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 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 (see script functions for preloading a project or a specific group!). 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.

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.

A quick overview of events, for 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 (increase mass = lowered 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 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. 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 layer0) 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" 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.

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, 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.

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.

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.

Physics sound

TODO: - ADD STUFF ABOUT MATERIALS.CFG AND HOW IT LINKS TO GROUPS, NAMES.

The physics system now lets the sound system handle any mixing/effects for the sounds played. This is done by sending one or more parameters to the played 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 (based on axes chosen in material) 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.

Feet sounds

TODO: - ADD STUFF ABOUT MATERIALS.CFG - ADD ABOUT GROUP, NAMING AND HOW IT IS LINKED TO MATERIALS.CFG

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 * 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.