Integrating Csound into Unreal Engine for Enhanced Game Audio

AuthorA[1] and AuthorB[2] *

[1] InstituteA
[2] InstituteB
your.email@yourdomain.com

Abstract. Unreal Engine is one of the most widely used game engines in the current market, thanks to its exceptional flexibility and strong graphical capabilities. Recently, the development team has introduced a new tool called MetaSounds, designed to facilitate sound synthesis, digital processing and sound design in a native way and within a node-based interface.
Despite its user-friendly interface, MetaSounds still lacks certain functionalities present in older sound engines such as Csound or SuperCollider. Currently, integrating Csound into Unreal needs the use of a middleware like FMOD or Wwise, along with Cabbage to export Csound code into a VST. However, a MetaSounds node that inherently incorporates Csound, without the use of external dependencies, and with MetaSounds adaptable, intuitive, and potent graphical interface would be a significant advancement. Thanks to Unreal Engine's support for C++ implementations and enabling developers to craft their own MetaSounds nodes, it can be possible to integrate Csound within a MetaSounds node through the Csound C++ API.

Keywords: Unreal Engine, game audio, video game, game engine, MetaSounds, Csound, plugin, MetaCsound, adaptative audio, adaptative music

1 Introduction

With the gaming industry expanding rapidly, it is not surprising that studios are constantly pushing boundaries once thought impossible. In the last decade, significant effort has been put into audio, specifically with realistic spatialization and adaptive audio, both in real-time. Adaptive audio is described as audio or music whose characteristics change in response to events in the game.

Game engines are one of the main tools in game development. They provide a set of commonly used tools in video games, such as rendering, physics, sound, and graphical user interfaces, all optimized for real-time execution. Several solutions have been created to allow adaptive audio in these engines. Developers usually rely on multiplatform middleware such as FMOD or Wwise, which can be integrated into the project or engine, although integration solutions have already been made for the most popular engines.

As Csound already has many features that are very useful for adaptive audio and music, this work focuses on creating a tool that enables Csound inside Unreal Engine.

2 Unreal Engine and Metasounds

Unreal Engine, developed by Epic Games, is one of the most popular engines at the moment. Unreal has established itself as an industry standard in areas such as realistic rendering and audio. Although it is coded in C++ and allows developers to code games in C++, it also supports the use of blueprints, a node-based programming interface. This makes the engine accessible to people who are not specialized in programming. Furthermore, it is free to use from the start and only charges developers once the game starts generating revenue.

Recently, Epic Games has released a new feature in their engine called MetaSounds. Currently in beta, MetaSounds is a high-performance audio system that provides audio designers with complete control over a digital signal processing graph to generate sound sources. Similar to Unreal's blueprints, it uses a node-based interface, making the connections between processing units, such as oscillators and filters, very easy and approachable. MetaSounds allows developers to perform complex digital signal processing (DSP) operations without the need for middlewares.

* Please place acknowledgement here.

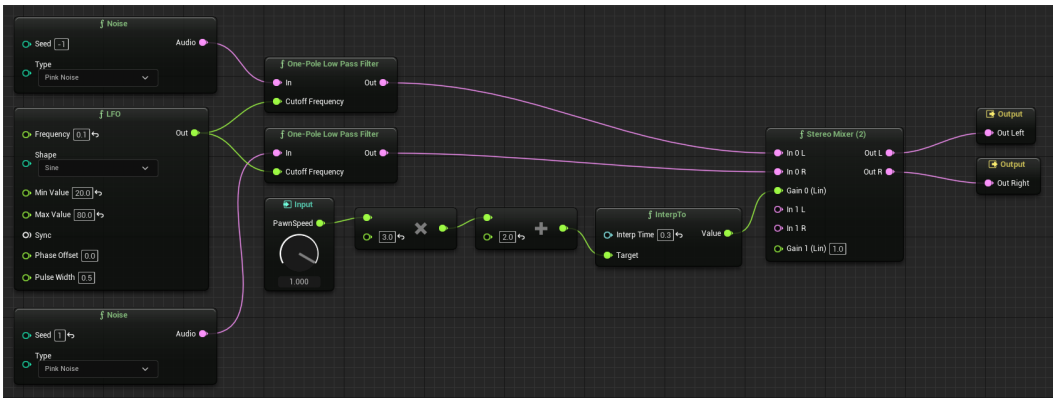

Fig. 1. MetaSounds graphic interface.

In a typical MetaSounds graph, the input nodes are created on the left side of the graph. These inputs can be graph variables that are written by other blueprints, making it easy to map them to events happening in the game, such as the player moving, a new enemy appearing, or a car passing by. In the center of the graph, the DSP nodes are placed, interconnected with each other, the graph inputs, and the graph variables. On the right side, the outputs of the graph are placed. Usually, there will be one audio output channel if the graph is meant for mono audio, or two output channels for stereo audio. An example of a MetaSounds graph is shown in Figure 1.

Despite the strong foundations and potential that MetaSounds has, it is still a very young environment and lacks many functionalities. While you can perform basic tasks such as adding a low-pass filter or pitch shift, it pales in comparison to older audio environments. However, if we could create a new MetaSounds node that runs an instance of Csound inside, receiving audio and control-rate inputs from other MetaSounds nodes and allowing outputs to be sent to other nodes, we could provide developers with far more options.

3 MetaCsound

The scope of this project is to create a new plugin for Unreal Engine, specifically for the MetaSounds module. The plugin, called MetaCsound, aims to integrate Csound into the MetaSounds. This is achieved by adding a new family of nodes. Since MetaSounds allows developers to create their own nodes in C++, and Csound offers a C++ API, integration is possible.

This family of nodes will have different vertex interfaces but will function the same at their core.

3.1 Vertex Interface

The vertex interface refers to the input and output pins in MetaSounds that can be interconnected from one node to other nodes. In terms of Csound, these will correspond to audio channels and control busses.

In Csound, score events are defined by a string, but there is no equivalent of an event like this in MetaSounds. Therefore, two MetaSounds vertexes will be needed to implement score events. One will be a string input containing the score event, and the other will be an input trigger that signals when the event described by the string should be triggered in Csound with sample-by-sample accuracy.

Additionally, one last input must be included in the module. This input will contain a reference to the Csound file to be executed in the node.

3.2 Node behaviour

When the node starts running inside the Unreal Environment, a new Csound instance will be created thanks to the Csound C++ API, and it will compile the file passed by reference. During each MetaSounds buffer call, the audio and control inputs will be read and written to the Csound input channels and buses accordingly. Then, the audio output channels and the control output buses from

Csound will be read and written to the MetaSounds output vertexes. Since MetaSounds and Csound may work with different buffer sizes, the Csound call to perform the next block will only be triggered by MetaSounds when the node detects that the buffer has been emptied and requires new data.

One key thing to note is that in order to ensure satisfactory integration between MetaSounds and Csound, they both have to operate with the same samplerate. Therefore, the samplerate of Csound will be overridden to match the samplerate of MetaSounds.

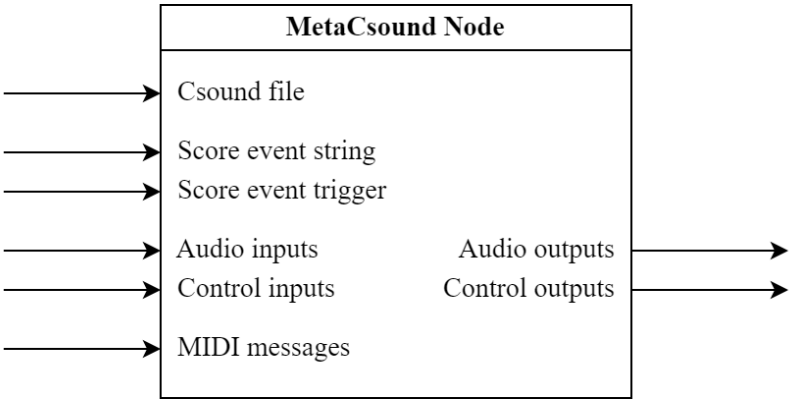

Fig. 2. New MetaCsound node layout.

3.3 Design iterations

When initially designing the project, the plan included a dynamic vertex interface. This involved compiling a new Csound file when detected to scan for every audio channel and control bus and then mapping them to MetaSound's respective vertex equivalents in the vertex interface. However, at the time of writing this paper, implementing this feature seems very challenging due to MetaSounds being in beta and the documentation being incomplete. To address this limitation, different variations of the node will be created, each with a different static vertex interface. This will allow developers to choose the version of the node that best suits their needs. However, this approach will require the control buses of the Csound file to be named with specific names so that the MetaSound node can map the control bus and the vertex accordingly.

Another potential improvement is to add MIDI Messages as inputs for the Csound file. Since MetaSounds allows for MIDI messages, it would be beneficial to send these message inputs to the Csound instance. The layout of the final design is shown in Figure 2.

4 Discussion

At the time of writing this paper, the node is already working and can be used without any major bug. In the example shown in Figure 3, we can see how background music is sent as the audio input to the Csound node. Csound will modify the output depending on the opcodes used in the Csound file and the score events triggered by the system.

The impact of this project could be significant on future game development. Traditionally, adaptive audio in games has been handled by middleware such as FMOD or Wwise. While these middleware are powerful, they consume a lot of resources, and developers have to pay fees when the shipped game surpasses certain revenue thresholds. Additionally, integrating middleware into the engine can be tricky at times. Since MetaSounds is a module of Unreal, the integration is already done, requiring fewer resources and less work.

Prior to this project, attempting to utilize the capabilities of Csound inside Unreal required the use of one of the previously mentioned middleware. Developers would also have to import a VST created with Cabbage, containing the Csound code. For Unity Engine, the other most popular engine besides Unreal, a less convoluted option already exists, CsoundUnity, which allows Csound to be

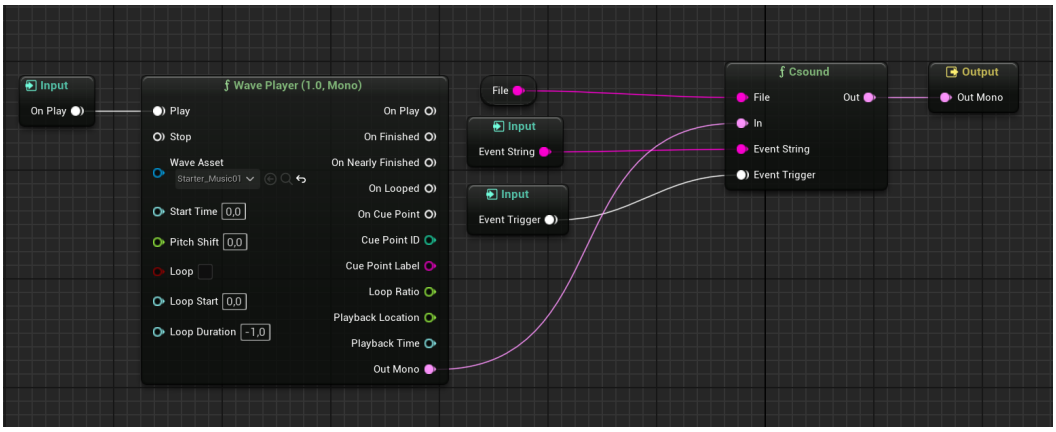

Fig. 3. Example of a MetasSounds graph using the new Csound node.

run inside the engine directly without the need for middleware and VST exports through Cabbage. Unreal Engine lacked a similar tool. With MetaCsound, Csound code can be easily run inside the MetaSounds module in a far more straightforward and user-friendly manner, making this project highly useful and relevant for both the Unreal and Csound communities.

5 Conclusions

At the time of this writing, the project is in a quite advanced state. The current prototype node is operational, with a Csound file passed as a string, mono audio input and output, and score event handling (see Fig 3). Next steps in the development process include implementing control buses, supporting multiple audio channels, integrating MIDI message inputs, and creating multiple versions of the node with different vertex interfaces while sharing the same core functionality. The project is already demonstrating its capabilities and great potential.

Hopefully, as MetaSounds establishes itself as a reliable tool and exits its current beta state, more developers will begin to use it, and the use of more convoluted paths such as middleware will be reconsidered for certain projects. If that is the case, MetaCsound could become a key tool for integrating complex DSP operations into the MetaSounds graph, making Csound one of the best audio programming options for adaptive sound in Unreal.

The source code of the project can be found in the following GitHub repository: `https://github.com/XXX/XXX`

References

1. Lazzarini, V., Yi, S., ffitch, J., Heintz, J., Brandtsegg, Ø., McCurdy, I.: Csound: A Sound and Music Computing System. Springer (2016)
2. Csound API documentation site, `https://csound.com/docs/api/index.html`
3. Epic Games: MetaSounds on Unreal Engine — Unreal Engine 5.1 Documentation, `https://docs.unrealengine.com/5.1/en-US/metasounds-in-unreal-engine`
4. Firelight Technologies: FMOD — FMOD for Unreal, `https://www.fmod.com/docs/2.02/unreal/welcome.html`
5. Walsh R.: CsoundUnity documentation site, `https://rorywalsh.github.io/CsoundUnity/`