Final Documentation

**Introduction**

The game that I selected for this project is the Unity first person shooter microgame, which has an overall sci-fi feel to it. The goal of the project was largely to recreate the sounds that currently existed, as well as adding in a few that felt necessary.

**Analysis/Requirements**

For this project, I wanted to incorporate enough of the existing mechanics of the game (weapons, enemies, etc.) so that I could have a full experience but still have a reasonable scope. My analysis of the game largely consisted of looking at the microgame’s sound effects as well as testing out the microgame so that I could see which ones existed throughout the experience. I determined that I wanted to implement two weapons (a shotgun and a blaster), weapon handling, a jetpack, basic movement, health and health pickups, notifications, ambience, and two enemies (the hoverbot and the turret). I initially wanted to incorporate music as well, but it seemed a bit out of scope since I have limited experience with composing. I did, however, add a few sounds for the interface as they were lacking in the original microgame.

**Asset Development**

For all of the assets, I used either freesound.org or zapsplat.com and modified them in Reaper. They appear in a random order (along with their corresponding source sound entries) since I did not approach them in a given order and instead designed the sounds that I felt most prepared to do first. There was a total of 88 tracks in Reaper used to develop the sounds necessary for the events, which included the following:

* Ambient: The level was largely outdoors on a planet with lava below it, so for ambient sound it made sense to have both wind and lava.
  + Ambience: For the ambience, I edited two wind sounds and a lava sound into seamless loops to be combined in the FMOD event.
* Hoverbot: The hoverbot is a small robot that flies around until it notices the player, at which point it starts to fire at the player; since this is the weaker of the two enemies, I wanted the sounds to appear less threatening.
  + HoverbotAlert: This uses a sample sound with a singular high-pitched robot beep with echo; I used the first beep part in a separate track that fades out to make it do a double beep instead, then added an equalizer to bring out the higher frequencies.
  + HoverbotAttack: This attack is less frequent than the turret’s attack, so I used a higher pitched sci-fi gun sound and added some equalizing to shape the sound and a ReaFir effect to reduce some excess noise.
  + HoverbotDeath: For the hoverbot’s death, I used a grenade explosion sound for the detonation and some miscellaneous robot noises that gave the impression of powering down. I made sure the robot sounds were much quieter during the first portion than the second portion, and I added an equalizer on both to shape the sound.
  + HoverbotMoving: For the moving noise, I created three seamless loops of drone noises: one that used an equalizer to heavily emphasize the lower frequencies and two higher frequency loops to make the FMOD event more noticeable.
* Pickups: I only ended up having to do a health pickup since I was able to give the player all of the other resources at the start.
  + HealthPickup: I used a high pitched sound with a trail off and split it into two parts; I then used an equalizer for each part so that the initial impact had most of the frequencies at full volume, while the trail off portion was much more reduced to make it a bit less distracting.
* Player: For the player’s sound effects, I had to encompass basic movement, jumping/landing, using a jetpack, and signals for getting on low health and dying.
  + Damage: This sound effect takes a punching damage noise (somewhat like hitting a punching bag) and uses an equalizer to shape the sound a bit.
  + Death: For the death noise, I found a dying grunt that seemed to do the trick; this was probably one of the only sounds that required little to no editing.
  + Footsteps: The floor in the game appeared to be heavy and solid, kind of like concrete or a sidewalk, so I obtained a track with footsteps similar to the floor type I was aiming for, then edited a few of the best ones with an equalizer to shape the sounds a bit and pitched them down a bit as well.
  + HeavyLanding: This takes one of the normal landing sounds that I created along with some bone crunches that I modified with an equalizer and combines them for a more impactful hit.
  + Jetpack: This was actually one of the sounds I was most proud of; I took some flamethrower sounds and used an equalizer to shape them to feel like a jetpack, then created a startup sound and a couple seamless loops.
  + Jump: Jumping sounds were a bit tougher to find than landing sounds, but I managed to find a portion of one of the landing sample sounds that felt good for a jump; I then modified them using an equalizer and pitched them down a bit.
  + Landing: The easier of the jump and land combo, I found a few good sounds of landing on concrete and used an equalizer to shape the sound a bit as well as pitching them down a tad.
  + LowHealth: For signaling the player is approaching low health, I obtained a heartbeat “lub dub” sound and sped it up a bit for the main effect. I also obtained a very quick sci-fi alarm that I slowed down, pitched down, and added distortion along with an equalizer to shape the sound.
* Turret: The turret enemy type is the boss of the two. It is stationary, but when alerted will rise up and begin rapid firing lasers at the player.
  + TurretAlert: This sound actually came from a CD player being opened and closed; with an equalizer to shape the sound a bit, it felt perfect for the turret rising up into the air to prepare to attack the player.
  + TurretAttack: The turret’s attack sound had to be suited for rapid firing, so it was a bit shorter than then hoverbot’s attack sound; I also wanted to use a different sci-fi laser sound and shape the sound with an equalizer in a way that would make it feel different from the hoverbot’s laser.
  + TurretDeath: Similar to the hoverbot’s death, this effect features both an explosion and some robot death type noises that are louder towards the end of the overall sound. I wanted to make them feel a bit different, so I used different robot noises and shaped the sounds in a different way with an equalizer.
* UI: These sounds included both the notifications in game and the various effects from interacting with the menus in the game and were designed to suit the sci-fi feel of the game.
  + Click: The clicking sound is meant to be a high pitched confirmation sound that responds to the player clicking on the interface; the two versions use equalizers to shape the sounds in slightly different ways.
  + Drag: The dragging sound is meant to be a very quick, low volume clicking sound, since this can happen very frequently and is meant to accompany the player dragging the slider along. Little editing needed to be done from the sample sound; this was another one that fit very well as it was.
  + Hover: This sound is hard to describe, but its high pitch notification is meant to signal that the player is hovering over an interactable UI element. This sound also had an equalizer used to mildly shape the sound.
  + ObjectiveComplete: In a similar way to the double beep of the hoverbot’s alert, this sound effect takes a victory noise and staggers them one after the other, with the second a slightly higher pitched and an overall equalizer for shaping the sound.
  + ObjectiveNotification: This notification sound also follows a similar pattern which takes a beep sound and staggers it one right after the other, with the first one being played at a slightly faster rate and both having an equalizer to shape the sound.
* Weapon: The weapons category is probably the one I struggled with the most; I had to redo the main weapon sounds completely and adjust the sounds for the other weapon-related events a bit throughout development.
  + BlasterAttack: This sound needed to be able to rapid fire (as the player can hold the mouse button to spray and pray); it takes a blaster shot sound and uses an equalizer to shape the sound, along with chopping off some of the excess tail to give a shorter blaster noise.
  + ShotgunAttack: The assets made for the shotgun simply consist of a heavier shotgun and a lighter shotgun at a lower volume to give a bit of layering to the sound; I decided it was fine to make this one less sci-fi, but eventually adjusted it in FMOD to make it feel more sci-fi.
  + WeaponCooldown: Since the reloading is essentially the weapon(s) cooling off in this game, the asset for this effect is steam made into a seamless loop with an equalizer to shape the sound and some volume adjustments to make it more consistent (otherwise the loop would have been much more noticeable).
  + WeaponDamage: This is just a subtle “tick” sound used to indicate the player has hit an enemy; it is actually taken from a clock and sped up, distorted slightly, and shaped with an equalizer to give it the feel I was looking for.
  + WeaponSwap: This effect was made from some pistol handling pitched down an octave and shaped a bit with an equalizer to give the impression of swapping which gun the player is holding.

**FMOD Development**

A large portion of the work to create the individual sounds were made in Reaper before importing them into FMOD, which made my work in the middleware a bit easier and allowed me to spend more time focusing on the sound effects that really needed the parameterization FMOD provides. Some entries may be a bit brief if there wasn’t much significant FMOD work beyond the Reaper editing. There were 27 FMOD events total, which includes the following:

* Ambient: The background noise to give the impression of being outside above a planet covered in lava. There were no other noises like this, so it gets its own category.
  + Ambience: This 2D event contains two wind loops and one lava loop with offsetting lengths that sustain throughout all of the gameplay. The ambience has a single outside parameter, where one is equivalent to being in the outdoors areas and zero means the player is not in those areas. During the outdoors area, the volume of the wind increases slightly, and the lava volume increases significantly.
* Hoverbot: The category of the sounds pertaining to the cute little robot of death.
  + HoverbotAlert: This event slightly randomizes the pitch and volume of the alert sound; it also has a large radius on the spatializer, so it is prominent even from a distance.
  + HoverbotAttack: This event uses two versions of the laser with slightly randomized pitch and volume; it also has a large radius on the spatializer to be very noticeable.
  + HoverbotDeath: This event slightly randomizes the pitch and volume of the death sound; it also has a large radius on the spatializer as it is important for the player to notice it.
  + HoverbotMoving: This event has a much smaller radius on the spatializer with a quicker drop off as well; it loops all three drone sounds while the bot is alive with a sustain point, and has an AHDSR for a bit of release time.
* Pickups: This only ended up needing the health pickup sound, but it was distinct enough that it deserved its own category.
  + HealthPickup: This 2D event uses slight randomization of pitch and volume when played.
* Player: All of the general player related sounds that didn’t seem to fit well in their own categories were brought together into this category.
  + Damage: This 2D event uses a multi instrument with the two versions of the sound along with a slight randomization of pitch and volume.
  + Death: This 2D event uses a slight randomization of volume (since it is a voice, messing with the pitch could have had adverse effects).
  + Footsteps: This event uses a multi instrument with the three versions of the sound along with a slight randomization of pitch and volume.
  + HeavyLanding: This event uses slight randomization of pitch and volume for the one-shot sound effect.
  + Jetpack: This 2D event starts with a singular startup sound that transitions into the full loop with a sustain point; a second jetpack loops also comes in with an AHDSR for a bit of attack time, and the entire event has an AHDSR for a bit of release time.
  + Jump: This event uses a multi instrument for the two versions of the jump sound along with a slight randomization of pitch and volume.
  + Landing: This event uses a multi instrument for the two versions of the landing sound along with a slight randomization of pitch and volume.
  + LowHealth: This 2D event plays persistently throughout the game and uses the health parameter to determine which sounds to play (if any). Once the player is at half health, a scatterer instrument plays one of the two heartbeat sounds with a slight offset for the spawn interval, no scatter distance, and slight pitch and volume randomization. As the health decreases, the volume and spawn rate of the heartbeat increase to a certain degree to make it faster and more intense. Once the player reaches thirty percent, the warning loop begins playing and increases in volume as the health decreases. The heartbeat has an AHDSR for a release time only, while the warning alarm has an AHDSR for both attack and release time.
* Turret: The sounds relating to the big bad boss turret, similar to the organization for the hoverbot.
  + TurretAlert: This event has an exceptionally large radius on the spatializer and plays the sound with a slight randomization in pitch and volume.
  + TurretAttack: This event has an exceptionally large radius on the spatializer and uses a multi instrument to play the two laser sounds with a slight randomization in pitch and volume.
  + TurretDeath: This event has an exceptionally large radius on the spatializer and plays the sound effect with a slight randomization in pitch and volume.
* UI: All of the interface sounds for the game, including both alerts and those triggered by the player’s interactions.
  + Click: This 2D event uses a multi instrument to play the two versions of the sound with a slight randomization in pitch and volume.
  + Drag: This 2D event uses a multi instrument to play the two versions of the sound with a slight randomization in pitch and volume; it also limits the number of sounds played to 7 so that Unity cannot trigger an excessive number of sounds at once.
  + Hover: This 2D event uses a multi instrument to play the two versions of the sound with a slight randomization in pitch and volume.
  + ObjectiveComplete: This 2D event plays the sound with a slight randomization in volume as well as a slight delay to account for the explosion that will be happening of defeating the final enemy.
  + ObjectiveNotification: This 2D event plays the sound with a slight randomization in pitch and volume.
* Weapon: All of the sounds related to the weapon, both including the firing of the weapons and the handling of them.
  + BlasterAttack: This event uses a multi instrument to play the two versions of the sound with a slight randomization in pitch and a bigger randomization in volume (since it will be playing very frequently).
  + ShotgunAttack: This event uses a multi instrument to play the two versions of the sound with a slight randomization in pitch and volume. It also plays the BlasterAttack event with a lower volume to give the weapon more of a sci-fi feel.
  + WeaponCooldown: This 2D event plays the loop with a slight randomization in pitch and volume, with an AHDSR for a slight release time. It also adds an AmmoRatio parameter that allows for the reducing of volume slightly and increasing of pitch slightly to give a satisfying ending hiss when completely reloaded.
  + WeaponDamage: This 2D event uses a multi instrument to play the two versions of the sound with a slight randomization in pitch and volume.
  + WeaponSwap: This 2D event uses a multi instrument to play the two versions of the sound with a slight randomization in pitch and volume.

**Mixing Work**

For mixing in this project, the sound effects were all split into three major groups: Ambient, SFX, and UI. Ambient included only the Ambience event, UI included all of the events in the UI category, and the rest of the events were a part of the SFX group. These groups were used in the PauseMenu snapshot to properly fade out the unnecessary sounds. When the player presses the tab button to open the menu mid-game, the volume of the SFX group drops significantly and the volume of the Ambient group drops somewhat. This allows the UI events to come through much more clearly and makes the ambience a bit of a more subtle background noise. When the player resumes, the snapshot turns off and the sounds return to their normal volume. Beyond this major grouping system, the adjustments to get the audio to mix well together were largely an active process throughout the development of the project; finishing touches were done when mastering the project.

**Mastering Work**

An original analysis of the game showed that it had -11.8 LUFS, and the maximum peak was 0.2 dB; this also included a lot of warnings showing up in the loudness meter used for analysis, indicating that it seemed to be nearing peaking quite often. Before going to work on mastering, I decided that my reference game would be Apex Legends, another first person shooter game which I play quite regularly. When turning the volume of Apex up to 100%, I found it to be extraordinarily loud; it had -12.0 LUFS, and it reached a maximum peak of 3.1 dB (largely due to a lot of grenades and gunfire). When I reduced it to my normal playing volume, it was much more bearable with -18.4 LUFS and a -0.7 dB maximum peak. I decided to try to get my LUFS somewhere near the -14 to -16 range and get the maximum peak down so that it was not going to peak at all. After further analyzing my footage, I found that a few sounds were overall too loud, particularly the shotgun and the ambience. After bringing those down, the loudness meter showed much better results, but it was quieter than I had hoped for. I played around with the volumes of different events and raised a few that seemed a tad quiet, then went back and analyzed it in the loudness meter; I continued this loop of analyzing the footage and adjusting until I was happy with the end result. The final result was -16.1 LUFS with a -0.4 dB maximum peak. This still has a bit of room for improvement, and there was a lot of variance due to the different actions taking place each playthrough of the game, but it reduced the warnings to only a few moments at most and felt a lot better overall.

**Final Notes**

Overall, I think that this project turned out really well and I was happy with the end result. Apologies that the documentation is so long; I went to write just a little bit about each event and did not realize how much space it would take up. The sources for the sounds will be available in a separate document.