Harmonix Plugin

An overview of the musical plugin.

Learn to use this **Experimental** feature, but use caution when shipping with it.

The **Harmonix** plugin provides the features, MetaSound nodes, and components we used to make Patchwork and Fortnite Festival. These tools utilize MIDI sequencing, audio analysis, and musical synthesis so you can create your own music-driven experiences.

You can enable the Harmonix plugin by doing the following:

1. Select **Edit > Plugins** to open the **Plugin** panel.
2. In the **Plugin** panel, use the search bar to find the Harmonix plugin.
3. Click the corresponding checkbox.
4. Restart Unreal Editor.

General Features

MIDI File Importing

You can import MIDI files into your project by dragging and dropping them into your **Content Browser**.

Fusion Patch Assets

Fusion Patches are assets that map a range of MIDI notes and velocities to Sound Wave assets. They drive the Fusion Sampler MetaSound node's instrument sampler functionality.

To create a Fusion Patch asset, do the following:

1. In the **Content Browser**, select each Sound Wave you want to map.
2. Right-click on one of the selected Sound Waves.
3. Select **Create Fusion Patch** from the context menu.
4. In the **New Fusion Patch Options** window, set the desired options.
5. Click **OK**.

There are example Fusion Sampler Patch assets in `Engine\Plugins\Runtime\Harmonix\Content\Examples\Patches`. You can find these in the Content Browser by enabling **Show Engine Content** and **Show Plugin Content** in the Content Browser's settings.

Peak Tamer

When working with audio-reactive visuals or gameplay, getting a smoothed value within the range [0.0, 1.0] from an audio-energy level or other sound-related analysis value is often useful. While it's possible to do this within MetaSounds using Compressor or Envelope Follower nodes, it can be expensive to do on the audio thread. The Peak Tamer works on the game thread to smooth and compress MetaSound Source outputs or other values.

There are two Peak Tamer types:

- `Harmonix::AudioReactivity::FPeakTamer`
- `UHarmonixPeakTamer`

Both types are functionally similar, but you can use `UHarmonixPeakTamer` from Blueprints. You can configure smoothing and compression settings for either Peak Tamer using `FHarmonixPeakTamerSettings` via the `Configure` method.

MetaSound Nodes

The Harmonix plugin contains a collection of MetaSound nodes for MIDI generation, filtering, and manipulation. The nodes are under **Functions > Harmonix** in the node selection context menu.

Learn more about each node and their pins by hovering over them and viewing their tooltips. You'll find additional details on a selection of the nodes below.

Fusion Sampler

The Fusion Sampler node functions as an instrument sampler by rendering an input MIDI stream using a given Fusion Sampler Patch. The patch maps MIDI notes to keyzones so the audio samples are played for the corresponding MIDI notes in the input stream.

> The Fusion Sampler node supports multi-threaded rendering. Click the expansion arrow at the bottom of the Fusion Sampler node to find the input pin.
>
> To support multi-threading, you must connect the node's audio and render sync outputs to a Fusion Synchronizer node and reference that node's audio outputs elsewhere in the graph.

Step Sequence Player

The Step Sequence Player node produces a stream of MIDI events by using a MIDI Clock, a Music Transport, and a MIDI Step Sequence asset. A MIDI Step Sequence is a 2D grid of cells where the rows represent a note, and the columns represent a unit of time.

> You can modify the MIDI Step Sequence asset while the Step Sequence Player node plays.

Clock-Synced Delay

The Clock-Synced Delay node applies a delay effect to the input audio signals in sync with a given MIDI clock. This differs from the default Delay MetaSound node by producing a musically-timed audio effect.

Components

The Harmonix plugin provides the following Blueprint-accessible components under the **MetaSoundMusic** category:

- Music Clock
- Music Tempometer

Music Clock

The Music Clock component drives musical timing and synchronization in C++ and Blueprint to support rhythmic gameplay systems, animations, and more.

A Music Clock is primarily used to synchronize gameplay events and graphics rendering to musical time rendered asynchronously in a MetaSound graph, but you can also synchronize with a basic tempo and time signature.

When driven by a MetaSound, the clock's time-related get methods provide one of the following time contexts:

- `AudioRenderTime` - Gives you a smoothed position of where the audio renderer is. This is useful for queuing up music events based on the current song time. This time does not account for how long the rendered music takes to arrive at the player's ears.
- `ExperienceTime` - When properly calibrated, tells you what the player is hearing and seeing. This is useful for scoring player input.
- `VideoRenderTime` - When properly calibrated, tells you what should be drawing right now to appear in sync with the music. This is useful for synchronizing animations, UI, and other visuals to the music.

Music Clocks have transports separate from the playing MetaSound. This is useful for pausing game systems synchronized to the clock while the underlying music continues to play. For example, you can pause the clock during a player interaction with the game UI and then continue the clock when the interaction is complete. In that case, when you continue, the clock catches back up to the current time rendered by the MetaSound.

Music Tempometer

The Music Tempometer component provides playback properties of a Music Clock component on its Actor and optionally updates a Material Parameter Collection.