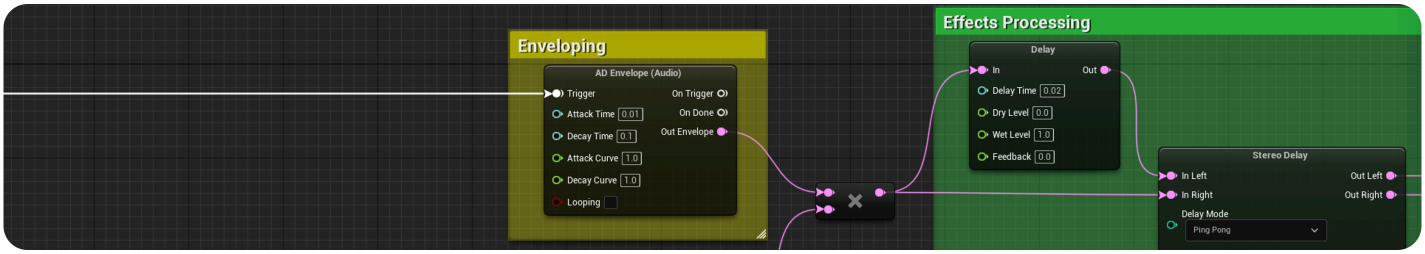
Developer
/ Documentation
/ Unreal Engine
/ Unreal Engine 5.4 Documentation
/ Working with Audio
/ Sound Sources
/ MetaSounds
/ MetaSounds: The Next Generation Sound Sources

MetaSounds: The Next Generation Sound Sources

An overview of MetaSounds in Unreal Engine 5.



Attack Time 0.01
Decay Time 0.1
Attack Curve 1.0
Decay Curve 1.0
Looping
Delay Time 0.02
Dry Level 0.0
Wet Level 1.0
Feedback 0.0
Delay Mode Ping Pong

Introducing MetaSounds

MetaSound is a high-performance audio system that provides audio designers complete control over a Digital Signal Processing (DSP) graph for generating sound sources.

MetaSounds offer user customization, third-party extensibility, graph re-use, and a powerful tool for in-editor sound design.

A Fully Procedural Audio Engine

Unlike Sound Cues, MetaSounds are fundamentally a Digital Signal Processing (DSP) rendering graph. With them, audio designers can construct powerful procedural audio systems that offer sample-accurate timing and control at the audio buffer level.

Using MetaSounds, audio designers can generate audio synthetically at runtime and freely mix and match procedurally generated sound with other audio sources.

We also designed MetaSounds to easily integrate with game data and player interactions to create immersive experiences triggered by gameplay events.

More Control for Audio Designers

Each MetaSound is its own audio rendering engine. They render in parallel with each other and have potentially independent rendering formats (e.g., sample rate, buffer size, channel count).

MetaSounds are created in a new **MetaSound Editor** where audio designers with no programming experience can create procedural sounds using a node-based interface. The editor enables the live previewing of all audio input parameters and contains several ready-made nodes that provide detailed control options for the entire audio rendering pipeline.

The MetaSound editor features a real-time meter on the output, in-graph widgets to control and visualize parameters (knobs and sliders), and buttons to interact with events in real-time.

Sample-Accurate Control of Audio

MetaSounds offer sample-accurate control of audio sources. Sample accuracy refers to the ability for timing to be on the order of a single audio sample. In other words, if the sample rate is 48,000 samples per second, a sample-accurate event would have a timing resolution of 1/48,000 seconds, or 0.02 milliseconds.

MetaSounds support sample-accurate control in several ways. MetaSound triggers execute sample-accurate events within the graph. Triggers can originate from gameplay events, MetaSound nodes, or the graph itself.

The MetaSound Wave Player node features sample-accurate concatenation. This means that when the playing sound wave finishes playing, it will seamlessly play the next queued-up sound wave without audible hitches or interruptions.

Many MetaSound node parameters can be modulated by audio buffers, allowing for "audio-rate" parametric modulation. This enables powerful synthesis and sound design techniques.

Improved Workflow

With MetaSounds, you can reuse and reference their graphs through presets. Presets reference an existing MetaSound graph, and you can override the graph's inputs.

This can result in a significant boost in productivity as you now have a single graph with different presets that reference a base graph. The presets will contain the unique differences for the same base graph. Without presets, you would have to manage hundreds of potentially similar graphs to account for different variations.

Another benefit of using presets is the ability to update the base graph and have the changes automatically propagate to any preset that references that graph. This can also result in significant productivity gains throughout the development process.

In addition to presets, MetaSounds also features graph composition, which is the ability to use MetaSounds directly inside other MetaSound graphs. Custom MetaSound nodes can also be created and used inside any other MetaSound graph. These custom nodes can define their inputs and outputs, and provide tooltip and versioning information.

This way, you can build up a library of common functionality nodes and reuse them across multiple graphs. Similar to presets, custom MetaSound nodes will automatically propagate their changes across any MetaSound that references them, further increasing productivity in a complex and changing project.

Increased Performance

MetaSounds are rendered asynchronously to the main Audio Mixer using the same architecture and tasks used for asynchronously decoding sound sources.

Each MetaSound DSP graph automatically converts to an optimized static, non-virtual C++ object, with data between nodes passed by reference and not copied. This avoids common disadvantages to this type of system, such as interpreted bytecode runtime, expensive virtual function overhead, and data copying.

Growing Node Library

MetaSounds ship with a growing library of MetaSound nodes, which provide various powerful options for procedural sound design and music.

The library features a rich Wave Player node, which supports seeking, loop points, sample-accurate concatenation, pitch-scale modulation, and reading from cue points in an audio file.

Other nodes include a diverse library of Trigger utilities, DSP math operations, DSP filters, dynamics processing, spatialization, real-time synthesis generators, and so on.

Portable and Extensible

The MetaSound system can be extended by third-party plugins using its C++ Node API.

Creating a new node for the MetaSound Editor involves creating a C++ class where the programmer defines the node's inputs and outputs, and an execution callback. This class also contains the actual audio rendering code and logic. You can write a new node in a single '.cpp' file in only a few hundred lines of code.

Rich Gameplay Interactions

MetaSounds can accept custom user input parameters similar to those used in Materials and Niagara VFX systems. These parameters can be connected to gameplay systems through the Audio Component parameter interface, either directly in Blueprint or gameplay C++ code.

MetaSounds have also been integrated with the Audio Modulation plugin, an optional plugin that provides a system to write into a modulation bus asset from any modulation source (including from Blueprint). MetaSounds can now read from the parameter buses, which means you can use them to modulate any MetaSound system.