Cinemachine Impulse User Guide

Overview

Impulse is a module for generating and managing **camera shakes** in response to game events. There are three parts to Impulse:

The Raw vibration signal. This is just a vibration curve in up to 6 dimensions: X, Y Z, pitch, roll, yaw. It vibrates as a function of time. The curve itself may be hand-crafted, recorded, or generated procedurally. Usually these raw signals are assets in the project.

The Impulse Source. This is a component that emits a raw vibration signal from a point in space. Game events such as collisions or explosions can trigger an Impulse Source to emit a signal from the spot where the event occurred. The impulse source makes use of a Raw signal and defines a time-envelope for the signal (attack, sustain, and decay) so that the signal fades in and out to the appropriate intensity and has a finite duration. It also defines how the signal dissipates with distance from the source: the farther away you go, the weaker it gets.

The Impulse Listener. Impulse Signals don't do anything on their own. They just exist, much like a magnetic field. The Listener is a component that can be attached to a virtual camera that listens for signals emitted by Impulse Sources. It then applies those signals to the camera's transform, causing the camera to vibrate in response to the signal. How the signal is interpreted can be tuned here. Some cameras are mounted more rigidly than others, and so may vibrate differently in response to the same impulse signal.

Using Impulse

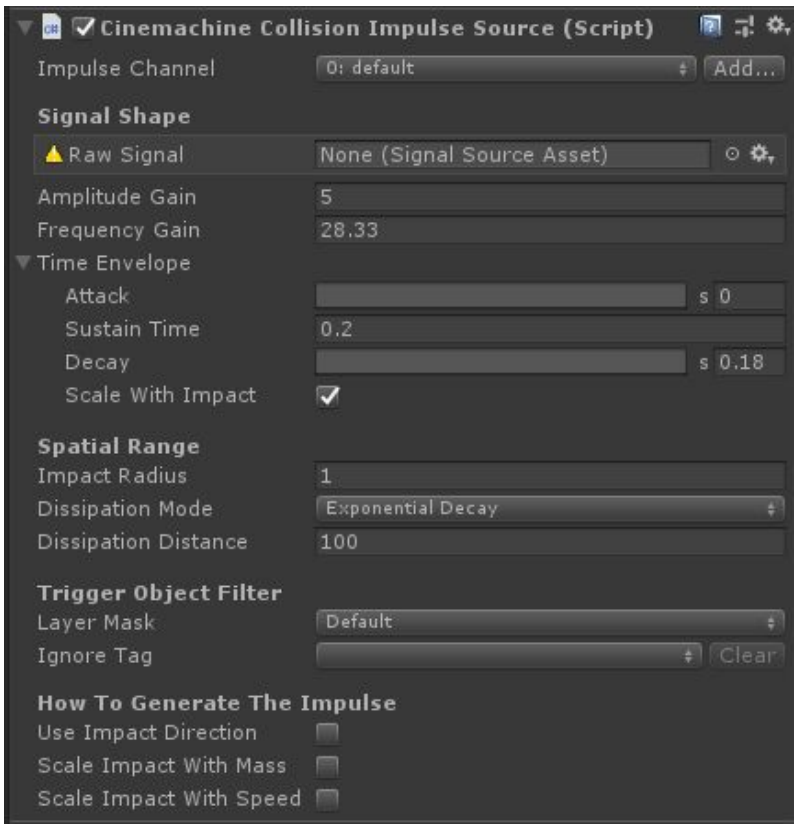
1. Create an Impulse Source

Each impulse source will emit impulses from its location in response to game events. You can have as many impulse sources as you like. For example, you could put them on the feet of a giant, so that the ground shakes when she walks, or on projectiles that explode when they hit, or on the surface of a gelatin planet, that wobbles whenever anything touches it. And so on.

Collision-based impulses

If you want an impulse to be generated when something collides with an object, or enters a trigger zone, attach a **CinemachineCollisionImpulseSource** component to the GameObject that has the collider. This component will generate an impulse when the physics system detects that an object collides with it, or enters the trigger zone (if the collider is a Trigger type). It works with Collider, and Collider2D as well.

Here is the inspector for CinemachineCollisionImpulseSource:

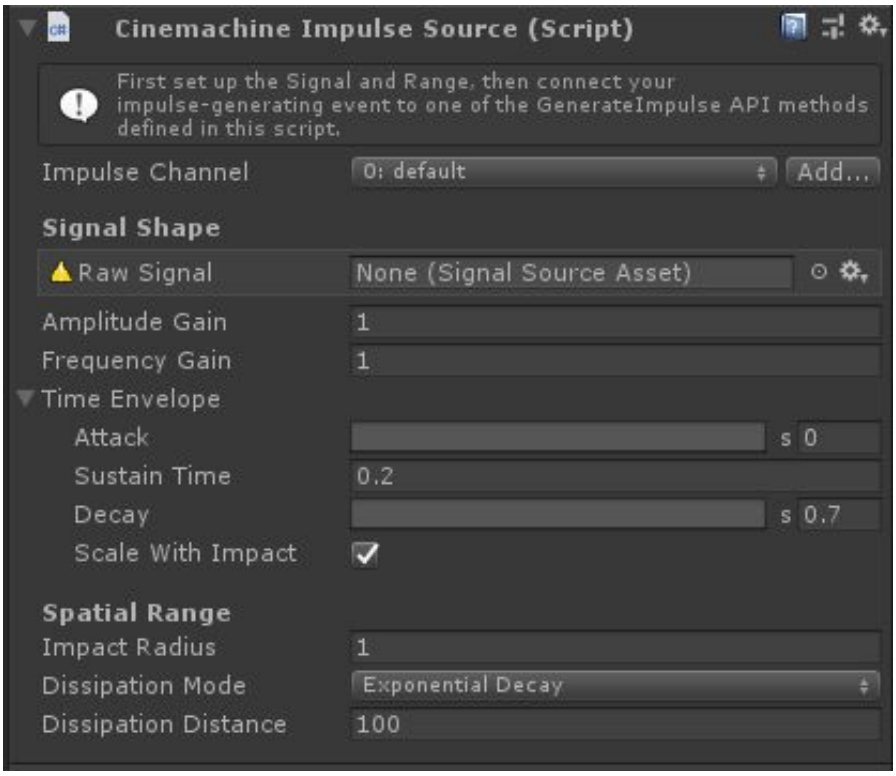


There are 5 sections in the inspector:

1. **Impulse Channel**. The Impulse Channels work like Camera Layers (but are distinct from them). You can broadcast Impulses on specific channels. Create new channels or edit them using the “Add…” button. Listeners can filter based on channels and so control which impulses they react to.
2. **Signal Shape.** Connect this to a Raw Signal source (more on that later) to provide the raw vibration curve. Next, set the amplitude and frequency Gain values to scale the strength of the curve and the curve’s time clock. Scaling the amplitude makes the curve move more, scaling the frequency makes it vibrate faster. A value of 1 means it’s unscaled. Finally, set up the time-envelope. This is a compound curve forming an envelope into which the raw signal is squeezed. That way you control fade-in, how long it lasts, and fade-out of the signal.
3. **Spatial Range.** The impact Radius is the radius around the impact point over which the signal is felt at full strength. Beyond that the signal strength fades out over the dissipation distance. After that, the strength is zero. So the total effect radius of the signal is impact radius + dissipation distance.
4. **Trigger Object Filter.** This allows you to control which objects will trigger the impulse when they collide or enter the trigger zone. Only objects on the Layer Mask will be noticed, except for objects tagged with the “Ignore Tag” value.
5. **How to Generate the Impulse.** This controls whether the mass and velocity of the impacting object are taken into account when generating the strength and direction of the signal. If they are not taken into account, then a constant signal will be generated, regardless of the impacting masses and velocities. Masses and velocities are taken from the RigidBody (or RigidBody2D) components of the colliding objects.

Event-driven impulses

Alternatively, if you want to generate impulses on events other than collisions or Collider triggers, you can use the simpler **CinemachineImpulseSource** component. This is a generic impulse source that contains an impulse definition (the first 3 inspector sections described above), and exposes a family of GenerateImpulse() API methods that you can call directly from your game logic, or hook up to UnityEvents in the inspector. Calling those methods will generate Impulses at the specified locations and with the specified velocities and strengths.



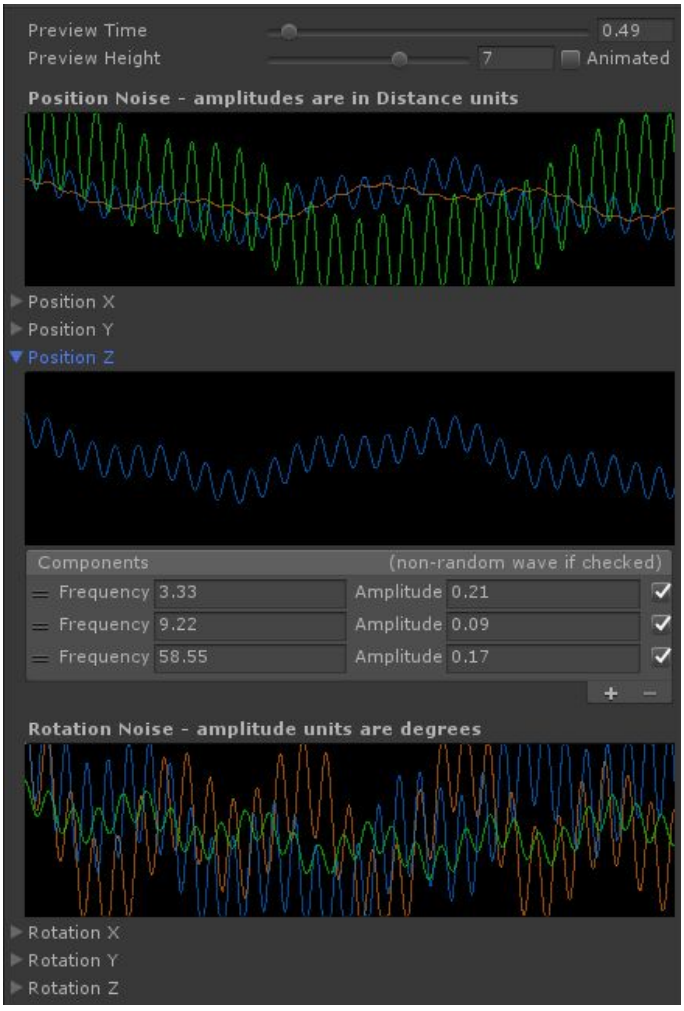
2. Create a Raw Signal Source

This is a signal that can be procedurally generated, recorded, or hand-crafted. Two asset classes for this are included with Cinemachine.

Procedural Signals

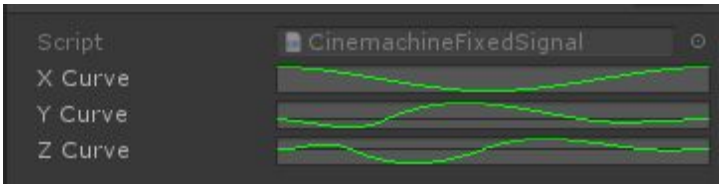
The **NoiseSettings** asset that comes with Cinemachine (used for Basic Multi-Channel Perlin noise on cameras) can be used as a procedural raw signal generator. Any camera NoiseSettings asset can be set as a raw signal, and it will pump its noise into the Impulse event.

However, most camera noise is defined only on the rotation axes. For Impulse, often you will want to have positional noise as well. You can create your own NoiseSettings that provide exactly the procedural signals you need. The NoiseSettings inspector is a very powerful tool for this:



Hand-crafted Signals

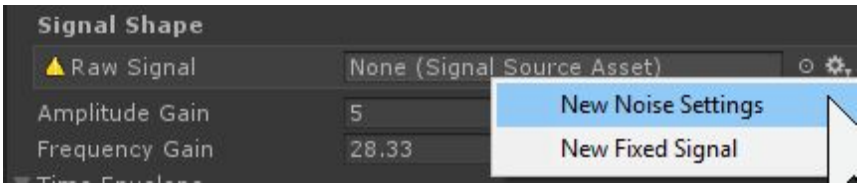
The **CinemachineFixedSignal** asset class is a simple 3D curve definition that lets you draw the curves yourself. The inspector UI is self-explanatory:



The Impulse generator will either stretch the drawn curve to fit the desired time envelope, or loop it, according to your specification in the impulse source.

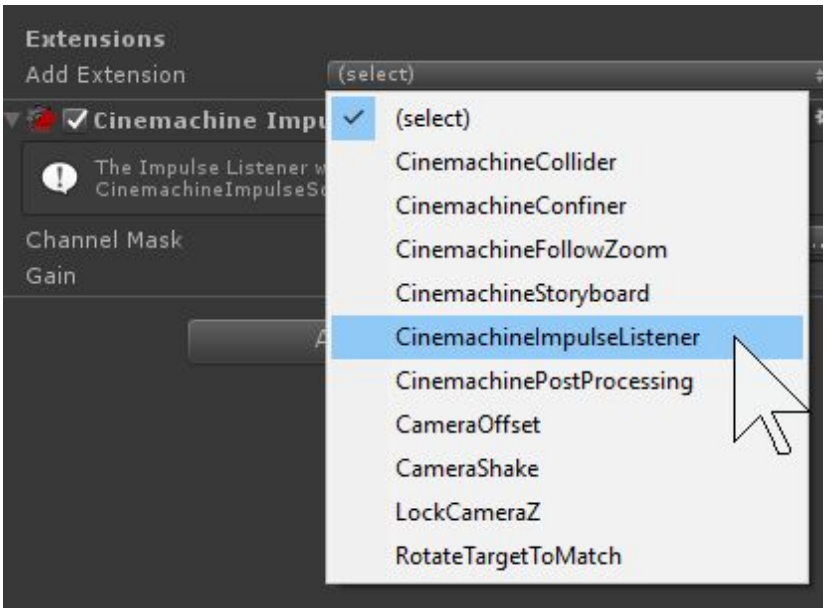
Custom Signals

You may create your own custom signals as well. If you derive them from the **CinemachineSignalSource** base class, they will appear automatically in the inspector menu alongside the built-in ones:

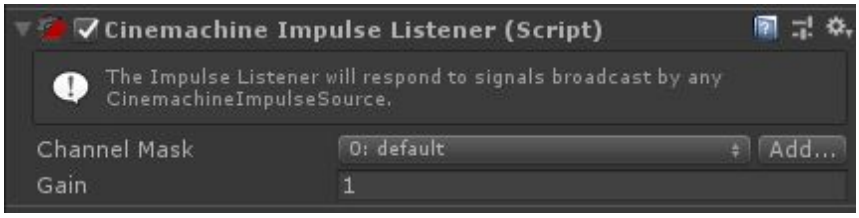


3. Create an Impulse Listener on your Virtual Camera

An Impulse Listener is a Virtual Camera extension. Add one to your virtual camera from the extensions menu in the inspector:



The listener has a channel mask that allows the virtual camera to filter which Impulse channels affect it. In the simplest case, the listener simply applies the signal verbatim to the camera's transform, causing it to shake. It also has parameters that control how the signal is interpreted in order to affect the camera's transform in a less literal fashion.



At present, the controls are quite limited (only Gain to control the intensity), but in the future this will be expanded. And as always, you can create your own listener to interpret the the signal any way you like.