

**Andre Steed
(2017852)**

Development

Element 2b

3724 Words

As a lover of video games and video game music, I have spent time in my past dabbling in the creation of both. So I had a lot of ideas when it came to starting this project. I considered many interesting ways I could marry music and gameplay together including rhymed based mechanics and a puzzle based on sound. I made attempts on both, however I found implementing them into a workable project given the time constraints to be not something within the scope of my ability. Instead I came up with the idea of a shooter where different guns manipulate sound using effects. So with my idea set I decided to define the different parts I would need to make and began to plan my game.

Key Goals

Once I had settled on an idea I decided to start putting together the logic of how things would work. The end result was to be a game where different guns have different powers, represented by sound effects such as Reverb and Delay, with different enemies requiring different guns to destroy. With that in mind I set out a few key goals that I would need to achieve in order to bring my vision to reality. These goals were as follows;

- Create a functioning First Person Shooter environment
- Design different sound elements to represent different actions within the game
- Create a system for triggering music tracks and sound effects

As I moved through these goals and my project, problems arose that changed my existing goals and also created new ones. I found as I got on, my idea evolved to work around the challenges I faced and the product I ended up with was similar, but not the same as my initial idea.

Building the Game

So as mentioned, the initial idea was a shooter with guns causing effects. So I needed to complete my first goal of creating the First Person Shooter environment. As I had never done this before I began researching on Youtube to see if I could find a guide to follow to create this simple environment for me to build up. I ended up finding a very in depth guide [insert Youtube reference] that allowed me to follow along in building my game and also understand the basics of coding within Unity. Thankfully, I am familiar with the basics of coding for games, having made a few games in my past using Game Maker Studio. I then moved to the Unity Asset Store to find assets in order to build my world. I decided on a space theme after finding some guns, robots, buildings and skies on the asset store and began building my world.

The theme of space defined my approach to composing the music for this game as music is a key part of a video game. I wanted to create a feeling of the future as space is often associated with such. In order to do this I decided to make the music synth heavy and less reliant on melody and more on the design of the sound. Whilst running into complications within my effect triggering system, which I will address later on in this essay, my idea of how the game was structured had changed. Initially, the game would be one big level with different guns being picked up to change the effects being triggered. This then moved to different levels where the player

begins with a different gun each level, with a different effect. With that in mind I had to compose different tracks for each level, each representing the effect that the gun produces.

The first level being the initial level has no effects. It serves somewhat as the control to get the player used to walking and shooting and not being distracted by the different effects. Most initial levels in games are usually the happiest sounding and are there to set the tone of the adventure of the game. As such I composed a simple, happy but futuristic sounding track.

The second level was the level that was based on the delay effect, so I reflected this in the composition. I used a slower tempo and gave the drums a half tempo feel. This was so I could use delay on the kick and snare without the composition ending up too crowded. The melody also features arpeggiated chords, also featuring a delay to them to further immerse the player with this delay world.

Level 3 was the reverb layer so I focused the composition on creating space. I wanted the track to feel bigger than it was by using reverbs of different sizes. This also ended up creating a bit of an unnerving sound which inspired my menu music, which I will get onto shortly. I created two versions of this track and discarded by original idea of using different reverbs and created a simpler version of the idea, limiting the tracks that had reverb on them.

Level 4 was where I took the theme of space and really ran with it as Level 4 was the level actually set on a different planet. To move away from the composition from the moment, this level featured a different gravity setting to the rest to reflect that the world was different. The music was more intense than the others as this was the final level and the climax of the players adventure. I also wanted to emulate being in a spacesuit with the sound so I had a challenge to create a track that would have enough going on that when I applied the high and low pass filters to emulate hearing a sound from outside a helmet, the music would not become plain and uneventful.

As my game evolved the feel I wanted the player to have changed. So the menu music needed to reflect the overall feeling of the game. Being in an unnatural environment, fighting enemies is unnerving so the menu music was made to reflect this. Multiple melodies that would clash and not resolve, along with a 6 bar loop which is pretty untraditional for most song structures allowed me to achieve this.

For each track I also created a more intense loop to be played when the player was spotted by an enemy in order to bring a sense of fear and urgency as "Music is best when it's not intrusive, but complements the gameplay. Sound effects help associate your brain with actions or threats, and gameplay would be difficult without these auditory cues." (Singh, 2018) So with all this created, I bounced each track down as loops of their different components ready for FMOD.

Designing the Sounds

After composing the music I moved onto sound effects. I could have used a sound library but I decided to create all the sounds myself. I wanted full control of exactly how every interaction felt and I could not do that using sounds I had not created. I created most of my sounds through granular synthesis using a plugin called Pigments 3 by Arturia. The footsteps were created through sampling a kick and a snare loop through the synthesizer in order to trigger pitch and speed of the sample. This allowed me to create the two main components of a footstep which is the heel touching the floor followed by the toe. I did a similar process to create the landing sound. The jump and death sounds were both created by me manipulating my own voice, shifting the pitch and stretching the audio file. The shot sound was created by feeding a simple sine wave through Ableton's sampler. I set a loop region and automated the length of the loop and the pitch of the sample in order to create my effect. I did this as I wanted my shot to sound more like a laser than a bullet and after analysing laser sounds through science fiction media I figured this was the best way to achieve that sound.

When creating the robot sounds I wanted a sound that was like an idle machine operating. So I turned to granular synthesis again and created a sound first through a wavetable synth which I then fed back through the synthesizer. Using granular synthesis I was able to create a looping portion of the original sound that had fast pitch modulation. I created a few beeps and bleeps very much inspired by R2-D2 from Star Wars and a separate shot sound from the players shooting noise.

My crates I wanted to have sounds that give a feeling of relief as they would replenish health and ammo. So I decided to use Maj7 chords for the main basis of the sound and built some synths that would sound warm and jubilant. From there I had all the sounds I needed so I moved on to programming them within FMOD.

Building the FMOD project

This was the first time I had ever used FMOD so before I started I read through [Ciaran's book] and highlighted the main chapters that would be useful for my creation. After I had a better idea of how the software worked I set about creating my first instance of my FMOD project. I started with the sound effects as they seemed to be the simplest to create. I decided to separate my player events into two with all the sounds I would not be manipulating in one folder (Footsteps, jump, land, death) with the shoot sound separate from these. I did this so I could set up sends to my effects from the group as outlined in the book and only the shoot sound would be using effects. I had realised that I had not made all these sounds level in my DAW so I had to adjust volumes within FMOD to balance things out.

With the robot sound I put the operating sound and the bleeps/bleeps into one event following instructions outlined in *Game Audio with FMOD and Unity* (Ciarán Robinson, 2019). By making multiple bleeps/bleeps I was able to make it feel like each robot was unique from the other as the random generation of those sounds would result in different sound patterns from different

robots. The shoot and death noises of the robot had their own separate events and since none of these noises would be affected individually by the level, I could group them all together. The crate sounds were also simple creating separate events for each sound with the sound that would emit from the crate object to signify it was active in a loop region so it would always play as long as the loop exists.

Doing the music for the level was much more interesting. I had debated in my head multiple ways I could structure these events. My goal was to, through my loops, create a track that had parts coming in and out at random points to make a constantly changing track that wouldn't feel stale. Initially I had each instrument that played throughout (main melody/bass/drums) in separate audio tracks with the elements that would come in and out randomly in one track. This track would then play one of the loops randomly each time the last loop ended.

I found with that I didn't get enough variation as it only meant one thing would be different each time which can feel repetitive. So to combat that I took some of those loops outside the instrument and put them in their own audio track. Then by using the probability function I was able to have these parts play or not play based on the percentage set. This meant instead of having one random element change I could have multiple random elements change. I then set the intense loop that was meant to play when the enemy spots the player within its own separate track with the intention of having an automation I could trigger to unmute it within Unity.

After a few times of listening back to the tracks I found it didn't feel like the song was changing randomly and certain sounds would always seem to play together. This reminded me of an article regarding Spotify's shuffle algorithm "If you just heard a song from a particular artist, that doesn't mean that the next song will be more likely from a different artist in a perfectly random order...they don't like perfect randomness."(Poláček, 2014). So from this I decided to not let these loops play in a true random fashion and instead manipulate the play percentage so that certain parts would play more often than others to create a random that the human brain could better resonate with.

After loading all my songs I went to the mixer and organised my sounds into the following signal flows.

- Level Music > Master
- Player Sounds > Master
- Shoot Sounds > FX Bus > Master
- Shoot Sounds > Master
- Robot Sounds > Master
- Crate Sounds > Master

I put my Reverb, Delay and EQ on the FX Bus with the idea of muting and unmuting each effect depending on the level loaded. I figured that this configuration was the simplest and easiest method I could use to achieve my goals.

Implementing the sounds into the game

Once I had FMOD sorted I imported it into Unity and let it take over all audio. I added all the player effect one shots into the First Person Controller Script and put the level music inside the script I made for the actual player character. Everything was working perfectly fine until I tried to have the intense loop trigger. Since I had no way to define, within the player script, that the enemy had spotted me I had to create a public state within the enemy that the player instance could call upon to check whether or not to trigger this loop to unmute or stay muted. So to make things easier I decided to create a game object and a separate script that would be in charge of controlling the level music and most of my FMOD triggers. This however, broke the music within the game.

After digging through the errors within Unity, I found that issue was due to my object not having the correct 3D properties to host a 3D Timeline event in it. I tried for a few hours to create the necessary properties with no avail so I decided to resolve the issue from the FMOD side of things. By creating a 2D Timeline within FMOD I was able to get my music working properly again. This problem was a good one for me as it allowed me to understand better when a 3D Timeline is needed versus a 2D one and that anything that is not coming from a specific emitter and isn't affected by the players position, could be a 2D Timeline. Instead of creating 2D Timelines for the rest I was able to convert them from 3D to 2D by getting rid of the one element in my Timeline that made it 3D. According to a post on the FMOD forums "The event has a Spatializer, 3D Object Spatializer, or a 3rd party spatializer on its master track." (FMOD Forums, 2019)

After spending some more time trying to get the intense music to trigger I decided to shelf the idea for now and work towards getting everything else working. The next task at hand was to get the enemy sounds to deploy. After doing some research I found the FMOD documentation (Fmod.com, 2021a) and the solution was to use Studio Event Emitter Script from FMOD to get the idle sounds to play, using the Object Enable and Object Disable parameters to start and stop the music. However, I found that this did not work. I wasn't too sure why as my enemy code was set to disable the enemy upon it's death but it seemed that didn't translate well over to FMOD. So I modified the code to destroy the object after a second and changed the Event Emitter to Object Start and Object destroy. Using this same method I was able to trigger the death sound using another Event Emitter and the Trigger Enter/Trigger Exit parameters. I thought that as the enemy dies after one bullet, that this would trigger the death. What I did not predict was that this would also trigger every time the player collided with the enemy resulting in the death noise playing whilst the enemy was still alive. This was not ideal so I went back to the documentation for FMOD. Within this documentation I found scripting examples (Fmod.com, 2021c) and the PlayOneShot command. I had used this command before but just copied and pasted some code without understanding it fully. Reading this through made it clearer to me that I could also use this command to trigger any event I needed and was able to make the enemy bullet sound and death sound work perfectly. Using everything I had learned before, the sounds for the crate became simple.

With the basic sounds created it was time to move onto the effects. My initial idea was to have the FX bus have effects that would be muted and unmuted based on the level loaded. This idea was scrapped when I found that there was no easy way to do this with a script. I had to find a new way to achieve this. I changed the routing by splitting each effect into its own Bus. This idea came about after finding a thread on the FMOD forums (FMOD Forums, 2016) which showed me that I could control Bus volume as well as mute and unmute busses individually. My new shoot sound signal routing had now changed to this.

- Shoot Sound > Master
- Shoot Sound > Delay Bus
- Shoot Sound > Reverb Bus

By simply declaring within my level music object if the correct effect bus was muted or unmuted, I was able to achieve the desired effect. The next challenge came with my Level 4 effect which was an EQ over everything to give the feel of hearing all sounds through a spacesuit helmet. Since I already knew I could not easily mute and unmute effects on FMOD I couldn't put all sounds in a group and nest the effect on there. Also, since sends are parallel processes, I couldn't use the same method as before as I would just be adding the affected bus on top of the unaffected sounds which would not create my effect. So I tried grouping all sounds and then adding a pre fader send to an effect bus with the EQ on it. I then added some code to turn the fader all the way down. The logic behind this was that if the send came before the fader, then the send would still have a signal going to the effect bus if the fader to the master was down. This unfortunately did not work. I found no sound would go to the send or the master and I was left with silence. Whilst technically that would be correct as in an airless environment such as space, sound would not travel. However, this would break the immersion of the game and the player would feel disconnected.

I was stuck on this issue for a while as I did not know exactly what to search for when doing research. I ended up stumbling across an article discussing Parameters within FMOD which states "FMOD Parameters are useful for making changes to FMOD events or snapshots using game variables. If we want to change our music dynamically or link sounds to certain variables: the use of parameters is important" (Famà, 2019) This sparked a new idea within me. What if I created an automated parameter on the frequency of the high pass and low pass filters and then used my level music game object to control this. So I did just that. After expanding my knowledge on parameters by reading the FMOD documentation (Fmod.com, 2021b) I grouped all the tracks together and threw an EQ on the bus for the group. I also created a new game object within my scene and attached the Set Global Parameter Script to it and was able to turn off the EQ parameter off for each scene that didn't need it and on exclusively for the Level 4 scene. Once this was done I needed to do some small tidying up and mixing and my game was functional.

Learnings and future

As I finished my project and looked back on it I have figured out some things I could have done differently. A way I could have made my effects work within each level is by creating a pickup which changes the gun my player holds. This pickup could set a public variable within my player script which is read by my level music object. The variable within the object would then dictate which bus to mute or what to set the global parameter level to.

I also had a few more ideas for triggering the intense music. I could have created three groups for my level music. One for the regular music, one for the intense loop and one which contained both. Since they were all set to the same tempo, if I started them at the same time within my level music object, they would be synched. I could then try one of two things. The first idea would be to set the intense loop to mute within the level music object and then have some code within the player script that referenced the FMOD Studio Listener inside the player object. The idea is that once the listener detects the external sound coming from the enemy, then the intense loop bus would unmute. I could see this potentially having conflict with other external sound sources in the game such as the crate sources. I would need to figure out how it would be able to differentiate between these sounds. Another conflict might be what would happen if two enemies spot me at the same time? Would the code trigger the sound to mute and unmute again in quick succession?

The second idea, and potentially the one I think I would go with would be to use a global parameter again. Placing an EQ on the intense loop bus and place a Set Global Parameter Script within the enemy object prefab. A simple line of code to be added to the update section of the enemy to set the global parameter to 0 or 1 over a period of time to fade in and out the sound as necessary.

After partaking in this project, I feel a lot more confident in my skills to use FMOD alongside Unity. I think from the lessons learned here, approaching this task in the future would produce a different result and a more complex, but tidy system to trigger music and sounds within a video game.

Reference List

Ciarán Robinson (2019). Game audio with FMOD and Unity. New York London Routledge.

Famà, A. (2019). Controlling FMOD Parameters in Unity. [online] Alessandro Famà. Available at:
https://alessandrofama.com/tutorials/fmod-unity/parameters/#Parameter_Types_in_FMOD_Studio [Accessed 18 Jul. 2021].

FMOD Forums. (2016). How do you script volume control for music and sfx buses. [online] Available at:
<https://qa.fmod.com/t/how-do-you-script-volume-control-for-music-and-sfx-buses/12885> [Accessed 18 Jul. 2021].

FMOD Forums. (2019). Convert 3D event to 2D.... simple, right? [online] Available at:
<https://qa.fmod.com/t/convert-3d-event-to-2d-simple-right/14577> [Accessed 18 Jul. 2021].

Fmod.com. (2021a). Game Components. [online] Available at:
<https://fmod.com/resources/documentation-unity?version=2.00&page=game-components.html>.

Fmod.com. (2021b). Parameters. [online] Available at:
<https://www.fmod.com/resources/documentation-studio?version=2.00&page=parameters.html> [Accessed 18 Jul. 2021].

Fmod.com. (2021c). Scripting Examples | Basic. [online] Available at:
<https://fmod.com/resources/documentation-unity?version=2.00&page=examples-basic.html>.

Poláček, L. (2014). How to shuffle songs? [online] Spotify Engineering. Available at:
<https://engineering.atspotify.com/2014/02/28/how-to-shuffle-songs/> [Accessed 18 Jul. 2021].

Singh, N. (2018). How does music affect player performance in your game? - Blue Label Labs. [online] Blue Label Labs. Available at:
<https://www.bluelabellabs.com/blog/do-users-play-better-with-music-sound-fx/>.