MetaMIDI Toolkit Manual

Updated 25/07/24

The MetaMIDI Toolkit is a collection of Blueprints for Unreal Engine 5.4 to provide a more intuitive way to create adaptive music for Game Designers, while still carving out a large space for composition and customization by Game Composers.

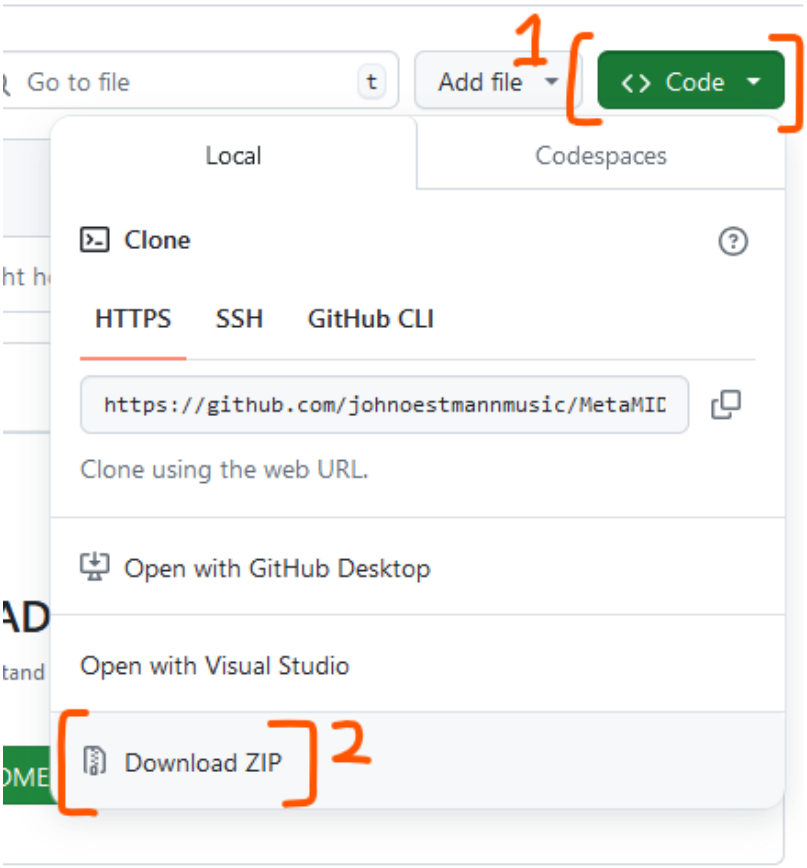
This manual is written aimed at a Game Designer without a music background. However additional notes have been included for the “Composer.” These notes go into more technical depth surrounding the musical and technical implementation of the Toolkit.

Table of Contents

Installation

1. Download Instructions

Go to https://github.com/johnoestmannmusic/MetaMIDI_Toolkit and download the toolkit by clicking the **Code** button, and then **Download ZIP**.

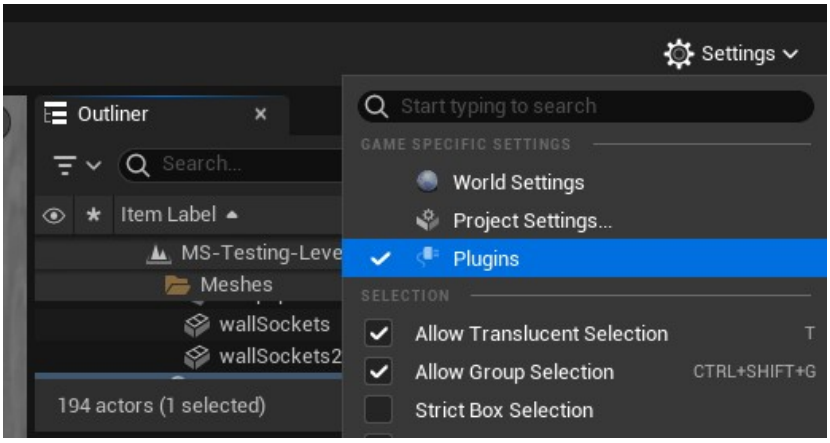


2. Enabling the Harmonix Plugin

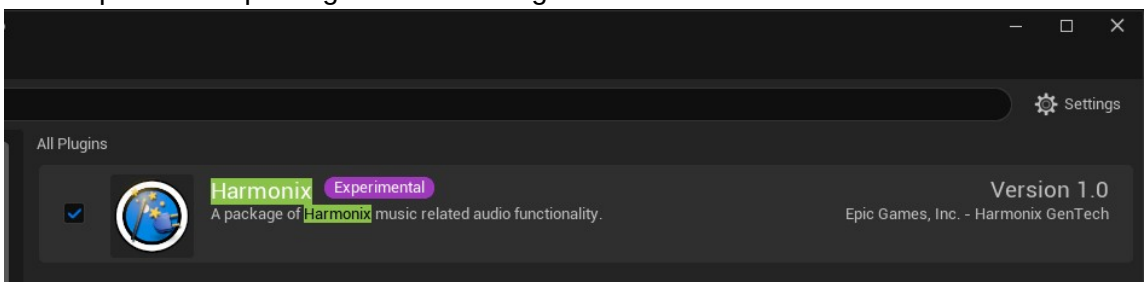
To use the MetaMIDI Toolkit, you will need to be using Unreal Engine 5.4 or later, and it needs to support the Harmonix plugin. In Unreal 5.4, this is included with the engine, however it needs to be turned on before you bring in the MetaMIDI files.

Turn this on by:

1. Clicking on **Settings > Plugins**



2. Searching for “Harmonix”, and turning it on. You may get a warning that it is still an experimental package and can’t be guaranteed to work in all situations.



3. Once that is ticked, bring the MetaMIDI folder into your project!

3. File Structure

Once you’ve added the **MetaMIDI_Toolkit** folder to your Unreal Engine project, you will find a number of files and folders inside. These include:

- **BP-METAMIDI-Player-Master:** This is the Blueprint which you will drag into any scene that you want the MetaMIDI music to play. All of the adjustable Macros will be on this Actor once in your scene. In this same folder is the ‘EmotionListEnum’ file.
  This is designed to be customized by the Game Designer.

- **MetaSounds/MS-METAMIDI-Player:** This is the core Metasounds patch, containing the playback, synthesis, and Macro functionality.
  This is designed to be customized by the Composer.

- **MetaSounds/MIDI-Files:** These are the music MIDI files that are played by the MetaMIDI Toolkit. Each music track in here aligns with one Emotion in this base Toolkit *(These have already been converted to ‘uasset’ files, but the original MIDI files can be found in the ‘MetaMIDI_Toolkit-Additional’ folder).*
  This is where the Composer can change the music files being played.

- **MetaSounds/Instruments:** This contains instrument synthesis Metasounds patches which are used within MS-METAMIDI-Player.
  This is where the Composer can change the core synthesizer functionality.

Playing the Music and Macros

The quickest way to play the music is to drag the **BP-METAMIDI-Player-Master** object into a scene as an Actor.
If you hit play on the scene, you will hear the MetaMIDI Toolkit playing the music for **Emotion-Mad** (the default).

Macros

Once it is in your scene, you can click on the Actor and find the MetaMIDI Macro Variables (**Macros** for short). These are designed to provide a simpler and more intuitive front-end for the Game Designer, but are connected to several musical and technical settings in **MS-METAMIDI-Player**.

See the next pages for Macro descriptions.

The Macros are as follows:

Macro	Values	Description	Additional Composer's notes
Emotion	Emotion-Mad, Emotion-Sad, Emotion-Sacred Emotion-Joyful Emotion-Powerful Emotion-Peaceful *Default: Emotion-Mad*	A dropdown list of 6 base emotions. Each emotion is tied to a different music track.	In this base Toolkit, when the Emotion is changed mid-game, it will load the new MIDI file and then play after a 0.5 second delay. This is to make sure the previous track stops before attempting to play the new one.
Speed Multiplier	0.1 to 2.0 *Default: 1.0*	Changes the speed of the music.	This changes the MIDI playback, and so it doesn't affect the pitch of the synthesizers.
Transposition	- 12.0 to 12.0 *Default: 0.0*	Changes the pitch of the music.	This float value is in semitones, and specifically affects the MIDI Note numbers which are fed into the synthesizers.
Smoothness	0.0 to 1.0 *Default: 0.1*	Changes the length of the notes for a more smooth or stuttery effect.	This changes the Attack and Decay of each note (the included FM Synth uses AD envelopes instead of ADSR).
Instability	0.0 to 1.0 *Default: 0.0*	A higher number makes the played notes sound more out of tune for an uncomfortable effect.	At the full 1.0, the resulting frequency of each note played is skewed by up to + or - 40hz.
Intensity	0.0 to 1.0 *Default: 0.0*	A higher number makes the instruments sound more harsh and intense.	This increases the FM Amount of each of the 3 Operators of the FM Synths by up to 800hz.

Complexity	0 to 6 Default: 4	This is the number of instruments that will play. The default is set to 4 as a good balance. Higher numbers can push the music towards feeling overwhelming, and lower numbers towards feeling simple.	This actually chooses which MIDI Channels play. E.g. when set to 3, Channels 1 to 3 play. When set to 5, Channels 1 to 5. The included MIDI music aims to have core harmony on the lower Channels, and flourishes on the higher Channels.
Volume	0.0 to 1.0 *Default: 1.0*	Changes the Volume of the music.	This changes the final volume internally in MS-METAMIDI-Player. This does not affect any volumes set of the audio object elsewhere.

Extending Macros Further

The base MetaMIDI Toolkit provided includes the above functionality. However, this project has been made open-source so that you can customize it as far as you'd like.
It is recommended that the composer and programmers spend some time exploring the Metasounds patches themselves to better understand how the Toolkit may be extended for your project.

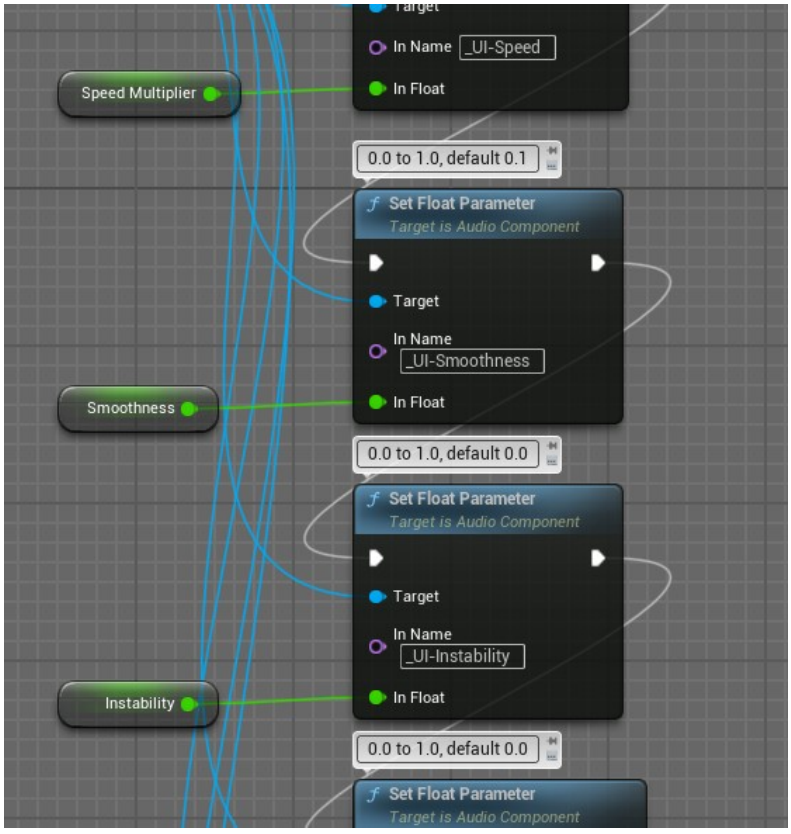
Implementing the Player-Reactive Music

While playing with the in-editor Macro sliders can be good for getting a sense of how different settings affect the music in different ways, the true power of this Toolkit comes from connecting the Macros to in-game changes.

For example, you might want the music Speed Multiplier to increase as time is running out in an arcade game. You may want the Intensity to increase as your player gets closer to their enemies. You might want the music to become more Muffled and less Complex as the player goes underwater.

Any combination of these changes are possible, simply by updating the variables inside the **BP-METAMIDI-Player-Master** Actor.

If you open the Blueprint for **BP-METAMIDI-Player-Master**, you'll see that each of the **Macro Variables** (Smoothness, Intensity, etc) are sent to the **Internal Metasounds Macros** (e.g. _UI-Smoothness, UI-Intensity, etc) every 0.5 seconds. If you simply add functionality which updates the Macro Variable themselves, the music will update itself for you.



Funding

This project has been assisted by APRA AMCOS and the Australian Government through Creative Australia, its arts funding and advisory body.

This project has also been financially supported by the University of South Australia.

MIT License