



Opening Notes (**PLEASE READ**):

1. *This step by step guide assumes that you are already familiar with the basics of getting around in Unity. You should already have Unity installed and configured on your computer as well. The version number is important. As of this writing (Sept 2022) the current version supported is Unity 2022.3.14 LTS. This version of the game lesson has not been tested in previous versions of Unity and GAI gives no guarantees of its effectiveness, especially when downgrading to an earlier version of Unity.*

This new version of Unity uses the Unity Hub to install the Editor and to open projects. So you'll download the Unity Hub app first, and then use the app to install or upgrade Unity as well as to open projects. If you've never installed Unity before, just search for a recent YouTube video on the basics of installing the Unity Hub and Editor. The basic download of Unity comes with a license for Unity Personal, which is free, and will work fine for this game lesson.

2. *Also, since this lesson focuses on using FMOD Studio, you will need to download the latest version of it here:<http://www.fmod.org/download/>. As of this writing(June 2024), the newest version tested with Unity is 2.02.11.*



Stop! You will need to install the FMOD Integration in Unity for this game lesson to correctly run. Please refer to the README _Install FMOD Integration PDF in your package. You can also watch [this video](#) as well.

3. *Lastly, this is not a course on the basics or concepts of FMOD Studio. This lesson will assume you're familiar with the various windows and sections in the editor. For a description of features, you can refer to the FMOD Studio User's Manual available in your downloaded FMOD Studio.*

GAME OVERVIEW

From the dawn of humanity, time has been defined and re-defined. Day to Night is a simple and fun adaptive exploration of creative audio design controlled by a single parameter. In this case, the parameter is time. 24 hours of time to be precise, just like real life. Your job will be to create music and ambient backgrounds that transition the player smoothly, from morning, to afternoon, to the dead of night, and back again. As the player moves around on the island time will progress, and the sun and moon will illuminate the landscape.

In this Unity/FMOD oriented level from GAI, we give you an opportunity to implement and integrate adaptive ambiances and music, tied to a simple game that moves between night and day. Once you finish this, you'll get a very good idea of how a changing game parameter can drive all sorts of sound effects and music in a game.

NOTE: In addition to this step by step guide there is a video tutorial series for this game lesson that can be found at this link:

<https://www.youtube.com/watch?v=HUWC0eKoEPQ&list=PLVKIvlgIn2rF1OxIfvdFERmei2QCOZjSQ>

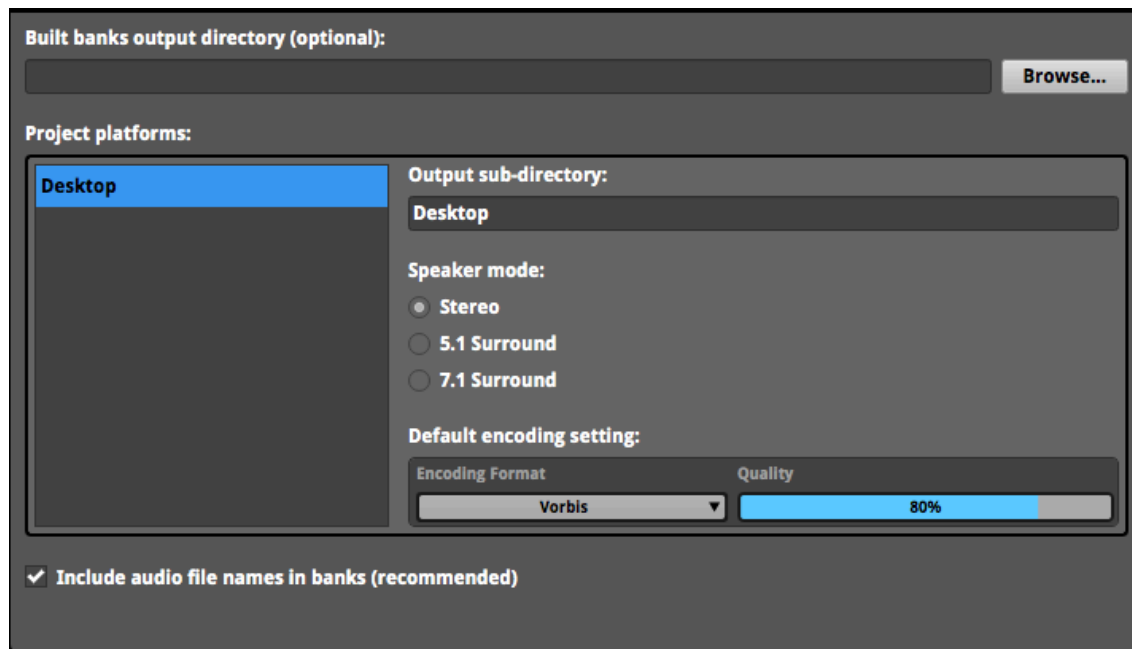
GETTING TO WORK

Fire FMOD Studio up. This starts up with a blank project, which is ideal for us. Let's save this as 'DayAndNight' somewhere convenient and easy for you to locate on a local drive.

Now we need to create two 2D Events. If you've got any prior experience in FMOD Studio, you'll know that right clicking (Ctrl click is okay too) is how you get things done. So right click in the Event Browser on the left side and select **Create 2D Event** to create an event. Do this again so that you now have two events. Name one of these events **Ambience** and the other one **Music**. We'll also cover creating a 3D Event to trigger common foot actions like running, jumping and landing.

We also need to make sure that the output format is set to stereo and not 5:1 surround. By default, all FMOD Studio projects are in 5.1 Surround. To change this you need to go to the FMOD Studio menu and select **Preferences**:

From this window select the tab labeled **Build**. Look below to see the installed platforms, showing only Desktop at the moment. You'll notice the format is set to 5.1. Change this setting to Stereo. Incidentally this window is also where you would add more platform support (such as for iOS or consoles) by clicking in the blank area and selecting from a list. We won't be doing this so we can move on.



Check an Event to make sure the change occurred by clicking on it and in the Editor select the Master track. In the Deck area down below and to the right you should see a single stereo panner

and a stereo meter to the right of this. If you see this, congrats, you're in stereo and we can now move on.

Structure Of the Project

The main theme of the project is that it is a simple Day and night Cycle over a terrain. Accordingly everything is driven by time. When these events are fully integrated in Unity you will have absolutely no control of the sound once it is in the game. The cycle runs automatically in a loop - all you can do in Unity is start or stop the level. Thus the structure in this case is fully adaptive in nature.

The structure is based on a 24 hour cycle that goes from 0.0 (midnight) to 24 (23.99 really), and then loops back to 0.0 and restarts the cycle again. Time in the game is roughly based on a 24 hour parameter line - there's no AM/PM designation. So in this case we are using more of a European, aviation, or military approach.

There are three significant areas in the cycle:

1. Midnight to Dawn (12:00 -6:00am or 0.0 to 6.0)
2. Dawn to Sunset (6:00am-6:00pm or 6.0 - 18.0)
3. Sunset to Midnight (6:00pm - 12:00am or 18.0 - 23.99)

In addition, special attention should be paid to the transition areas around dawn and sunset so that they sound more dramatic than the other time blocks.

Next, you'll need to source or create a number of assets to complete this assignment. We don't have a specific asset list per se because you can import a number of sounds and tracks of music to fit the bill. But here's a basic minimum list:

Ambiences

One or more Night Ambience Sounds. These might include:

- wind sounds
- insect sounds
- frog sounds

One or more Day Ambience Sounds

- birds
- insects
- animal sounds

You should generally keep these ambiances fairly short. Around 20 sec to 1 minute max is usually fine. Also, it's a good idea to check and make sure there's no clicking and popping in your loops and also no obvious sound that easily identifies one specific point in the loop. you want the loop to fade into the background while listening. you can also use short 1-3 sec non-looping sound clips in generator modules like the Scatterer to get more variety.

Music Clips

- Midnight to Early morning music loop - quiet and contemplative
- Short transition to morning from this loop
- Morning to Sunset (Day music) loops -a bit more active
- Short Transition at sunset to evening
- Sunset to Midnight (Evening music) loops - even more active

The Day and Evening loops will each be sets of loops and these should be relatively short as well - less than a minute, in most cases. Each loop in a set will have to be the same length, but the Day and Evening sets should differ from each other in terms of compositional material. So all Day music loops must be the same length and be based on the same material, and all Evening loops would be different material with all loops being of a different but identical length of their own.

Example - say the Day music is 8 bars at 112 bpm. This means that all Day music loops must be 8 bars long at that exact tempo.

Then let's say the Evening loops are faster in tempo - like 130 bpm - and are 12 bars long. This means all Evening tracks must correspond to this exact length and tempo. But Day and Evening music will differ from each other, because they play at different times of day.

Once composed in your DAW you should bounce these tracks out as individual instrument loops. In this method you simply solo each instrument and bounce it out as a separate part. In this way you can dynamically mix the volume and combinations of your tracks with FMOD Studio.

Last, all audio assets for music and ambiences should be bounced out as stereo WAV files at 44.1KHz / 16Bit, and imported into your project's Audio Bin.

Footsteps and Foot Actions

Footsteps are extremely common in games of all types and can range from simple repeated sounds for retro platformers to extremely complex systems covering a variety of surfaces for left and right foot parts like the heel and toe (such as the indie game Limbo used). In this game lesson, we'll be doing a reasonably simple version of a footstep implementation.

First a bit of conceptual background as to the approach. If you normally download footstep sounds via a website like [Freesound.org](https://freesound.org), you will more than likely get several sounds playing in succession - that is, 'footsteps'. To use these in a game situation you would have to figure out a way to have them play continuously. You could loop them, but this is very impractical and would only work for certain specific cases where you can't see the feet.

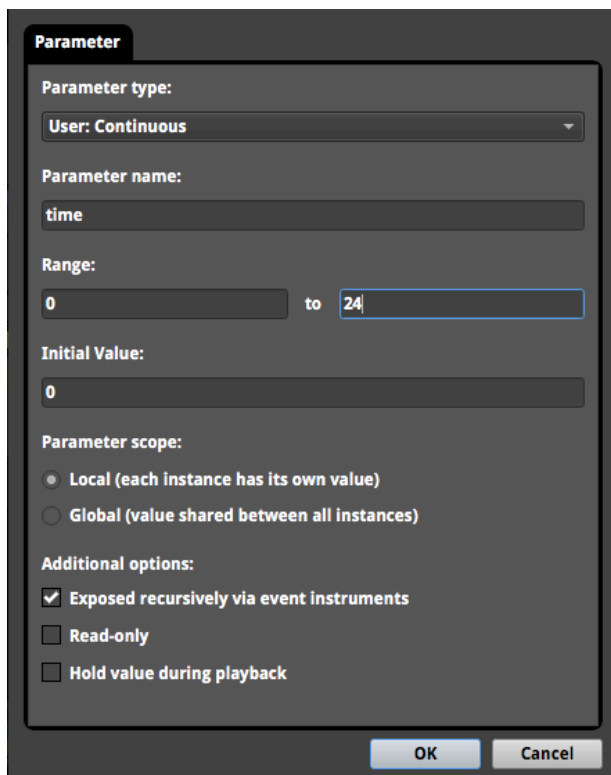
So the solution in this case is to use your sound editor (Audacity or whatever) to cut out each individual footstep sound as a separate audio file. Then they can be imported into FMOD Studio and dropped into a Multi Instrument where the sounds can be triggered individually via a random or a shuffle based configuration. This way the game can decide based on its internal logic how slow or fast to trigger the sounds.

Asset Editing tips - Footstep audio assets should be roughly the same length for each step, but it's most important that the beginning of the sound is immediate and nearly the same in the waveform editor. If there's too much silence at the beginning of one file as opposed to the rest, the results can sound strange and uneven as if you're suddenly staggering around drunk or you just stumbled on something.

We'll cover how to set things up with footsteps and foot actions later. For now let's get to working in FMOD Studio on the Ambience Event first.

Configuring Ambience

Let's tackle Ambience first - for this event we want to put our sound modules on a parameter line, not on the timeline. We have a 24 hour day so let's first create a parameter line called 'time'. Look to the right of the Timeline tab in the Editor for the plus symbol. Clicking this will allow you to create a custom parameter or use one of the many internal parameters available. In this case we want a custom parameter so select the **Add Parameter Sheet** choice when the menu drops down. Configure this parameter as shown below.



The screenshot shows the 'Parameter' configuration window in FMOD Studio. It is titled 'Parameter' and contains the following settings:

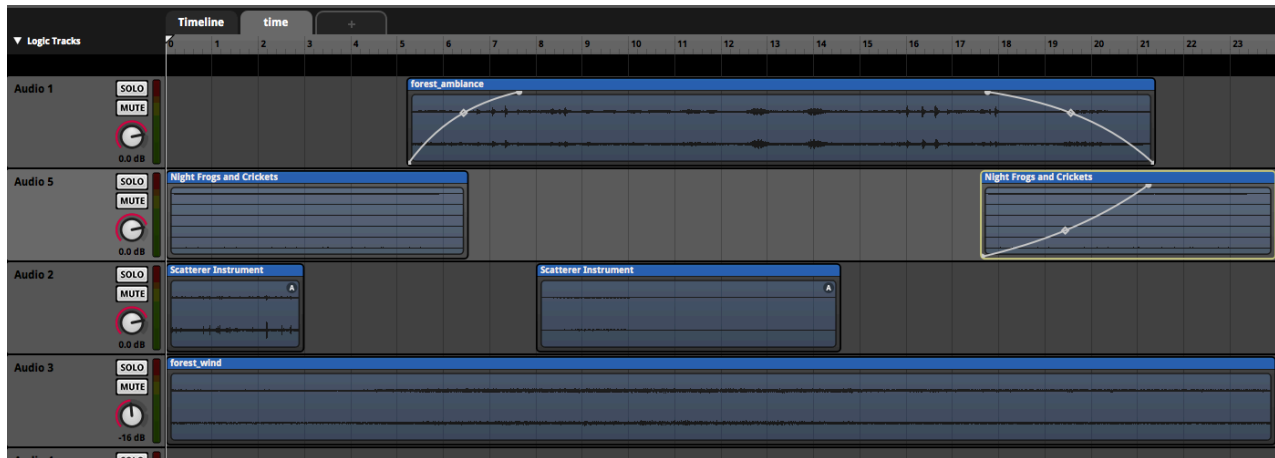
- Parameter type:** User: Continuous (selected from a dropdown)
- Parameter name:** time (text input)
- Range:** 0 to 24 (two text input fields with a 'to' separator)
- Initial Value:** 0 (text input)
- Parameter scope:**
 - ☒ Local (each instance has its own value)
 - ☐ Global (value shared between all instances)
- Additional options:**
 - ☒ Exposed recursively via event instruments
 - ☐ Read-only
 - ☐ Hold value during playback

At the bottom are 'OK' and 'Cancel' buttons.

Now you might be wondering why we include 24 as the size when technically the number won't actually happen. Basically, there's not much difference between the value of 23.99 and 24.0 especially when the game will then immediately revert back to 0.0. So it's just simpler this way.

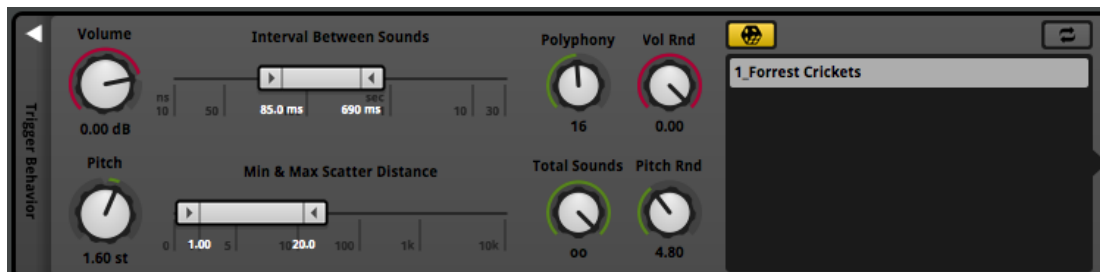
Now that we have a parameter line we should start putting in some ambiances. In terms of organizing the basic flow of the ambiances, you're going to want a sort of **A B A** structure, where A is the more night-oriented ambience and B is the daylight oriented ambience. How you specifically organize this is up to you, but you only want sound effects in this event - no music clips of any kind.

Here's a basic screenshot of how you could organize the ambiences:



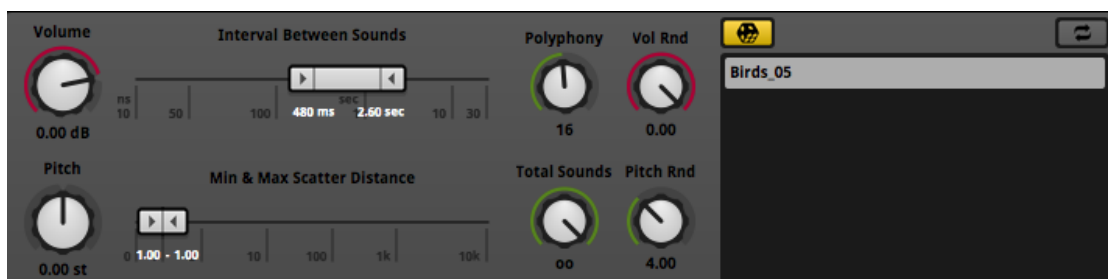
This is actually a quite simple approach using no volume automation or mixing and only four audio tracks. The second track is an ambience of frogs and crickets. Starting around 5:00am we begin to crossfade our night ambience with our daylight ambience which has more birds and such. At about 5:30pm (17.5) we crossfade back to our crickets and frogs and this continues to the end.

Our lowest track simply contains background wind that plays throughout the level at a constant volume serving as a bed for the rest of the SFX. You can certainly feel free to automate the volume of this if you desire. The third track contains a couple of Scatterer Sounds, which are very useful for creating complex ambiences. Here's an example of a very simple application of the Scatterer using only a single sound file as the source.



Feel free to add more sounds to contribute variety to the Scatterer's output.

Towards the morning and middle of the day, about 8:00am-2:30pm another Scatterer is configured to provide a more dense arrangement of bird sounds.



Again, in a more serious production setting there would be a lot more bird sounds available in the list - this is merely a quick setup to test. The *Pitch Rnd* control helps to keep everything from sounding too identical however. Also note that I'm basically eliminating distance by setting those controls at the bottom to 1.0 meaning that everything will sound 1 meter away.

Last, we want to make sure that this ambience has no actual location in the game space. This means it's a 2D Event. To configure this you'll click on the Master Track, and look down below at the Deck area under the Master tab. Highlight the 3D Panner, right click on it and select 'Delete'. This will eliminate the 3D characteristic from your sound in the game.

Setting Up The Music Event

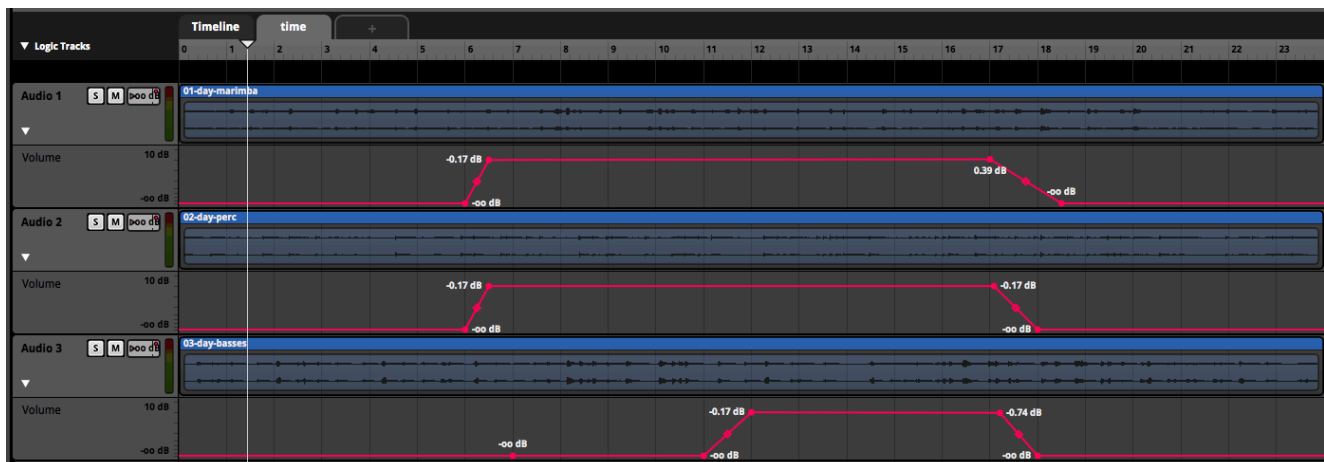
For the Music Event you can take one of two different approaches. Try out Method 2 first but if you find yourself getting confused by the configuration, feel free to switch to Method 1 which should be a lot easier.

Method 1 using only the Parameter line - this method is nearly identical to the method you have been already using for the Ambience Event. In this method you take all of your music tracks for the Day music and Night music themes plus any other pieces or clips of music, and place every element in its own Track. Depending on how many clips and loops you're using in total this might result in a lot of tracks needing to be created. Each looped clip should have only one track assigned on it, and should be stretched out to the full range of the **time** parameter, otherwise there's a good chance that the tracks will not be properly synchronized. Looping should obviously be turned on for that clip in the Deck area. For the non-looping clips you may be able to use a few of these on one track to save space.

You will then put in Volume automation for whatever period of time that the clip covers. This will mean that some tracks will have automation for the daylight hours and be silent (zero automation volume) in the nighttime hours, and for the night music tracks the situation will be reversed with daylight music being silent and the nighttime areas with greater volume.

Additionally for Day and Night music tracks respectively you need to develop a staggered mix of intensities so that over the day various intensity levels are heard until all parts for Day or Night are playing. As an example let's take the Day music. Say it has 4 levels of intensity. So you'll create 4 tracks for each level, put the loop you want on each one, and set the loop in the Deck area for each one.

Next, we draw the automation, we know that daylight doesn't start until about 6 am. So the volume of all Day tracks will be silent (all the way at the bottom) until 6 on the **time** parameter. Next you need to think about which instrument/s you want to hear first and how the mix will develop. For Day music the loops should be roughly from 6 to 18 (or 6am to 6pm) so that's the range covered.



For Night music the range will be much smaller - from 6 pm until 11pm. This is a much shorter range so the entrances will be much closer together. It will likely have to fade out for that last hour before the entire cycle starts over again.

The music from 0 to 6 will also be a loop but does not have to have multiple levels of tension. It should be fairly subdued and quiet however.

Method 2 - Use the Timeline and call Day and Night Music Events Separately

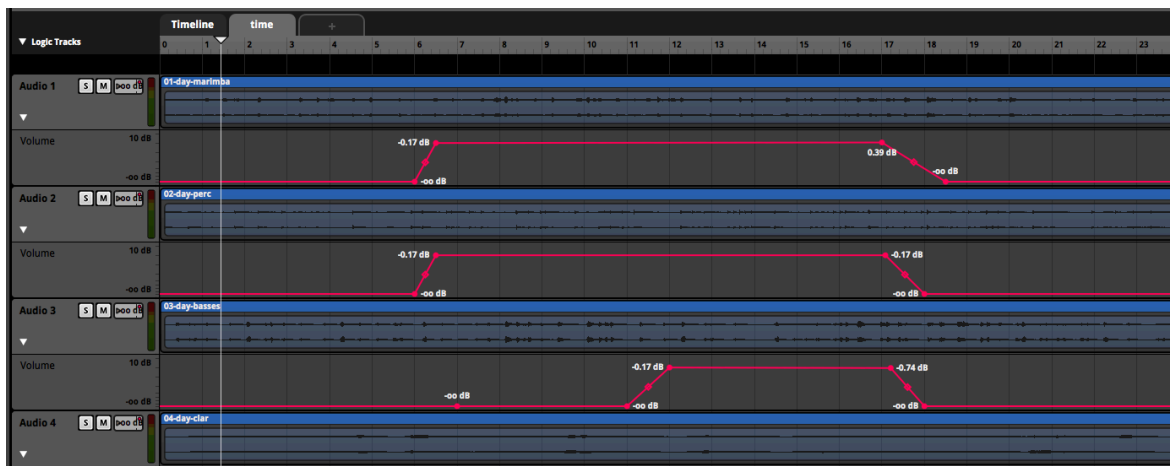
For this method we're going to use a different method to organize the Music event, namely the Timeline, but we will need to create some sub-events (Reference Events) first. Do the following:

Right click on the Event Browser and select **New Folder** from the list. Name the folder 'Music Loops'. Under this folder create an event called 'DayLoops'. Add the existing **time** parameter you created for the Ambience Event. Do this by clicking on the plus symbol and selecting the parameter from the existing list.

Copy the DayLoops Event by right clicking it and selecting **Duplicate** from the menu. Rename the duplicate to 'NightLoops'. We now have 2 Events that we can reference in the Timeline to control our music, and both will be using the same **time** parameter.

Okay, now let's make our Master Music Event. Create a top level (not in a folder in other words) Event called 'Music'. You'll also want to use the existing **time** parameter you did with the Ambience and other Music events. This will sync all of the actions for all the Event in the project to one parameter which is what we want. Leave this behind for now. Let's look at the DayLoops Event first. Click this to select it.

Next, we're going to need to add some tracks. Right click on the audio track or in the Editor on the left, and choose **Add Audio Track**. Add the amount of tracks that you bounced out for the Day Music.



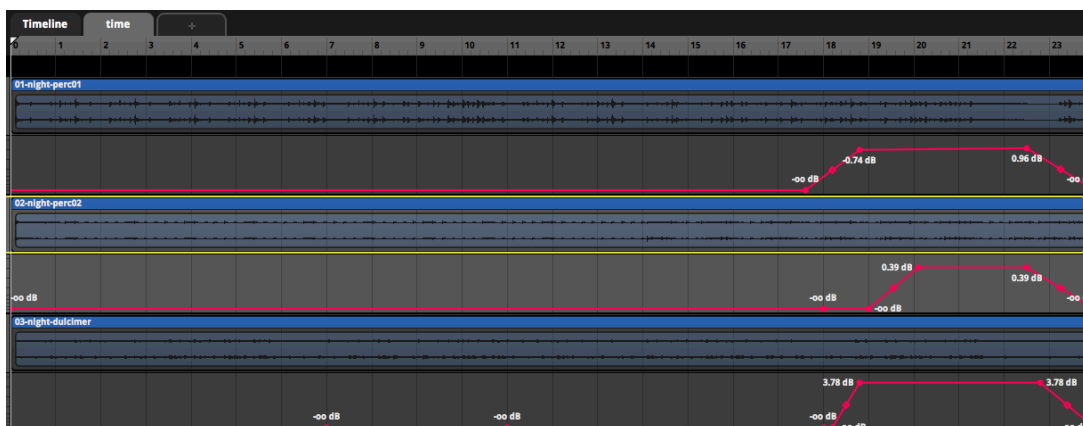
This example has several, but 3 tracks is probably sufficient enough for the concept.

To get these in you can simply drag them in from the Audio Bin and drop them on the track. Remember to make them loop by selecting the option from the Deck area in the upper right corner. Otherwise they will only play once. Remember also to stretch them so that they cover the entire parameter range. The reason for this is to keep all parts in sync with each other. Otherwise parts would start and stop at various times unrelated to the tempo.

To add automation for a track you'll need to right click on the volume control on each track and select **Add Automation**. Clicking in the Editor area then allows you to add automation points and adjust curves as needed.

Now a really important thing to keep in mind here, is that time is our master, or more precisely '*time*' is, actually. Everything needs to happen from the beginning of the game level and the time parameter drives everything. Since this music is the Day Music, we need to make sure it happens at the appropriate time. Dawn is about 6am in this setting or 6.0 on the number line. so you'll notice the volumes of all instruments rise after 6.0 and fall by about 18.0 or so - that's sunset, or 6pm. so you'll make your music mix do the same.

Now let's look at the Evening music by comparison:

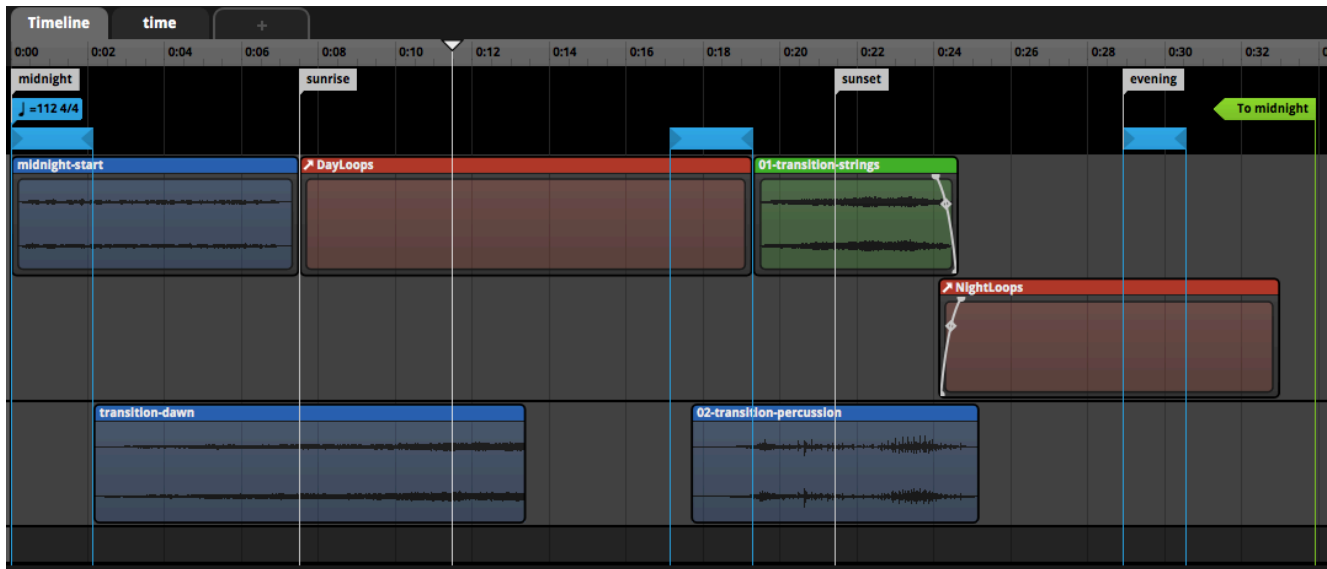


The method you're going to use for configuring the Night tracks is the same but note how brief a space we have for our evening music. We had twelve virtual hours for our day loops but here we barely have six! So keep this in mind - It won't be playing for very long before midnight hits and we return to the beginning.

Additionally we will be needing some kind of music for the area at the start of the game at 0 or midnight until 6 am when the Day Music begins. Note that this doesn't have to be a loop with multiple intensity levels.

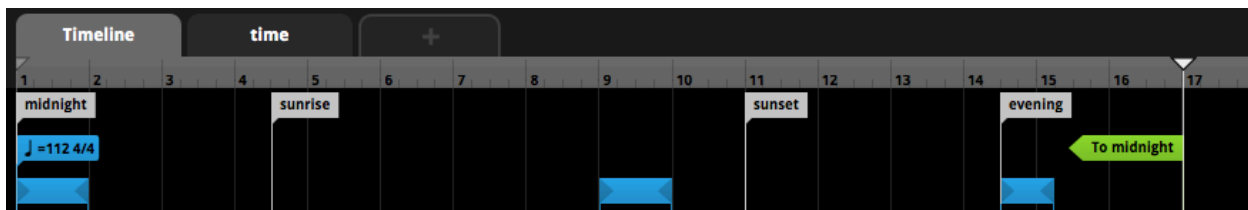
Setting up the Music Event on the Timeline

Okay with the work on our sub events underway, we're getting to our master Music Event now. Let's take a look at how this works:



This may look a little bit strange, but let's just take it a bit at a time. First let's start with adding our Reference Events. To add a reference event you'll need to drag over the Event you want to trigger. When you do this you'll notice an arrow in the upper left corner, indicating that the clip in question is a Referenced Event. Do the same for the Night Event. Place the DayMusicLoops on the left and the EveningMusicLoops event on the right. We'll position them more precisely later on.

Next let's look at the roadmap in the Logic area at the top of our Timeline:



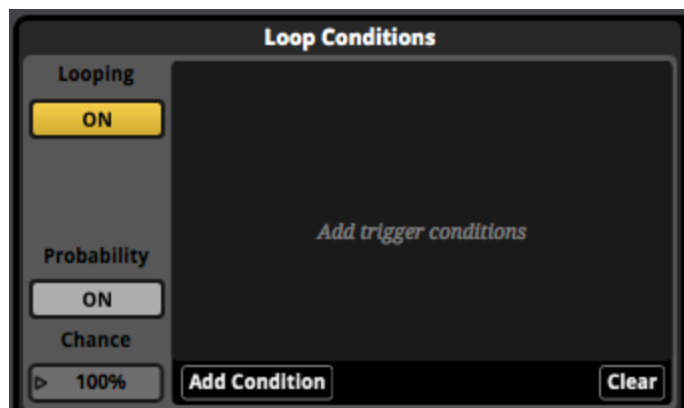
First thing to mention here is that the Timeline is misleading. It does not represent time as it is measured in the Unity game level during gameplay. Let's look at the grey colored tags first. These are Markers and they can play an important role in directing traffic in an event. Most of these markers here are purely for show though. They are indicating roughly where in the Timeline a significant event

occurs. The one really functioning Marker is our first one called *midnight*. The event gets to the end when it reaches a Transition - that's the green arrow on the far right. The transition then calls the *midnight* location and the cycle repeats. We have to put this in because the cycle in game keeps looping and without that return to the *midnight* marker it would keep on playing the timeline.

Create the Markers and Transitions by right clicking in the Logic area (the black band) and selecting the appropriate item. Place these in various points on the timeline in the order shown. They don't have to be exactly the same position - this is rough placement - but it should be in the same order.

Next let's cover the blue Loop Regions. There are three of these at various places on the Timeline. To create a Loop Region right click in the Logic area and select **Loop Region**. Now you can position and resize the region accordingly.

Now technically without any change, if you played the event from scratch it would just hit the first Loop Region and stay there until you stopped the event. However, appearances can be deceiving. Loop Regions can be conditional. Click on the leftmost Loop Region and look below in the Deck. You'll see something like the image on the right:



From here, select **Add Condition** and from the available list select **time**. You'll see something like this:



Now, using the adjustable range slider we can set a range that will activate this loop, but more importantly when the parameter value exceeds that set in the condition, the loop is canceled, allowing the Timeline to proceed. So set this loop up in the manner shown, and configure the other two Loop Regions like so:

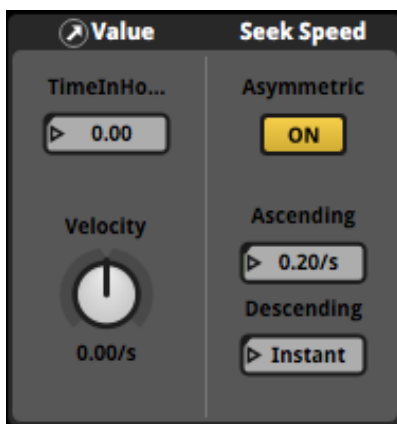


Note that we don't want to leave too much of a gap between ranges. Also note that these are the settings that are good for this version, and they might need adjusting depending on your individual approach.

Okay so what's happening with the Timeline? Basically we stated earlier that the parameter *time* is our master and it runs constantly. So the value will increase constantly, until it restarts the day cycle. So what happens is the sound progresses until it hits a Loop Region. It will then progress while inside

the loop until the *time* parameter exceeds the range specified. After this it breaks out and then progresses smoothly until the next loop which catches it, and so on and so on. Technically we don't really need Loop Regions, but they are very useful to know about, and they do save a lot of horizontal space. Each of our Reference Events for Day and Night will also progress on their parameter lines, because they are slaved to the same *time* parameter. Make sense?

So at this point we need to put in our Loop for the midnight to morning music followed by our various transition points - sunrise, sunset, etc. This is going to vary a lot based on your individual material. The most important thing right now is knowing how to test the event in FMOD which is a bit tricky. If you use the simple method we outlined earlier for the Music Event or you're working with the Ambience Event you can use the following method to utilize the Seek Speed function to move the parameter value smoothly. For this, you'll want to click on the parameter tab and look below in the Deck area:



Turn on the Asymmetric option as shown and set the Ascending value to 0.20. Leave the Descending value alone - since the game parameter automatically changes to the next day after 23.99 back to zero (midnight), you want to change instantly back to 0.

To work with this, have your Editor area zoomed out to the maximum parameter value range, where you can't zoom out any farther. Then start playback and quickly click the highest number value on the parameter line. If you've done it correctly you should see the cursor slowly moving to the right at about the pace of the Unity project.

If you've chosen the more complex Music Event based on the Timeline and using Referenced Day and Night Events, unfortunately this method will NOT work for you due to the internal design of parameters in the program. You'll have to play the Event and then click on the *time* parameter tab and move the control slowly towards the right by hand. If you hear the changes of your music begin to change based on the parameter value you know you're doing it right.

Implementing Foot Action Event

In addition to the Ambience and Music Events we've already covered you'll need to implement and integrate a single Event that will handle different foot actions. There are three actions that will be triggerable by the First Person Controller object and script in the game. These are:

- Footsteps
- Jump sound
- Landing sound

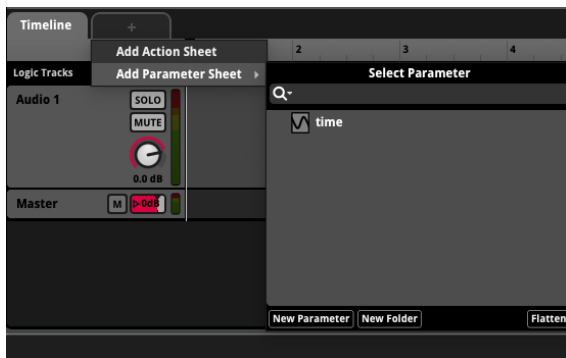
Because there isn't a triggering conflict for each sound here (jumping and landing don't trigger footsteps at the same time), we can easily put all three sounds in one FMOD Studio Event and then

using a parameter, switch between the desired sound we want when that action is triggered in the game.

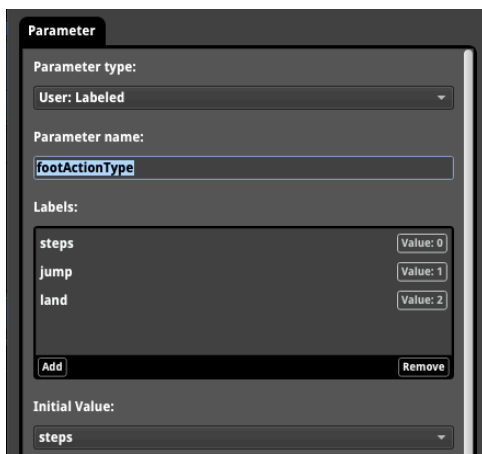
To get started you'll need to create a 3D Timeline Event in your DayAndNight FMOD Studio Project. In this case it's named **Footsteps**. A 3D Event means your Master track will have a Spatializer component added to it as shown here in the screenshot:



Next we need to add a parameter, so click the + tab to the right of the Timeline tab to add one.



Click the **New Parameter** button at the bottom, and you'll get the dialog to set the parameter name and other characteristics:



For the parameter type, select **User:Labeled**. This makes it easy to see how the parameter is sectioned off with three labeled choices. Note that all of the remaining settings will reset if you change the parameter type, so be sure to change this one **FIRST**.

Next, for the parameter name, you **MUST** enter in **footActionType**, exactly as shown in the screenshot. This is

because the setting is hardwired into the script handling the foot events.

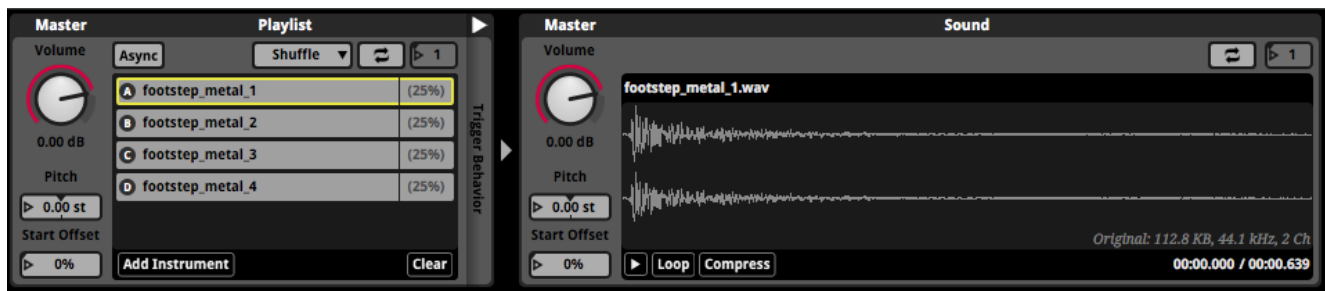
After this you will need to edit the labels as shown, the parameter values here are more for your understanding - FMOD Studio actually doesn't use them. Instead what it uses are the values printed on the right side. So the game engine will always be sending number values (usually floating point numbers) when triggering parameter values on the Event.

Once you're done here click OK and we'll go into setting up the Event itself.

First we'll set up the **steps** section/value on the parameter line we created (not the Timeline). For this you'll have to have imported individual edited step sound files into your project. Then select all the steps you want to, and drag these over the area underneath the label. FMOD Studio will then create a Multi Instrument and add the sounds you've selected to it.

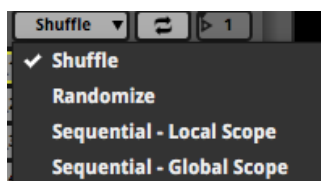


This Instrument will play back a trigger from a list of sounds, rather than from one single sound. There are a couple of options you can configure here. Look below in the Deck area:



Each one of these playlist items has a unique footstep sound asset. There are three sections to the trigger area:

- **Master section:** In this section, you'll find the volume and pitch controls. These controls can be modulated in several ways and you are able to add a parameter for automation as well.
- **Playlist section:** In this section, you'll see all of the sounds available in the playlist. To change the order of any of the sounds, simply click and drag on the item in question and move it to the desired location., Take a look at the menu underneath the Playlist label. As it is set currently, the sounds in the list will trigger in a random and non-repeating order.

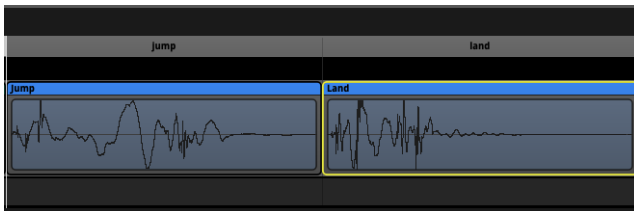


There are also three more optional settings:

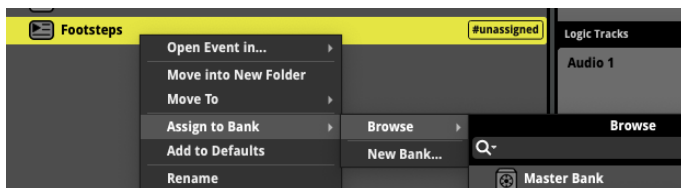
- **Randomize** - will play random item but will allow repeats

- **Sequential - Local Scope / Sequential Global Scope** - This will play sounds in order. Local and Global Scope are a bit too much to explain at the moment, but just leave this set at Shuffle and we'll move on.

Next we'll move into dragging over the landing and jumping sound. These can definitely also be Multi Instruments if desired, if you have multiple jumping and landing sounds you want to use. In this example there's only one sound for each section/value since it's not going to be heard often, but randomizing pitch and volume can help keep things fresh when moving around.



Okay now that we have that set up we just need to add the Event to the bank and Build it. To add the Event to the bank, right click on it and select **Assign To Bank** and then browse for the Master Bank and select it.



After this go to the File menu and select Build or press F7. If your FMOD Studio project and the Unity project are open and connected to each other you should get an indication that the Banks have been updated.

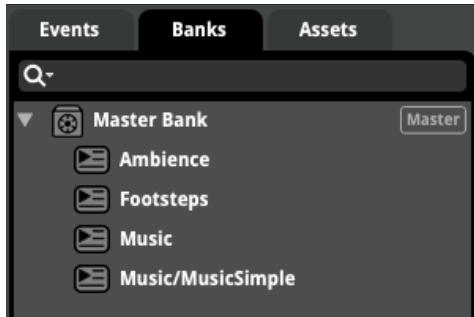
Okay if everything works well here the next step is to configure it directly in the scene, so let's do that next.

Exporting Your Project

Now that we have configured everything in FMOD, it's time to get things ready to export so we can get it into Unity. We just have a few more tasks that need doing.

First click on the Music event and make absolutely sure that the Seek parameter is set to 0.0 so that it is off. This is extremely important, because our time parameter reaches a maximum amount (23.99) and immediately returns to zero (0.0). If Seek is still enabled the time value will result in very slow movement and your Event will not work properly. In the game we must make sure that any form of gliding from one value to the next is turned off, because the game level itself already changes those values slowly for us.

Next, we need to assign all the events to a bank. That means the two sub events in the Music folder as well (if you chose the more advanced Music implementation option). To do this simply right click on the Event and choose **Assign To Bank** and then **Master Bank**. Now, to verify that the Events were assigned properly go to the Banks tab on the left side in the Browser area.

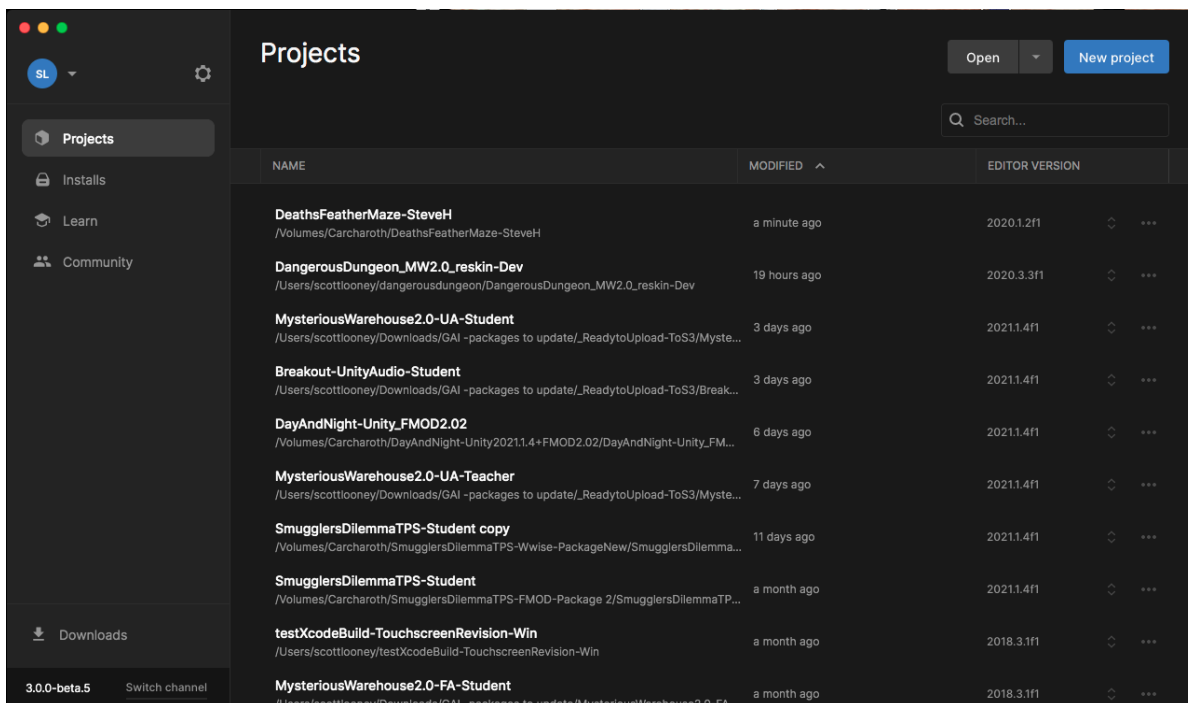


On the left you see the Master Bank bank, which is always present. Clicking the triangle there will reveal the list of Events below. Note that I am also including all referenced events, as well as the top level events. Note also that if you wanted to add support for building banks on iOS or Android, or add support for XboxOne or PS4, you would do it in the Build tab of the FMOD Preferences as shown earlier when we covered the switch from 5.1 to stereo. Since everything looks good here we can move on. Let's build a bank.

Building a bank in FMOD Studio is a very simple and common process. All you do is go to the **File** menu and choose **Build**. That's it! All the settings of the bank were already set up so once that's finished the building process is fairly direct. Once the Build process has completed, let's move on to Unity.

Opening the Unity Project in the Hub

For this you should already have the project folder in the game lesson uncompressed. It's called 'DayAndNight-Unity'. To Open this in Unity, you must first open the Unity Hub application. and then from the Projects Heading, click the dropdown menu next to the **Open** button and select **Add Project From Disk** Then navigate to the project folder you want to add, and click Open. Then click the heading that appears in the Hub to open Unity and the project.



Unity may ask to update the project. This is usually safe to do, so you can confirm this.

The project and specified scene will now open up in Unity. Check to make sure your project has an **FMOD** menu heading at the top as shown here. Now we just need to import the banks.



First Time Run FMOD integration

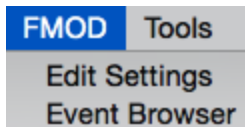


Stop! You will need to install the FMOD Integration for this game lesson to correctly run. Please refer to the README _Install FMOD Integration PDF in your package for directions.

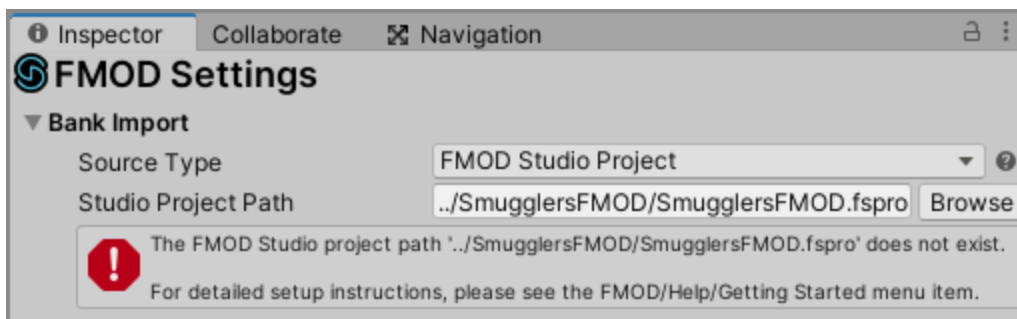
Connecting The Integration to Your FMOD Project

Almost there! Now that we have opened the project, we need to connect the FMOD integration with your current FMOD Studio project so it knows where to find all of the Event and Audio data. If you have installed the integration but not configured the connection to the project, continue following the directions here.

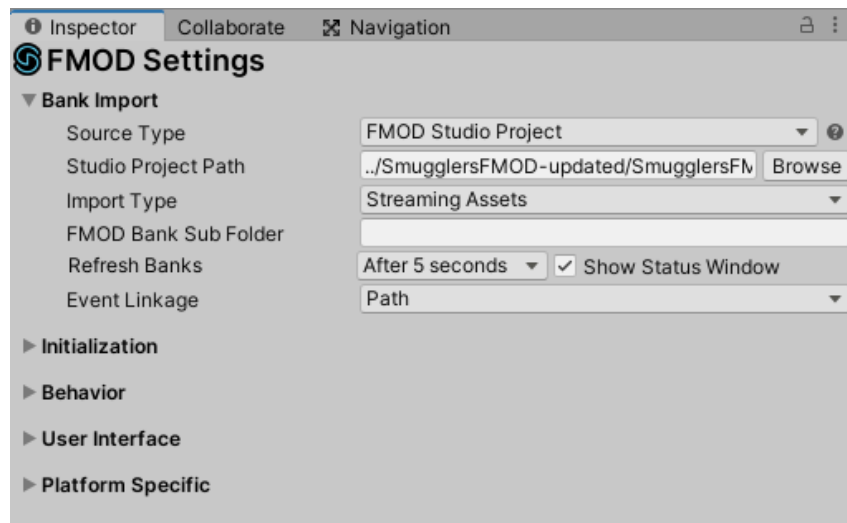
- To integrate Unity with FMOD Studio go to the **FMOD** menu and select **Edit Settings**.



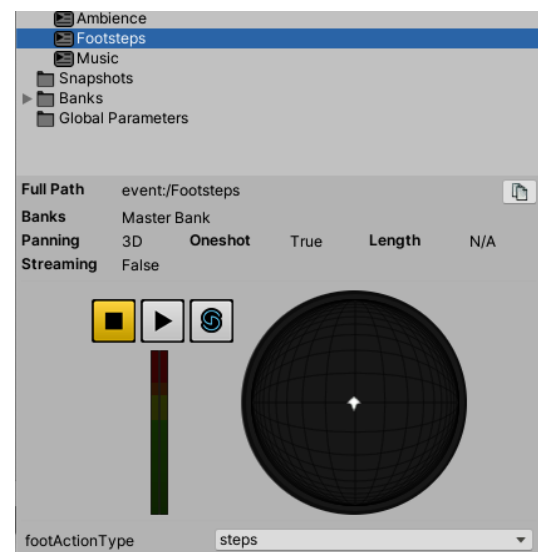
- The settings should show up in the Inspector like so:



- By default it's set to integrate with an FMOD project but it can also simply use one or more banks, via the Source Type dropdown menu. With any middleware that uses banks, it's really the bank that gets copied to the game engine. It doesn't usually care about the project itself except as a means to find a bank to use.
- Now that you know this, click the **Browse** button and navigate in the dialog to inside your new FMOD Studio project and select the fspro project file inside.



- It is possible that you may still get some error messages in the Console, indicating that it was unable to find a bank to copy over. If so you will need to make sure that you've built a bank and added Events to it.
- After a short bit the importing should conclude successfully. To verify this, go to the FMOD menu and select the **Event Browser** setting. You'll get this window which will allow you to browse events in the FMOD project directly from Unity.
 - You can even preview these events to see if they will work properly. For example, if you click the **Music** event, you will see its information plus the ability to play the event and change its parameter values.
 - Click the Play button to trigger the playback of the event.
 - Move the **time** slider to the right. The effect should be quite similar to sweeping through the same parameter in FMOD Studio. If it seems to take too long for the music to respond, you may have the Seek still enabled, so you should check your FMOD Studio project to make sure.
 - You can also check the Footstep Event here as well, as shown in the screenshot.

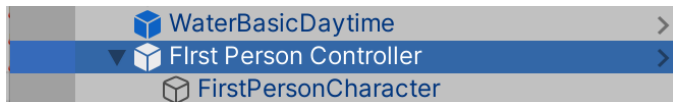


If all sounds about right here, it means that your FMOD project's audio is correctly configured for those Events and we're ready to start configuring things in Unity itself. If there's an issue cropping up here it could mean that the Event is not set up correctly in FMOD Studio. One tip to know for 3D sounds is that Previewing any 3D Built-in parameters will not audition this way. This must be done in the game level itself. We don't have to worry about this for this project, but it's good to keep in mind.

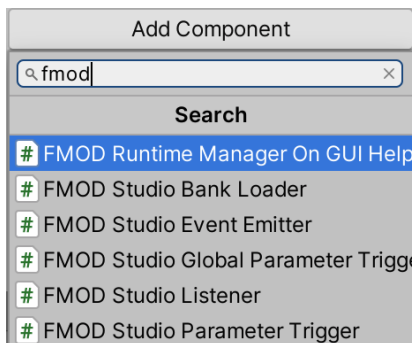
Configuring FMOD Events in the Unity Project

Now we need to make sure everything is set correctly inside the Scene itself. Because the FMOD Studio integration uses a completely separate audio setup than Unity's, it's very important that we verify that all of the parts are present that we need to route audio to the FMOD Studio engine.

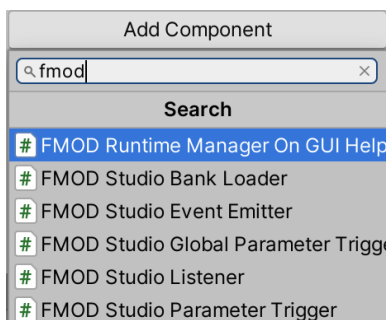
1. First, we do need to make sure that the right scene is open. In Unity after opening the project the screen may be blank, resulting in an empty scene, so we need to load the right scene to get started. In the Editor, look in the Project View under **Assets/Day-Night Cycle/Scenes/** and find the **DayNight-student.unity** scene file and double click it to open it.
2. First, we need to add a listener, so look in the Hierarchy for the **First Person Controller** object and then the **FirstPersonCharacter** object underneath it.



3. Once the object comes up, click on it. At the bottom of the Inspector click the **Add Component** button. In the search field, type 'FMOD' and find the script marked *FMOD Studio Listener*.

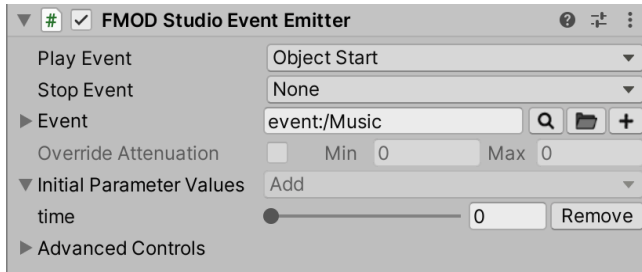


4. Select this and click the **FMOD Studio Listener** listing near the bottom. This will add the Listener Component which will handle FMOD Studio's independent output. When you're ready, move to the next section below.
5. Now that the Listener is in, we just simply go to our Hierarchy and look for two objects that are located underneath the parent object **Day_Night Circle_FMOD**. One is labeled Ambience and the other Music. Both of these objects are blank. Select both of these objects (with Shift click) and click the **Add Component** button. Since our search term should still be there, so we can see all of the FMOD components we will need:

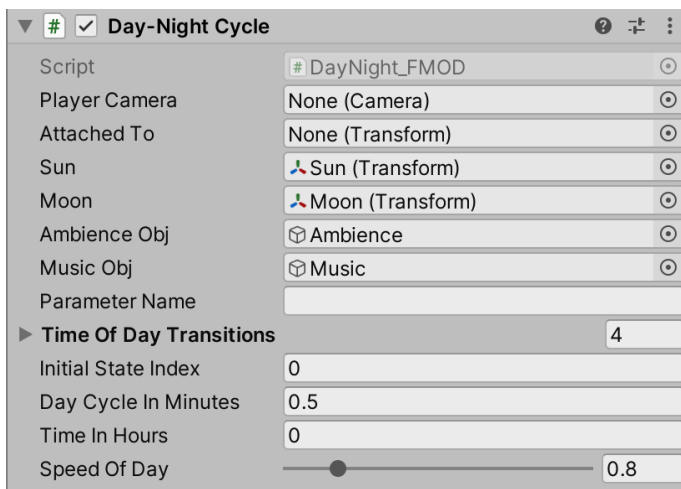


6. Select **FMOD Studio Event Emitter** by clicking on it. This should add the same component to both objects.

- Let's take a look first at the **Music** object and The Studio Event Emitter:



- Configure the Component as shown above. For the Event value click on the magnifying glass icon to select it from the list. The folder button opens the Browser window where you will have to drag the listed events, and the plus sign adds an Event to FMOD Studio (not needed in this case). Last, to add the parameter you'll click the **Add** bar and from there select the **time** parameter. A control will appear to set the parameter value. Leave it at 0.
- Do all of these same steps (6 and 7) for the **Ambience** Event as well.
- Okay one last thing. We've configured the events but there's currently no direct hookup from the script controlling the time change to change the **time** parameter in FMOD. Let's look at that. Find the **Day Night Circle_FMOD** object and click on it. Then look on in the Inspector for the *Day-Night Cycle* script:

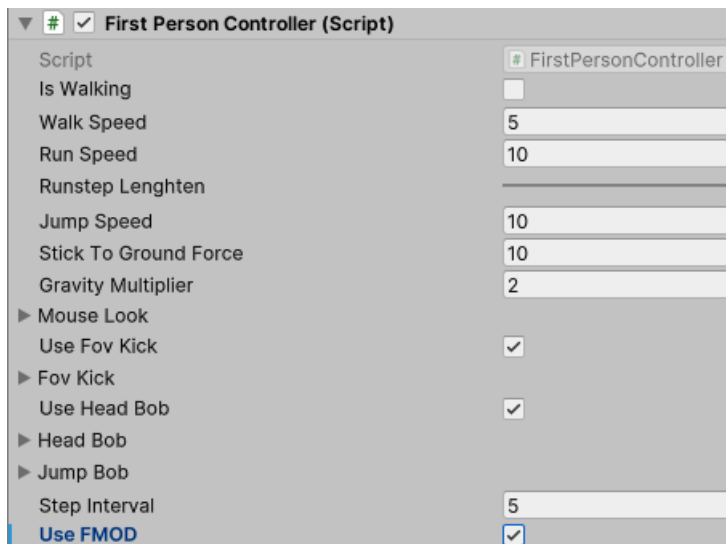


- Look at the entry labeled *Parameter Name* - What you need to do is to put the name you used in FMOD Studio for the time parameter. In this case we just called it **time** in FMOD Studio so that's what we'll enter.



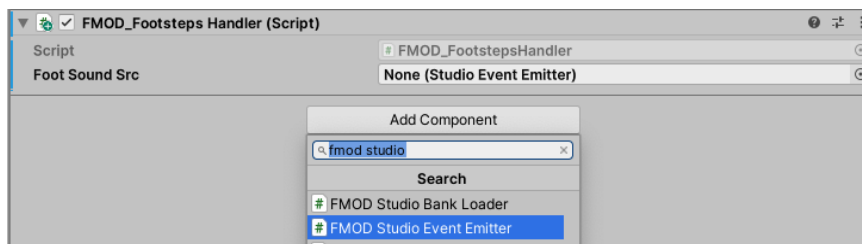
Integrating the Footstep Event

To integrate this Event, look for the **First Person Controller** object in the scene and examine the First Person Controller component:

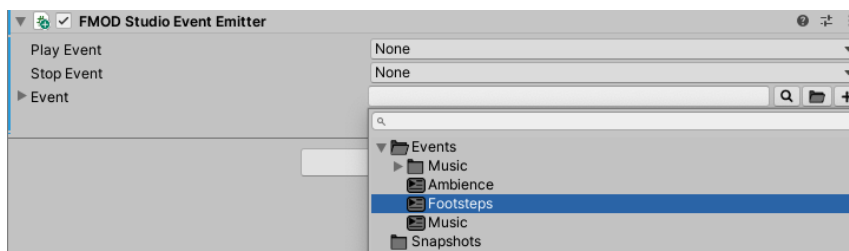


All we really need to do here is make sure the **Use FMOD** Boolean has been enabled here. This means that the foot actions that are normally wired to Unity Audio Sources and Audio Clips will now be handled by the FMOD Studio integration instead.

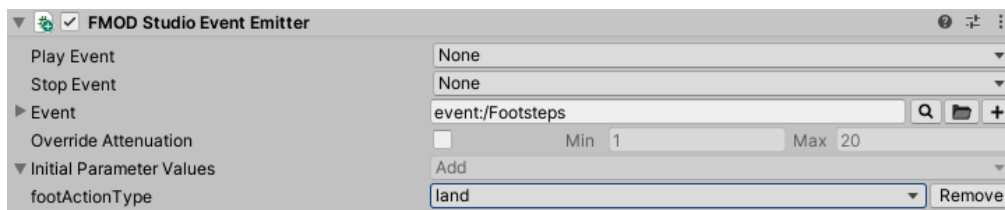
Next we need to examine the script below this, called **FMOD_FootstepsHandler**. This script, like its name suggests, handles all of the foot-related sounds in the game. The only variable it needs to be configured is to add a Studio Event Emitter Component. So for that you'll need to add this via the Add Component button and menu like so:



On the Emitter component you'll search for the **Footsteps** Event (or whatever you named it) and add that to the Event field.



Once that is set you'll have to make sure to initialize parameter values. If you don't do this there's a good chance the Emitter will not change the parameter, even if the script is telling it to do so. In this case i'm also setting **footActionType** parameter to **land** as that's the first sound that will be heard since the player drops in the scene initially.



Ready To Test

Okay, we should be ready to go at this point. Save what you have and give the scene a playback. You should hear something at this point. If you have any difficulties, go to the Troubleshooting section and see if your issue shows up there. If not, walk around the level and pay attention to the ambience and music and ask yourself these questions:

- Is the music and ambience lining up with sunrise and sunset, as well as the DayMusicLoops and NightMusicLoops reference events within your Music Event? Are they early? Are they too late?
- Is the transition between the end of the day and the beginning of the next (remember that's at midnight or 0.0, not at sunrise) appropriately smooth?

Other things you might want to work on is the balance of the volume of your events and the smoothness of your timing of the music over the entire day cycle.

To check what time of day is occurring during playback, make sure you have not maximized your Scene View on playback and check the Inspector on the right. Look at the **Time In Hours** value. It should be moving continuously while playing back the scene. When it gets to the end of the 24 (23.99 really) hour cycle it will restart at 0.0. Pay attention to the smoothness of that transition. Does the end of the Night music get cut off prematurely?

In general modifications should be done in FMOD Studio unless the timing is obviously either mostly too fast or too slow. The .20 Seek time in FMOD when we tested is a little bit faster than the speed of the day cycle in Unity, but it's pretty close. So if some events are lining up but others are not it's a problem to fix in FMOD. But if everything seems to happen too early in comparison to the day cycle that could be addressed in Unity, using the **Speed of Day** value at the bottom. This will not require a lot of adjusting to hear fairly substantial differences. It's currently set at .80, so setting it to .67 will make the day cycle longer, and setting it to .87 will make it shorter. Currently .80 yields just about a 2 minute day cycle.

Updating Your Event In FMOD

Since it's very likely that you may overlook something in your first pass, or you need to revise your music or ambience, we need to talk about how you iterate between FMOD and Unity. The first thing to understand is that updating your FMOD Studio project itself will do nothing. Unity is only interested in updating the Event through a revised bank build. So when you make changes in FMOD you should save your project first, but then you must go to the **File** menu and choose **Build**. At that point FMOD will detect a change in the configuration and the new bank will be automatically copied to the StreamingAssets folder. You can keep editing and updating your project until you are satisfied with the result.

Resetting your Project Connection

If you move the project folder to another location on your computer or to another disk or another computer (certainly not unheard of in a school environment) you will have to reconnect to the Build folder in your FMOD Studio project because you have changed the path.

Troubleshooting

Q. Help - I'm getting errors! No sound happens and a lot of Null Reference errors are printed to the Console.

Ans 1. Check your Main Camera object to be sure that the FMOD Studio Listener Component has been added to it. You'll definitely get errors otherwise.

Ans 2. Check that your events are correctly associated in the main Day Night Circle_FMOD object.

Ans 3. Check that your parameter name is totally consistent with your FMOD project

Q. I'm not hearing one or more of my events

Ans1. Make sure that all of these events in FMOD are 2D Events. You must delete the 3D Panner module from these events to make sure they will play in 2D (Page 6)

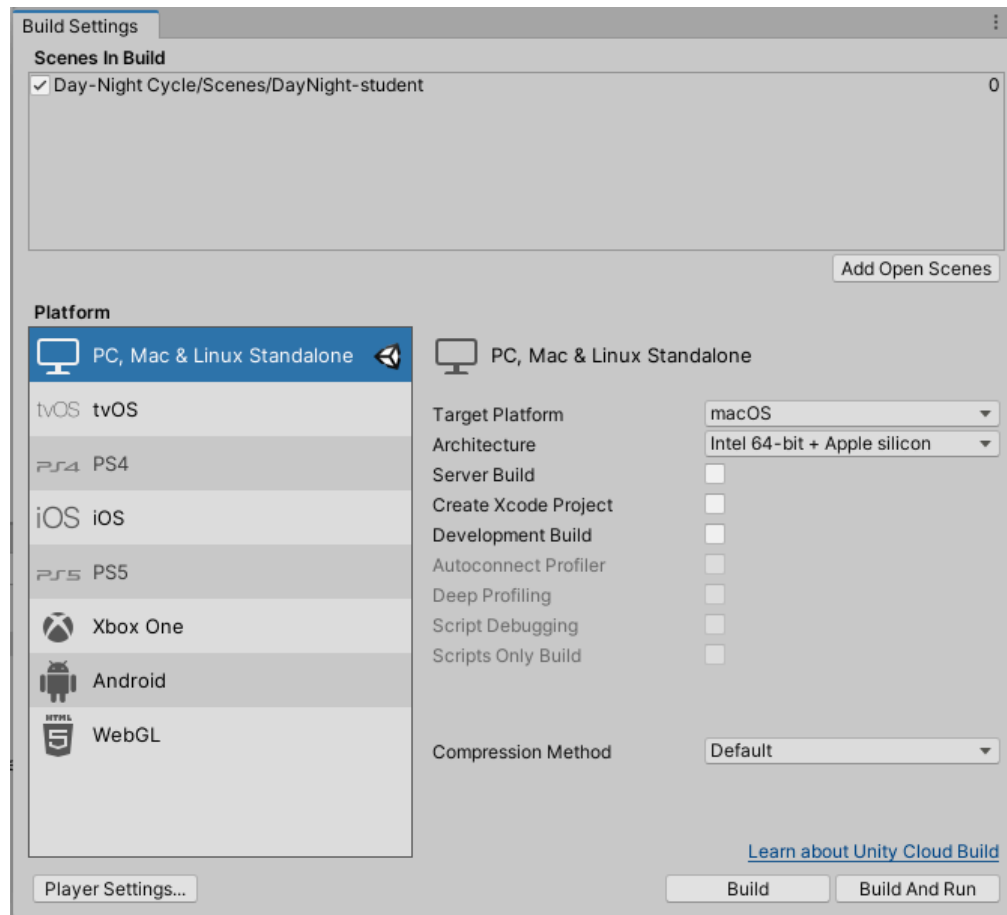
Okay, once you have successfully overcome any obstacles and have your Events properly triggered and synced with the parameters it's time to create a build of the project.

Creating a Standalone Build Of Your Game

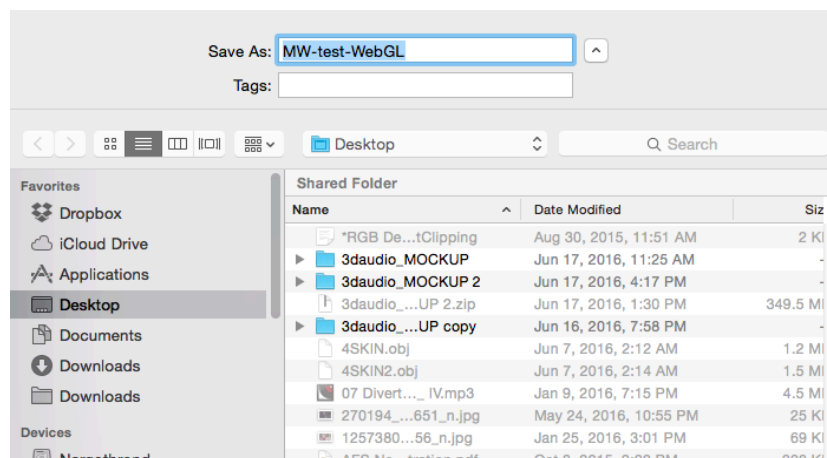
Let's talk briefly about building a game application you can submit. Unity can build a Standalone version so that you can play it as a complete application that doesn't require Unity.

By default Unity should install Standalone support (Mac/PC/Linux). This will be specific to your platform, so if you're on a Mac it will be an app file and if you're on a PC it will be a folder with an exe file in it. Follow these steps to get a Standalone build working:

1. Go to the File menu and choose Build Settings.



2. Click the Add Open Scenes button to add the current scene to the build. This is vital.
3. Click the Build button at the bottom of the window. You will get a dialog to save your file within the Project folder. Do NOT save the build in the project folder! This can cause issues in your project. Use something else like the Desktop or your Documents folder. Type a name for your build, and click Save.



4. For a Mac the result will be a .app application. For Windows, it will be a folder with several files in it including the .EXE executable file which will run it.

5. Once you click **Save**, the build will commence. Once the build is complete you will find an executable application or folder at that location. On Mac, test this out by double clicking on the app file. On Windows look inside the build folder for the .EXE file and doubleclick that to open the application.
6. Select whatever choices you wish and then click the Play! button on the lower right. If all has gone well the experience should be similar to what you encountered when playing it in the Editor. To quit the application simply use Cmd-Q (Mac) or Alt-F4(Windows).

That's it! We hope you've enjoyed working on this level and getting to know a bit about how to configure and implement adaptive music and sound events based on a parameter in FMOD Studio and Unity. Stay tuned for more exciting products from the Game Audio Institute by going to our webpage and subscribing to our newsletter at gameaudioinstitute.com, or you can visit our Facebook page as well.