Sound

FMOD

HPL3 uses FMOD to handle all sound playback. See fmod.org for more details, documentation and help. When working with sounds it is mainly the FMOD Ex Designer tool that will be of interest, the easiest way to learn how it works is to check the tutorials in FMOD's YouTube channel.

General

It is important to plan and think through how to split up your sounds into different projects. The game will load sounds based on this, it will help you find your sounds, it will ease preloading if you need to use that and so on.

Project

Create a project for large chunks of sound types, for example a project for each character, level or for physics. Each project can contain many groups of sound events, preferably a hierarchy of 1-2 sub-groups will make it organized and easy to work with when for example optimizing. More details about this is found under FMOD Designer Events. Some type of sounds need to follow a special structure and naming convention! See the sections below, for example physics and footsteps.

Banks

A bank stores all the sounds for a project, you can have many banks but you should normally only need two. One bank for sounds that you load into memory to play and one bank that streams sounds from the harddrive and plays them. Special cases where you need to have more banks would be if you have sounds that are localized to different languages, where you have one bank for each language.

Sounds to be streamed are usually longer sounds, such as a looping ambient track or voices. Be cautious with using streamed sounds, streams depend on the speed of the harddrive, so what works well for you might not be so for another user. Try to keep the number of simultaneous streams to something low, perhaps max 6 streams. Remember that music and dialog streams, so could easily be 3 streams if you have a cross-fade of music going when a voice is played.

Example files

Our releases always have all the final FMOD project files in the game sound/ folder. The actual source sound files are not included, but you can view the projects and see how they are organized and what settings are used for reference.

FMOD Designer Sound Defs

A sound def is the settings for a sound (or set of sounds for variation) that you can use in your Multi-track events. There is nothing specific to keep in mind for these, do as suitable and follow the FMOD documentation.

FMOD Designer Events

The Events are what you actually use when playing sounds in the engine. The events are what replaces our old system of playing a SNT file, which essentially worked in the same way as Simple Events. When working with events there are a few things to keep in mind, mainly to organize events into easy to navigate groups and to use categories to send the events to be processed (or not processed) by the proper global effects.

Simple Event

Nothing in particular to keep in mind beyond the basic usage as specified by the FMOD documentation. Just check our general important notes for events and performance concerns for info on why you should use simple events and any minor specific setting you might need to set.

Multi-track Event

These events allow for parameters to affect the sound in a detailed manner, for example changing the sound of an impact depending on the speed of the object or much more advanced mixes. Make sure to view Performance concerns for Events to not overdo it. These events replace what previously was set with a lot of parameters in the snt editor, model editor and level editor for sounds. The move to FMOD makes it possible for a sound designer to do much more without any engine specific knowledge or assistance from a programer/scripter.

Following is a quick overview of events, for more details check FMOD documentation and tutorials.

An event is created with layers, a layer can contain one or many Sound Defs. At first glance it is much like any sound editor, but as you begin to make your first event you notice that what normally is a representation of time is not so in the context of making FMOD Events.

To be able to move your sounds around you will need to right-click the top of the event and add a parameter. The parameter can represent numerous things, such as time or the engine sending in a value with the speed of an object. These parameters can be used to make envelope mixes and different sound def play depending on the value of a parameter or mix of parameters. For example see physics, the engine sends in the speed and mass of an object, the speed sets the volume and what type of sound def to play and the mass is used to change the pitch of the sounds (increased mass = lower pitched sound).

Let's say we have a machine in the game and the player will be able to start and stop this machine. A single event can be created to be used for this, where we first play a sound as the machine starts up, then loops a sound while it is running and finally plays a sound as it stops.

1. Right-click the top and add a parameter to the event, the name does not matter.
2. Right-click the parameter and choose parameter properties, set Velocity to 1. This will make the event play like in a typical sound editor.
3. Right click on your layer and add in three sounds defs. This should be a start sound, a loop sound and a stop sound.
4. Right-click on the start and stop sound defs and choose Sound instance properties… Set Loop mode to oneshot (as in do not loop).
5. Play the event, make sure it plays the start, loop and then stop sounds in correct order as you like. If you like, drag the sound defs a bit over each other and they will automatically cross-fade.
6. To test envelopes, right-click the section to the left (where it says layer00) and Add effect. Choose Volume.
7. Right-click on the red envelope line to add points that you can use to fade the sound. Play the sound and notice that it fades the sound after the envelope, just like in a sound editor.
8. Right-click the “timeline” (parameter value line) and add a sustain point somewhere in the looping sound. Play the sound and notice that it stays and loops the looping sound at the sustain point.
9. Press the key off button and notice how the event move along and plays the stop sound and then ends.
10. To make it work in game, add a User Property (right-click anywhere in the property section) called KeyOffParamName, type text and add a value with the name of the parameter containing the sustain point.

From this simple example you can see (hear) the possibilities that are available to you as the sound designer to create interesting and easy to use sound events. This event would allow the person scripting to simply play the sound as the player starts the machine, it would automatically begin to loop for as long as the machine is running and then as the player stops the machine the script would stop the sound (which triggers key off) and the sound stops with playing the stop sound as you designed it.

Envelopes, volume and 3D parameters

There are three special parameters for 3D that you can use for events, right-click the parameter bar and you'll see Add 3D … parameter. These allow you to do typical things, such as fading a sound depending on the distance from the source. This you most of the time don't want to do, you rather simply use the 3D Min Distance and 3D Max Distance values for the event as a whole. But say you want a sound that fades out, but as it reaches a certain distance it stays at a certain volume, for this you have to do your own envelopes.

If you make your own envelopes you have to change 3D Rolloff to Custom or else the envelopes for Volume and/or Occlusion will not be used. You can also leave the 3D Min/Max distance values as they are, they are not used.

Don't forget to change the parameter property to have the correct value for the distance you want, the default is 0 to 1, which ingame would mean that the sound only reaches 1 meter. Right-click on the distance parameter, choose Parameter Properties and set the Minimum/Maximum value to what you want them to be.

Sustain Points

A sustain point can be added to a parameter, this makes the game play a sound and when it reaches the sustain point it stays and loops there. Then when you stop the sound it will continue playing the event from the sustain point to the end. See the above list for making a multi-track event for an example of the usage.

Important. To use sustain points you must add a User Property to the sound event. You do this by right-clicking in the property sections, Add/Edit user properties and then create a property with these specifications.

Name:	KeyOffParamName	The name of the parameter that the engine will look for when checking if a sustain points exists.
Description:	blank	Only a description, no need to fill in.
Type:	text	The name of the parameter that has the sustain point, typically param00.

If using script to play and stop the sound event, you must set the fade on Sound_Stop() to 0 to have the event continue playing and doing the stop sequence. If you have a fade time, the sound will stop and fade with the loop at the sustain point location instead.

Groups

Groups are almost only for organizing your project. However it is useful to think it through how to group and sub-group your events, as you can use a script function to preload group(s). Preloading is useful if you for example notice that you want to play many sounds at the same time and the game stutters from this. By preloading the sounds the engine have the sounds ready in memory for usage.

Categories

All sounds must be in one of two categories. These categories you have to create in each new project you make. It is only a matter of making a new category and naming it properly, there are no specific settings needed.

- “world” - All sounds that should be affected by global effects, such as a reverb. For example if the player goes below surface then all the sounds around him should change.
- “gui” - All sounds that should not be affected by global effects. Simply no matter where the player is or what he does, these sounds are always the same. For example button clicking sounds for the menu.

FMOD files & how engine access them

When you have a project of sounds that you want to use in the game you use the Build… option in the project menu. You can uncheck everything except that banks that you want to build. After building the project you take the .fsb (bank) files and the .fev (project) file and put them in the games sound/ folder at a suitable location.

Most of the time sounds will be picked from menus, using the tools for the engine (that is the plan, at the time of this writing this functionality does not exist.). If you manually have to specify a sound to be played, in for example a config file, a script file or in a input field allowing only to type in text this is how you do it.

Specify the full path to the sound, where it begins with the name of the project (the .fev file) and ends with the name of the event. For example for a physics impact that would be “physics/metal/compact/impact”. Footsteps defined in the material.cfg file has an exception from this!

See the Footstep sounds section.

Important notes for Events

- Make sure that all events that should be looped are set to “Oneshot” **false**. If not, there might be all sort of strangeness in the sound playback.
- Use unique names for all projects, as the .fev file it creates with the same name, must be unique for it to work correctly.
- For looping (non “Oneshot”) sounds that are to be in the environment, like fire from a torch, always use “just fail if quietest” max playback behavior. This will make sure that if there are many sounds of the same type around the player, it will be the loudest once that are heard. Other options are possible, but using any of the “steal” ones are almost always a bad choice.
- Must use gui or world category, see categories section.

Performance concerns for Events

- Use simple events as much as possible, they are faster than multi-track events.
- Keep Max Playbacks to as few as possible, often the default 1 should suffice.
- Do not add lots of effects to all events, it takes CPU. In particular never use reverb or similar, those should only be global and amount used by setting the dry/wet mix values (or use envelopes for dry/wet).
- Preload projects/groups if needed to remove ingame stuttering when trying to play multiple sounds simultaneous.

Physics sound

The physics system lets the sound system (FMOD) handle any mixing/effects for the sounds played. This is done by sending one or more parameters to the Multi-track Event for a sound. These are as follows:

Scrape
The event must have ONE param, name does not matter. The physics sends the linear speed to this param. Normal values 0 - 10. Is looping.

Roll
The event must have ONE param, name does not matter. The physics sends the angular speed to this param. Normal values 0 - 10. Is looping.

Impact
The event must have two params named “mass” and “speed”. “Speed” is sent the speed of impact (normal values 0 - 20). “Mass” is sent mass of the colliding object (normal values 0 - 200, but rarely over ~30) Note that mass is 0 for any static object no matter their size.

Joint
Move and Impact must have ONE param, where the speed is sent. Normal values are 0 - 5.

Materials.cfg

This file is located in the root (top) folder of the game installation, next to the game executable. The file can be edited in a text editor and it lists all the different types of material available to use by textures and objects in the game (ie makes something that looks like gravel behave and sound like gravel).

Parameters that are for a sound that you can define are always ending with SoundName = “”. Currently there are three sounds: Roll, Scrape and Impact. As you can see they are specified by giving the full path to a sound event, beginning with the project name and ending with the event name.

The parameter called StepType has the same value as the name of the event to play. For this to work you have to make sure that the footstep events are put in the correct project, in the correct groups and with the correct names. See Footstep sounds section.

Specific settings for physics

There are some specific parameter values to keep in mind for physics sounds.

General

- Max playbacks should be 2, possibly higher if you experience issues with sounds cutting out. Keep as low as possible.
- Mode is 3D.
- Min, Max Distance values should not be very far. Min: 0.5, Max: at most 30.
- Priority should be low for physics, it is not important to have sounds from physics if channels run out, things like voices are much more important.

Scrape

- Fade in/Fade out Time must be 100-300ms to avoid stuttering sounds.

Roll

- Fade in/Fade out Time must be 300-500ms to avoid stuttering sounds.

Footstep sounds

Footsteps are defined in the material.cfg file, with the StepType parameter. The footsteps and script for it has been designed to work in the following manner:

- Events are in a “player” project.
- At the top there is an event group called “footsteps”, in it are event groups called “barefoot”, “sneaker” and “default”.
- Events for each type of material, with the same name as the StepType parameter value are located as needed in each type of foot type (barefoot, sneaker, default).
- A parameter called movement in each event, with a value from 0 to 3.
- Sound defs for sneak, walk and run.
- The three step types has been laid out evenly, sounds should be reaching like this: sneak 0-1,

walk 1-2 and run 2-3.

The engine checks what material the player is in contact with and what type of footwear the player has. This is then put together to a string and appends the value of the StepType parameter from the materials.cfg file. It sends the string, which is the location for a footstep event (ie player/footsteps/barefoot/stone), and also checks the current player movement (sneak, walk or run) value and sends that as the parameter value. This allows FMOD to play the different movement sounds for that particular material.

Note: This is game specific and can be easily changed by editing the script files for the player's current move state

SNT (old sound system)

The old system of .snt files (see HPL2) is still in there and can be used, **it is not further developed and support might be dropped at anytime**.

Supported audio file formats

You can use wav, mp3 or ogg audio files. OGG is what we used for Penumbra and Amnesia and should be considered the best choice due to its small size and is license free to use (unlike mp3).

SntEditor.exe

Use this tool to create and edit .snt files. After you have run the tool the first time and selected Options → Register Application, you will get an option in the right-click menu if you click on a sound file in Windows Explorer. You can select one or many sounds when using the “Create Sound Entity file”-option and the editor will create several .snt files for each sound. If sounds are named the same, but with different numbers appended to the name, the editor will create a single .snt file for the group of sounds and the sounds will be added to the pool of sounds it randomly plays. *While the editor has input for Start and Stop sound these are actually not supported or working in HPL3.*

Editor Settings

General		
Main Sound	List the sound(s) that is played each time the .snt file is called by the game.	
Properties		
Loop	If the .snt should loop.	
Use 3D	When active a sound has a position in the world and moves around as the player moves around. If not set the sound is played at a static position, for example user interface sounds.	

Stream	If stream, a sound will be streamed and played instantly. If not streaming, then it will first be loaded to memory and then played. Long sounds should use stream and short sounds should not. Do not use too many Streams, as they will crack and drop out if there are too many (at most 6 channels, that is 6 mono sounds).	
Blockable	If an object is set to be blockable, the volume a sound located behind it will be lower when Blockable is active.	
Volume	Volume of the sound, 0 to 1.	
Min Distance	While the player is within this distance in meters from the sound source the volume will remain a the defined Volume, 0 to inf.	
Max Distance	When the player is this far away in meters from the sound source the volume will be 0, 0 to inf.	
Block Volume	Multiplier for the Volume. Block Volume<	nowiki> *%% Volume = the sound volume when blocked by an object, if Blockable is active.
Priority	If the game runs out of sound channels, then lower priority sounds will be dropped in favor of higher priority sounds. A dialog should have high priority, but physics should have low priority.	