



FMOD MUSIC & SOUND EFFECT SYSTEM

Parisa Eslambolchi
Hamilton Institute



Contents

- Introduction
 - FMOD features
 - Comparison
- Software**
- The basics
 - Getting started
 - Samples, Streams and Songs
 - 3D sound
 - Music Synchronization
 - DSP Engine



About FMOD

FMOD has been designed by Firelight Technology at 2001 in Germany.

FMOD is the fastest, most powerful and easiest to use sound system on Windows, Linux, and Windows CE there is, and now Macintosh, GameCube, PS2 & XBox!



FMOD Features

Sound output API Support and 3D Sound

1. FMOD does not actually require DirectSound of any version. It can work on any machine that has a 16 bit sound card!
2. DirectSound, DirectSound3D voice management support.
3. Creative EAX 2.0 and EAX 3.0 reverb with occlusion and obstructions.
4. Multimedia WaveOut (for legacy and lower latency NT support)
5. Linux OSS (Open Sound System), ESD (Enlightenment Sound Daemon) and ALSA (Advanced Linux Sound Architecture) support.
6. ASIO Support for windows.
7. Macintosh SoundManager support.



FMOD Features

Compiler Support

1. Microsoft Visual Studio 5, 6 and .NET
2. Watcom
3. Borland
4. LCC-WIN32
5. Metrowerks/Codewarrior
6. GCC
7. MingW & CygWin
8. Delphi, Kylix
9. Visual Basic



FMOD Features

Software Engine

1. ***THE*** fastest mixing routines in the world.
2. Quality 32bit interpolating mixers.
3. MMX Support.
4. Volume Ramping / Click Removal when there are sudden pan or volume changes.
5. Fake surround sound per channel.
6. All mixers support stereo samples.
7. Sounds can be played backwards.
8. Software 3D Engine including true Doppler, logarithmic volume attenuation and 3D panning.
9. Spectrum analyzer support through 1 simple function call.



FMOD Features

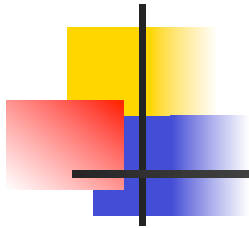
Music Playback

1. Native ADPCM/ACM support for any platform, without needing any codecs.
2. Shoutcast / IceCast / HTTP internet streaming support(juts pass a URL to FMOD instead of a filename).
3. ID3v1, ID3v2, ogg vorbis, ASF tag support.
4. WAV/MIDI/WMA/ASF/AIFF/ MP2/MP3/etc Support.
5. Multiple song system.You can play multiple sounds or streams at once, cross fade them in and out etc. Also you can set priority for your sounds, streams and songs.
6. Stream and sound music synchronization. A variety of commands are available to synchronize your graphics with the music.
7. CD Playback using FSOUND's FASTCD system (NO POLLING).
8. And there's more cool stuff inside...!



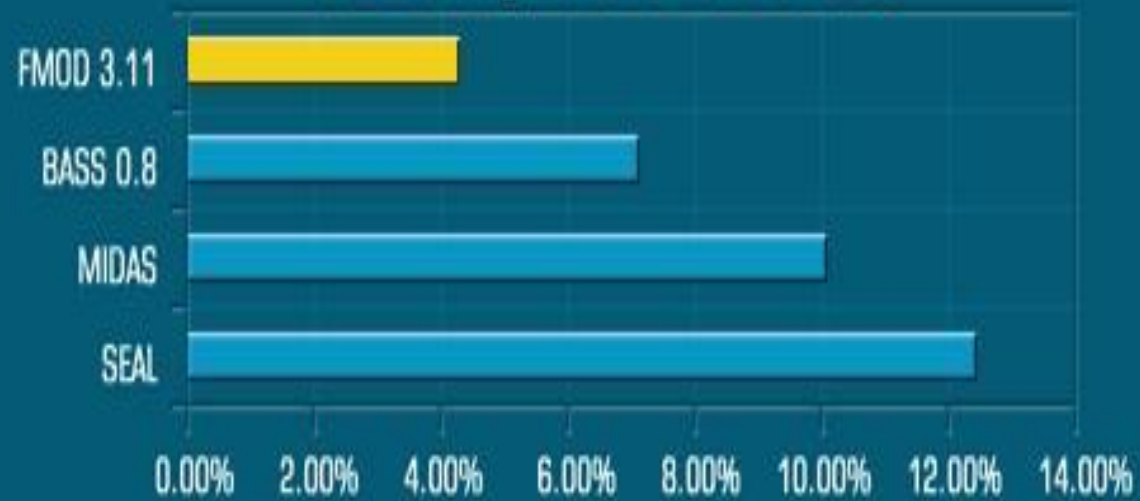
Comparison

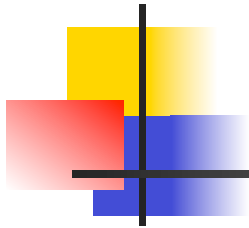
Here are some comparisons with FMOD against other player programs, for mixing 32, 8 or 16 bit sounds together at once. Note 'FMOD Normal' means non MMX accelerated.



8 bit source data with interpolation (quality mode)

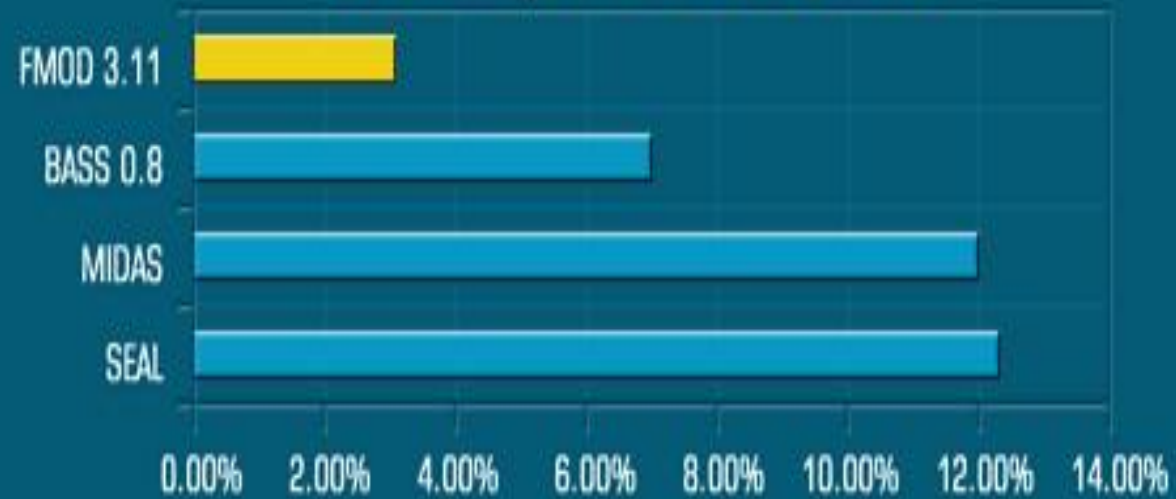
CPU usage (%) on a P3-500

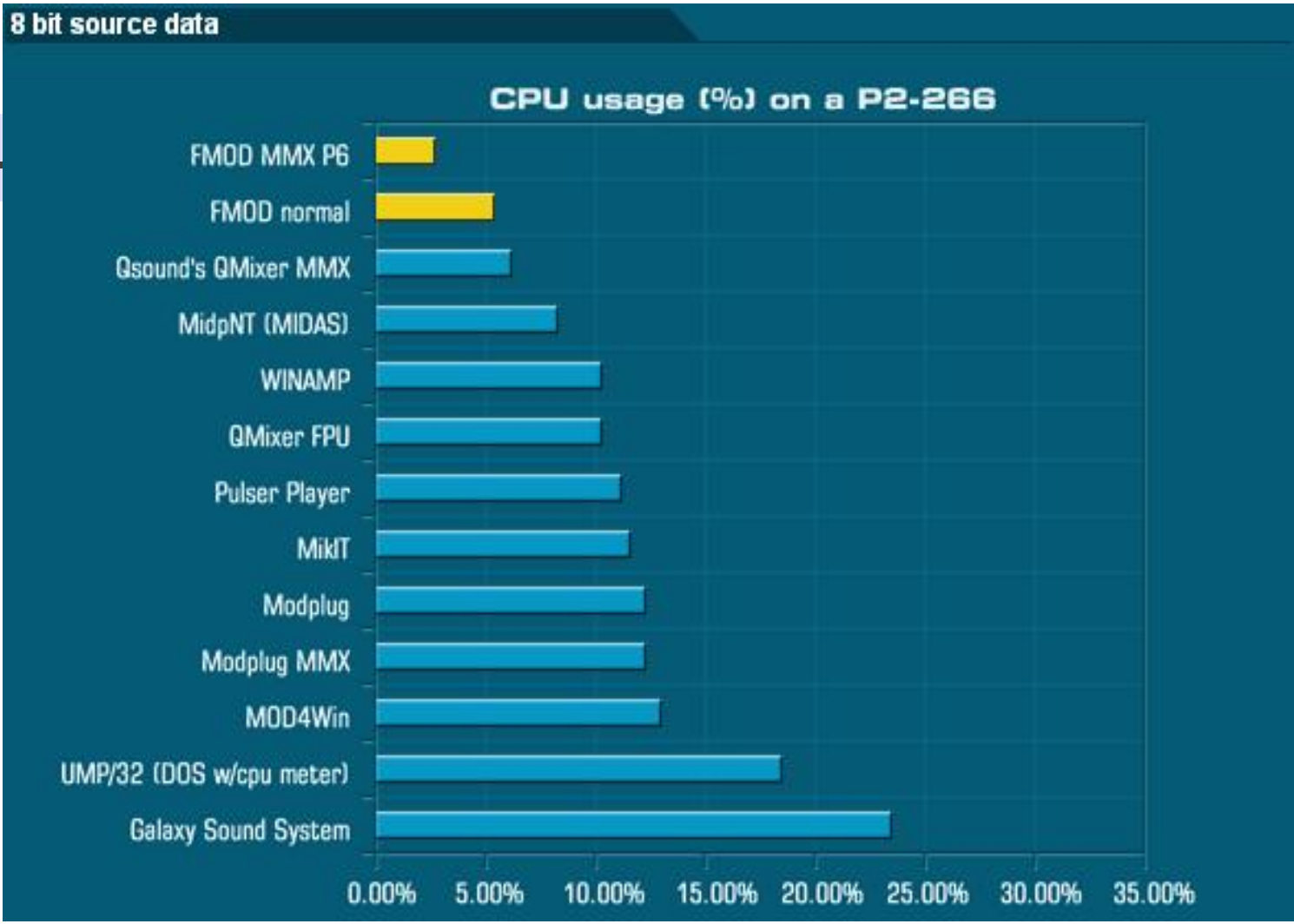




16 bit source data with interpolation (quality mode)

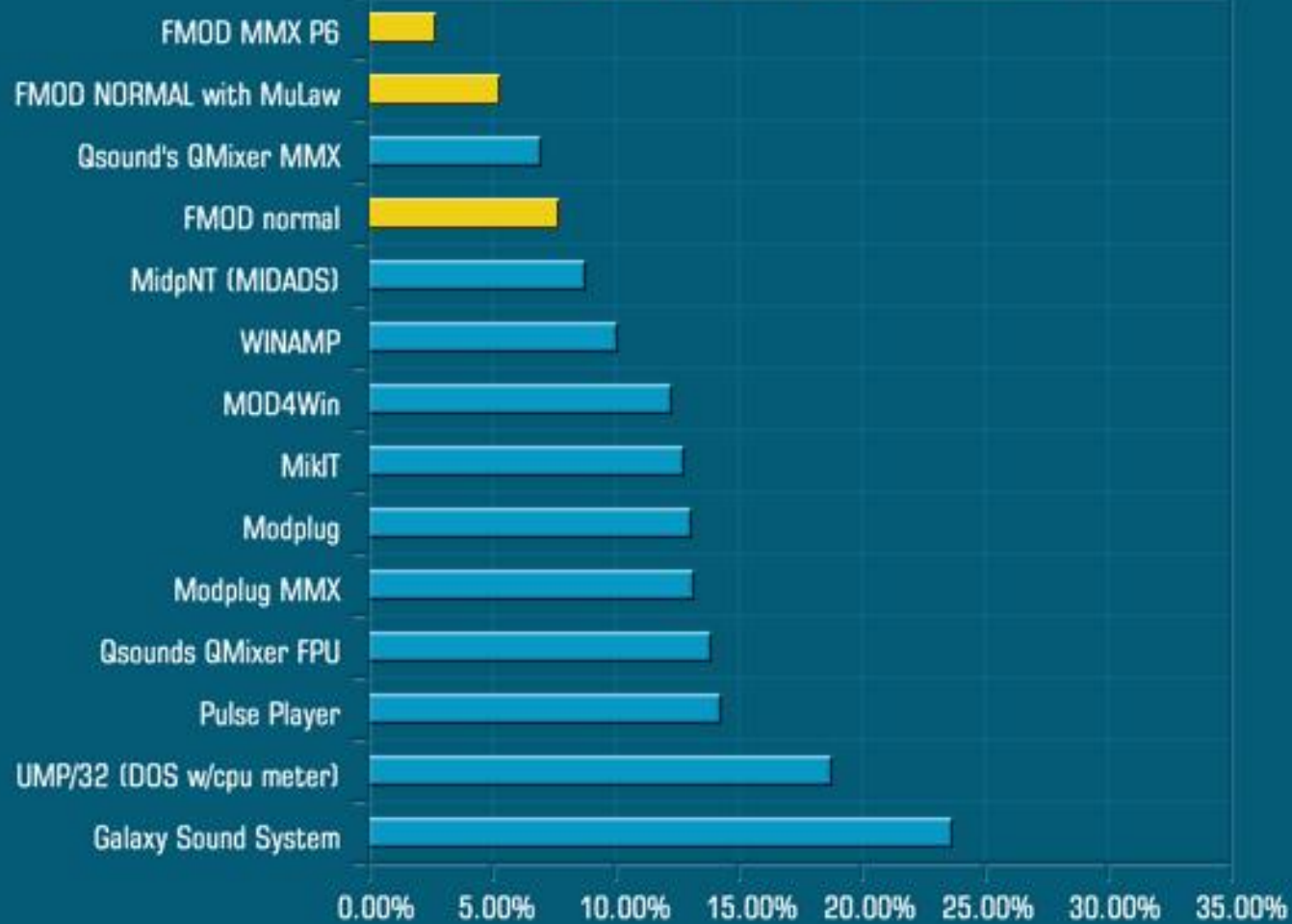
CPU usage (%) on a P3-500

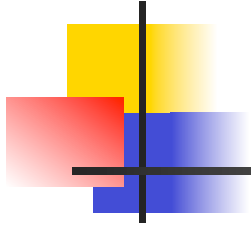




16 bit source data

CPU usage (%) on a P2-266





FMOD API



The basics

The package of **FMOD Programmers API** includes API, Documentation, Media, and Samples folders.

In **API** folder, you can find necessary header,lib and dll files due to your workspace (win32,wince,linux,...).

In **Documentation** folder you can find a help file.

Media holds some wav files and songs that the examples use.

Samples is for C programmers and contains a variety of samples to demonstrate 3D sound, DSPeffects, recording, streams and more.



Getting started

- The first thing you need to do is make sure FMOD.DLL is accessible from your program.
- To start coding, you need to initialize FMOD first! You do this *once* at the start of your program.

Do this with **FSOUND_Init**(int *mixrate*, int *maxsoftwarechannels*, unsigned int *flags*).

mixrate :Output rate in hz between 4000 and 65535. Any thing outside this will cause the function to fail and return FALSE.

maxsoftwarechannels : Maximum number of SOFTWARE channels available.

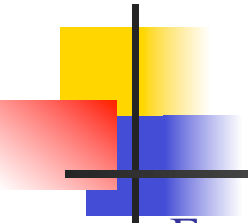
The number of HARDWARE channels is autodetected.

Having a large number of **maxchannels** does not adversely affect CPU usage, but it means it has the POTENTIAL to mix a large number of channels, which can have an adverse effect on CPU usage.1024 is the highest number that can be set. Anything higher will return an error.



Samples, Streams, and Songs

- If input is a sequenced music file such as **.MOD**, **.MID**, ... then use the **FMUSIC API**. You will want to use [FMUSIC_LoadSong](#) to load it into memory. When this gives you a valid return handle, use that handle with [FMUSIC_PlaySong](#) to play it!
- If it is a PCM based or compressed file such as **.WAV**, **.MP2**, **.MP3**, then you have a choice to make:
 - For small sounds such as sound effects that you want to trigger multiple times at once (for example a gunshot), then you will want to use [FSOUND_Sample_Load](#). When it gives you a valid return handle, use that handle with [FSOUND_PlaySound](#) or [FSOUND_PlaySoundEx](#). This is the fastest way to replay sounds, as they are decompressed into memory (at load time) first before being played.



-For big sounds, and does not need to be played multiple times at, then the other choice is to use FSOUND_Stream_Open. This opens a file, and prepares it for playing. When it gives you a valid return handle, use that handle with FSOUND_Stream_Play or FSOUND_Stream_PlayEx. This streams the file from the disk in real-time and decompresses the file on the fly. This option uses less memory than samples when the file is past a certain size. Note that this option uses a lot more CPU time than a sample as it has to access the disk, and if necessary, decompress the data on the fly. FMOD decompression routines are usually fairly efficient but it can be significant on slow machines (especially CE devices), or if multiple streams are played together at once. You can of course measure this with FSOUND_GetCPUUsage or just see how it affects your frame rate.



3D Sound

- 3D Sound in FMOD is accomplished through a various different API's which are handled transparently to you through one unified interface. These are DirectSound 3D with EAX 2.0 & 3.0 support, or FMOD callback software 3D stereo engine.

Hardware Support

- Allocating samples with **FSOUND_HW3D** is the key to getting your sounds to use 3D hardware acceleration. You do this when loading wav, raw, mp3, etc samples, through the following function.
FSOUND_Sample_Load(int index, char *filename, unsigned int mode, int memlength);
- If you have the correct hardware, the sound will be loaded into hardware mode. If not, the sample will be loaded in software mode, transparently to the programmer, but it will not be accelerated or take advantage of advanced 3d algorithms and EAX.



3D Sound

Velocity parameter

- Velocity is only required if you want Doppler effects. Otherwise you can pass NULL to both FSOUND_3D_Listener_SetAttributes and FSOUND_3D_SetAttributes for the velocity parameter, and no Doppler effect will be heard.
- It is important that the velocity passed to FMOD is **METERS PER SECOND** and *NOT* meters per FRAME.



3D Sound

Orientation and coordinate systems

- Getting the correct orientation set up is essential if you want the source to move around you in 3d space. FMOD Uses a **left handed coordinate system**, (x = right, y = up, z = forwards), which is the same as DirectSound3D.
- If you use a different coordinate system, then you will need to flip certain axis or even swap them around inside the call to [FSOUND_3D_Listener_SetAttributes](#).
- Take the right handed coordinate system, where Z points backwards, or comes out of the screen at you. To convert this to FMOD coordinate system simply negate the Z coordinate of the listener up and forward vector.



3D Sound

Channel resource management and low end cards

Some soundcards only support 4 or 8 3D hardware channels, whereas other soundcards support 32 and 96 hardware 3D channels.

What do you do in your game if you need 16 hardware channels but you just have 4 hardware channels?

The solution is to use [FSOUND_SetMinHardwareChannels](#). This function is called once before calling [FSOUND_Init](#).

What this allows you to do is **assume** a number of channels will either ALL be played in hardware, or NONE in hardware (therefore in software).



Music Synchronization

Synchronizing graphics with sound using FMOD functions usually involves either polling against a music value to trigger your effect, or getting a callback from FMOD.

Streams

The best way to synchronize a stream, such as a wav or mp3 is by using [FSOUND_Stream_SetSynchCallback](#).

The next best way to synchronize a stream is by using custom synch points. You can add your markers manually with [FSOUND_Stream_AddSynchPoint](#). Add your markers manually with this and set your synch point callback with [FSOUND_Stream_SetSynchCallback](#).

FMOD is a real-time system and the amount of time spent in a callback has to be low, or you could cause buffer under run or stuttering.



DSP Engine

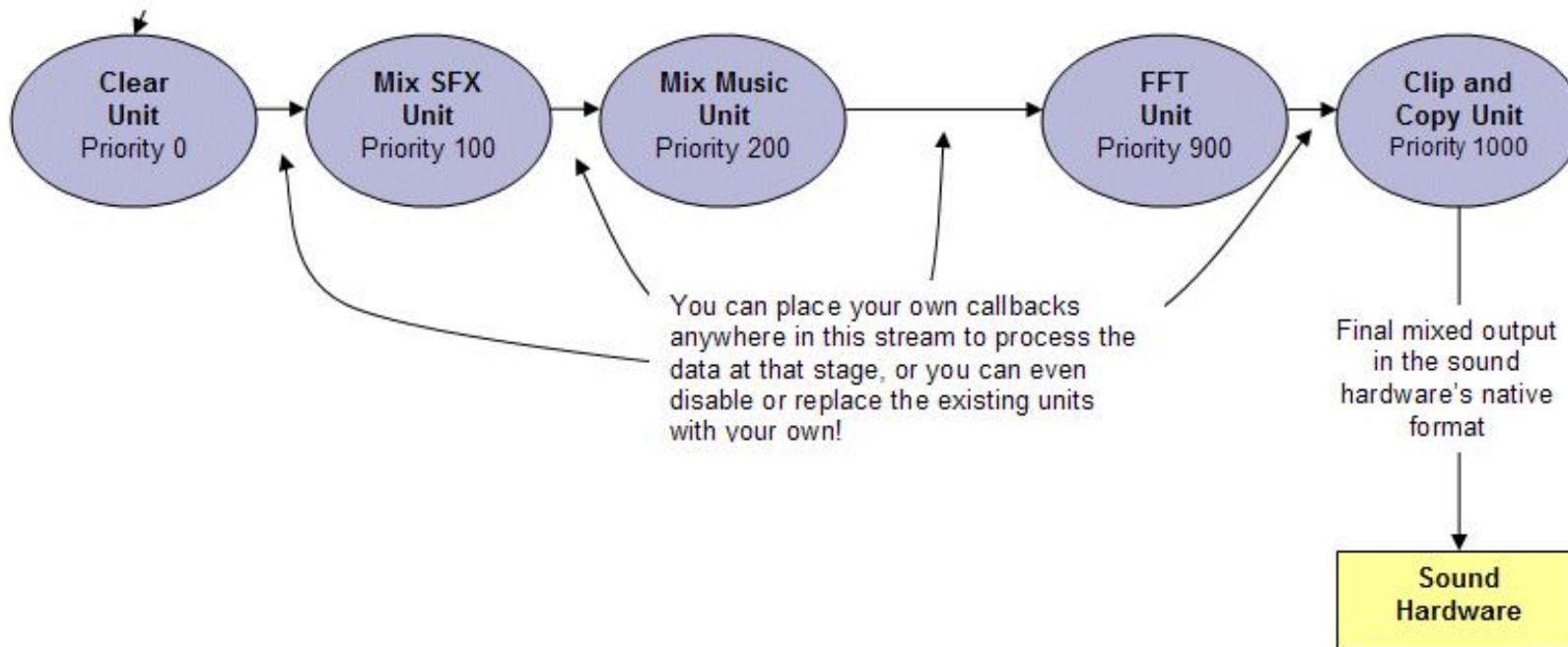
The DSP system drives FMOD's software engine. The DSP engine allows you to read - and process the mixed sound data as it travels to the soundcard in its various phases.

It allows you to filter the data if you like. You could add your own reverb, distortion, analyzing or even just plotting sound data as an oscilloscope for effect!

Roughly every 25 milliseconds (platform dependant), FMOD mixes a batch of data to be send to the sound device. To do this it executes a chain of algorithms to produce the sound. This chain is executed every time and performs the various jobs such as mixing, clipping and calling user callbacks to process the sound data. At the low level, it is simply a linked list of callbacks. Each time one of these callbacks is called it is passed in the mixer buffer. The function does this is [FSOUND_DSPCALLBACK](#).

MixBuffer – 16bit stereo or 32bit float depending on FSOUND_GetMixer()

This contains data from the last mix, but is about to be cleared.





More Information

FMOD WEB PAGE

www.fmod.org