fmod® studio

Getting Started Guide

www.fmod.com

Table of Contents

Welcome to FMOD Studio

Sound for film and television is scripted during production, and the playback is predictable and linear. For interactive media such as games, the sequence of events cannot be predicted and so the audio playback must adapt to the game play.

FMOD Studio is a new breed of tool that aims to bridge the gap between linear and adaptive sounds and music. If you've used a DAW before, you should feel right at home. If you're new to audio, we've worked hard to make the UI straight-forward and intuitive.

FMOD Studio includes an Audio Engine to be used by your game. If you're an Unreal Engine 4 or Unity developer we have FMOD Studio plug-ins for both. For all other developers, see our engine documentation for instructions on how to use FMOD Studio in your game.

In addition to this guide there are a couple of other useful learning resources:

- Our YouTube channel has step-by-step videos on how to use FMOD Studio
- Our community run Q&A site is a great place to look for help and ask questions

We hope you enjoy discovering what FMOD Studio can do, and can't wait to hear what you create with it.

- The FMOD Team.

How to Read This Guide

This guide is designed to get you working with FMOD Studio as soon as possible. It doesn't cover every one of FMOD Studio's features, and it doesn't describe any feature in great detail. Instead, it focuses on the most fundamental and commonly-used features, provides step-by-step instructions for performing common tasks, and explains only the bare minimum required to get started creating content.

When working your way through this guide, you should start by reading this section, and the tutorial titled "Creating a Project." Once you've finished that tutorial, you should be able to work through the remaining tutorials in any order.

In order to make these tutorials easier to follow, we've followed some conventions:

Text that looks like this is an instruction for you to follow. Do what it says.

Any text that looks like this is a description of what you should see on your screen or hear from your speakers or headphones, assuming you've followed all the preceding steps.

Text that looks like this is an explanation of something. These usually contain background information or detail about the topic that isn't obvious just from following the instructions.

Warning: Boxes like this contain information about actions that may alter your project’s performance and behaviour in ways that aren’t immediately obvious. Pay careful attention to these, as they may save you time and effort.

If you see text that contains a ‘>’ character, also known as a right angle bracket, it indicates a specific command or entry of a menu. “Select File > New,” for example, is an instruction to open the FMOD Studio’s ‘File’ menu and select the ‘New’ command.

Learning by Experiment

Feel free to experiment. You can learn a lot that way, and you can always come back to these tutorials later.

A good way to start is by right-clicking on everything. FMOD Studio’s context-sensitive menus contain the vast majority of its capabilities.

Don’t worry, you can’t break anything. Unless you change the content of the Audio Assets tab, FMOD Studio performs only non-destructive editing to audio files. As a result, the only thing you risk changing when you edit an FMOD Studio project is that same FMOD Studio project, and even that will be be safe if you don’t save. Also, if you do make a change that you decide you don’t want, you can undo most actions by selecting “Edit > Undo”.

Further Reading

This guide isn’t the only way to learn about FMOD Studio. If you have any questions, post them at http://www.fmod.org/support/qa/, or for urgent and confidential problems contact support@fmod.org. We will respond to you as soon as possible.

If you prefer to learn by seeing what other people have done, there are many user-made video tutorials and demonstrations of FMOD Studio on YouTube. We’ve posted a few of our own at https://www.youtube.com/user/FMODTV.

Finally, the FMOD Studio User Manual, the FMOD Studio Programmer’s API Documentation and the Low Level Documentation each contain a wealth of information about their respective topics.

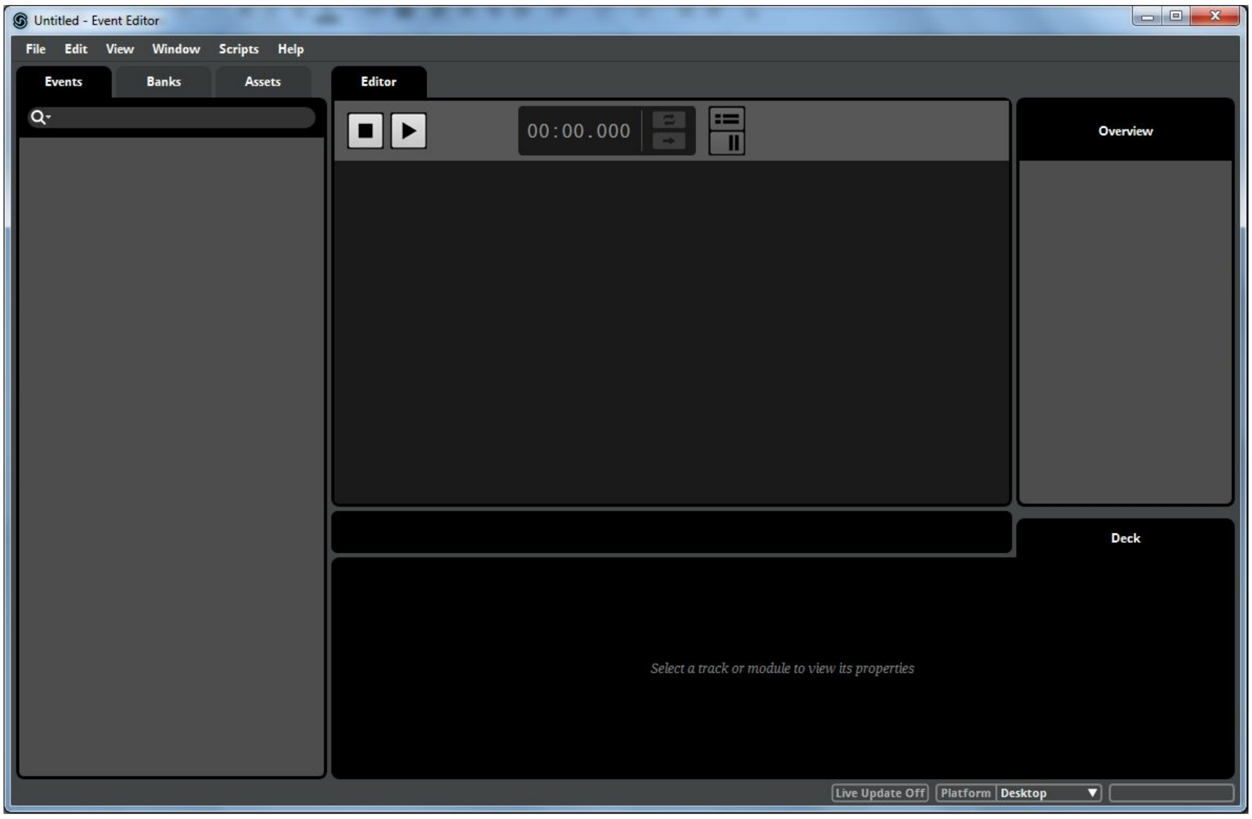
Tutorial: Creating a project

All content created in FMOD Studio is organised into “projects.” Generally, a game requires exactly one FMOD Studio project. This one project contains all the events, music, and mixer data for your game. (Our legacy product, FMOD Designer, often required users to create multiple projects per game project. Studio is not limited in this way.)

Before we do anything else, we should create a new project. There’s a very simple way to do this:

Launch FMOD Studio.

FMOD Studio opens, and the Event Editor Window displays a new, blank project.



You can start editing this new project immediately.

Warning: FMOD Studio projects are not automatically saved to disk when first created. Until you manually save a project for the first time, any audio files it uses are loaded from the directories you imported them from instead of copied into the project’s internal asset directory. If these externally linked files are changed in any way, your project may break or begin to act in ways that differ from what you expect.

Saving a project early prevents these problems from occurring by copying all existing audio files into the project’s internal asset directory, and by ensuring that any files imported after saving are automatically copied instead of externally linked.

For information on importing audio files, see the “Importing Audio Files” subsection, below.

If you want to create a new project and FMOD Studio is already running, you can do it from the ‘File’ menu:

Select ‘File > New’.

A new Event Editor window opens and displays a new, blank project. All existing open FMOD Studio windows continue to exist in the background.

You can start editing this new project immediately. The most common reason for creating a new project while working on an existing one is wanting to test an idea before implementing it in the project you are working on.

For more information on working with multiple FMOD Studio projects simultaneously, see the FMOD Studio reference manual.

Creating Events

The main building blocks of an FMOD Studio project are “events.” Events are important because they can be triggered, and their parameters manipulated, from your game’s code; In turn, events can trigger and manipulate their modules to produce, alter or reroute audio signals. In other words, events define how in-game occurrences change what comes out of the speakers.

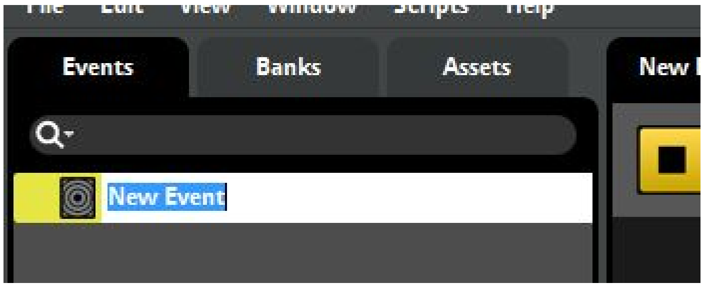
They’re called events because they correspond to occurrences in your game. For example, a game that features explosive barrels will probably have an “explosion” game event in its code and a corresponding event in its FMOD Studio project, and every explosion that occurs in the game will cause that “explosion” event to be triggered. To avoid confusion between in-game occurrences and FMOD Studio events, this guide calls in-game occurrences “game events,” and uses the word “events” on its own to mean events created in FMOD Studio.

Multiple instances of any given event can be playing simultaneously. When multiple instances of an event are playing, each instance has its own parameter values, independent of the parameter values of all other instances. For example, in our hypothetical game that features explosions, the game has only a single explosion event, but triggers a new instance of that event for each explosion that occurs, and assigns each instance its own timeline and parameter values depending on the circumstances of the particular game event that triggered it.

The project we made is currently empty. It will need at least one event if it’s going to be of any use to us, so let’s make one:

Right-click on the Events Browser to open the context-sensitive menu, then select the ‘New Event’ menu item.

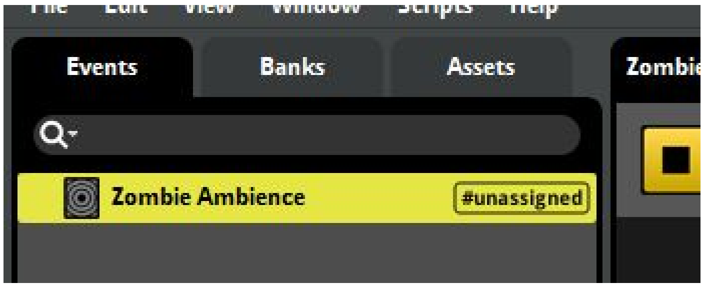
A new event appears, ready to be named.



Type “Zombie Ambience” and press the ‘Enter’ key.

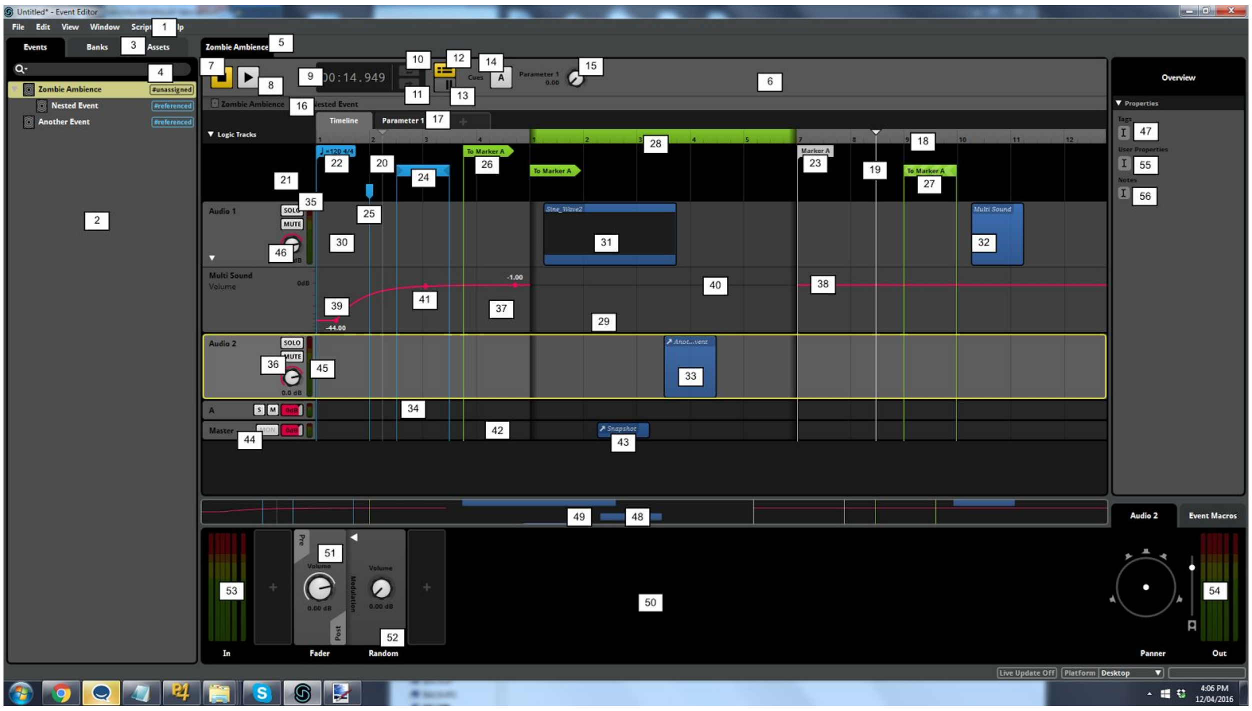
The event is now named Zombie Ambience.

(If your event is named ‘New Event’ instead of Zombie Ambience, it’s because it somehow lost keyboard focus before the new name was entered. Just double-click on the event name and you’ll be able to rename it.)



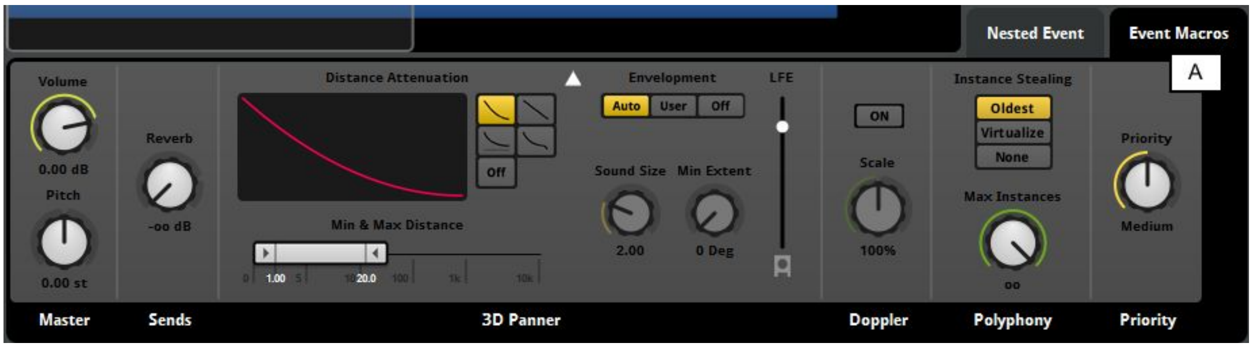
Besides being a handy reminder of what an event is, the name of an event can be used to identify and instantiate the event in your game’s code.

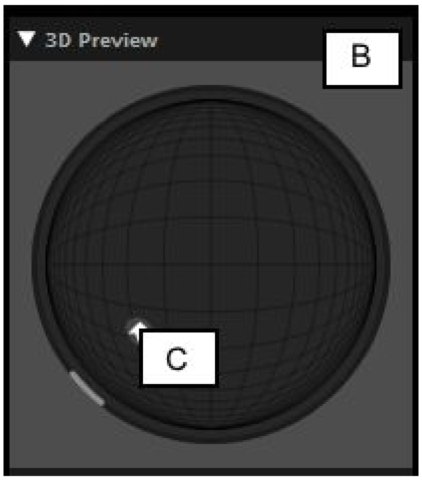
Understanding the Event Editor Window



1. Menu bar
2. Events browser
3. Browser tabs
4. Browser search bar
5. Editor
6. Transport bar
7. Stop button
8. Play button
9. Time indicator
10. Loop playback toggle button
11. Follow cursor toggle button
12. Tracks View button
13. Strips View button
14. Cue button
15. Parameter value knob
16. Breadcrumbs
17. Parameter tabs
18. Parameter ruler
19. Parameter cursor
20. Ghost cursor
21. Logic tracks
22. Tempo marker
23. Destination Marker
24. Loop region
25. Sustain point
26. Transition marker
27. Transition region

28. Transition timeline
29. Event tracks
30. Audio track
31. Timelocked sound module trigger region
32. Non-timelocked Sound module trigger region
33. Event reference sound module trigger region
34. Return Track
35. Solo toggle button
36. Mute toggle button
37. Automation track
38. Automation curve
39. Automation point
40. Automation track current value indicator line
41. Diamond handle
42. Master Track
43. Snapshot trigger region
44. Monitoring button
45. Track meter
46. Volume knob
47. Tags
48. Birdseye View
49. Birdseye View Reticule
50. Deck
51. Fader module
52. Drawer
53. Input meters
54. Output meters
55. User Properties
56. Notes





A. Event macro controls
B. 3D Preview
C. Emitter icon

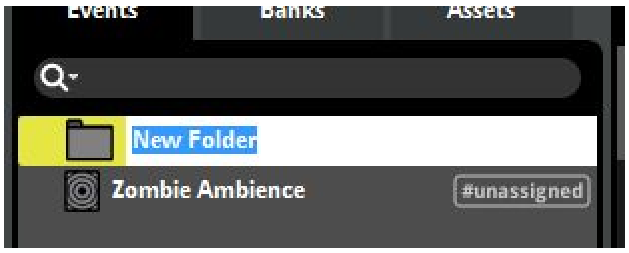
Organising Events

As well as events, you can create folders in the events browser. These folders can be used to sort other folders and events, allowing you to organise your events and folders in ways that suit your workflow.

When an event is in a folder or series of folders, the name of that folder or series forms part of the path for that event. That path can be used to trigger the event in your game's code. Folders have no other effect on your project's behaviour.

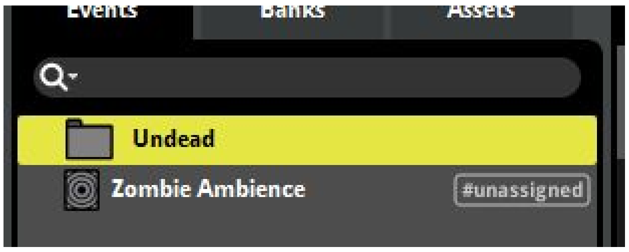
Right-click on the Events Browser to open the context-sensitive menu, then select the 'New Folder' menu item.

A new folder is created, ready to be named.



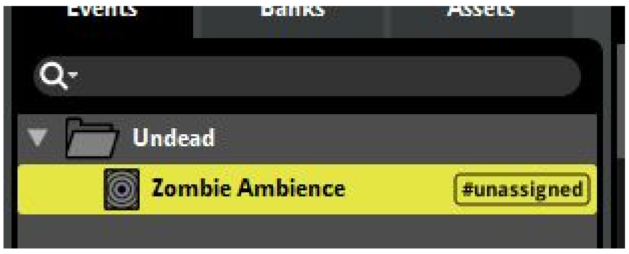
Type "Undead" then press the 'Enter' key.

The folder is named "Undead." (If your event is named "New Folder" instead of "Undead," the folder somehow lost keyboard focus before the new name was entered. Just double-click on the folder name and you'll be able to rename it.)



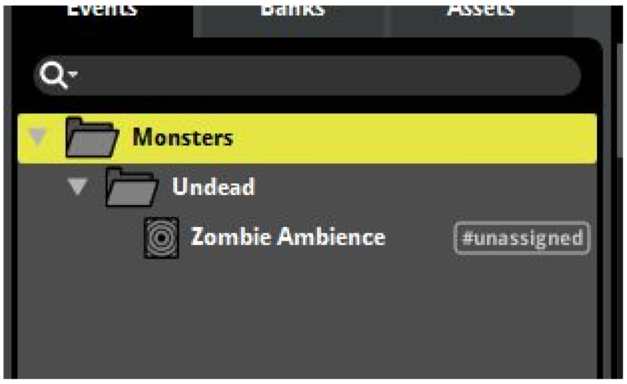
Click and drag the "Zombie Ambience" event onto the "Undead" folder.

The "Zombie Ambience" event is now slightly to the right of the "Undead" folder. This indicates that it is contained within the folder.



Right-click on the "Undead" folder to open the context-sensitive menu, then select the 'Move into New Folder' menu item. Name this new folder "Monsters".

The "Undead" folder is now inside the new "Monsters" folder.



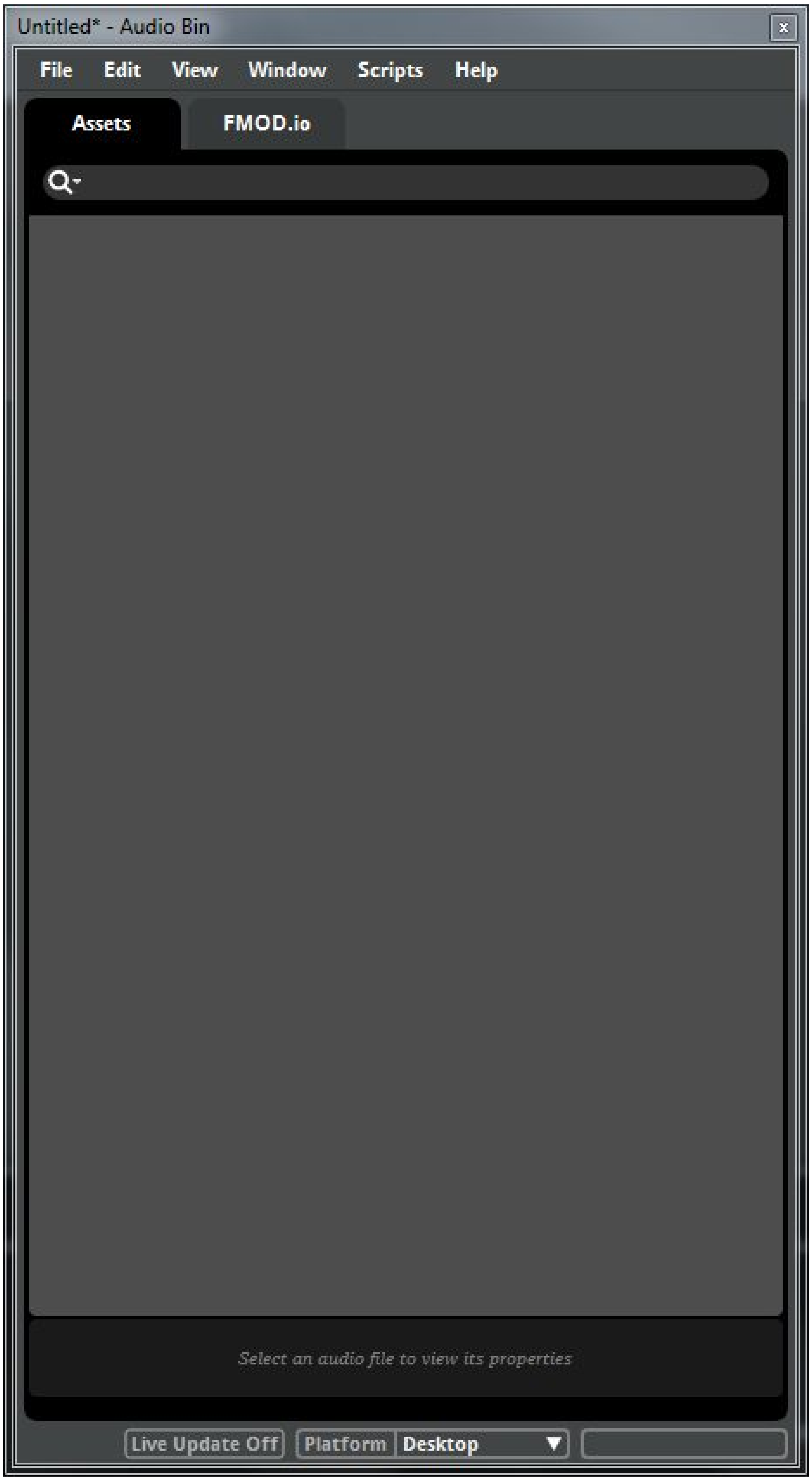
Obviously, there's no benefit to creating a multi-level folder structure for just one event, but it will come in handy if we add more events later.

Importing Audio Files

In order for an event to produce an output, you need to import some audio files into your project. You also need to add a sound module that uses those files to the event.

Select 'Window > Audio Bin.'

The Audio Bin window appears.

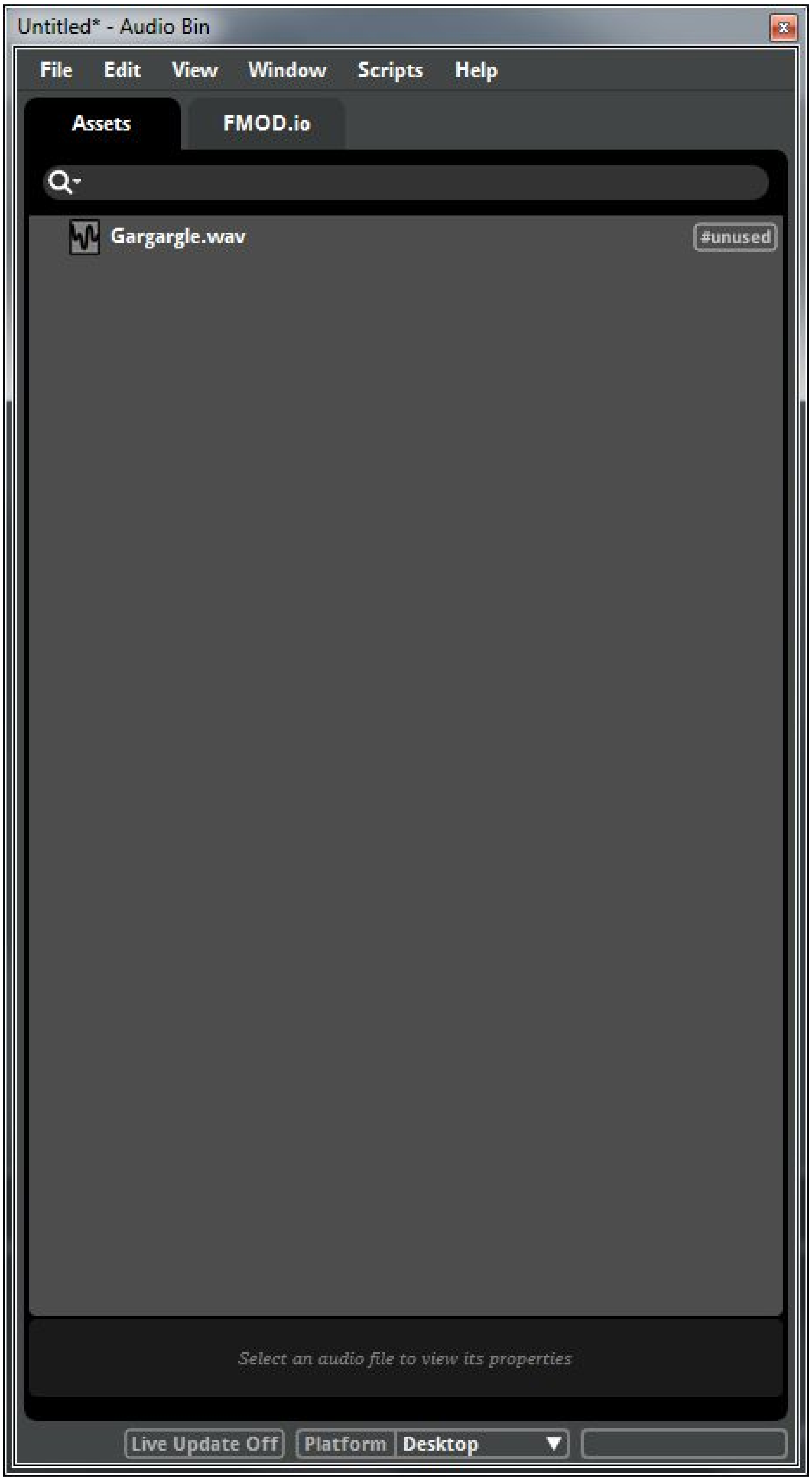


All files that have been imported into an FMOD Studio project are listed in its audio bin. From the Audio Bin, it’s possible to update files, replace them, and add them to other events.

In Windows Explorer or Finder, navigate to and select the audio file you want to import.

Click and drag the selected audio file onto the Audio Bin.

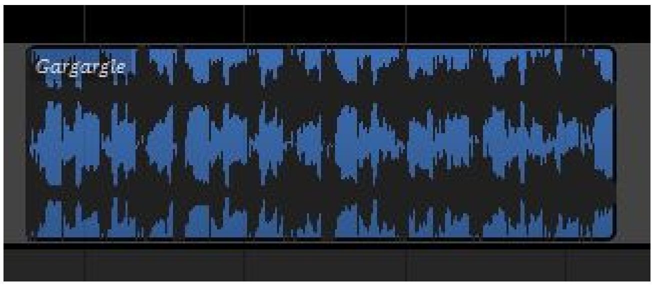
The audio file appears in the Audio Bin’s browser.


Untitled* - Audio Bin
File
Edit
View
Window
Scripts
Help
Assets
FMOD.io
Gargargle.wav
#unused
Select an audio file to view its properties
Live Update Off
Platform
Desktop

You’ve just added an audio file to your project. All files that have been imported into an FMOD Studio project are listed in its audio bin. From the Audio Bin, it’s possible to update files, replace them, and add them to other events.

Click and drag the audio file from the Audio Bin onto one of the tracks in the event editor.

A blue trigger region with a waveform graphic appears on the audio track.



We could also have just dragged the audio file directly onto the audio track, and the file would have been added to the audio bin for us. It’s far more common to add audio files to events by dragging them from the audio bin, though, especially if we’re using the same audio file in more than one place.

Warning: Importing audio files into a saved project more than once may result in your project using multiple identical versions of your source files. This can make your project larger than it would otherwise be. It may also increase the number of steps involved in updating and editing the audio files used in your project.

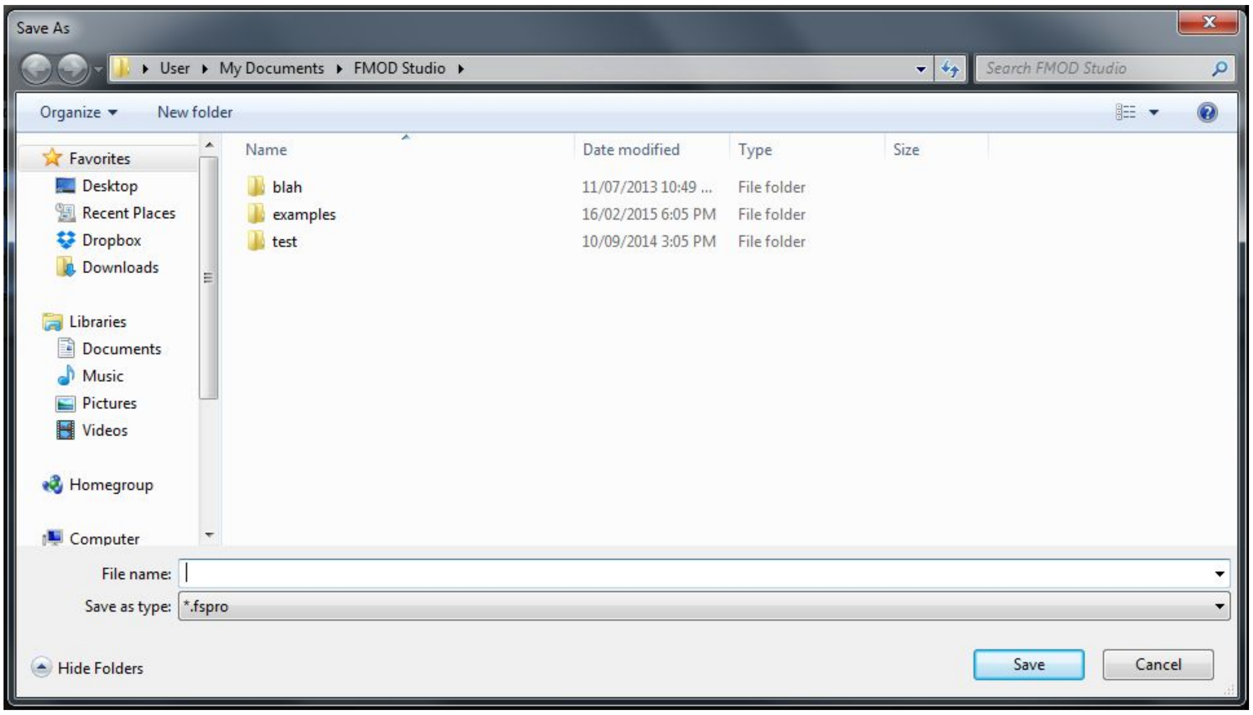
To avoid importing the same file more than once, drag sounds from the Audio Bin instead of Finder or Windows Explorer whenever possible.

Saving a Project

Currently, we have a project, but it hasn’t been saved to the hard disk yet. We should do that now.

Select ‘File > Save As...’.

A ‘Save As’ window appears.



Name	Date modified	Type	Size
blah	11/07/2013 10:49 ...	File folder	
examples	16/02/2015 6:05 PM	File folder	
test	10/09/2014 3:05 PM	File folder	

File name:

Save as type: *.fspro

The exact appearance and behavior of the ‘Save As’ window depends on your operating system. The example depicted above is from Windows 7.

No matter what appearance this window has, it allows you to specify a name and location under which a project is to be saved.

Navigate to an appropriate location for the project.

Specify a name for the project.

Click the ‘Save’ button.

The ‘Save As’ window closes, and the project name appears in the titlebar.

The project is now saved.

A saved FMOD Studio project isn’t just a single file. Instead, it’s a whole directory structure containing numerous files and folders that correspond to different parts of the project. When you make changes to a project, FMOD Studio only edits files that contain or reference the changed content.

Warning: Do not attempt to manually edit or reorganise an FMOD Studio project folder or any of its contents! The files and folders that a project folder contains are interdependent and interconnected, and making changes outside of FMOD Studio may cause your project to become corrupted or unusable.

If for some reason you do need to manually edit any of the files in a project folder, contact support@fmod.org for advice on how to do so.

After saving a project for the first time, you can select ‘File > Save’ to save without having to specify a new location and name.

Warning: Saving a project that has already been saved to disk by selecting ‘File > Save’ overwrites the version of the project on disk.

Working with Single Sound Modules

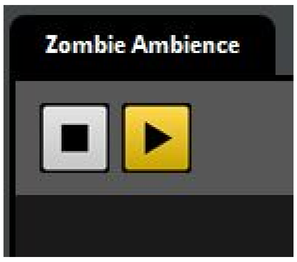
Earlier, when you dragged an audio file onto an event track, you created a ‘sound module.’ A ‘module’ is a component of a signal chain characterised by its ability to be be displayed in the deck when active, allowing you to edit its properties and settings. ‘Sound modules’ are modules from which the audio in an event originates.

Sound modules are represented in the editor by colored boxes on event tracks called ‘trigger regions.’ The position and size of a trigger region represents the circumstances in which the associated sound module is triggered.

Close the Audio Bin window.

The audio bin window disappears.

Click on the play button in the transport bar.

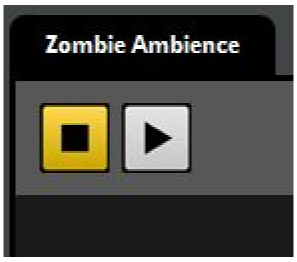


The play button is highlighted, and the cursor advances from left to right. When the cursor encounters a blue trigger region, the sound file you imported earlier plays. When the cursor leaves the trigger region, the sound file stops.

When the parameter cursor intercepts a trigger region, the associated sound module is ‘triggered’ and outputs a signal to the audio track. When the cursor leaves the trigger region again, the sound module stops outputting a signal.

Click on the stop button in the transport bar.

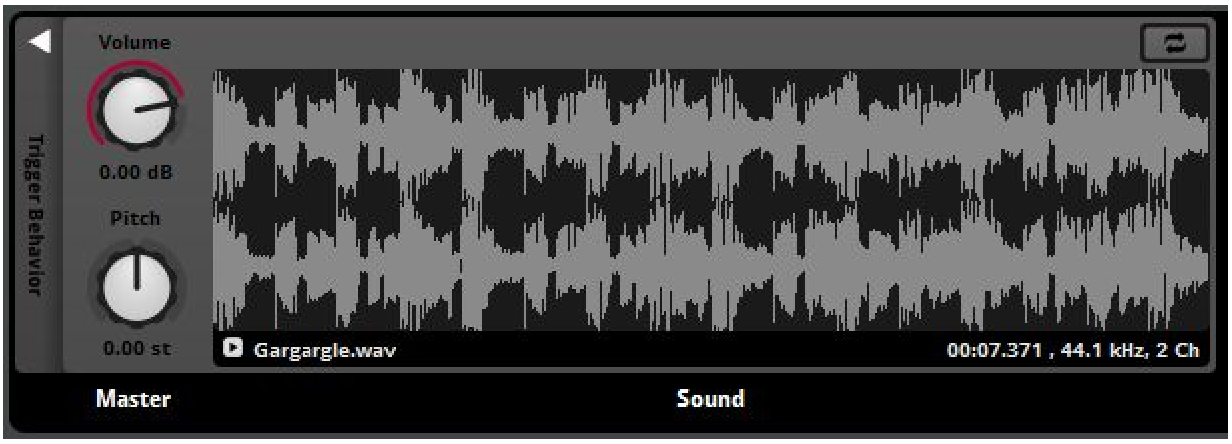
The cursor ceases to advance. The stop button is highlighted, and the play button is no longer highlighted.



The event is now paused.

Click on the trigger region.

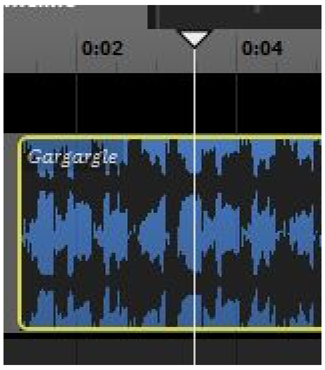
The trigger region is highlighted in gold and the associated module appears in the deck.



There are a number of different kinds of sound module in FMOD Studio. This particular module is a ‘single’ sound module. Single sound modules are so called because each can be associated with a single audio file.

Click on a part of the parameter ruler that’s directly above the trigger region.

The cursor is positioned overlapping the trigger region.



Click on the play button.

The cursor advances from its current position, and the sound module starts to play from part-way through its audio file.

This particular single sound module is also ‘timelocked.’ Timelocked sound modules have waveform graphics on their trigger regions. When a timelocked sound module is triggered, the part of the visible waveform that the parameter cursor overlaps is the part that plays.

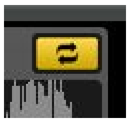
This behaviour is useful sometimes, but not always. Sometimes, you want an audio file to start playing from the start no matter what. Fortunately, not all sound modules are timelocked, and it’s easy to unlock a timelocked sound module.

Click on the trigger region.

The trigger region is highlighted in gold and the associated module appears in the deck.

In the Deck, click on the Loop toggle button.

The Loop toggle button is highlighted in gold.



Click on a part of the parameter ruler that’s directly above the trigger region.

The cursor is positioned overlapping the trigger region, and starts to advance from there. The sound module begins to play from the beginning of its audio file. The sound module stops producing output as soon as the cursor leaves the trigger region.

Sound modules that aren’t timelocked are still controlled by the cursor, but don’t behave differently depending on which part of the trigger region the cursor overlaps: As long as the cursor is on the trigger region, the sound module is triggered, and if the cursor isn’t on the trigger module, it isn’t.

Working with Multi Sound Modules

Single sound modules aren’t the only kind of sound module. ‘Multi’ sound modules can’t be timelocked, but aren’t limited to always playing the same audio file when triggered.

We’re going to need a fresh slate for this, so let’s create a new event.

In the Events Browser, right-click on the "Undead" folder to open the context-sensitive menu, then select the 'New Event' menu item.

A new event appears, ready to be named.

Type "Vampire Dance" and press the 'Enter' key.

The event is now named "Vampire Dance."

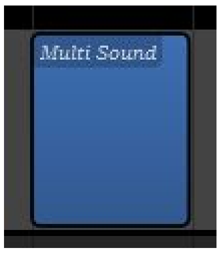
Click on the "Vampire Dance" event to activate it.

The Vampire Dance event is displayed in the editor.

Now that you have a new event ready, it's time to create a multi sound module.

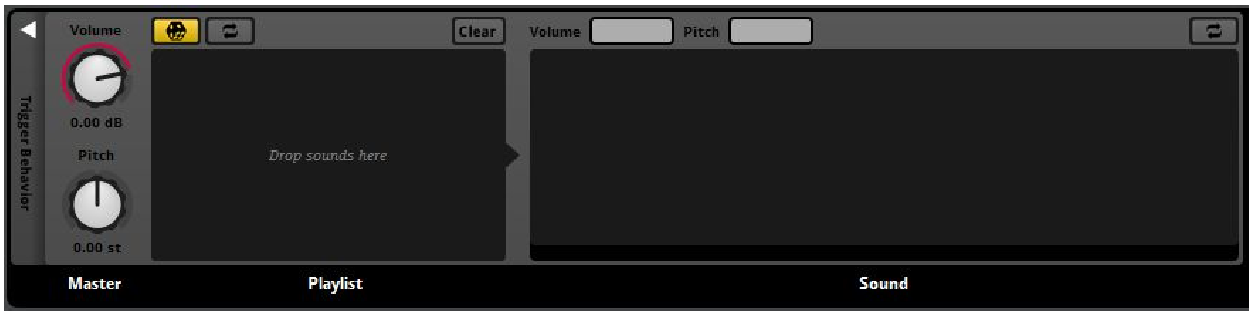
In the editor, right-click on the audio track to open the context-sensitive menu, then select the 'Add Multi Sound' menu item.

A new trigger region appears.



Click on the trigger region.

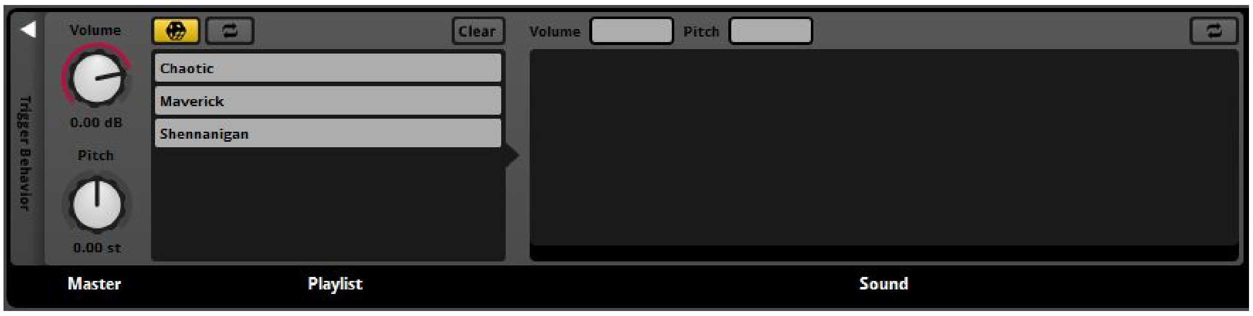
The sound module associated with the trigger region appears in the Deck.



And there you have it, a multi sound module. It looks a lot like a single sound module, but it has an extra 'Playlist' section that determines how it selects files to be played. Currently that playlist is empty, which means that the module won't produce any output when triggered - but that's easy to fix.

In Finder or Windows Explorer, navigate to and select some audio files you haven't imported yet. Click and drag these files onto the multi sound module's playlist.

The files are listed in the multi sound module's playlist.



And now the multi sound module is ready.

Click on the play button in the transport bar.

The cursor advances from left to right. When it encounters a blue trigger region, one of the audio files in the playlist starts to play. The module continues to produce output until the sound file reaches its end.

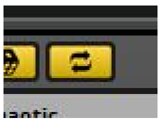
By default, a multi sound module selects which sound file it should play at random each time it is triggered. We'll cover how to change this later.

Multi sound modules can never be timelocked, so there's no waveform drawn on the trigger region.

Unlike the non-timelocked single sound module in the "Zombie Ambience" event, this module doesn't stop outputting a signal when the cursor leaves its trigger region. Whether a module falls silent when untriggered usually depends on whether that module is set to loop. In this case, the module isn't set to loop, so the sound plays out to its end even if the module is untriggered.

In the deck, click on the loop toggle button directly above the playlist.

The loop toggle button is highlighted in gold.



Click on the play button in the transport bar.

The cursor advances from left to right. When it encounters a blue trigger region, one of the audio files in the playlist starts to play. The sound module stops producing output as soon as the cursor leaves the trigger region.

Besides affecting what a module does when untriggered, loop mode determines whether a module loops.

Click on the stop button in the transport bar twice.

The event is paused and the cursor returns to its starting position.

Click and drag the left edge of the trigger region as far to the left as it will go.

The trigger region is resized so that it starts at 00:00:000.

Click and drag the right edge of the trigger region until the editor starts to scroll.

The trigger region stretches out until the mouse button is released.

Click on the play button in the transport bar.

The cursor advances from left to right. When it encounters a blue trigger region, one of the audio files in the playlist starts to play. When the file reaches its end, another file from the playlist starts to play. When the cursor leaves the trigger region, it stops playing sounds immediately.

As long as the cursor stays inside its trigger region, a looping sound module continues to play new sounds forever.

What Next?

Events, sound modules, imported audio files and trigger regions are the basis of everything you can do with FMOD Studio. Once you know how to create, edit and audition them, and understand how being timelocked or looping changes how they behave, then you're ready to start creating most common kinds of events.

That said, we have only scratched the surface of what you can do with FMOD Studio. If you want to make events and music that change depending on what happens in your game, or to create a mix for your project, or to connect FMOD Studio to your game live, the other sections of this guide are ready and waiting.

Tutorial: Making Adaptive Events

Not all events are fire-and-forget. Often, an event needs to react to changing circumstances after it has already begun playing. In FMOD Studio, the primary way to make an event adaptive is to give it parameters.

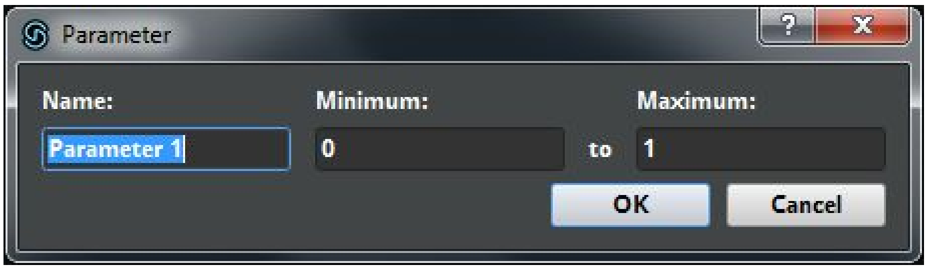
Parameters are numerical variables that can be set and updated from your game's code, and that are associated with particular events. An event's timeline is actually a special kind of parameter that automatically advances, and that all events have by default.

All instances of a given event have the same parameters, but the current values of those parameters can vary from one instance to the next. This allows each instance of an event to have a unique state.

Create or open an FMOD Studio project, create a new event, and make it active.

In the Editor, right-click on the parameter tab area and select 'Add Parameter...'

A window titled 'Parameter' appears, with fields labelled 'Name,' 'Minimum' and 'Maximum.'



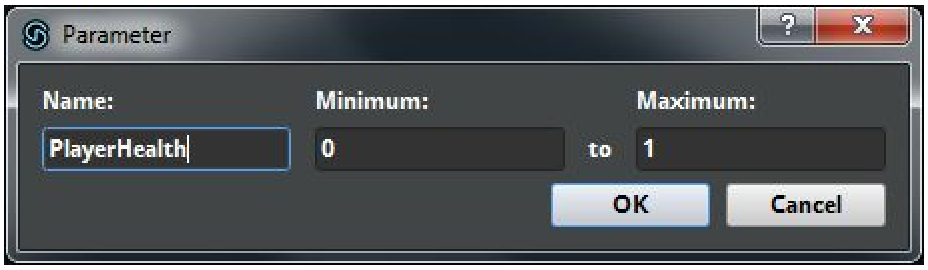
In this case, the parameter window has appeared because you're creating a new parameter, but it also appears when you edit an existing one.

Select the contents of the 'Name' field, then press the 'Delete' key to delete them.

The 'Name' field is empty.

Type "PlayerHealth".

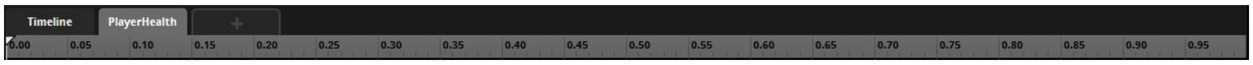
The name field now contains "PlayerHealth."



Parameter names are used when setting the value of a parameter in your game's code, so it's important to name them something that makes sense in the context of your game.

Click on the 'OK' button.

The parameter window disappears, and the new parameter is visible in the editor.

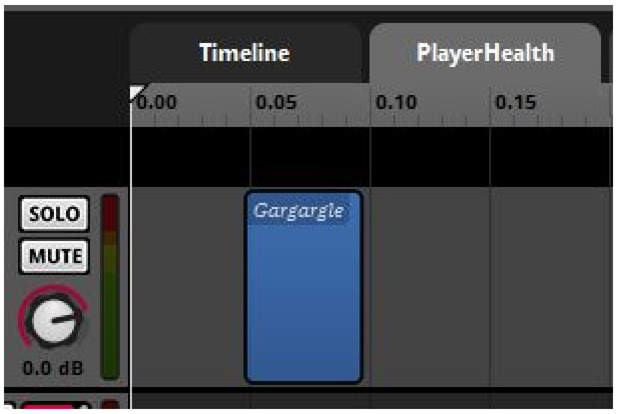


And now the event has a new parameter named 'PlayerHealth' which runs from 0 to 1. Of course, depending on the needs of your game you might want a parameter that runs from 0 to 100, or from -10 to 420, but 0 to 1 will work for this tutorial.

This parameter can be updated in code, allowing you to control the event event after it has been triggered. Of course, since the event currently doesn't have any content, changing the parameter won't make any difference, but that will change once we've added some content to the event.

Drag an audio file onto the track.

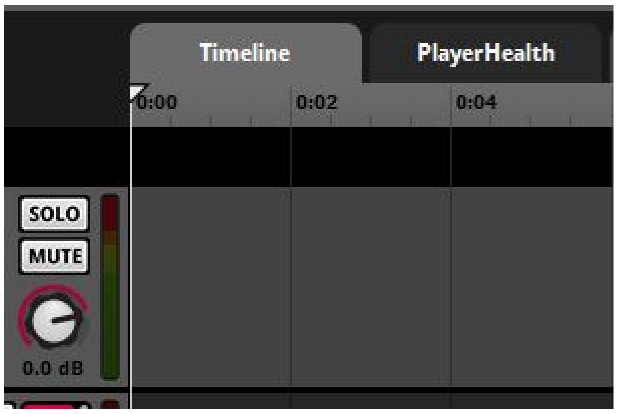
A sound module trigger region appears on the track.



This trigger region might look normal, but there's a trick to it.

Click on the Timeline parameter tab.

The trigger region disappears.



When you create a trigger region, it only exists on the parameter you created it on. A sound module can be triggered only by the parameter that holds its trigger region. In this case, you created the trigger region on the ‘PlayerHealth’ parameter; This means that its sound module is triggered by the ‘PlayerHealth’ parameter’s cursor, instead of the Timeline’s cursor, and won’t show up on the Timeline or any other parameter you create.

Because each sound module has exactly one trigger region, it’s not possible for a sound module to have trigger regions on more than one parameter of an event. There is no limit to how many parameters of an event can contain trigger regions, however, nor to how many trigger regions a parameter can contain.

Note that while only one parameter of an event can be viewed at a time, all of them exist simultaneously. This means that parameters that aren’t currently displayed can still trigger the trigger regions they contain.

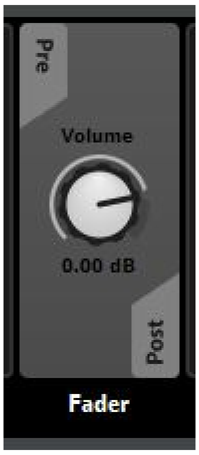
Automating Properties

Parameters can be used for more than just triggering sound modules. They can also be used to control most properties of tracks, effects, and any other module that can appear in the deck. The primary means of using parameters in this way is automation.

To automate a property, start by locating the property to be automated.

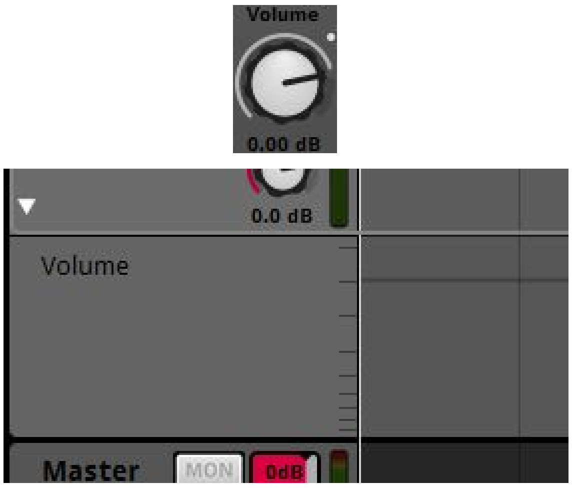
Click on the ‘Audio 1’ audio track head to make it the active item.

The ‘Audio 1’ audio track’s module appears in the deck.



In the Deck, right-click on the Volume knob to open the context-sensitive menu, then select the 'Add Automation' menu item.

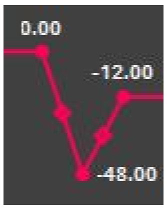
A small filled circle appears next to the Volume knob, and a new automation track appears in the event editor.



The filled circle simply indicates that the property is subject to automation. The automation track is the interface used to view and edit the property values that will correspond to particular parameter values.

Click and drag a few parts of the horizontal line in the automation track.

The line turns red, and points appear in the locations you click, connected to each other by the line.



Whenever the parameter cursor intersects the red line and the event is playing, the property is set to the value indicated by the red line at that point.

Click on the 'PlayerHealth' parameter tab.

The 'PlayerHealth' parameter is displayed. The automation curve disappears, though the automation track remains visible.

Automation curves, like trigger regions, only exist on a single parameter, and are only affected by that parameter's cursor.

Note that, when automating the properties of a sound module, it is entirely possible to place the trigger region of that sound module on one parameter, and to automate a property of that sound module on a different parameter.

Almost any property that is represented by a knob or dial in the Deck can be automated.

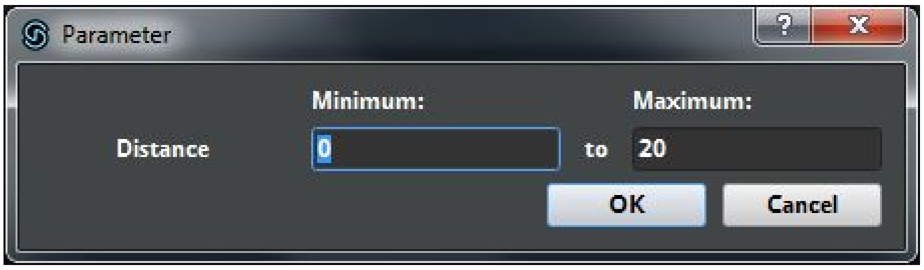
The Advantages of Built-in Parameters

Generally, when an event in your project includes a parameter, your game code needs to include function calls that update the value of that parameter. However, FMOD Studio includes a small number of 'built-in parameters' that are automatically updated based on information routinely fed to FMOD Studio's geometry system. Using these built-in parameters makes sense if you need an event to change, attenuate or otherwise react based on its position relative to the listener.

One of the most common applications of built-in parameters is to customise the way an event sounds different at different distances.

Right-click on the '+' parameter tab to open the context-sensitive menu, then select 'Add Built-in Parameter > Distance...'

The 'Parameter' window appears.


Parameter

Distance Minimum: 0 to Maximum: 20

There is no 'Name:' field in the Parameter window, as built-in parameters use fixed names.

Each built-in parameter has different minimum and maximum values, based on the range of values that that particular parameter is most likely to need. For the purposes of this tutorial, we will assume that the default minimum and maximum values of 0 and 20 are what this particular Distance parameter needs.

Click on the ‘OK’ button.

The ‘Parameter’ window closes, and the new parameter appears in the editor.

Once a built-in parameter is created, it behaves like any other parameter, with two exceptions: It cannot be renamed, and its value automatically updates based on the position of the event relative to the listener.

Click on various parts of the ‘3D Preview’ to reposition the event in different locations.

The value of the ‘Distance’ parameter updates accordingly.

More information about the various available built-in parameters can be found in the FMOD Studio manual.

Adding Modulators

Another tool for making events adaptive is Modulators. Like automation, modulators allow the value of a property to change depending on the circumstances. Different modulators have different effects.

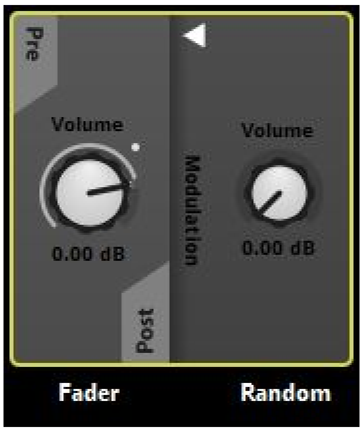
Adding a modulator to a property is very similar to the process of adding automation.

Click on the ‘Audio 1’ audio track head to make it the active item.

The ‘Audio 1’ audio track’s module appears in the deck.

In the Deck, right-click on the Volume knob, then select ‘Add Modulation > Random’ in the context-sensitive menu.

A modulation drawer appears to the right of the module.



The modulation drawer appears to the right of a module if any of that module's properties are modulated. By default, it is closed, meaning that none of the modulators applied to that module are displayed.

While the modulation drawer is expanded, the properties of the modulators can be adjusted. In this particular case, there is only one property, and it describes the range of values the fader value could be attenuated by when the event is triggered.

Modulation drawers are expanded by default, but can be collapsed by clicking on their disclosure triangles. Whether a modulation drawer is expanded or collapsed has no effect on its behaviour.

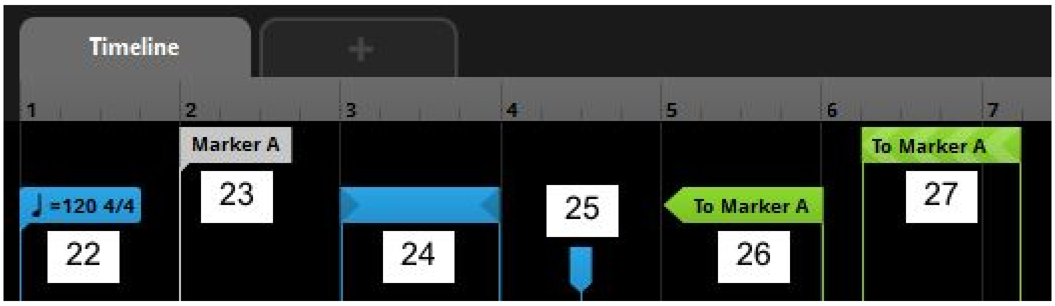
Information about the various kinds of modulators can be found in the FMOD Studio manual.

Tutorial: Making Adaptive Music

Music that changes in response to in-game events can contribute to almost any game.

When creating music that changes in reaction to in-game events, it is often easiest to use timelocked sound modules, as the ability to see the waveform can make the process of determining appropriate positioning for trigger regions and markers much easier.

Reading the Logic Track



22. Tempo marker
23. Destination Marker
24. Loop region
25. Sustain point
26. Transition marker
27. Transition region

The logic track can hold a number of markers that can alter the behaviour of an event's timeline in various ways. In general, when the timeline cursor reaches a logic marker, it is affected by that marker. When the cursor reaches multiple markers at the same time, it is affected by the highest one first, then the second highest, then the third, and so on in that order, until one of the markers moves the cursor to a different position or it runs out of markers at that position.

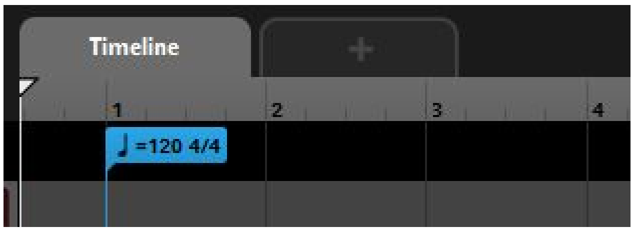
Setting Tempo

When creating music events, it is often vital to establish the tempo to which the piece is set.

Create or open an FMOD Studio project, create a new event, and make it active.

Right-click on the logic track and select the 'Add Tempo Marker' menu item.

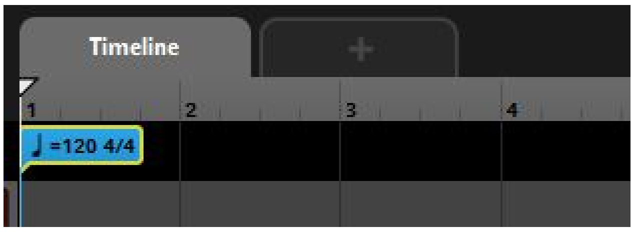
A blue tempo marker appears. The ruler changes to show bar numbers instead of timecodes.



Tempo markers set the tempo of the section of timeline on the right of the marker. We want the entire event to use a single tempo, so it needs to be to the left of everything else.

Click and drag the tempo marker to the extreme left end of the timeline.

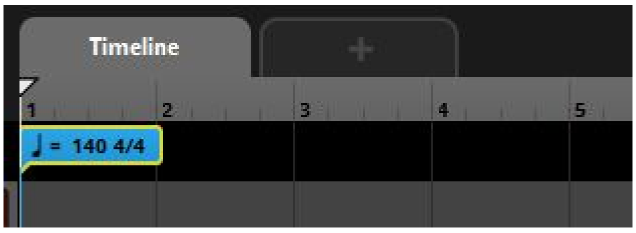
The tempo marker is now located at the left edge of the timeline. The timeline and everything on it is to the tempo marker's right.



By default, newly created tempo markers set the tempo to 120 beats per minute and 4/4 time. Not all music events use these settings, however, so it's often necessary to change them.

Double-click on the tempo marker, then type '140' and press 'Enter.'

The marker is now set to 140 beats per minute and 4/4 time.



By itself, a tempo marker does nothing. However, its presence does affect two things. One of these is the 'Snap to Ruler' behaviour.

Add a sound module of any kind to the 'Audio 1' track.

Click and drag the sound module left or right to reposition it.

The sound module jumps between specific bar and beat positions instead of sliding smoothly between all possible points along the timeline.

The 'snap to ruler' behaviour causes dragged objects to be aligned with the notches displayed on the parameter ruler. It can be overridden by holding down the 'Ctrl' key, or turned off by selecting 'View > Snap to Ruler.' In any case, 'Snap to Ruler' mode has no effect on how events behave in-game.

The other thing affected by the presence of a tempo marker is the behaviour of quantization in the 'Delay & Quantization' section of the 'Trigger Behaviour' drawer.

Create a new parameter named "Slot."

Click on the 'Slot' parameter tab.

Drag an audio file onto the 'Audio 1' audio track.

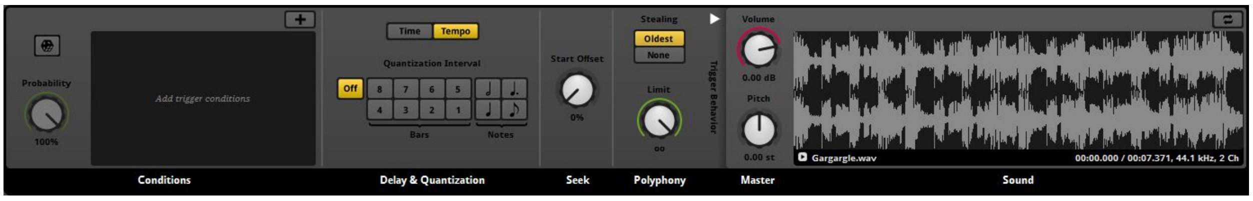
A single sound module triggered by the 'Slot' parameter is created on the 'Audio 1' audio track.

Click on the sound module's trigger region.

The sound module appears in the deck.

Click on the 'Trigger Behaviour' drawer's disclosure triangle to expand it.

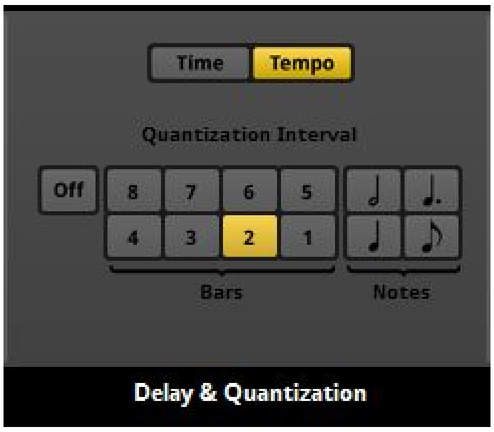
The trigger behaviour drawer expands, displaying its contents.



This drawer contains a number of properties that determine when and how the associated sound module is triggered. The section we're interested in is labelled 'Delay & Quantisation.'

In the 'Delay & Quantization' section, click on the quantization interval button labelled '2,' in the 'Bars' group.

The '2' button lights up.



Quantization interval is a way of constraining when sound modules and logic markers can start to specific points in time. By specifying two bars as the quantization interval of this trigger region, we ensure that it will only start playing at the start of an odd-numbered bar. Other than that, the behaviour of the sound module is unchanged; The sound module will only be triggered if the cursor overlaps its trigger region, and will behave as an ordinary single sound module once triggered.

Click on the play button to audition the event.

The event begins auditioning.

Click on the ruler to position the cursor overlapping the trigger region.

After a short delay, the sound module plays.

The slight delay is because of the quantization setting; The sound module was waiting until the start of an odd-numbered bar.

There is no limit to the number of tempo markers an event can contain. When there is more than one tempo marker in an event, the current tempo is set by the closest tempo marker to the left of the timeline cursor's current position.

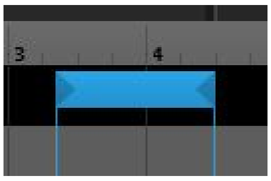
Creating Loop Regions

All the events we've dealt with so far have had linear timelines: The timeline cursor always moved from left to right. When designing music, however, it's often useful to loop certain sections of the composition so that they play repeatedly. The tool for this in FMOD Studio is the loop region.

Click on the Timeline parameter tab to return to the Timeline.

Right-click on the logic track and select the 'Add Loop Region' menu item.

A blue loop region appears.



As you might expect, loop regions cause a region of the timeline to be looped.

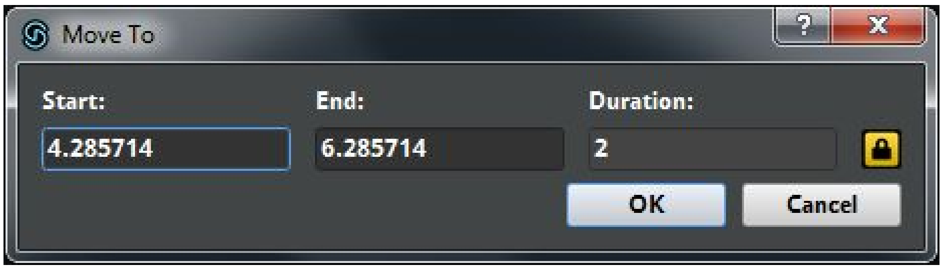
Click on the play button to audition the event.

The timeline cursor starts to advance. When it reaches the right end of the loop region, it returns to the left end.

By default, new loop regions are two seconds long, but they're easy to resize.

Right-click on the loop region and select the 'Move To...' menu item.

The 'Move To' window appears.



Start: 4.285714
End: 6.285714
Duration: 2

Select the contents of the ‘Start:’ field, then press the ‘Delete’ key to delete them.

The ‘Start:’ field is empty.

Type “4”.

The ‘Start:’ field is set to ‘4.’

The values in these fields are in seconds, so we’ve just set the loop region to start at 00:04:000.

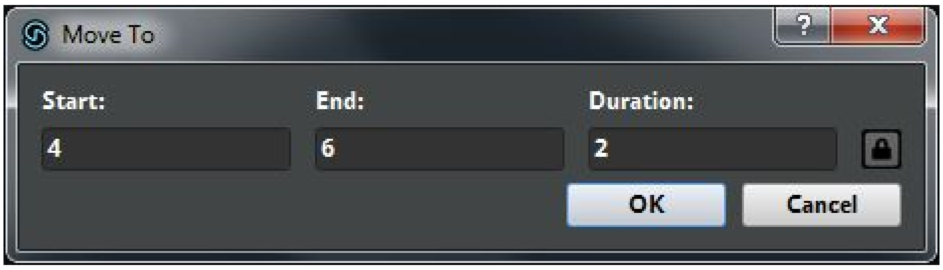
Click on the ‘End:’ field.

The ‘End:’ field is automatically set to ‘6.’

The value of the ‘End:’ field updated itself because the ‘Duration:’ field is locked. The duration field shows the difference between the ‘End:’ and ‘Start:’ fields, so as long as it remains locked, changing the value of either ‘Start:’ or ‘End:’ also changes the other field by the same amount.

Click on the lock icon.

The lock icon is unhighlighted, and the ‘Duration:’ field is unghosted.



Start: 4
End: 6
Duration: 2

With the ‘Duration:’ field unlocked, we can edit its value, and can edit each of the ‘Start:’ and ‘End:’ fields without the other also changing.

Change the value of the ‘Duration’ field to ‘1.’

Click on the ‘OK’ button.

The ’Move To’ window disappears, and the loop region is moved and resized to start at 00:04:000 and end at 00:05:000.

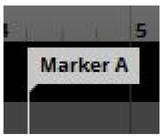
You can also resize a loop region by clicking and dragging its edges in the logic track.

Creating Transitions

Loop regions work well if we want to send the cursor back to an earlier point on the timeline, but sometimes we want to send it forwards, instead. For that, we have transition and destination markers.

Right-click on the logic track and select the ‘Add Marker’ menu item.

A destination marker labelled ‘Marker A’ appears on the logic track.

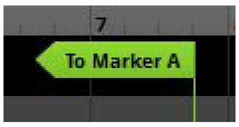


Destination markers can be renamed by double-clicking on them.

Destination markers do very little on their own. They simply mark positions on the timeline. To make them useful, you need to create a transition marker with the destination marker as its destination.

Right-click on the logic track and select ‘Add Transition To > Marker A’ from the context-sensitive menu.

A transition marker appears, labelled ‘To Marker A.’



The names of transition markers are based on the names of the destination markers they send the cursor to, and cannot be manually changed.

Click on the play button to restart the audition of the event.

The cursor advances from left to right. When it reaches the ‘To Marker A’ transition marker, it disappears and reappears at the ‘Marker A’ destination marker.

Warning: Double-clicking on a transition marker creates a transition timeline. Transition timelines are not covered by this tutorial, but can be defined as short timelines that the timeline cursor travels along between disappearing at a transition marker and reappearing at the associated destination marker.

If you ever notice that the timeline cursor disappears when it reaches a transition marker and does not immediately reappear, that transition marker may have a transition timeline.

To remove a transition timeline, right-click on the transition marker that has the transition timeline and select 'Remove Transition Timeline' from the context-sensitive menu.

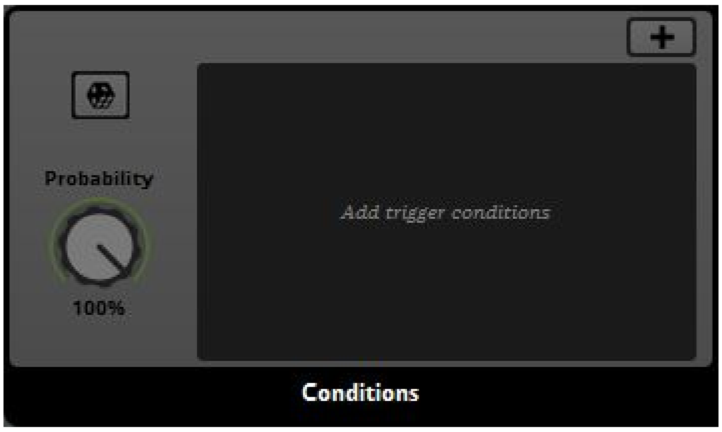
Remember that a sound module that isn't timelocked does not care which part of its trigger region the cursor overlaps; It is not possible to 'seek' inside a sound module that isn't timelocked by having the cursor enter mid-way through the trigger region.

Setting Conditional Logic

By default, a logic marker is 'always on' and affects the timeline cursor the same way every time they overlap. However, there are many situations where it is beneficial for a marker to only function while certain conditions are met. To achieve this, you can select a marker and specify the conditional logic that you want it to follow.

Click on the transition marker.

The transition marker is highlighted, and its panel appears in the deck.

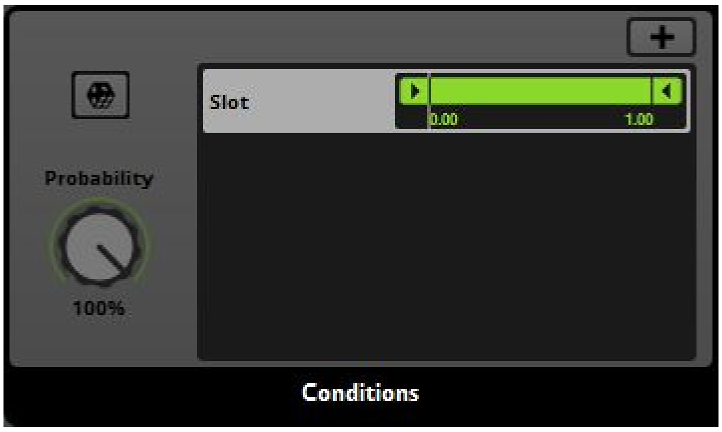


This panel has two parts. We won't be using the left side in this tutorial.

The right side allows you to specify a 'Parameter Condition,' a range of parameter values that must be met for the logic marker to function.

Right-click on the '+' icon, then select the 'Add Parameter Condition > Slot' menu item.

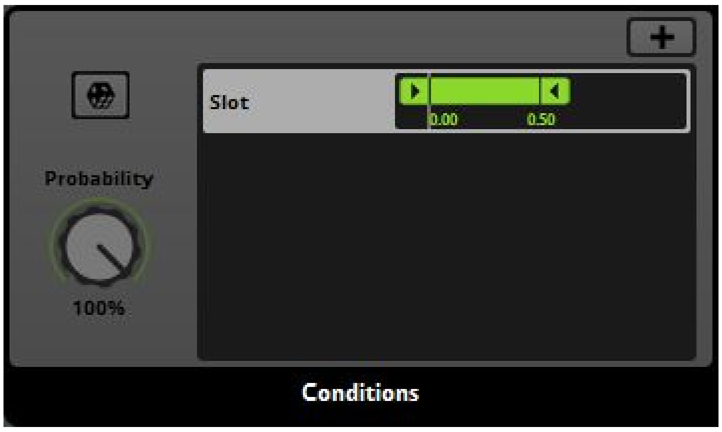
A ribbon slider appears on the Logic panel.



This ribbon slider describes a range of values that the Slot parameter can have. By default, this range includes all possible values of that parameter.

Click and drag the right handle of the ribbon slider until the number displayed beneath it is 0.50.

The ribbon slider resizes to fill only half of the range of possible parameter values.



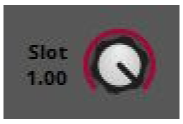
Click on the play transport button.

The cursor advances from left to right. When it reaches the 'To Marker A' transition marker, it disappears and reappears at the 'Marker A' destination marker.

As long as the parameter value is within the range of 0 to 0.5 described by the ribbon slider, the associated logic marker functions normally.

Turn the Slot parameter value knob to the right as far as it can go.

The parameter's value is set to 1.



Click on the play transport button.

The audition restarts, and the cursor advances from left to right. When it reaches the ‘To Marker A’ transition marker, it continues along the timeline without being affected.

When the parameter value is outside the range of 0 to 0.5, the marker is ignored, and the timeline cursor behaves as if that logic marker is not present.

Parameter-based conditional logic can be assigned to loop regions, transition markers, transition regions, and all kinds of sound modules. Because it is possible to update the value of a parameter from within your game’s code, it is possible to create ongoing events that change their behaviour depending on in-game situations.

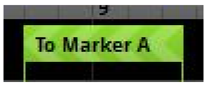
Understanding Transition Regions

While transition markers are useful, sometimes events require that a transition take place somewhere within a region of the timeline, but not necessarily at a single specific and predictable point. Transition regions fulfil this need by describing a range of values within which a transition could occur.

Aside from covering a range of values instead of a single point, transition regions function exactly like transition markers, and are created and used in a very similar way.

Right-click on the logic track and select ‘Add Transition Region To > Marker A’ from the context-sensitive menu.

A transition region appears, labelled ‘To Marker A.’



It’s generally not useful for a transition region to overlap with its associated destination marker, so if Marker A is inside the region of timeline described by your new transition region, you should click and drag either the marker or the region to a different location.

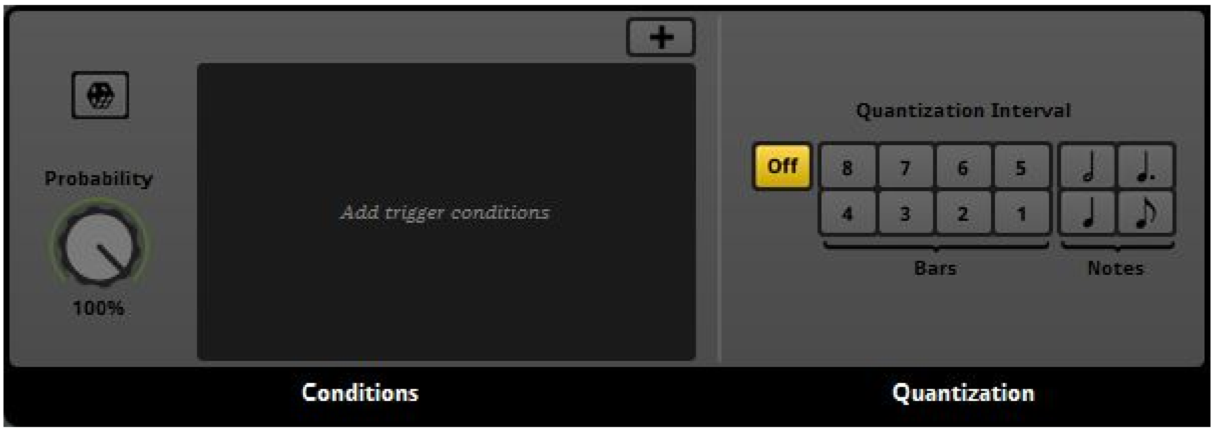
Click on the play transport button.

The cursor advances from left to right. When it reaches the ‘To Marker A’ transition region, it disappears and reappears at the ‘Marker A’ destination marker.

Generally speaking, transition regions behave in exactly the same way as transition markers, except that they cover a stretch of the timeline instead of a single point. This is particularly useful when using conditional logic, as the length of the region provides additional chances for the parameter condition to be met after the cursor initially enters it. There is one exception, however.

Click on the 'To Marker A' transition region.

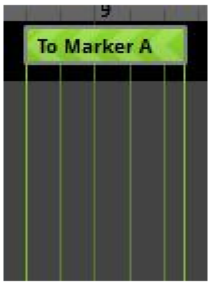
The region's panel appears in the deck.



This panel is very much like that of a transition marker, but contains the additional 'Quantization' section. This section allows you to limit the points within the region at which a transition can start to specific intervals based on the tempo.

Click on the Quantization section's 'quaver' button.

The 'quaver' button is highlighted, and a series of vertical lines appears under the transition region.



Click on the play transport button.

The cursor advances from left to right. When it reaches one of the vertical lines under the 'To Marker A' transition region, it disappears and reappears at the 'Marker A' destination marker.

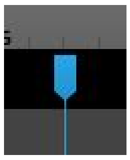
The pale green lines under the transition region represent the points within the region at which a transition can take place. Note that the bright green lines at the right and left edge of the transition region serve to mark the edges of that transition region and are not necessarily points at which a transition can take place.

Using Sustain Points

There is only one other type of logic marker in FMOD Studio. Sustain points cause the timeline cursor to maintain its position instead of moving past them.

Right-click on the logic track and select 'Add Sustain Point' from the context-sensitive menu.

A sustain point appears.



The exact way your event reacts to having its timeline cursor stop moving depends on its content it contains. Stopping the cursor at a single point on the timeline is not the same as pausing the event: Content on parameters other than the timeline will not be affected, and content on the timeline does not necessarily fall silent.

Drag an audio file onto the 'Audio 1' track to create a timelocked single sound module in a position where it is overlapped by the sustain point.

Add a second audio track to the event.

Drag an audio file onto the 'Audio 2' track to create a second single sound module in a position where it is overlapped by the sustain point.

Set this second sound module to loop, making it no longer timelocked.

Click on the play transport button.

The cursor advances from left to right. When it reaches the sustain point, it stops. The timelocked sound module stops producing output, while the other sound module continues to produce output.

Because timelocked sound modules always play the part of the waveform overlapped by the cursor, they cease producing any output when the cursor stops moving. (When the voltage of an audio signal becomes constant, it no longer produces audible output.) No other content of an event stops as a result of the timeline cursor stopping: Content on other parameters is completely unaffected, sound modules that aren't timelocked continue to output signals, the tails of effects and modulators continue to play out, and automation continues to have its effect. Of course, if the timeline cursor is not moving, the automation and trigger regions it intersects never change.

Tutorial: Routing and Mixing

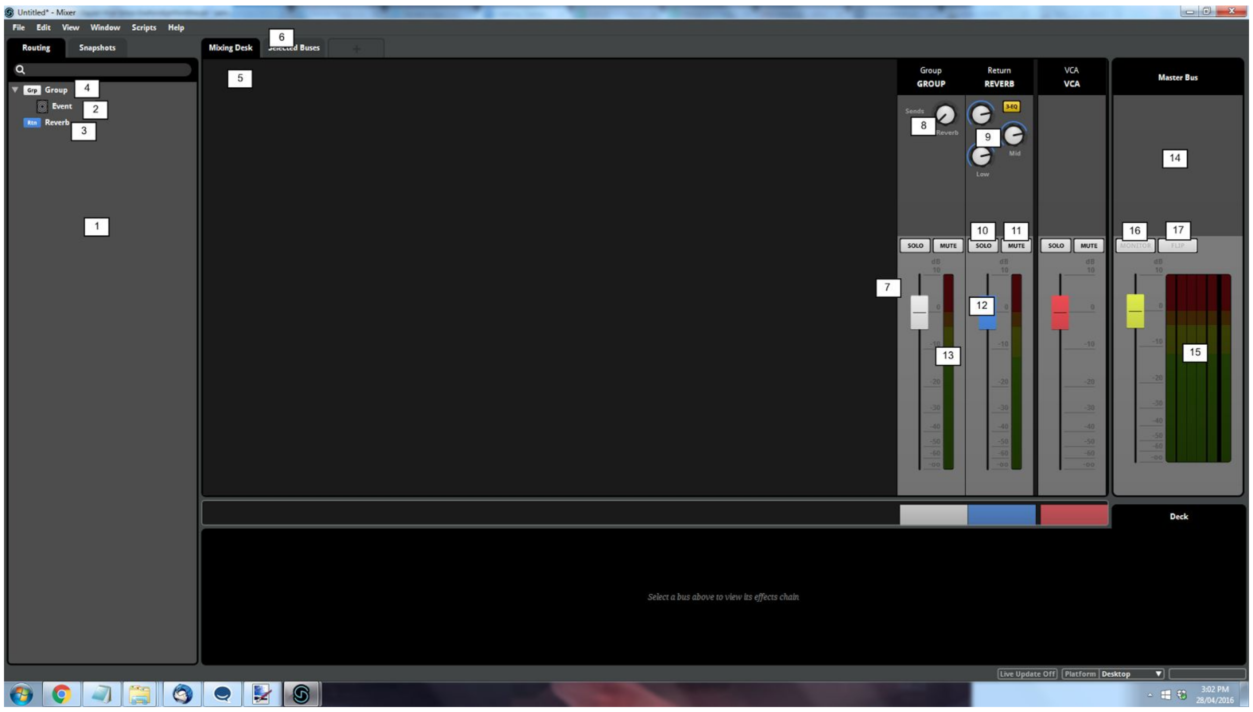
In films and music, the goal of mixing is to ensure that the elements of a mix are balanced against each other in order to achieve a desired overall effect. This is equally true in games - but complicated by the fact that you cannot accurately predict which elements will be present at any given moment.

The key to dealing with this dynamism is creating a mix that automatically evolves and changes to suit whichever sounds are currently present. FMOD Studio therefore includes a number of tools designed specifically to define how in-game events should change your project's mix.

Of course, FMOD Studio also includes the tools you need to mix a project more traditionally.

Understanding the Mixer Window

Many features in the mixer window are identical or nearly identical to features found in the event editor window. These features have not been labelled in the below diagrams.



1. Routing browser
2. Event
3. Return bus

4. Group bus
5. Mixing desk
6. Mixing desk tabs
7. Mixer strips
8. Send
9. 3-EQ
10. Solo button
11. Mute button
12. Slider
13. Meter
14. Master bus
15. Master bus meters
16. Monitor button
17. Flip button

Routing Signals

Most mixing in FMOD Studio requires that all events in the project are routed into group buses.

Generally, a project requires one group bus for each group of events that you want to manipulate as a group. For example, if your game has separate volume settings for music, sound effects and character voices, it needs to contain one group bus for each of those groups of events - and if you want to manipulate specific subgroups of the events within those three broad groups, you will most likely need additional group buses for each subdivision.

Open the examples.fspro project

The “examples” project that ships with FMOD Studio will be used throughout this tutorial.

Select ‘Window > Mixer.’

The Mixer window appears. The Routing Browser at the left of the window shows the buses and events that route into the project’s master bus, and the Mixing Desk shows the mixer strips for various buses in the project.

To start with, we only need to make a small change: Setting the “Command” event to route into the “Character” group bus.

The “Character/Radio/Command” event appears in the Routing Browser. It is not inside any group bus.

The Routing Browser displays the full path of the “Command” event, instead of just its name. This is because event folders do not affect a project’s routing, and so aren’t shown in this browser as folders. Instead, the Routing Browser displays group buses.

Any event or bus that appears to be ‘inside’ a group bus, routes into that group bus. Buses and events that are outside all group buses instead route directly to the project’s master bus. In other words, in the Routing Browser, the signal flows from right to left.

Click on the disclosure triangle next to the “SFX” group bus.

The buses that route into the “SFX” bus are displayed. The “Character” group bus is among them.

There’s the group bus we want our event to route into. Now we just need to make the change.

Click and drag the “Character/Radio/Command” event onto the “Character” group bus.

The events that route into the “Character” group bus are displayed. The “Character/Radio/Command” event is now among them instead of in its original position.

Now every time our game creates an instance of the “Command” event, the output of that event instance will route into the “Character” group bus, where it will be mixed with the outputs of all the other event instances that route into that group bus.

Mixing Using the Mixing Desk

Once the various events in your project are routed into group buses, the Mixing Desk can be used to attenuate or boost specific group buses as per your project’s needs.

Note that under most circumstances, mixing is done while the project’s events are playing through the sandbox or in-game. However, for this part of the tutorial, we will be mixing without auditioning.

We want to reduce the volume of ambiences relative to other sounds in our game. All ambience events are routed through the “Ambience” bus, so we need to adjust that bus’ volume property.

Each bus in the project is represented in the Mixing Desk as a mixer strip. Each mixer strip is labelled with a bus type and name, and features a volume slider.

The names shown on the mixer strips are abbreviated versions of the names shown in the routing browser. We need to find the mixer strip that corresponds to the “Ambience” bus.

Hover the mouse cursor on the word “AMBINC” where it appears on a mixer strip.

After a short delay, a tooltip appears. The tooltip says “Ambience,” the non-abbreviated version of the mixer strip’s name.

This is the mixer strip we need. All we need to do now is reduce its volume.

Click and drag the slider of the “AMBINC” mixer strip down to about -10 dB.

The “Ambience” bus’ signal chain appears in the Deck. In the Deck, the bus’ volume property is displayed as a knob labelled ‘Fader.’.

It’s also possible to adjust the sliders of multiple mixer strips at once.

Click on the label of the “AMBINC” mixer strip.

The “AMBINC” mixer strip is highlighted in gold.

Hold down the ‘Shift’ key and click on the “WEAPNS” mixer strip.

All the mixer strips between “AMBINC” and “WEAPNS” are highlighted in gold.

The mixer strips are now part of a multi-selection. Moving the slider of any mixer strip in this multi-selection will also move the sliders of other mixer strips in the multi-selection.

Click and drag the slider of the “AMBINC” mixer strip up to 0 dB.

The “AMBINC” slider is set to 0 dB, and the other sliders in the multi-selection are set to +10 dB.

The values of the other sliders in the multi-selection have been adjusted so that they retain the same values relative to the dragged slider. It is also possible to adjust them to the same absolute value as the dragged slider.

Hold down the “Shift” key and drag the slider of the “CHRCTR” mixer strip down to -10 dB.

All mixer strips in the multi-selection are set to -10 dB.

On second thought, we shouldn’t have made all these adjustments. Let’s put everything back the way it was.

Double-click on any slider in the multi-selection.

All the sliders in the multi-selection are set to 0 dB.

Double-clicking on a controller sets it back to its default value. Double-clicking a controller in a multi-selection sets every controller in the multi-selection back to its default value.

Effects and Signal Chains

Each bus in the project has a chain of effects that it applies to the signal, as well as a fader that adjusts the signals' volume. By adding effects modules to a bus' signal chain, you can change the signal at that point.

In the Mixing Desk, click on the "CHRCTR" mixer strip.

The "CHRCTR" mixer strip is highlighted in gold. The deck displays the signal chain of the "Character" bus, currently empty except for the 'Fader' panel.

A newly created group bus has a mostly-empty signal track and a fader value of 0 dB, meaning that it does not alter the signals it receives as input in any way other than mixing them. We can change this by adding an effect.

Right-click on the '+' indicator to the right of the 'Fader' panel to open the context-sensitive menu, select the 'Add Effect' menu item, then select the 'FMOD Flanger' submenu item.

A 'Flanger' effect module appears to the right of the 'Fader' panel.

Now the "Character" group bus is subject to a Flanger effect, meaning that the signal that passes through it will be subject to that effect. Four events route into the character bus, so instances of any of those events will sound different as a result of this effect.

Right-click on the empty space to the right of the 'Flanger' effect module to open the context-sensitive menu, select the 'Add Effect' menu item, then select the 'FMOD Lowpass Simple' submenu item.

A 'Lowpass Simple' effect module appears to the right of the 'Flanger' effect module.

When more than one effect appears in a signal chain, the signal flows through them from left to right. In this signal chain, the signal is affected by the Fader, then the 'Flanger' effect, and then the 'Lowpass simple' effect. In practice, the precise order of effects in a signal chain rarely matters.

One situation where it can matter is when a signal chain includes a 'send.' A send creates a copy of the signal at the point where it exists, and outputs that copy to a 'return bus' elsewhere in the project.

A return bus functions the same way as a group bus, except that you cannot route other buses into it. Instead, it receives its input from sends throughout the project.

Examples.fspro already contains one return bus, named “Reverb”. Let’s create a send to that bus.

Right-click on the ‘+’ to the left of the ‘Fader’ panel to open the context-sensitive menu, then select “Add Send > SFX > Reverb”.

A send module appears in the deck to the left of the ‘Fader’ panel. A “Reverb” send knob appears on the “CHRCTR” mixer strip.

At the moment, this send is to the left of the Fader and other modules in the signal chain. This means that it sends a copy of the signal as it is before it is affected by the fader or effects modules.

If we instead want to send from another point in the signal chain, we can simply move the send to a different place.

Click on an empty part of the send module, and drag it between the ‘Flanger’ and ‘Lowpass Simple’ modules.

The send module is now between the ‘Flanger’ and ‘Lowpass Simple’ modules.

Now the send creates a copy of the signal as it is after it is affected by the fader and ‘Flanger’ module, but before it is affected by the ‘Lowpass SImple’ module.

There’s no limit to the number of effects and sends that can be added to a signal chain. However, effects modules and sends do require some system resources to function, so projects with many effects modules may perform poorly on some platforms.

Creating Snapshots

Because the situation in a game can change based on a player’s actions, it’s often necessary to change the mix to suit whatever situation the player is in. For this purpose, FMOD Studio includes snapshots.

Click on the ‘Snapshots’ tab.

The Snapshots browser is displayed.

The snapshots browser lists all the snapshots currently in the project in order of priority. We’ll learn about snapshot priority a little later; For now, it’s time to create a new snapshot.

Right-click on the snapshots browser to open the context-sensitive menu, then select the ‘New Overriding Snapshot’ menu item.

A new snapshot appears. In the Mixing Desk and Deck, all properties and sliders are displayed with dashed outlines instead of their knobs and slider handles.

Those dashed outlines indicate properties that aren't 'scoped in' to the snapshot. Only properties that are scoped in to a snapshot can be affected by that snapshot; Other properties are ignored and remain at the their default values or the values assigned to them by other active snapshots in the project.

Right-click on the dashed outline of the "CHRCTR" mixer strip's slider handle to open the context-sensitive menu, then select the 'Scope In' menu item.

The slider handle appears in place of its dashed outline. The Fader knob appears in the deck in place of its dashed outline.

Now the "CHRCTR" bus' volume property is scoped in to the "New Snapshot" snapshot. It's also the only property scoped in to this snapshot, but it's the only property we need it to affect.

Note that while it is theoretically possible to create a snapshot whose scope includes every property in the mixer, there are very few situations in which this would be useful. As a result, most snapshots only include a small number of properties.

When a snapshot is active, it acts like a mask, replacing the normal values of its scoped-in properties with the values specified in the snapshot. When the snapshot becomes inactive again, the properties return to their normal values. Currently, when the "New Snapshot" snapshot is active, it sets the "CHRCTR" bus' volume slider to 0 dB. This isn't very useful, as it's the same value the "CHRCTR" bus' volume slider has normally.

Click and drag the slider of the "CHRCTR" mixer strip down to about -10 dB.

Now the snapshot sets the volume of the "CHRCTR" bus to -10 dB, a significant change. We can compare the snapshot to the unaffected mixer to see if it looks right.

Click on the 'Mixing Desk' breadcrumb.

The mixing desk stops displaying the snapshot, and instead displays the normal mix for the project. The "CHRCTR" mixer strip's slider is set to 0 dB.

Changing the value of a property in a snapshot doesn't affect the normal value of the property.

In the Snapshots browser, click on the "New Snapshot" snapshot.

The Mixing Desk displays the "New Snapshot" snapshot.

While the snapshot's currently displayed in the Mixing Desk, that doesn't mean we'd be able to hear it if we auditioned an event. To do that, we have to audition the snapshot. Auditioning the snapshot is something we can do quite easily.

In the MIxing Desk's transport controls, click on the play button.

The snapshot starts auditioning.

With the snapshot auditioning, any event that routes into the buses it affects will sound different when auditioned.

Click on the stop button twice to pause and stop the snapshot.

When made active (or auditioned), a snapshot sets each of the properties in its scope to the values specified in the snapshot. When the snapshot ceases to be active, those properties all return to their normal values. In either case, the scoped-in properties' values usually change instantly, but we can change this by using a feature snapshots have in common with events: Timelines and parameters.

In the transport controls, click on the tracks button.

The Mixing Desk displays the tracks view. The buses scoped in to the snapshot and the master bus are shown as tracks.

The tracks view is only available in the Mixing Desk when viewing a snapshot. It allows us to automate and modulate the properties scoped into a snapshot just like we would the properties of an event.

In our case, we want to ensure the snapshot doesn't result in abrupt changes to the sound of the project when it becomes active or becomes inactive.

In the Deck, click on the Event Macros tab to view the snapshot's macro controls.

The Desk displays the snapshot's Intensity property.

The Intensity of a snapshot can be thought of as the wet/dry mix between the normal values of the properties scoped in to the snapshot and the values specified in the snapshot. An intensity of 100% means that the snapshot's values will completely replace the normal values, an intensity of 0% means that the snapshot does not change the properties' values, and an intensity of 50% means that the scoped-in property values are set to halfway between their normal values and the values specified in the snapshot.

In our case, we want the intensity of the snapshot to start at 0 when the snapshot is first made active, then to ramp up to 100% intensity over a the next second. We also want it to ramp down from 100% Intensity to 0% Intensity when the snapshot is made inactive again. To do this, we just need an AHDSR modulator.

Right-click on the Intensity knob to open the context-sensitive menu, then select 'Add Modulation> AHDSR.'

A modulation drawer containing an AHDSR modulator appears in the deck.

There, now the AHDSR modulator will ensure the snapshot takes a full second to reach peak intensity, and a full second to fade back to normal when the snapshot ceases to be active.

Snapshots can be triggered from your game's code, just as events can be. For more information on triggering snapshots from your game's code, see the FMOD Studio Programmer's API documentation. Alternatively, snapshots can be triggered from events, just like event sound modules and event reference sound modules, by creating a snapshot trigger region in an event.

Organising Snapshots

As mentioned above, each snapshot only alters a small number of the properties in a project. A property not scoped in to a particular snapshot is in no way affected by that snapshot, and can therefore be freely altered by other snapshots. However, it's also possible for multiple snapshots (or multiple instances of the same snapshot) to simultaneously alter a single property. When this happens, there are some simple rules that determine how the property is affected.

Right-click on the snapshots browser to open the context-sensitive menu, then select the 'New Overriding Snapshot' menu item. Name the snapshot "Undercurrent".

A snapshot named "Undercurrent" appears in the snapshots browser and in the Mixing Desk.

Click and drag the dashed outline of the "CHRCTR" mixer strip's slider handle down to about -30 dB.

The dashed outline is replaced by the slider handle, which is dragged to the appropriate position.

There are now multiple snapshots in the project that can change the "CHRCTR" bus' volume. What happens when more than one are active at once? Well, these are "Overriding" snapshots, so the answer is that one "overrides" the other. To find out which takes priority, just look at the order they appear in in the Snapshots browser: Active snapshots are applied in the reverse of the order in which the Snapshots browser lists them; Consequently, the snapshots at the top of the Snapshots browser are applied last, overriding all the snapshots from lower down.

In the snapshots browser, the “New Snapshot” snapshot is higher priority than the “Undercurrent” snapshot, but lower priority than the “IngamePause” snapshot.

That means that if both the “Undercurrent” and “New Snapshot” snapshots are active at the same time, the “New Snapshot” snapshot is the one that sets the value of the “CHRCTR” parameter. The “IngamePause” snapshot always has the last word whenever it is active, because it is the highest snapshot in the Snapshots Browser.

Note that all active snapshots are applied, even if the property values set by one snapshot are being overridden by another snapshot. This is because if a snapshot instance’s Intensity property is set to anything less than 100%, it alters properties to hold values partway between their original values and the values specified in the snapshot. As the snapshots are applied in order, this means that an active snapshot with an intensity of less than 100%, positioned higher in the Snapshots browser than another active snapshot, will set property values to points partway between the values specified in the higher snapshot and the values specified in the lower snapshot.

Note that besides “overriding” snapshots, there are also “blending” snapshots. These operate exactly the same way as overriding snapshots, except when dealing with volume. Instead overriding each other’s volume property values, blending snapshots combine those values.

Tutorial: Building the Project

The end goal of creating content in Studio is, of course, to use that content in your game. Before you can implement content in your game, however, it needs to be “built”: Output in a format that your project can use.

Open the “examples.fspro” project that came with FMOD Studio.

You can build a project as many times as you want at any stage during its development. For this tutorial, we’ll make use of the “examples” project that comes with FMOD Studio.

Assigning Events to Banks

FMOD Studio builds events into “banks” of content. Before you can trigger an event in your game, at least one bank that contains that event has to be loaded. Therefore, if you want an event to actually appear in your game, you need to assign it to one or more banks.

In the Events Browser, click on the disclosure triangle of the “UI” folder to expand it.

The contents of the UI folder is displayed. The “#unassigned” tag appears next to the “IngamePause” event.

The “#unassigned” tag appears next to events that are not yet assigned to any bank. We want to build all of our events, so we need to assign this one to a bank.

Right-click on the “IngamePause” event to open the context-sensitive menu, select the ‘Assign to Bank’ menu item, then select the “UI_Menu” submenu item.

The “#unassigned” tag disappears.

The “IngamePause” event is now assigned to the “UI_Menu” bank, and so Studio has automatically removed its “#unassigned” tag.

There are a few other unassigned events in the project. It would be time-consuming to find them manually, so it’s easier to search.

In the search bar of the Events Browser, click on the loupe icon, and select ‘Filter by Unassigned Events.’

The Events Browser displays a subset of all events and folders in the project.

Selecting ‘Filter by Unassigned Events’ or typing “#unassigned” in the search bar causes the Event Browser to only display events if they have the “#unassigned” tag.

Expand all of the folders by clicking on their disclosure triangles.

Assign both the “Music/Basic/Random Layered” and “Music/Complex/Situation_Oriental” events to the “Music” bank.

Note that the two events we just assigned to banks remain visible, even though we used the search bar to filter by unassigned events. We could refresh the results by searching again, but for now it’s not important.

We have now assigned all the events we need to build to banks. The next step is to actually build.

Building your Whole Project

The simplest way to build a project is to build the entire thing at once.

Select “File > Build…”

A progress bar appears, and FMOD Studio builds the project.

Once the progress bar disappears, the built project files are ready to be used in your game.

Finding the Built Banks

The built project files can be found in your project's build directory. We therefore need to know what the project's build directory is.

Select "Edit > Preferences…"

The preferences window appears.

Click on the 'Build' tab.

The content of the Build tab is displayed. The built banks output directory field is blank.

This field usually displays the folder in which built bank files are placed. If this field is blank, it defaults to the "Build" subdirectory of the project folder.

In Finder or Windows Explorer, navigate to the examples project folder.

Open the "Build" subdirectory.

Inside is another subdirectory called "Desktop."

The "Desktop" folder contains all the built banks for the Desktop platform. Note that if we had defined build settings for more than one platform, they would appear as separate folders here.

What Building Produces

Open the "Desktop" folder.

The folder contains one .bank file for every bank in the project, as well as a .strings.bank file.

With the exception of the .strings.bank file, each .bank file contains the compressed assets and data of the events assigned to it. Before playing any event can be played in your game, at least one bank file that contains that event needs to be loaded into memory.

The .strings.bank file contains the names and paths of every event in your project, as well as their associated GUIDs. Certain convenience functions in the FMOD Studio programmer's API make use of these, allowing your game's code to call events by name and path. If the .strings.bank file is not loaded by your game, you can only call events by GUID. Most games keep this file loaded at all times.

For more information on loading and using banks, see the FMOD Studio programmer's API documentation.

Tutorial: Live Update

When mixing and editing a project, it often helps to be able to hear the results in the game. Live Update allows you to connect FMOD Studio to your game as it runs, make changes to your project, and hear the results in real time.

This tutorial makes use of the "FMOD Studio UE4 Integration for Windows/Android" and "UE4 Tutorial Video Assets files - Completed FMOD Studio project and UE4 level", both available from http://www.fmod.org/download/, as well as Unreal Engine 4, available from https://www.unrealengine.com.

Connecting to a Game

Usually, if you want to use Live Update, your game's code needs to set the FMOD_STUDIO_INIT_LIVEUPDATE flag when it calls Studio::System::Initialize. For more information about flags in FMOD Studio, see the FMOD Studio Programmer's API. However, the FMOD Studio UE4 Integration comes preconfigured for live update, so we don't need to worry about that for this tutorial.

Open "FMOD Tute.fspro" in FMOD Studio.

Open "FMOD_UE4_Tutorial" in Unreal Editor 4.

In Unreal Editor 4, play the level.

In FMOD Studio, select "File > Connect to Game…"

The 'Connect to Game' window appears.

The 'Connect to Game' window lets you specify an IP address, so that we can connect to a game over a network. This is particularly useful when running a game on a console development kit. When running the game on a local machine, we can simply use the default "localhost".

Click on the 'Connect' button.

The 'Connect to Game' window disappears. A green 'Live Update | Enabled' indicator is visible in the status bar.

With that, we're connected to the game as it plays in Unreal Editor.

Live editing

Almost every aspect of an FMOD Studio project can be edited during live update. There is only one exception: It is not possible to add audio assets to or remove audio assets from a project. If you want to add or remove audio assets, you need to do so while not connected, and will have to rebuild your project and put the new banks into your game before you can hear the assets in play.

Note that the built banks of a project are not automatically updated with the changes you make during live update. This means that if you end Live Update and restart your game, you will no longer be able to hear any changes made during the Live Update session. To hear the changes made during a Live Update session in future sessions of your game, save and rebuild your FMOD Studio project, then add the updated banks to your game project.

Understanding the Profiler

Select 'Window > Profiler.'

The Profiler window appears.

Creating a Profiler Session

The profiler allows you to record "sessions" live update. Profiler sessions are useful when investigating all sorts of issues that can arise in the audio of a game.

Right-click in the Sessions browser to open the context-sensitive menu, and select 'New Session.'

A session named 'New Session' appears in the Sessions Browser. The 'New Session' session is displayed in the Editor.

The editor displays some of your project's buses as tracks. By default, it only includes the Master Bus, but you can 'scope in' other buses at any time.

Select 'Window > Mixer Routing.'

The Mixer Routing window appears.

Click and drag the 'Character' group bus from the Mixer Routing window onto the Profiler window's Editor.

The 'Character' bus appears as a track in the Profiler window's Editor.

There’s no limit to the number of buses that can be scoped into a session. However, in most cases it is only necessary to scope in the specific buses in which we have an interest.

Close the Mixer Routing window.

Now that we’ve scoped in the bus we need, we can simply record the output of the game. A recorded session contains both the audio output of the game and the API calls that were made, allowing the session to be replayed and inspected in detail even when no longer using Live Update.

In the Editor’s transport controls, click on the record button.

The record button is highlighted in red. In the tracks view, the timeline cursor advances along the timeline, and graphical data starts to appear on each track and sub-track.

In the Unreal Engine 4 editor, perform some actions in the game.

For the purposes of this tutorial, it doesn’t matter exactly what you do in the game. In other situations, you may wish to emulate an ordinary play session, or reproduce a particular sequence of events that you wish to investigate.

In the Profiler window, click on the stop transport button.

A number of tracks appear in the tracks view.

When you stop recording a session, FMOD Studio automatically scopes in any event that was played at any point during that session. This lets you know which events were active during a session.

Unlike events, group and return buses are not automatically scoped in when you stop recording a session. This is because group and return buses are technically present in every recorded session, so there is no way for FMOD Studio to predict which of them are relevant.

Select ‘File > Disconnect from Game.’

The record button ceases to be highlighted, the timeline cursor stops moving, and new graphical data stops appearing.

Now that we have recorded a session, we can examine it in detail. But first, as we are done recording the session, we no longer need to be running the game.

In the Unreal Engine 4 editor, click on the Stop button.

The game simulation ends.

Note that Live Update and recording sessions using the profiler do themselves consume resources, and may result in your game running slightly slower than would otherwise be the case. Recorded profiler sessions are nonetheless a valuable diagnostic tool.

Learning from the profiler

The profiler is currently displaying the CPU useage of each thing scoped into the session. This is just one of nine possible graphical representations of recorded data that the profiler can display. You can change which is displayed by clicking on the six categories in the transport bar.

Click on the Memory category.

The first circle next to the Memory category label is filled. The memory use of each bus and event is displayed on each track.

Click on the Memory category again.

The text of the Memory category changes to 'File I/O,' and the second circle next to it is filled. The File I/O of each bus and event is displayed on each track.

Each category displays different information about the project.

CPU indicates the CPU usage of each track as a percentage.

Memory indicates the the memory used by each bus measured in KB, and File I/O indicates the amount of data read from files, also in KB.

Levels indicates the levels of each track.

Voices indicates the number of voices playing through a track. Note that a single event instance can have multiple voices. Voices (Self) includes only the voices originating in an event, but not the voices originating in other events referenced by that event. Voices (Total) includes both the voices originating in an event and the voices originating in events references by that event.

Lifespans represents each instance an event as a line. Note that this information is only displayed on the tracks of events, and not on the tracks of buses.

Instances indicates the total number of instances of each event. Instances (self) includes only the instances of an event, and not the event instances of the events it references. Voices (Total) includes both the instances of the event and those of the events it references.

Once a session has been recorded, it's possible to play it back.

Click on the play transport button.

The timeline cursor moves along the timeline. The audio produced during the recorded session plays back exactly as it occurred.

This allows you to listen to the audio in a session. This is most useful when trying to isolate the possible causes of audible bugs in your game.

While editing your project, it is also useful to be able to make changes and see how they affect the recorded mix. To do so, we need to use API playback mode.

Click on the stop transport button twice.

The profiler session stops playing back and the timeline cursor resets to the beginning of the session.

Click on the API button to the right of the time indicator.

The API button it highlighted in yellow.

Click on the play transport button.

The timeline cursor moves along the timeline. The events that were triggered and parameter changes that were made during the session play back.

The profiler is now in ‘Playback with API Capture’ mode. In this mode, instead of playing back the audio captured during the recorded session, Studio plays the API callbacks captured during the recorded session.

Using Playback with API Capture mode, you can make changes to your project and then play back the profiler session to hear the effect of the changes.

For more information about the profiler, see the FMOD Studio User Manual.