Koreographer FMOD Studio Integration Overview

for v1.6.1

Table of Contents

Overview **3**
Supported FMOD Studio Versions 3
FMOD Studio Project Settings **3**
FMOD Studio Live Update Support **4**
Generating Koreography **5**
The FMOD Koreography Set **5**
The Koreographed Event Emitter **6**
Configurable Fields 6
Using the Koreographed Event Emitter 6
The FMOD Event Description Visor **7**
Configurable Fields 7
Using the FMOD Event Description Visor 7
Handling FMOD Studio and FMOD Event Callbacks **8**
Enabling Koreographer's Music Time APIs **9**
FMOD Studio Integration Class Namespace **9**
Integration Support **10**

Overview

The FMOD Studio Integration for Koreographer enables you to use Koreography and the Koreographer system with audio that is played back through Firelight Technologies' FMOD Studio sound engine. The integration provides a straightforward and lightweight integration path that enables both the Koreography Event triggering system as well as access to Koreographer's Music Time APIs.

The integration contains two new components:

1. **Koreographed Event Emitter** - A custom version of the *Studio Event Emitter* component provided in the default FMOD for Unity integration that enables Koreographer support for the specified Event. Includes optional support for the Music Time API.
2. **FMOD Event Description Visor** - Enables Koreographer support for any audio played back via any specified Events. As these are Event Descriptions under the hood (as opposed to Event Instances), any time the specified Event is played back by any means it will include Koreographer support. Includes optional support for the Music Time API.

The integration contains one new asset type:

1. **FMOD Koreography Set** - A collection of Koreography. Internally this asset contains the 'glue' information that connects the Koreography to the FMOD Studio system. Both the *Koreographed Event Emitter* and the *FMOD Event Description Visor* use this asset to specify Koreography for event triggering.

Koreographer's FMOD Studio Integration requires audio file names be present in the built project (discussed in detail below). Further, the FMOD Studio integration currently requires that all Koreography be loaded directly in one of the two components listed above. This allows Koreographer to communicate properly with the FMOD Studio runtime to retrieve state information about the position of audio.

Supported FMOD Studio Versions

The original FMOD Studio Integration was built against FMOD Studio 2.2 and its Unity integration. The original integration was built against version 2.1 and was backwards compatible with 2.0. Due to minor API changes in 2.2, the integration is only directly compatible with 2.2 and up. If you require support for FMOD Studio 2.0 or 2.1, please reach out to us.

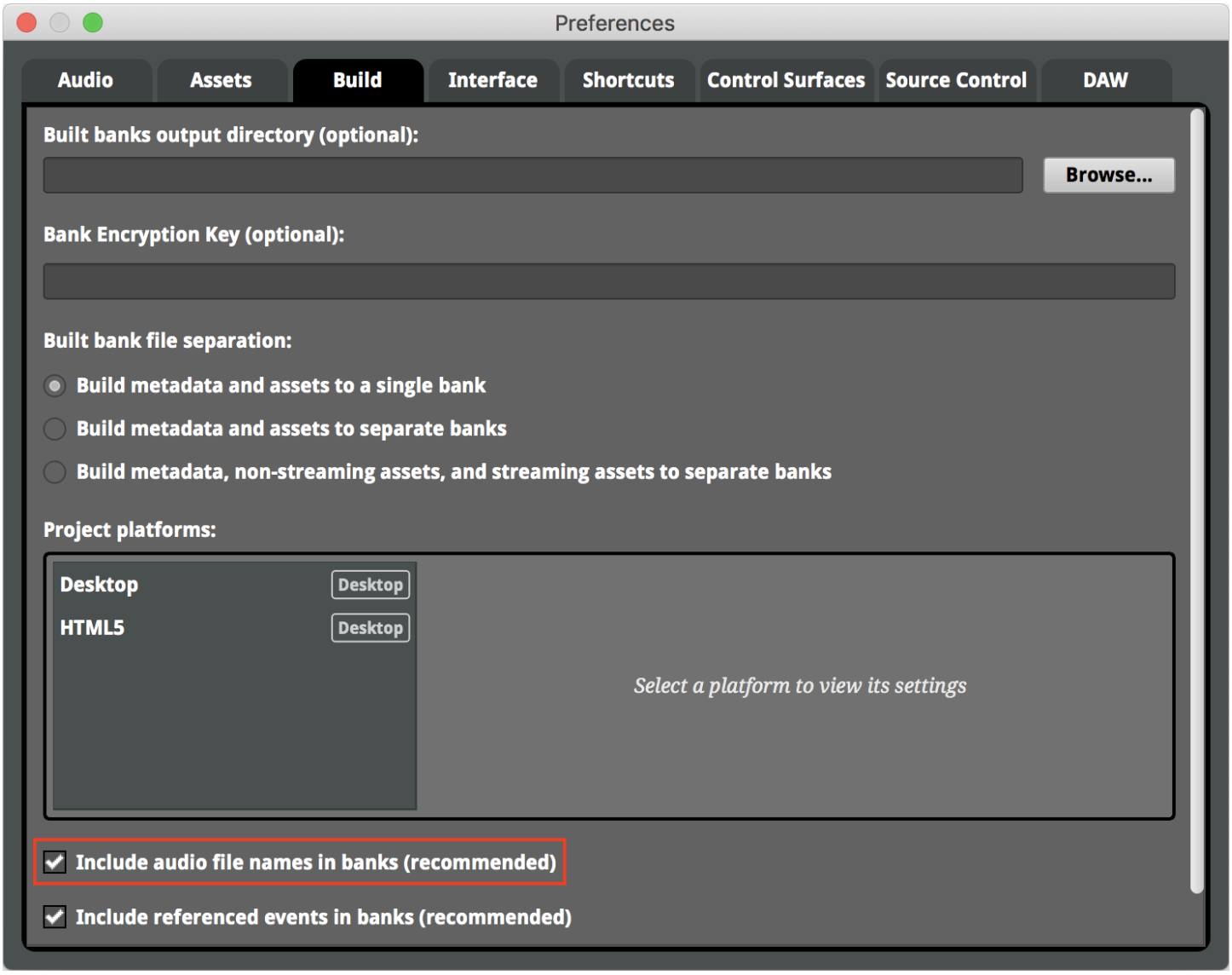
FMOD Studio Project Settings

When FMOD Studio generates Banks one of the steps it can perform is the removal of filenames from the audio assets. This might be done to reclaim a modicum of memory. This process, however, makes identifying individual sounds at runtime impossible.

FMOD Studio projects default to include the audio file names in the Banks, though it can be manually disabled. Please be aware that the integration will fail to work if audio file names are **not** included in the Banks.

To ensure that the file names are included in the built Banks:

1. Open your FMOD Studio project in the FMOD Studio application.
2. Open *Preferences*.
3. Open the *Build* tab.
4. Towards the bottom of the settings, ensure that the following option is checked:
    - **Include audio file names in banks (recommended)**



FMOD Studio project Preferences with the necessary option checked

FMOD Studio Live Update Support

FMOD Studio supports editing your audio project while your game runs in the Unity Editor with a feature called Live Update. While Koreographer's FMOD Studio Integration fully supports Live Update, special care must be taken with your audio assets to ensure that everything functions properly.

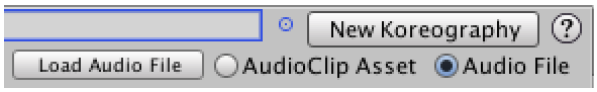
Specifically, any Koreographed audio file must have the filename *without* the file extension added to the audio file's "Title" field in its metadata. Example:

- **Filename:** `lvl_1_intro.wav`
- **File "Title" Metadata:** `lvl_1_intro`

Many applications can read and modify this metadata. VLC is one such application. If this metadata is not correctly set then the Koreography connection may break when the Live Update feature is used.

Generating Koreography

By default, the Koreography Editor works with Unity AudioClip assets to source audio information and provide the Koreography-to-Audio link. Audio files in FMOD Studio projects, however, are never directly imported into Unity and, thus, are not converted into AudioClips. To support generating Koreography for audio used in an FMOD Studio project, the FMOD Studio Integration adds an **Audio File** option that, when selected, enables a **Load Audio File** button.



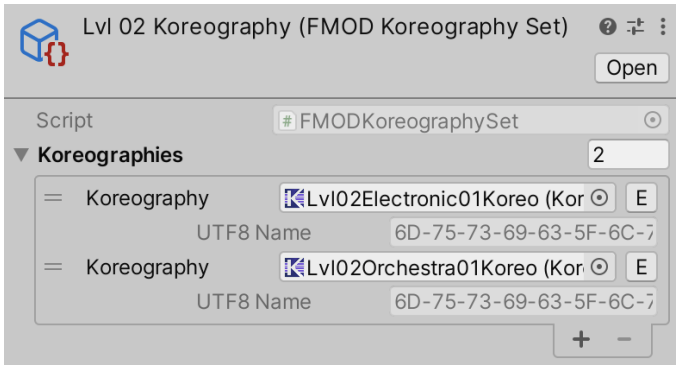
The Koreography Editor's new *Audio File* option and *Load Audio File* button

The *Load Audio File* button will allow you to specify any audio file you wish, including those outside of your Unity project. This allows you to load the same audio file that you use in your FMOD Studio project to generate Koreography.

> **Note:** Audio files loaded in this manner are ***not*** imported into the Unity project.

The FMOD Koreography Set

The *FMOD Koreography Set* is a Unity asset that contains references to **Koreography** (a standard asset type in the main Koreographer package) in its single configurable field called **Koreographies**. Behind the scenes, this asset stores the name of the audio file specified in the Koreography as an array of UTF8 bytes in order to enable Koreographer's Event System and Music Time APIs. The creation of the array is handled automatically and is therefore not editable in the *FMOD Koreography Set*.



An example *FMOD Koreography Set* asset with two Koreography specified

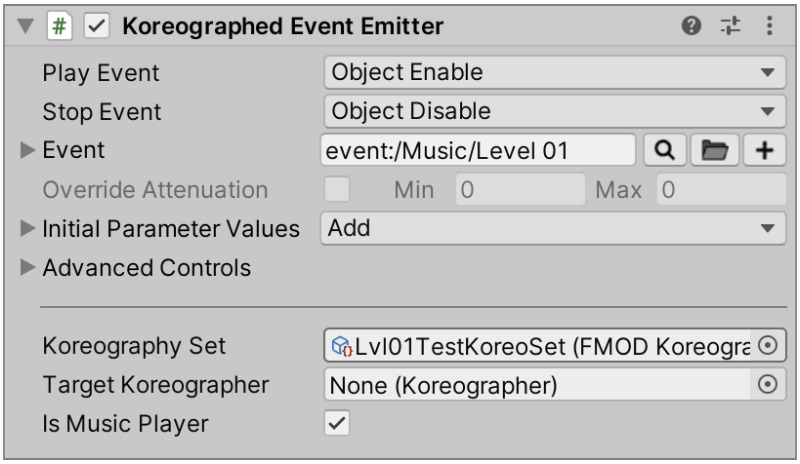
As with other assets in Unity, *FMOD Koreography Set* assets are created through the application menus:

Assets→Create→FMOD Koreography Set

The Koreographed Event Emitter

The *Koreographed Event Emitter* is a component that must be added to a GameObject in the scene. Add this component wherever you would normally add a *Studio Event Emitter*.

The *Koreographed Event Emitter* can be found in the *Add Component* menu under *Koreographer→FMOD Studio→Koreographed Event Emitter*.



The *Koreographed Event Emitter* component Inspector with an example Koreography Set specified

Configurable Fields

In addition to all of the fields available to the default *Studio Event Emitter*, the *Koreographed Event Emitter* has the following configurable fields:

- **Koreography Set:** A reference to an *FMOD Koreography Set* asset. The Koreography contained in this asset are used by the *Koreographed Event Emitter* to process Koreography for sounds played through the specified *Event*.
- **Target Koreographer:** If configured, the *Koreographed Event Emitter* will use the specified Koreographer component for event triggering. If not specified, the singleton Koreographer will be used.
- **Is Music Player:** If checked, the *Koreographed Event Emitter* will register itself as a "music player controller" with the Koreographer component specified by the **Target Koreographer** field (including the singleton fallback for that field).

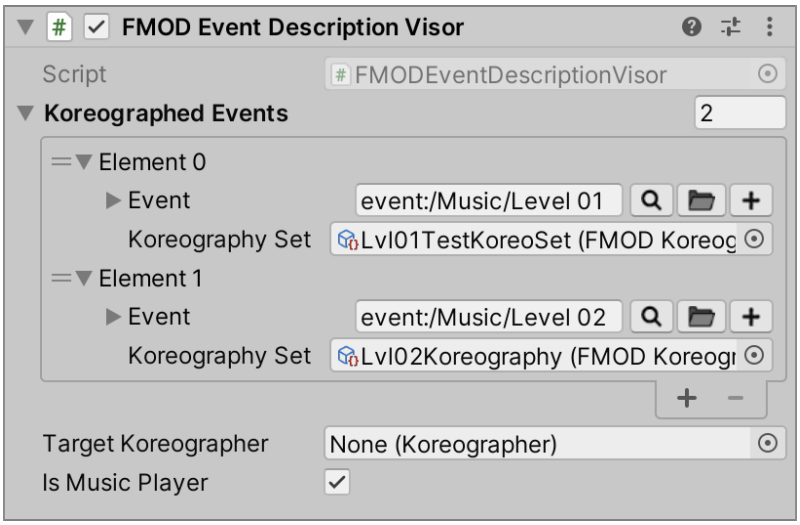
Using the Koreographed Event Emitter

To begin, ensure that FMOD Studio is working correctly in your Unity project. Once verified, add a *Koreographer* component and the *Koreographed Event Emitter* to your scene. Create an *FMOD Koreography Set* asset and add at least one Koreography to its **Koreographies** field. Add a reference to that asset to the *Koreographed Event Emitter*'s **Koreography Set** field. Koreography Events will begin to trigger when the configured **Play Event** triggers and results in the playback of an audio file for which there is Koreography loaded in the *Koreographed Event Emitter*'s **Koreography Set**.

The FMOD Event Description Visor

The *FMOD Event Description Visor* is a component that must be added to a GameObject in the scene. To keep things organized, it is recommended to add the component to a GameObject called “Koreographer”.

The *FMOD Event Description Visor* can be found in the *Add Component* menu under *Koreographer→FMOD Studio→FMOD Event Description Visor*.



The *FMOD Event Description Visor* component Inspector configured with two Koreographed Events

Configurable Fields

The *FMOD Event Description Visor* has the following configurable fields:

- **Koreographed Events:** A list of FMOD Studio Events that this visor should monitor. Each entry consists of:
    - **Event:** The FMOD Studio Event to monitor.
    - **Koreography Set:** A reference to an *FMOD Koreography Set* asset. The Koreography contained in this asset are used by the *FMOD Event Description Visor* to process Koreography any time the paired **Event** is played.
- **Target Koreographer:** If configured, the *FMOD Event Description Visor* will use the specified Koreographer component for event triggering. If not specified, the singleton Koreographer will be used.
- **Is Music Player:** If checked, the *FMOD Event Description Visor* will register itself as a "music player controller" with the Koreographer component specified by the **Target Koreographer** field (including the singleton fallback for that field).

Using the FMOD Event Description Visor

To begin, ensure that FMOD Studio is working correctly in your Unity project. Once verified, add a *Koreographer* component and the *FMOD Event Description Visor* to your scene. Set the number of **Koreographed Events** to the number of FMOD Studio Events that you want Koreographed. For each such Koreographed Event, select the FMOD Studio Event you wish to be Koreographed in the **Event** field and specify a corresponding **Koreography Set** asset that contains Koreography appropriate to the sounds played through that FMOD Studio Event.

Elsewhere in your Scene, initialize the commands (via script or the *Studio Event Emitter* component) to play one or more of the Events specified in the **Koreographed Events** list. Koreography Events will begin to trigger whenever any Event specified in the **Koreographed Events** list begins to play and results in the playback of an audio file for which there is Koreography loaded in the Event's paired **Koreography Set**.

Handling FMOD Studio and FMOD Event Callbacks

Koreographer's FMOD Studio Integration requires access to certain callbacks from the FMOD Studio system. Specifically these are:

- **Studio System Callbacks** - Callbacks registered with the Studio System.
  - `POSTUPDATE` - Used to determine an appropriate time to scan an `EventInstance` for newly created `Channel` instances.
- **Event Callbacks** - Callbacks registered with `EventDescription`s or `EventInstance`s.
  - `CREATED` - Used to identify `EventInstance` candidates to watch by the *FMOD Event Description Visor*.
  - `DESTROYED` - Used to identify when to clean up resources related to watching an `EventInstance`.
  - `STOPPED` - Used to identify when to clean up resources related to watching an `EventInstance`.
  - `SOUND_PLAYED` - Used to determine if a new audio file of interest has started playing within an `EventInstance`.

Each individual FMOD Studio resource only supports the registration of a single callback function. Koreographer's FMOD Studio Integration requires registration on the following resources:

1. The Studio System.
2. `EventDescription` instances for any Event configured in the *FMOD Event Description Visor*'s **Koreographed Events** field.
3. Every `EventInstance` instance being monitored by the *Koreographed Event Emitter*, the *FMOD Event Description Visor*, or otherwise manually by you (e.g. via the `FMODEventInstanceVisor` class).

If you need to receive callbacks for any of the above resources, then you must forward the necessary callbacks (outlined above) to Koreographer's FMOD Studio Integration code. For simplicity (and to properly handle AOT platforms), callback handling has been centralized in the `FMODCallbackHandler.cs` script. How you properly forward the necessary events to the proper components depends upon the callback target in question. See:

1. The Studio System
   - **Callbacks:** `POSTUPDATE`
   - **Explanation:** The integration needs to know when the FMOD Studio System has completed its main update phase so that it can scan for newly created `Channel` instances to monitor.
   - **Override Instructions:** Please set the static `FMODCallbackHandler.UseThirdPartyPostUpdateCallback` field to `true` and ensure that you add the `POSTUPDATE` bit to your callback mask. When your custom handler receives the `POSTUPDATE` callback type, please call `FMODCallbackHandler.Instance.DoFMODStudioPostUpdate()`.
2. `EventDescriptions`
   - **Callbacks:** `CREATED`
   - **Explanation:** Callbacks registered with `EventDescriptions` are automatically added to their generated `EventInstances`. By registering a callback function for the `CREATED` event with specific `EventDescriptions` of interest, the integration is able to monitor all `EventInstances`

generated from those `EventDescriptions`. When this happens, the callback is immediately overridden to support the standard `EventInstance` callback workflow outlined below.
    - **Override Instructions:** Please set the static `FMODCallbackHandler.UseThirdPartyEventDescriptionCallback` field to `true` and ensure that you add the `CREATED` bit to your callback mask. When your custom handler receives the `CREATED` callback type, please call `FMODCallbackHandler.Instance.HandleEventDescriptionCallback()`.
3. `EventInstances`
    - **Callbacks:** `DESTROYED`, `STOPPED`, `SOUND_PLAYED`
    - **Explanation:** The integration uses the `SOUND_PLAYED` callbacks to determine when to scan for new sounds of interest to monitor. The `DESTROYED` and `STOPPED` callbacks are used to ensure proper cleanup of resources when a monitored `EventInstance` ends.
    - **Override Instructions:** Please set the static `FMODCallbackHandler.UseThirdPartyEventInstanceCallback` field to `true` and ensure that you add the `DESTROYED`, `STOPPED`, and `SOUND_PLAYED` bits to your callback mask. For convenience, you can simply use the `FMODCallbackHandler.EVENT_CALLBACK_TYPE_MASK` mask to specify all necessary callbacks at once. When your custom handler receives any of the specified callback types, please call `FMODCallbackHandler.Instance.HandleEventInstanceCallback()`.

Enabling Koreographer's Music Time APIs

Koreographer uses information it receives from a "Music Player" to enable access to the Music Time APIs. In the default Unity integration, this is automatically handled by the included music player components (*Simple Music Player* and *Multi Music Player*).

The Music Time APIs are enabled by connecting a script that implements the `IKoreographedPlayer` interface to a Koreographer component instance at runtime (generally in the `Start()` method). It is through this interface that the Koreographer system understands which of the currently playing audio files (and, if loaded, its Koreography) should be used as the Music Time source.

In Koreographer's FMOD Studio Integration, both the *Koreographed Event Emitter* and the *FMOD Event Description Visor* components may optionally be configured to act as a "music player". To make use of the Music Time API functionality, you must check the "**Is Music Player**" option on either of the aforementioned components in your scene. Please be aware that only one "music player" may be configured for any given Koreographer component at a time.

FMOD Studio Integration Class Namespace

Koreographer's FMOD Studio Integration classes can be found in the **SonicBloom.Koreo.Players.FMODStudio** namespace. Additionally, Editor-only support classes are located in the **SonicBloom.Koreo.EditorUI.FMODStudioTools** namespace.

Integration Support

Need help getting Koreographer's FMOD Studio Integration up and running? Have a use case that isn't covered by the integration? ***Please*** reach out to us for support! We are here to help you get moving!