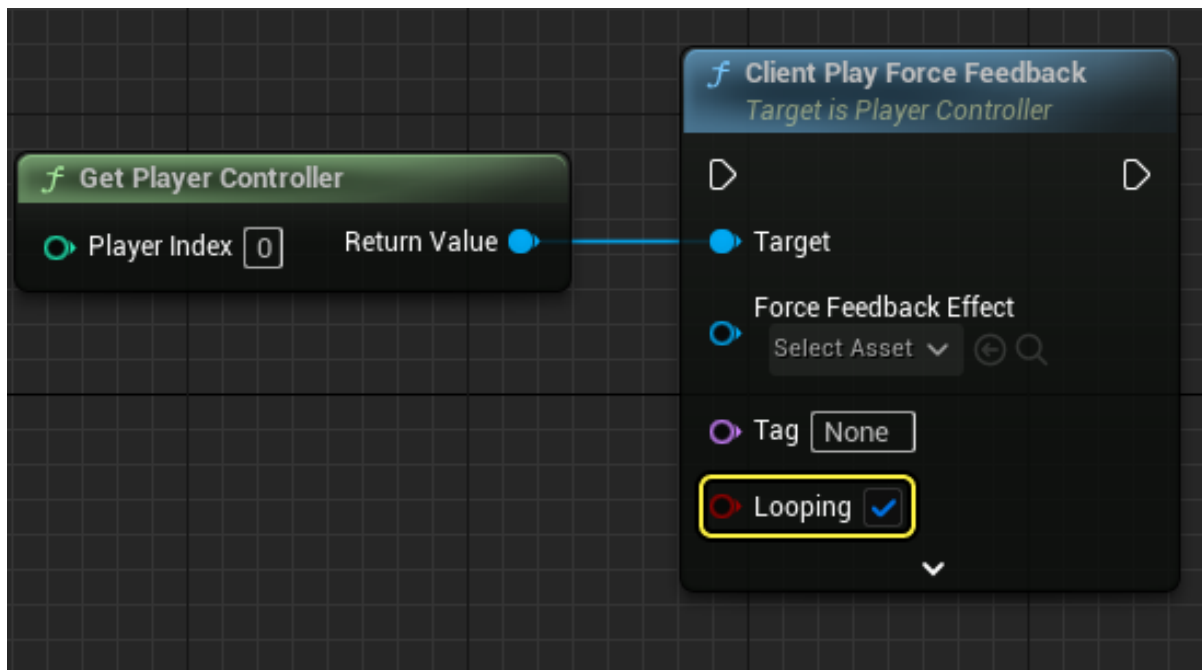


 The force feedback will be replicated to the owning client if called on the server.

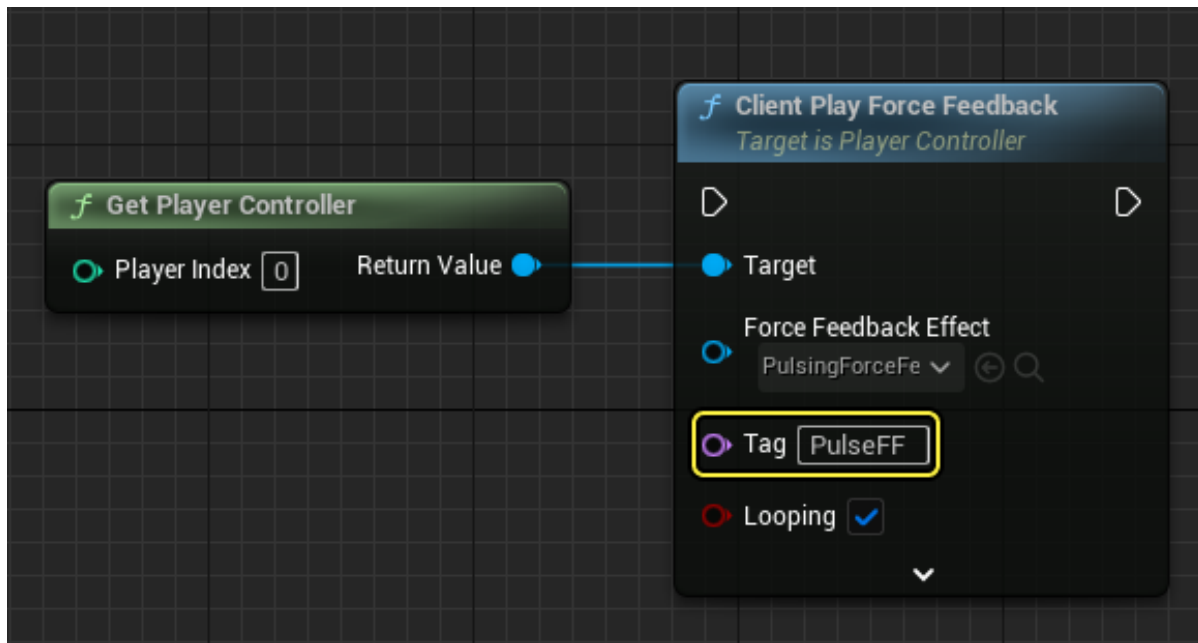
3. Specify the Force Feedback Effect to use directly on the node or with a connected variable.



4. Check **Looping** if you want the effect to loop.



5. Optionally, set a unique name for the effect in the Tag field. This feature allows you to stop the effect; if an effect with the same name is already playing, it is stopped and the new effect plays instead.



## Playing Force Feedback in C++

You can call `ClientPlayForceFeedback` on the local Player Controller.

```
1 void ClientPlayForceFeedback
2 (
3     class UForceFeedbackEffect * ForceFeedbackEffect,
4     FForceFeedbackParameters Params
5 )
6
```

 Copy full snippet

You can then use the Force Feedback Effect, specifying whether or not the effect should loop, and optionally choose a name for the effect. If a name is provided, and another Force Feedback Effect with the same name is played before the original effect ends, the original effect stops immediately and the new effect plays instead.

## At a World Location

You can place a [Force Feedback Component](#) in the world location of the intended source effect. This plays a Force Feedback Effect that changes intensity based on distance from the player observing it.

A Force Feedback component plays a Force Feedback Effect on command, but also has a physical location in the world. Like sound or light, the intensity of the force experienced by the player changes with the player's distance from the source according to a data-defined attenuation setting.

Force Feedback Components can be attached to any Actor from the source code or Blueprint and added dynamically during gameplay. You can also use the following utility functions:

- `UGameplayStatics::SpawnForceFeedbackAtLocation`: spawn at a given world location
- `UGameplayStatics::SpawnForceFeedbackAttached`: attach to a specific pre-existing component

These functions return the spawned Force Feedback Component, so you can continue to manipulate it. However, if you have no use for the component after it finishes playing the effect, use the Auto Destroy boolean option to remove the component once the effect ends.