Tutorial 09
FMOD Designer 101
The basics of using FMOD Designer

written by Stephan Schütze

Introduction

Fmod is a sound engine and authoring system used to create interactive audio environments for games on a variety of platforms. FMOD comprises of two main parts, the programmers API, which allows programmers to integrate the code base for controlling audio in a game program, and FMOD Designer, a tool that allows composers and sound designers to create audio environments from raw sound assets and implement them into a game system.

This tutorial will go through many of the basic concepts of FMOD designer and demonstrate how to create some simple audio events. FMOD Designer can be a powerful tool for creating interactive audio environments, but it can be daunting for anyone unused to how its workflow progresses. This tute aims to demystify the process and get users working in FMOD as quickly as possible. This tutorial can be followed through with any available sound files or you can use the provided ones by using this link.

FMOD 101 tutorial assets
FMOD Designer is available for free download from www.FMOD.org

Getting Started

The first thing to understand about FMOD Designer is that it does far more than just play simple sound files. While it is perfectly capable of taking a single wav file and making it available as an event to be accessed by the game code, this is a waste of its capabilities and I would argue that in the era of next gen consoles and powerful operating systems this is just lazy sound designing. As creative professionals I believe we have a responsibility to our audience to produce audio environments worthy of the incredible graphics and game-play in modern games.

The second thing to understand, and perhaps the most important, is that FMOD Designer is not a linear sound editor. In many ways users need to unlearn many of the methods of sound creation they may have learnt over the years as traditional thinking can often get in the way of creating effective audio environments.

Sound Definition Page

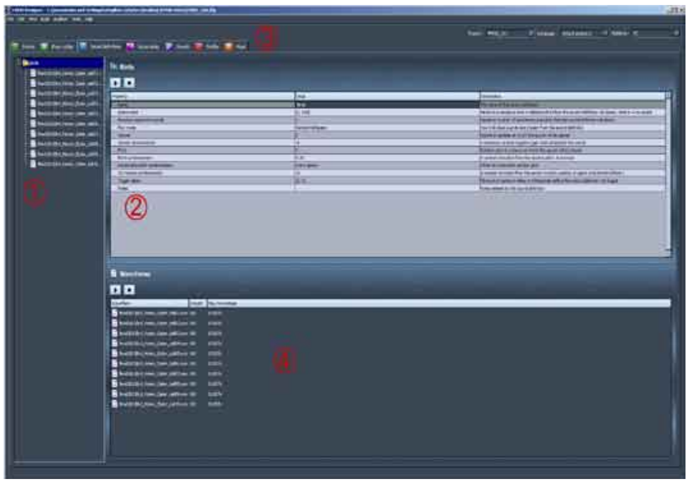


Figure 1. Sound definition page.

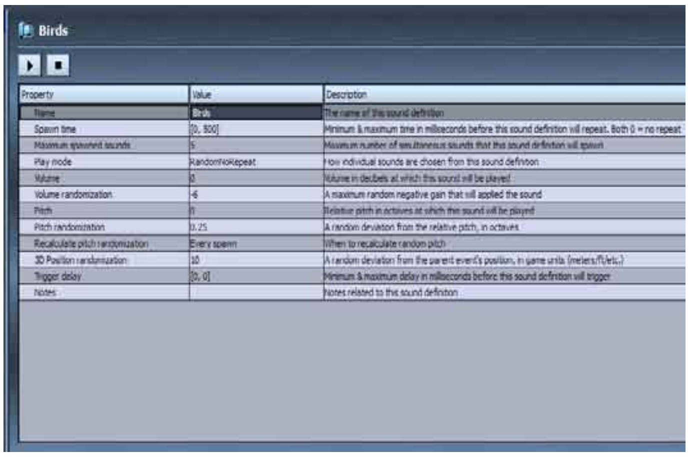
1. Sound definition list area. This is where Sound definitions and their associated sound files are listed.
2. Sound definition parameters. The individual parameters for each Sound definition are displayed and can be altered in this section. A Sound definition can be auditioned in this window with all the appropriate parameters applied.
3. Tabs to other areas of Fmod Designer
4. Waveforms window. This displays the contents of the Sound definition selected in the Sound definition list area. The sound files can be auditioned individually in this window. The parameters of the Sound definition are bypassed for the purposes of auditioning in this window.

To start with, create a new project in FMOD Designer. I like to assemble all the elements of a project into a new project folder before I start. It is always possible to add to the project folder as progress is made, but I find there are a variety of reasons to put each project and its assets in a dedicated folder. It is very important to make copies of all the sound files you plan to use and add them to your project folder. The primary reason for this is to avoid your original source files from being corrupted or altered when included in a project. In this way you are completely free to alter, edit or resample any of your material, safe in the knowledge that you have your original source as a backup if anything goes wrong. It also means that all the assets to create a project exist in one location which is useful for backup and security purposes as well as if someone else needs to work on the project. Under the **File** menu in Designer, create a **New Project** and save it into your project folder with an appropriate name. Next we will import some sound files into FMOD Designer. Add your sound files to your project folder if you have not already done so.

Creating Sound Definitions

A Sound definition is far more than a simple sound file, although it is possible for it to consist of just a single sound. Go to the Sound definitions tab in FMOD Designer, the area down the left hand side is where all the Sound definitions are listed, at this stage it should be empty. Right click in this area and select
Add empty sound definition… a new window should pop up, type in the name for your new sound definition. For this tutorial I am going to create a sound event of birds in a forest, so I am going to label this sound definition Birds. An empty Sound definition is effectively a folder into which we will place sound files, now that you have created the definition folder you can see a list of parameters for the Sound definition in the main part of the window. We will work through these parameters in a second; first we must add some sound files to our Sound definition.

Importing sounds into FMOD Designer is a simple process of dragging and dropping them. If you have created an empty Sound definition you can simply drag the sound files from your windows folder into the Sound definitions folder. If you drag a sound file directly into the Sound definitions list area with no folder FMOD will automatically create a new Sound definition folder containing your sound file. It will also automatically name the Sound definition to match the name of the sound file. If you drag and drop multiple sound files into the list area FMOD will create a new definition for each file. This can be handy for quickly creating a large number of Sound definitions that will only contain one sound file each. As we want to create a single Sound definition with multiple sound files in it we have created an empty definition first. The tutorial assets include sound files called Bird_brown_honey_eater_call01-09; drag all nine files into the Birds Sound definition now.



Property	Value	Description
Name	Birds	The name of this sound definition
Spawn time	[0, 500]	Minimum & maximum time in milliseconds before this sound definition will repeat. Both 0 = no repeat
Maximum spawned sounds	5	Maximum number of simultaneous sounds that this sound definition will spawn
Play mode	RandomNoRepeat	How individual sounds are chosen from this sound definition
Volume	0	Volume in decibels at which this sound will be played
Volume randomization	-6	A maximum random negative gain that will applied the sound
Pitch	0	Relative pitch in octaves at which this sound will be played
Pitch randomization	0.25	A random deviation from the relative pitch, in octaves
Recalculate pitch randomization	Every spawn	When to recalculate random pitch
3D Position randomization	10	A random deviation from the parent event's position, in game units (meters/ft/etc.)
Trigger delay	[0, 0]	Minimum & maximum delay in milliseconds before this sound definition will trigger
Notes		Notes related to this sound definition

Figure 2. Sound definition parameters

Sound definition parameters

The Sound definition parameters allow for a series of variations to be applied to the sound files each time they are triggered in an event. These variations can be subtle or extreme depending on your desired end result. The parameters work in the following manner.

Name
This is simply a label you have applied to this definition. The name can be anything you want, but it is useful to create names that help you in creating your events. The more information you put in a Sound definition name the easier it is to work with them later if you are creating complex events.

Spawn time
This function allows you to set a Sound definition to repeat itself. There are two values that can be altered with the slider controls. These values represent the minimum and maximum time in milliseconds in which the sound will repeat. Effectively you are defining a time window in which the definition can repeat. So a value of 0 for minimum and 1000 for maximum would mean the Sound definition would wait up to one second before repeating. (1000 milliseconds = 1 second) If you wanted it to wait exactly one second then you would set both the minimum and maximum values to 1000 milliseconds.
Being able to define values that randomise how a sound is played is very important in creating realistic sound environments. Almost every sound in nature and a great many man made sounds are very random in how they are generated and avoiding repetitive sounds is an important part of good audio design.

Maximum spawned sounds
This value works as a modifier to the Spawn time function. It allows you to control how many sounds will play simultaneously. Its default of 1 will mean that only one sound will play at a time, so even if you set the Spawn time values very short a sound will only replay once the currently playing instance has finished. If you set the number to a higher value it will allow multiple instances of the sound to play simultaneously. Tweaking this value while listening to the sound playing will allow you to achieve a balance between not enough sounds playing at once, and too many playing which may create an overly cluttered effect.

Play mode
I consider this to be the most important parameter to set correctly if you are using a sound definition that contains more than one sound file. This function defines how the Sound definition chooses to play its sound files. They can be played sequentially or randomised in a variety of ways. I almost always create Sound definitions with multiple sound files.

Even the most regular sounds in the real world have a level of variation when analysed.
Try bouncing a ball and really listen to the sound it creates. Each bounce is subtly different in pitch, tone and amplitude. When creating sound for games it is desirable to recreate this difference in how sound is generated. I will usually use the Random no repeat value, this truly randomises the available sound with the single exception that it will never play the same sound twice in a row. I find this produces the result I am happiest with. The values work in the following manner.

Sequential:
This plays the sound files in the order in which they appear in the Sound definition as a list. Once it has played through all the files it returns to the top of the list.

SequentialEventRestart:
This plays the files as above but each time the event is triggered it returns to the top of the list, so it will only works down through the list if the Sound definition is played multiple times within one triggering of the event.

Random:
Completely random selection of the available files.

RandomNoRepeat:
As above but will not play the same sound file twice in a row.

Shuffle:
This randomises the list and then plays through it so it will not repeat any sound file until they all have played.

ProgrammerSelect:
This allows a coder to define how the list is accessed in the game code.

ShuffleGlobal:
This randomises the list as Shuffle does, but no sound file will be repeated in any project event until all sounds have played through.

Volume
This is simply the volume at which the sound files within a Sound definition are
played. This is useful for fine tuning the volume of a Sound definition in an event that contains multiple Sound definitions.

Volume randomization
This function allows for variation in the playback volume between 0 dB (100%) and –60dB (effectively 0%) Each time a sound file is played it will be randomly assigned a volume level between 100% and the defined lowest value. By combining this function with the Volume function it is possible to create a volume window in which the sound files will be played. Setting the Volume to 80% and the Volume randomization to 60% would cause all sounds to be played somewhere randomly between those two values.

Pitch
This allows you to change the relative pitch of the Sound definition by up to 4 octaves. Pitch alteration at extreme levels will not just add variation to a sound but can considerably alter how a sound file will sound when played.

Pitch randomization
This functions in the same way as Volume randomization but applies to a sound’s pitch. Its range of +-4 octaves allows for a large range of variation. Also like the Volume functions, pitch randomisation can be combined with Pitch to create a defined window of variation.

Recalculate pitch randomization
This is a function that allows further control over how the pitch randomization is applied. It allows for the pitch randomization to alter every time the sound is spawned, every time the event containing the Sound definition is retriggered or for it to be controlled through a specific parameter inside an event containing the Sound definition. This last option allows for the same sound definition to be used in multiple events without the pitch randomisation being triggered unless specified within an event.

3D Position randomization
This allows the user to define an area larger than a single point in which the 3D sounds will be spawned. The function will often not be obvious until a sound is implemented into a 3D game environment. Often sounds in games are given static points from which they are emitted to represent environmental sounds such as a factory machine, the ocean or a forest. In a situation such as a forest the sound Is usually an ambient representation of the birds, insects and animals within the forest creating sounds. In reality these sounds do not all come from a single point in the centre of the forest, but with a traditional method of sound implementation this is often the result. The value of this function provides a diameter distance away from the chosen emitter point within which the sounds can be spawned.

Trigger delay
Trigger delay looks and acts similar to the Spawn time function except it defines the window of time before the Sound definition is triggered. This is very useful for creating events with multiple Sound definitions in it. Adding 4 or 5 instances of the same Sound definition that have a Trigger delay value to an event will make the sounds play with a randomised delay between them creating a cascade effect. This is very effective for creating chaotic events such as explosions, crashes, rockslides and other dramatic sounds.

Creating a forest with just one Sound definition
Using the parameters we will now create a sound to represent a forest. This simple approach is very light on memory and asset use and so is ideal to use on platforms where saving resources is an important part of sound design.

The eleven sounds used in this tutorial are all 48KHz 16bit uncompressed wav files. These are quite high quality for an average computer game. Combined they take up about 1.5 meg of memory or disk space. By way of comparison, at 48KHz a 10 second long sound is over 900Kb of memory. Creating a forest ambience sound out of a single 10 second long looping sound is straight forward to do, but at 10 seconds long it would become very obvious that the sound was looping unless it contained very ambiguous sound material. The sound we are about to create will have very distinctive bird calls, but for less than 1.6 meg of memory we will create a sound that can play indefinitely and will never loop repeat.

I have used the following values

Name	Birds
Spawn time	[200,500]
Maximum spawned sounds	2
Play mode	RandomNo Repeat
Volume	-6
Volume randomization	-6
Pitch	0
Pitch randomization	0.2
Recalculate pitch randomization	Every spawn
3D Position randomization	3
Trigger Delay	[0,0]

The values can always be varied to suit the user's needs and personal tastes, and different source material will support a wider range of parameters. As I created this sound I was constantly tweaking the values of the above parameters. I do this for two reasons. Firstly to try and achieve the best result for the sound I am creating. Altering the values while listening to them is the only way to really capture the feeling of an alive sound, but secondly I always want to know how far I can push the parameters before things really start to sound wrong. I started with the pitch randomisation at +-.33 but it sounded too extreme, I was getting bird calls that sounded too shrill and too deep, so I narrowed the range for the final sound, but extending the range still further allows me to listen and see what other effects I might have created. This is often how new ideas for sounds come about. Accidently creating a new sound by breaking the sound you are currently working on can be really useful for later projects.

With the Spawn time values I have used, there is a high spawn rate and the end result sounds like a tree with a large number of noisy birds in it; similar to dusk when all the birds are nesting for the night. Increasing the Spawn time numbers will thin out the rate of sounds and create a less populated forest. Increasing both the 3D Position randomization and the Volume randomisation will create the effect of the birds being spread over a larger area with calls more in the distance. Adding an overall drop in Pitch to the entire sound could create an unreal effect of giant birds or other monstrous creatures inhabiting some far off strange environment. Playing with the values can produce a wide range of end results and add more flexibility for the end user. Remember that all of these alterations and combinations of sounds is still only using about 800Kb of sound files so this could be used on something as limited as an iphone, all the way up to any of the next gen consoles.

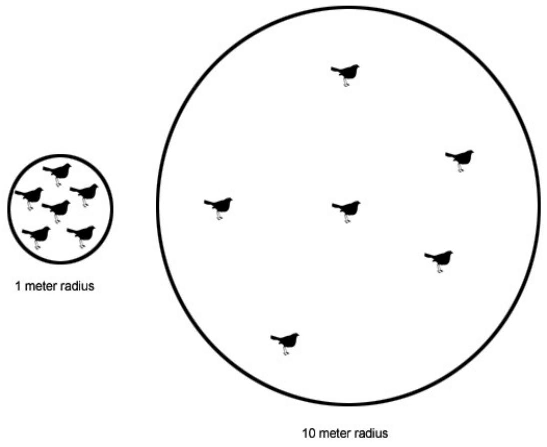


Figure 3. Different values for 3D position randomisation, but equal values for Spawn time and Maximum spawned sounds.

With the spawn time the same, increasing the 3D position randomization will increase the area over which the sounds will be generated in the 3D world. But as the same number of sounds is being generated over a larger area the effect will be of a sparser distribution of sound sources. (In this case the same number of birds in a larger area)

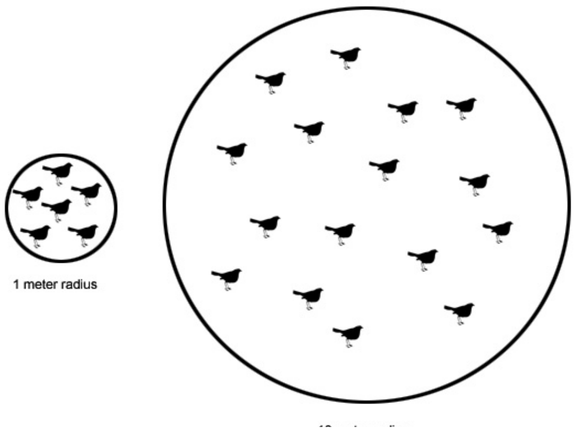


Figure 4. Increasing the values for 3D position randomisation requires increased values for Spawn time and Maximum spawned sounds to achieve the same density of sound.

In this case, to achieve the same level of density in the 10 meter area the number of spawned sounds has been increased as well. Decreasing the values in the Spawn time function will mean that a new sound will be generated sooner, but you need to increase the maximum number of spawned sounds to allow more sounds to play simultaneously.

At this stage we have only created a Sound definition. Although it might be a sound we can use in a game it is not yet ready to be used by a programmer. For this we need to create an Event.

Events Page

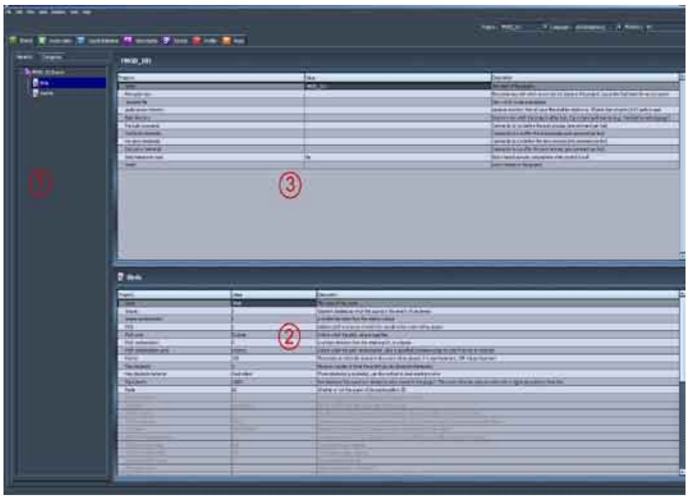


Figure 5. Events page

1. Event list window. All created events will be displayed in this area. Right click to create a new event. Events can be sorted into Event Group folders to help manage them.
2. Event Parameters window. Each event has its own parameters displayed in this window. Common parameters can be group edited by selecting two or more Events prior to editing in the same manner as Sound definitions.
3. Project settings window. These parameters apply to the entire project and define information such as the location of the source files directories and the desired directory in which to build export files.

The Events page is similar to the Sound definition page in that it has a list window and parameters window. The parameters for Events are independent to, but combine with the parameters for Sound definitions, so lowering the volume on an Event will lower the overall volume of that event regardless of the parameters of the individual Sound definitions within that it, but it will not effect those definitions if they are used in other events. There are a lot more parameters for Events than for Sound definitions as these provide important information to how the Event will work in a game environment.

Figure 5. also shows two Events, one named birds and one named Insects (although I changed the name of this second one to “Birds and Insects”). Right click in the Events list window and create these two Events, we will get to the second Event later.

For the purposes of this tutorial we are going to leave all the Event parameters at their default settings except for one. Find the **Mode** parameter and change it from 2D to 3D. We have done this so that we can more accurately audition how our bird sound is going to work. 3D allows FMOD to simulate the 3D position of our Event when we listen in stereo. Combined with the 3D position randomization parameter we set in the Sound definition this will create a stereo image of our event. Otherwise our bird sound will be completely mono and sound very flat.

Now we will add sounds to our Event.

Event Editor Page

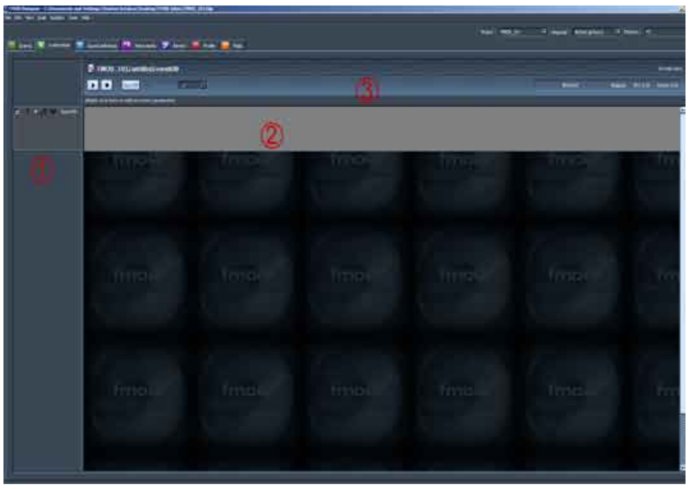


Figure 5. Event editor page

1. Layers Panel. Right click to create new layers. Right click within a layer heading in this panel to add effects or to modify the selected layer.
2. Editing window. This is where Sound definitions are added to layers and edited to create the desired sound event. Right click within a layer to add or replace a Sound definition.
3. Parameter Panel. Right click to add a user defined parameter to your events. Parameters are powerful controllers that allow for events in a game environment to change how your sound is played.

The Event editor page is where sound design really begins and it is also where many of the old methods of design need to be rethought. Firstly, and most importantly I think, it is important to understand that this is NOT a linear sound editor. The horizontal axis is not a measurement of time. There is a line cursor that can be positioned within the window, but it does not chase from left to right counting seconds like a traditional editor when you press play. This often confuses people and unfortunately can result in them not getting the best out of FMOD. Designer will play any Sound definition that is under the curser line, but by default it will simply play every Sound definition present in the event.

First let's create the most simple event possible. Right click in the Edit window in layer00 and select **Add sound…** a pop up window will list your available Sound definitions, in this case we only have one, so select Birds and add it to your event. Make sure you choose the **One Shot** tick box in the pop up window. There should now be a Sound definition titled /Birds that fills the entire layer from left to right in the edit window. If you press play now you should hear the same sound we created in the Sound definition window earlier, except there should be one noticeable difference. Because we selected 3D under the Event parameters and we also defined a 3D position randomization value of 3, the sound when played through the event editor is stereo in effect. The birds sound like they are all around your head in different positions. (This is much easier to notice if you listen through headphones).

When played in a 3D game environment, this event would be triggered at a specific location. The 3D position randomisation function means that there is a variation on that position of 3 meters, so the birds inhabit an area larger than a single point in space and as a result sound more realistic. If you go back to the Event parameters and change **Mode** back to 2D, the event will return to mono for the purposes of auditioning and sound exactly the same as the Sound definition. Try this step if you want to hear the difference. We have now created a simple event that can be added to a game environment by a programmer.

Cleaning up

One thing that might be noticable is that there is a slight hiss in each of the bird sound files. This is background wind that was present when the birds were recorded on location. It is very rare to be able to record animals outside without there being some wind presense. If we had a single long recording it would sound quite natural, but because we are selecting from a series of smaller files that often have a period of silence between when they spawn, this noise, or rather its absense, becomes apparent when we trigger our Event. there is a technique from film production that we can use to fix this issue.

In film production they often have a similar problem when recording people speaking. Microphones in different locations, or lines rerecorded later in the studio often lack the background sounds of the filming location. Traffic hum, airconditioner and wind noise are often present when filming occurs. To counter this, the sound crew will usually record some extra material they call Atmos. This is a recording of the background environment without any other sounds in it. This can then be layered into any edit to provide an underlying layer similar to that present on the original location shoot.

We are now going to add an Atmos layer to our birds. In the Sound defintion window create a new Definition by dragging the Atmos.wav file from the provided sound files for this tute. This will be a Sound definition with a single file, we don't need to alter most of the parameters, in fact the only parameter we are going to alter is the Volume. Set it to -18, this will set the Atmos wind noise to a good level to blend with the bird sounds.

Now in the Event editor page right click in the left hand panel under Layer00 and select **Add Layer**. Click in main window in the new layer, select **Add new Sound...** and add the new Atmos Sound defintion. This time leave Looping selected as we want the Atmos sound to loop continuously.

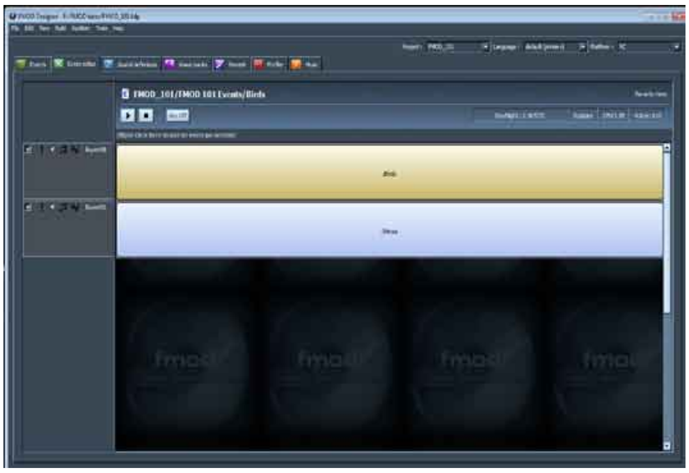


Figure 6. Our event now has two layers. Note that looping sounds are displayed in blue and one shot sound are displayed in yellow.

The Atmos sound helps cover the gaps of complete silence between instances of the birds sounds triggering and adds a gentle wind ambience to the overall sound event. This sound event could now be placed in a game environment to represent a forest populated by birds.

One step further

We are going to take this one step further and in the process demonstrate how to make an ambient environment for a game that is very light on resources and allows for day/night changes. Firstly I am going to edit a source sound to use in this project.

The sound I am going to use is the sound of crickets in country Japan that I recorded a few years ago. Many insects produce a cyclic call and this is very useful when creating sound environemnts. For the purposes of a computer game anywhere that allows you to save on memory and storage space is useful.

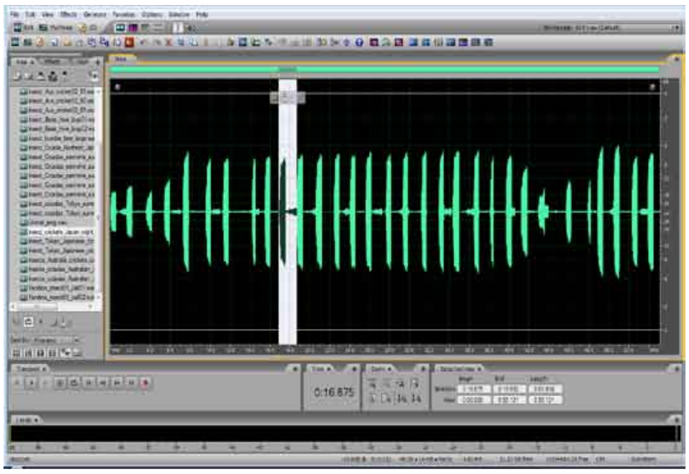


Figure 7. The sound of crickets in a wav file. Many insects produce a repeating pattern sound.
Japanese crickets

Figure 7. shows a sound file recording of the Japanese cricket. The link will allow you to listen to the original source sound .The sound is 55 seconds long at 44.1KHz 16bit sample rate. This results in a sound file of 4.6 megabytes which is far too large to use in any game for a simple background ambient element. The highlit area is a single cycle of the insects repeated call. When isolated and copied to a new file we get the following.

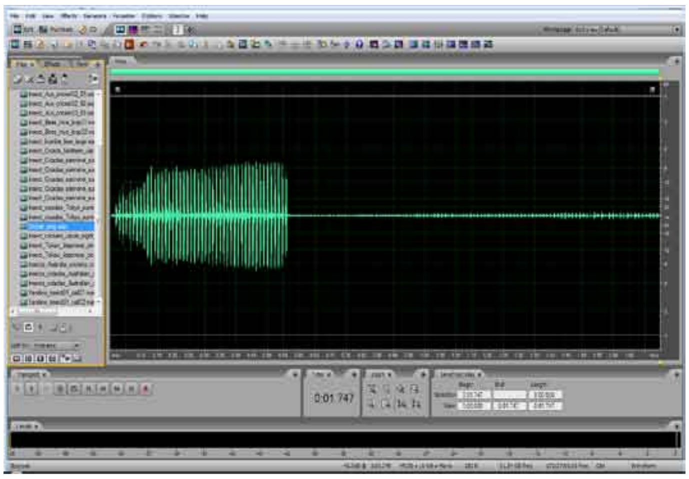
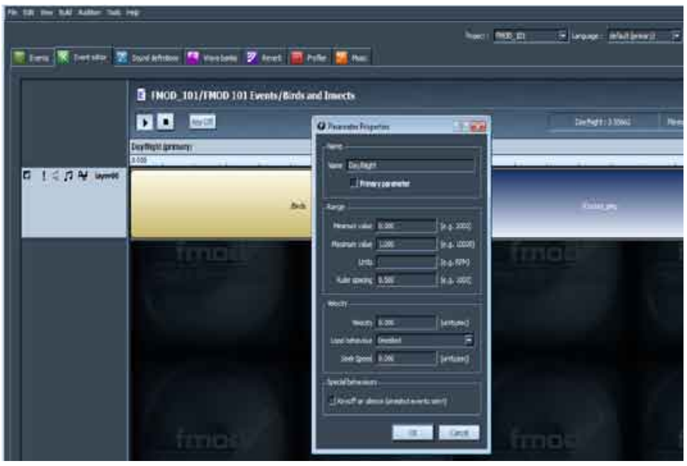


Figure 8. A single cycle of the Cricket’s sound in isolation

Figure 8. shows a single instance of the crickets repeated call, it is only 1.7 seconds long and is only 150Kb. We can use this very small file to create an ambience using FMOD. The link will play our new single isolated cycle. It still sounds a lot like a cricket, but we will modify its behaviour slightly when we implement it in FMOD to make it sound more realistic.
Crickets isolated cycle
Crickets isolated cycle looped

Create a new Sound defintion in FMOD with the Cricket sound file. As we only have a single sound file we do not need to worry about ranomizing the play order. I am also not going to randomize the volume or pitch as I want the sound to be regular. I have set the 3D position randomziation to 5 and after a bit of experimenting I settled on a spawn time of [350,600]. These values can also be tweaked to suit individual taste, but I think these values are a good starting point.

Now create a new Event. I called mine “Birds and Insects” which seemed appropriate. Make sure you set the Mode to 3D in the same way as we did with the Birds event. In the Event editor for the new event right click in the Parameter bar to create a new parameter, then right click and choose Parameter properties.
I have called this parameter Day/night, I left the default value range from 0.00 to 1.00, but I changed the ruler spacing to .500, this will divide our parameter into two main sections. We will use one to represent daytime and one for night time. Click ok to close the property window and return to the main editor window.



Name: Day/Night
Minimum value: 0.000
Maximum value: 1.000
Ruler spacing: 0.500
Velocity: 0.000
Loop behaviour: Oneshot
Seek Speed: 0.000

Figure 9. Event editor Parameter properties window

Right click on layer one and add the birds sound to the left hand side and then the crickets sound to the right hand side. By default when there is a parameter present the Sound definitions will be displayed as small boxes, click drag the edges of the boxes so they each take up one half of layer00. Finally drag the inside edges of each sound def box across the other definition to create a cross fade area between the two Sound definitions.

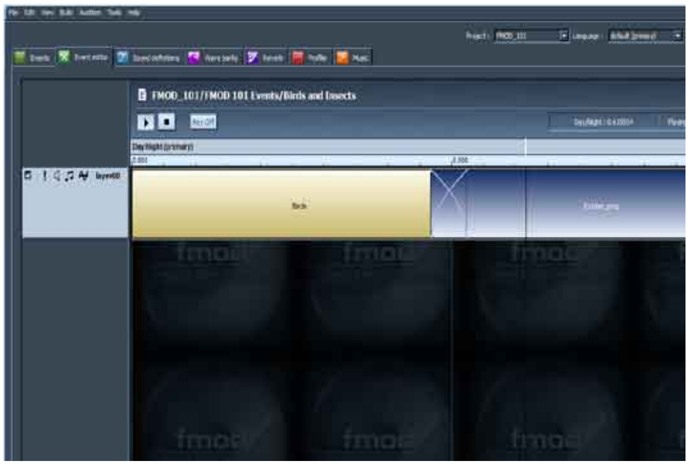


Figure 10. Multiple Sound definitions in a single Event layer. Note the cross fade in the center.

Now if you play the Birds and Insects event you should hear the sounds of the birds. This is because the parameter curser line defaults to the left hand edge. If you click and drag it across to the right side of the screen the sound should change to that of the crickets. In the Event with the mode set to 3D you can hear the crickets playing in different locations so it sounds more like an environment with crickets in it. Moving the slider back and forth will change the active Sound definition. This parameter value is what a programmer would use in game. The Event would be located at a certain point in the game environment and then the game would alter the Day/Night parameter at appropriate times. This results in our sound Event altering between birds during the day and crickets at night time.

Just to be clever

Lets add one more sound to our Event. Copy the Birds Sound defintion in the sound definition list window with the right click menu and past a second copy of Birds underneath the other definitions. Next rename this new definition to Dawn Birds. Now we will change the spawn time to [0,500] and spawn count value to 10. This will a create version of the Birds sound definition with a greater density of bird calls. Now return to the Event editor page and creat a new layer by right clicking in the layer panel under layer00 and choosing **New layer**. In layer01 right click and add our new Sound defintion Dawn birds. Position it and adjust the edges so it covers the cross fade between the two defintions in layer00 and occupies some of the area into the daytime side. Finally add the Atmos layer to the event in the same way we did for the Birds Event.

Now position the cursor in the night side and press **Play**. The crickets will be audible as if it was night time in our game. If you now slowly drag the curser towards the left of the window we now have a far greater presense of birds as you reach the cross fade area. This simulates the sound of birds awakening in the early morning. As the curser moves closer to the 0.00 parameter value the Dawn birds should cease to play leaving us with a typical daytime bird effect.

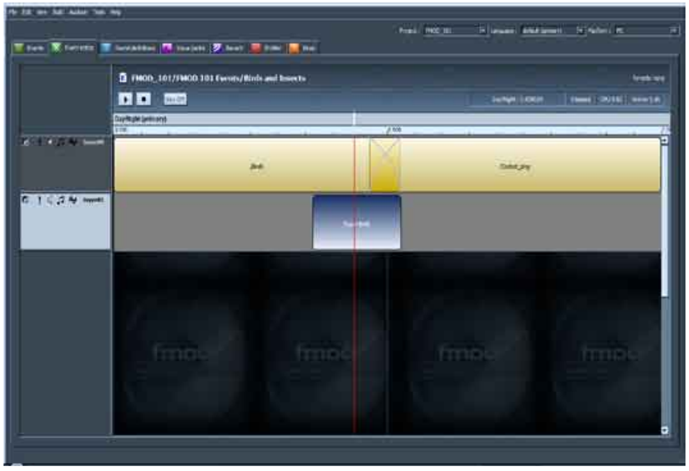


Figure 11. Multiple layers and Sound definitions

Adding Effects

If we look at the overall parameter and its values and consider 0.00 to be midday and 1.00 to be midnight, then we could set the game code to slowly cycle back and forth between the two values. At dawn and dusk we cross fade between the two ambient effects, with the added layer giving the impression of noisy birds waking and going to sleep. If we add Volume effects to each layer we can enhance this effect.

Right click on Layer00 in the layer panel and select **Add effect**. Choose **Volume** from the pop up window. Repeat this step for all three layers. There should now be a red line across the top of all three of the layers. This is a controller that allows us to alter the volume of each layer in the event. Right click anywhere on the red line and select **Add point**, this add a control point that allows you to drag a section of the red line to a new position. In this manner you can create volume fades. Approximate the positons of the Volume effects lines in Figure 12.

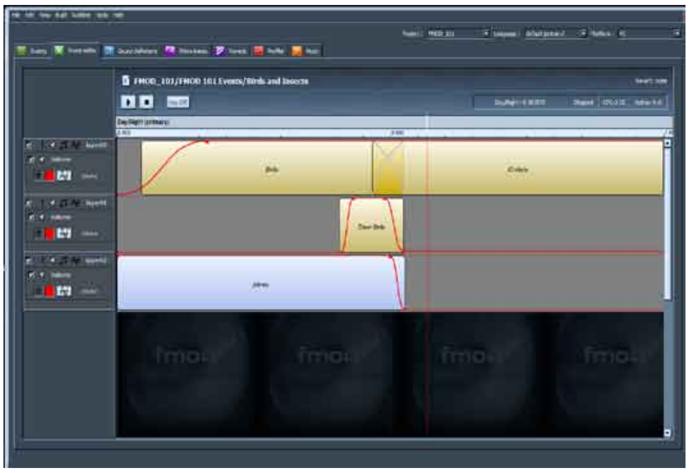


Figure 12. Adding Volume Effects to all the layers

The positions or shapes of the curves do not need to be exact, what we are simulating is the following.
Assuming we have set our game code so time of day is linked to our event parameter we now have a day night cycle.

As the parameter moves from night to day there will be a cross fade between the crickets and birds in layer00. At approximaitely the same time as this cross fade, both the Dawn birds and the atmos layers will fade in. After a short while the Dawn birds will then fade out simulating the end to their noisy awakening. As the curser nears the extreme left end the birds begin to fade out leaving only the Atmos layer playing. This simulates high noon and all our birds are off hiding from the midday sun. Once the curser hits the left hand edge parameter of 0.00 it would reverse and head back towards evening and night time.

This entire demonstration has used about 1.6 meg of assets to create an environment that alters from day to night and includes a dusk and dawn highlight. It uses very simple event structure and limited resources. Designing an effective audio environment should not need huge amounts of resources. Although sometimes this is unavoidable, often it is also possible to work simply. Even on large projects, any place where you can use resources more efficiently frees the extra resources up to be used elsewhere.