F12001 Vehicle Sound*

By John (Busy Bee) Bayley

Overview

This describes the EngineSoundData (hereafter referred to as ESD) section of the SFX file. Each "block" of "code" is defined within braces {}. Therefore, the ESD is itself a block of code. At the time of this document, the game's sound engine supports up to 5 sound blocks:

- EngineSound=0 (Engine idle sound)
- EngineSound=1 (Engine low rpm sound)
- EngineSound=2 (Engine medium rpm sound)
- EngineSound=3 (Engine high rpm sound)
- EngineSound=4 (Engine maximum rpm sound)

These are contained within the ESD block. Two more variables are defined within the ESD block are MinPitch and MaxPitch. These variables define the total range (in 1/96th of an octave) of the sound blocks (as a whole) defined within the ESD. Each EngineSound block contains 3 variables in a range of 0.000000 to 1.000000:

- Min (Minimum value in which this sound will begin and pitch bend down)
- Max (Maximum value in which this sound will end and pitch bend up)
- Natural (Value between Min/Max at which the sound is considered to be its Natural pitch)

The hard thing to understand is that everything is relative to an "imaginary line" defined as 0.000000 to 1.000000. I refer to any value within this imaginary line (i.e. EngineSound data block variable values) as a Sound Frequency Definition (SFD) and the entire line as the SFD range. Each sound block uses this imaginary line (or SFD range) to determine where each sound is played and where each sound ends as well as the natural pitch for the sound. MinPitch and MaxPitch set the "scale" for the line and not only determines the entire pitch range for the line, but also determines how much pitch "bend" will occur as a sound is faded in and out (depending on the scale).

So, for example, if you have 1 octave (range=96 - explained later) defined by MinPitch and MaxPitch, then each EngineSound data block will be used to spread the sound(s) out over 1 octave from the Engine Idle sound to the Engine Max Rpm sound. This is not a true octave since the 8th note is never played. The precision loss is 1 semitone per octave. So in order to hear a true octave you would add 8 (8=1semitone) to 96, 16 to 192 and so on. 0.000000 would represent the first frequency (or note) in the octave and 1.00000 the highest frequency. So you would likely have the idle sound closer to the beginning of the imaginary line and the max rpm closer to the end (and any other sounds in between). Note that values can overlap which allow for sounds to be blended (one fading out as the other is fading in) giving you a unique blend of two sounds as you cross over from one to the next (keep in mind the crossover between the pitches of two sounds which can cause a detuning effect). I'll go into more depth on each entity then show how it's pieced together at the end.

Note: F12002 does not have a sound resource manager and hence does not support multiple ESD sections. What this means is that only one EngineSoundData block can be used for all vehicles currently running in the game. Therefore even though you can have different engine sounds for each vehicle they must all use the same ESD section. The problem this causes is that all vehicles will have their sounds played along the same point of the Sound Frequency Definition Range with the same amount of pitch bend step for each sound defined. So you should keep this in mind when creating your vehicle sounds since you will have to keep the pitch within the same defined range. For example, let's say you have a Vette and a Porsche... both their low rpm sounds will be played at the same point and time along the SFD line given their current RPM. Needless to say if you have set your ESD to sound great for your sampled Vette sounds, your Porsche sounds may sound too low or high in pitch or not blend well from one sound to the next so you will need to resample your Porsche sounds to a pitch where it is compatible with your existing ESD settings. Also note that the game only looks for the f1sounds.sfx file so your ESD must exist in that file.

<u>The Sound Frequency Definition Range (0.000000 - 1.000000):</u>

Imagine a line whose length is fixed at 1.0cm. Along that line you have to define the entire pitch range of your vehicle (whether using 1 sound or multiple ordinal sounds). This pitch range is defined using the MinPitch and MaxPitch values (or I should say the difference between the two as I will explain later). Since MinPitch and MaxPitch define the pitch range, the larger the range the more points along the line you will have resulting in each sound having its pitch "bent" more. The sounds will still be played at the same point along the line (although the sounds will be played in quicker succession), but the amount of pitch bend will increase from the point the sound is faded in to its Natural pitch, and to the point it's faded out from its Natural pitch. This is because the number of frequency steps increase as the total pitch range increases.

A good example of this effect is that if you were only using 1 sound file and set the Natural to 0.0 and the Min to 0.0 and the Max to 1.0, by the time you got up to max rpm with a very large pitch range, the sound would have its pitch bent so much that it may no longer sound like a car but more like a squealing pig. On the other side, having the pitch range too small would mean that the original sound would be bent so little that as you increase in rpm the sound would hardly change and not sound like what you'd expect max rpm to sound like. Keeping the pitch range under the 2 octave mark (192.0) for most vehicles is a fairly easy range to work with. It gives you enough range to express much of the timbre of the vehicle's different sounds as it goes through each sound step without creating too much or too little pitch bend.

Don't assume that this line has much to do with your accelerator. Just because you have the throttle all the way to the floor doesn't mean that the max rpm sound is playing at its max pitch. Take for example the case of being in 6th gear doing 60Kph...if you push the accelerator all the way down it will take some time before the vehicle reaches max rpm and thus max rpm sound pitch.

So what is the triggering factor? The vehicles rpm mainly. Just as MinPitch and MaxPitch are applied to the SFD range to determine the total pitch range, the vehicles rpm is applied in much the same manner. If a vehicle has a max rpm of 10,000rpm, that rpm range is applied across the same SFD line. In that case, when the vehicle reaches 5,000rpm, the sound defined within the 0.500000 range of the line will be played. If a vehicle has a max rpm of 7,000rpm, then when the vehicle reaches 3,500rpm, the sound defined within the 0.500000 range of the line is played, and so on. At which pitch those sounds are played depends on the pitch range of MinPitch and MaxPitch, and the EngineSound block data that contains the range for a specific sound to be played. So if you have different vehicles in a set that will be racing at the same time, and they have different rpm ranges, you will have to take into account the pitch of the samples they use so that they will all work using the same EngineSoundData definition. So basically the SFD line is used as a lookup (index) value for the current rpm so that the game knows exactly what sound to play, and at what pitch to play it for the vehicles current rpm. The basic calculation for the game would be:

(1.0 / VehicleMaxRpm) * VehicleCurrentRpm = SFDValue

So given a vehicle with a MaxRpm of 8,000 currently running at 4,000rpm would have a SFDValue of 0.500000. The game would then lookup the EngineSound block that has a range which contains 0.5 (there could be more than one sound if they overlap), access that sound(s), alter the pitch depending on the Min/Max values, and mix the sound with any other sounds currently playing. Of course, if the throttle is OFF, the backfire sound might play instead so there are a few other factors involved as well. This is just a general overview and assumption.

The EngineSound block:

This block contains 3 variables; Min, Max, and Natural. The key value is Natural since Min & Max are purely relative to whatever is defined as the Natural value. Here is an example from the actual f1sounds.sfx file for the idle and low rpm sound:

```
EngineSound=0
{
Min=0.000000
Max=0.089835
Natural=0.000000
}
EngineSound=1
{
Min=0.433367
Max=0.081605
Natural=0.439400
}
```

Natural:

The Natural value defines when (or where along the SFD range) the sound is played at is Natural frequency (meaning unchanged from what you'd hear if you played the sound in Windows Media Player for example). Given the example above where Natural=0.439400, this means that as the vehicle's rpm range hit close to the half way mark, EngineSound=1 would be played at its natural frequency. How soon the sound begins to play and how soon it ends is defined by Min/Max respectively.

Min & Max:

These are absolutely relative to the value defined in Natural. They determine where (along the SFD range) the sound starts to play and when it ends as well as how much pitch bend is applied to the sound (by using the pitch range as a scale factor). To determine the actual value along the SFD where the sound will start to play you take the Natural value and subtract the Min value from it. To determine when the sound will stop playing, add the Max value to Natural. Let's take the example above and determine where along the line the low rpm sound will start and end playing.

To determine when the sound will start playing, use the following equation:

Natural(0.439400) - Min(0.433367) = 0.006033

So here the low rpm sound will start playing when we reach the above value but also important to note is that the pitch of the sound will be bent in pitch down from Natural to this value. The amount of bend depends on the pitch range defined by MinPitch and MaxPitch. Again, the higher the pitch range the more frequency steps there are between 0.006033 to 0.439400.

To determine when the sound will stop playing use the following equation:

Natural(0.439400) + Max(0.081605) = 0.521005

Here you can see the sound will be faded out by the time the vehicle reaches just over half its rpm and pitch range.

Notice where EngineSound=0 (the idle sound) ends playing at 0.089835 and where the EngineSound=1 starts at 0.006033. You can see that the idle sound is bent up to its Max value and the low rpm sound begins bent down to its Min value and the two overlap by 0.083802. So in this case if you were accelerating, as the idle sound is increased in pitch (bent up to Max) and the low rpm sound is increased in pitch (bent up from Min) the two are mixed together to create a nice crossover from one sound to the next.

TIP: If you've ever used a music keyboard with a pitch bender you can imagine the key you press is the Natural and moving the bender all the way up will be the Max and moving the bender all the way down will be the Min. Another important factor is, as with a keyboard where you can set the range of the pitch bender (even though the bender itself still moves the same distance), that is basically how MinPitch and MaxPitch are applied to the entire SFD range (that's the entire range, not the individual range for each sound).

MinPitch & MaxPitch:

Let's look at the original values that come with the game:

MinPitch=-134.0

MaxPitch=58.0

What is important here is not the actual numbers themselves, but the difference between the two. In this case, 192 is the difference between them and is what defines the pitch range for the entire SFD line. The above is exactly identical to this:

MinPitch= 0.0

MaxPitch=192.0

So what does 192 mean? Well, since each value is expressed as 1/96th of an octave, 192 = 2 octaves minus 2 semitones (as mentioned, the precision lose is 1 semitone per octave). 96.0 would be 1 octave minus 1 semitone, 8.0 would be 1 semitone and so on. The easiest way to determine the number of musical steps is to divide the pitch range by 8.0. This will give you the number of semitones in the pitch range.

One thing to note is that with EngineSound=1 (low rpm sound), the sound is actually bent below Min to give the effect of the small drop in rpm as the vehicle goes from idle to rev (much like if you let the clutch out and drive the rpm drops slightly below idle rpm before increasing).

So given the above, we'd apply this pitch range to the SFD range. Remember the ruler? Now imagine we have the 1.0cm ruler and we now have 192.0 "ticks" or marks along that ruler. This is how the pitch range is applied. And along that ruler you would define points where each sound is played (but using the millimeter ticks on the ruler instead). Using the Natural as the point where the actual Natural sound is played, and the Min/Max on either side where the sound starts/ends being played. Remember the sound starts bent down to Min and bends up to Natural and continues to bend up past Natural till (Natural+Max) is reached. The amount of pitch bend depends on the pitch range which is 192.0 points in this case. If we used 1 octave (96.0), we'd only have 96 points along the ruler and therefore the amount of pitch bend for each sound from Min to Natural and Natural to Max would be less.

Here's the math... with the ruler (our SFD range) being 1.0cm, and the pitch range being 192.0 (24 semitones) this means that each mark on the ruler would be 0.0052083 apart. To determine this, use the following method:

1.0 / PitchRange = PitchStepSize

For 1 octave (96.0) the PitchStepSize would be 0.0104166. You can see how the step size doubled. This also means that the amount of pitch bend will be half of what it was when it was 192.0.

Let's take a look again at an original EngineSound block... the one for low rpm sound:

MinPitch=-134.0

MaxPitch=58.0

```
EngineSound=1
{
Min=0.433367
Max=0.081605
Natural=0.439400
}
```

Let's see what happens to this sound and how it is played. The pitch range here is 192.0. The Natural of the sound (the way the sound would be heard if played in Windows Media Player) will be heard at 0.439400. Regardless of the pitch range, the Natural of the sound is always played at this point (its pitch is NOT affected by the pitch range, only the amount of pitch bend to get to the Min/Max values is effected). The sound will start playing at 0.006033 (Natural-Min). The pitch at which it starts playing is defined by the pitch range, so in this case with Min set to 0.433367, and the pitch range set to 192.0, the amount of pitch bend will be:

Min / PitchStepSize = Pitchbend

Or

0.439400 / 0.0052083 = 84.4

Remember the PitchStepSize of 192.0 is 0.0052083. So the amount of pitch bend (rounded up) will be 84.4 (expressed in 96ths of an octave) or roughly 10 1/2 semitones down from its Natural sound. This works out to 10.55 semitones of negative pitch bend (84.4 / 8.0) since 8.0 equals 1 tone in the pitch range scale.

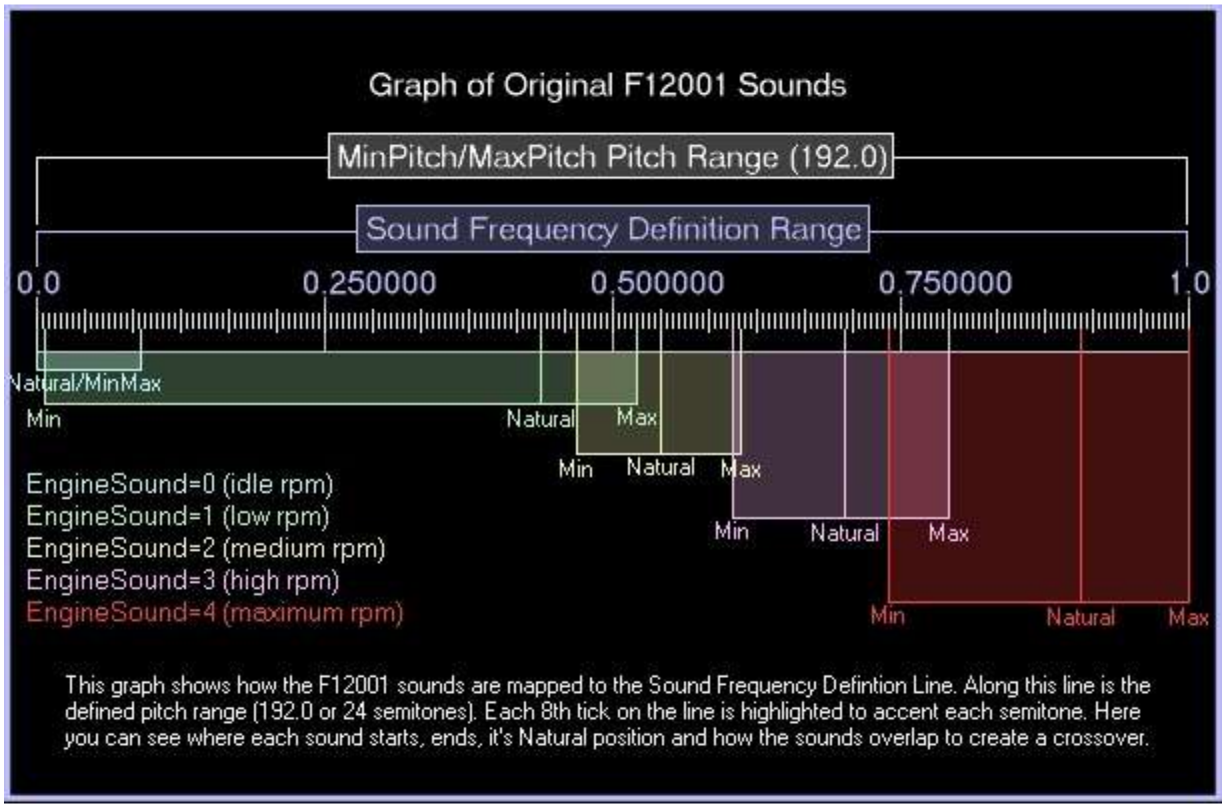
Now let's look at how much the sound will be bent up in pitch before fading out.

Max / PitchStepSize = PitchBend

Rounded up, the pitch bend would be 15.7/96th of an octave or roughly 2 semitones.

Now let's look what happens to all this when we change the pitch range with MinPitch & MaxPitch to 96.0 (1 octave). The PitchStepSize is now 0.0104166. Therefore the negative pitch bend (bend down in pitch) will be only 41.6/96th octave or roughly 5 semitones, and the positive pitch bend will only be roughly 1 semitone. Now you can see how the pitch range defined by MinPitch/MaxPitch effects how much the Natural sound is bent down/up when applied to the Min/Max values in the EngineSound block.

A picture may help clarify how the sounds are mapped.


Graph of Original F12001 Sounds
MinPitch/MaxPitch Pitch Range (192.0)
Sound Frequency Definition Range
0.0
0.250000
0.500000
0.750000
1.0
Natural/MinMax
Min
Natural
Max
Min
Natural
Max
Min
Natural
Max
Min
Natural
Max
EngineSound=0 (idle rpm)
EngineSound=1 (low rpm)
EngineSound=2 (medium rpm)
EngineSound=3 (high rpm)
EngineSound=4 (maximum rpm)
This graph shows how the F12001 sounds are mapped to the Sound Frequency Defintion Line. Along this line is the defined pitch range (192.0 or 24 semitones). Each 8th tick on the line is highlighted to accent each semitone. Here you can see where each sound starts, ends, it's Natural position and how the sounds overlap to create a crossover.

Using this information you can calculate the data values you will need in order to set the amount of negative/positive bend for each sound, and how it can also be used to overlap sounds to create crossovers from one sound to the next. Since you can calculate what the pitch of a sound will be (knowing its Natural pitch) when the sound begins and when it ends, you can cross fade two sounds together either in near perfect pitch or slightly detuned to create interesting effects.

UPDATE: Much of the above info is useful, but the usage within the game is somewhat obsolete. There can now be up to 20 engine samples: inside/outside (2) * rpm samples (5) * throttle on/off (2). 2*5*2=20. And the rpm sample ranges are defined by RPM, rather than the arbitrary 0.0-1.0 scale.

An interesting fact is that you can calculate the frequency that an engine sound should be knowing some basic information about the engine. Specifically:

Frequency = (RPM / 60) * (# of cylinders / 2)

The reason we divide by 60 is to convert from a per-minute value to a per-second value. The reason we divide by 2 is because most car engines are 4-strokes – there is an ignition every OTHER time the piston comes by. Just for reference, middle-A on a correctly-tuned keyboard is 440Hz. If you want a correctly-pitched engine sound for a V8, then you will need to tune it to middle-A when it is running at 6600 RPM.

The other fact that this equation brings up is that engine frequency is LINEAR with RPM. This is not obvious to everyone, but it is a natural fact.

You will notice in the vehicle SFX files that engine RPM samples can be tuned by hand (and their RPM blend range set) by sections looking like this:

```
      EngineRPMCoastInside=0
      {
        MinimumRPM=5.00         // above zero, but low enough to hear engine stall
        MaximumRPM=3100.00      // must overlap properly
        NaturalRPM=1838.5499    // engine RPM at which sample was recorded
      }
```

In this example, this is the first RPM sample (index 0) for an off-throttle engine sound (or coast). It's audible between 5 and 3100 RPM (it will automatically be blended with other engine sounds that overlap this range). The natural RPM should be the RPM at which the sound was recorded. Sometimes this is not known, so you can use the frequency example above to determine the engine's pitch.

Even if you knew the RPM at which every engine sample was recorded at, small errors would probably make the engine sounds out-of-tune with each other as they were blended through the RPM range. The solution is to use the Control-Y engine tuner tool. This will help you get each engine sample tuned to each other at the same RPM.

Regarding on/off throttle samples (also known as Power and Coast), you will need to use a line like the following example in your vehicle SFX file:

```
EngineLoadBlendInside=(0.1,0.7)   // power sound starts to blend in at 10% throttle,
coast sound blends out at 70%
```

This line blends the power and coast inside engine sounds based on throttle position. For this example, only the coast engine sound will be audible between throttle positions of 0-10%. Between 10% and 70%, the coast and power engine sounds will be blended. Between 70% and 100%, only the power engine sound will be heard.

With all of this RPM and throttle blending going on, there could potentially be 4 engine samples being played at one time for one car. I would suggest using the on/off-throttle distinction ONLY for the inside sounds. Otherwise, a group of 15 AI cars around you could potentially be creating 60 different engine sounds –

overwhelming for many computer systems (and near the peak of what our game allows), and this is before any other sounds, such as tire squealing, rumblestrip-riding, windnoise, shifting, or backfires can be played.

You may also be wondering how the following lines fit into all this:

```
playerEngineVolumeMinimum=0.3
playerEngineVolumeThrottleFraction=0.5
playerEngineVolumeRevFraction=0.2
```

This is just one extra volume multiplier applied to all current engine samples being played. Depending on throttle position and fraction of full revs, this example could be playing the engine between 30% and 100% of full volume.

One final volume control, of course, is the engine samples themselves! Keep in mind all the possibilities and constraints of our system as you design those excellent sounds.

UPDATE 2 (T):
First off, our sample SFX file had a mistake - that *FadeRadius stuff under ATTENUATION is old, and is not used (ATTENUATION itself is just a comment and is not read by the game). Instead, it should be replaced by:

```
EngineRange=145.0
EngineShape=0.960
EngineAmbient=1.00

ShiftRange=145.0
ShiftShape=0.910
ShiftAmbient=1.00

OtherRange=25.0
OtherShape=1.0
OtherAmbient=1.00
```

* The Engine parameters affect engine sounds and horns.

* The Shift parameters affect shifts, backfires, TC, engine starting, launch and speed limiter clicks.

* The Other parameters obviously affect other sounds.

* The Range parameter isn't really the complete range, it's just a multiplier where a bigger number will make the sounds audible and louder further away.

* The Shape parameter affects the drop-off curve. I believe low numbers make a somewhat linear (generally unrealistic) drop-off in volume, and high numbers are more curved as in real-life.

* Finally the Ambient parameter defines within what range the volume is at 100%. **Note:** Set this first, because it may or may not have odd affects on the range and shape.

Onto the next section:
The ENGINE VOLUME MIX numbers MUST add up to 1.0, for both the player and the non-player. The current engine volume is calculated something like this (ignoring attenuation):

volume = Minimum + (current_throttle * ThrottleFraction) + (rev_pct * RevFraction)

As an example, if you are coasting at 3000 RPMs and suddenly nail the throttle, the volume will increase due to the ThrottleFraction parameter. And then as the revs increase, the volume will increase further due to the RevFraction parameter.

Finally, you cannot change the amplitude of samples within the *.sfx file. It must be done within the WAV file itself.