GitHub Repo Link: <https://github.com/txa7348/671-FinalProject/>

The game I designed audio for is a small prototype I made called Azrael’s Journey. Azrael’s Journey is a top-down adventure game based on The Legend of Zelda for the NES. The player has several different abilities, including a melee attack, ranged attack, charge attack, and block. Their goal is to progress through the dungeon, defeating enemies and grabbing pickups to survive. When the player enters a new room, the doors stay locked until they achieve a specific unlock condition, like defeating all the enemies or pressing all the switches. Some doors will require a key to unlock, which the player must find somewhere else in the dungeon. If the player is able to reach the end of the dungeon and defeat the boss located there, they will win the game. To give the game somewhat immersive audio, a lot of sounds needed to be designed for it. Each of the game’s main states required its own music, including the title screen, dungeon, boss fight, win screen, and loss screen. The game also needed interface sounds for its menus to reinforce the player’s actions for things like changing or confirming their current selection. In addition to these sounds, the game needed effects for nearly everything that happens during actual gameplay. These sound effects are used to inform the player that something important has happened, including whenever the player moves, attacks, takes damage, grabs a pickup, or presses a switch. The game needed similar sounds for its enemies, which tell the player when an enemy attacks, takes damage, or is defeated. It also needed environment sounds to tell the player something has happened, like when a room’s doors open or key is spawned, as well as indicators for whether the player has won or lost the game.

Once I identified the sounds I needed to make, I sorted them into groups based on whether the sound was an interface sound, sound effect, or music, with sound effects being further sorted by the source of the sound effect (player, enemy, or game). I started designing the game’s interface sounds, which included the sounds for changing or confirming an option in the menu, as well as a heartbeat noise to inform the player when their health is low. The menu sounds originally consisted or electronic pings to match the 8-bit aesthetic of the game, however I later remade these sounds to use wooden clicks instead so they would better fit the game’s more realistic audio. In FMOD, these sounds used a simple single instrument to play the sounds as one-shots in-game. For the ‘low health’ sound, I split heartbeat sounds into their two parts (called the systole and diastole), then layered the matching parts of each beat. In FMOD, I used a scatterer instrument with the beats and a silence instrument to simulate an actual heartbeat, then added automation to increase the volume and spawn rate of the instrument as the player loses health. I also added a snapshot instrument that lowers the volume of the music and sound effects and limits their higher frequencies using 3-EQ, increasing the effect the less health the player has. This automation is all controlled through a “PlayerHealth” parameter, which is controlled by the Player script in Unity and represents the percent of health the player has left.

After making the interface sounds, I began designing the game’s sound effects. Starting with the player, I made several ‘attack’ sounds, including a melee attack using a whoosh and metal swinging noise, a ranged attack using bow and arrow noises, and a charge attack using a slower and lower-pitch version of the melee attack sound. For the player’s ‘block’ effect, I originally created a metal-clinking sound that represented something hitting a shield. However, since the player uses a set of magic scales to block and not a shield, I remade the effect by layering several different “magic” sounds. I also made a key pickup effect for the player that used the sound of clinking coins, and a potion pickup effect that used the sound of someone drinking. Since all of these player effects consisted of only one audio file, I added them in FMOD as a single instrument to their respective events. The rest of the player’s sounds were more complex though. I created the ‘take damage’ sound by taking several grunting noises and adjusting their pitch and formants using ReaPitch. In FMOD, I added the grunts to a multi instrument, so a different groan would be triggered each time the player was damaged. I used a similar process to make a ‘jar breaking’ sound, changing the pitch of the ceramic-smashing sounds and adding them to a multi instrument. In Reaper, I designed the player’s ‘walk’ sound the same way I made these previous sounds, but edited it differently in FMOD. After isolating several footprints, I added them to a scatterer instrument in FMOD. I set the scatter distance to 0 so the footsteps would all be played next to the listener, then adjusted the spawn interval until the footsteps felt accurate. Rather than playing the sound as a one-shot, I added a loop region around the instrument to allow it to continue playing as long as the player was moving. For almost every ‘player’ sound effect, I added random pitch modulation to help each instance of the sound feel distinct and lifelike, since the player would trigger these sounds repeatedly over the course of the game.

After finishing the player’s sounds, I worked on creating sound effects for the enemies. These were made almost identically to the player’s sounds, with the content of the sounds being specific to each type of enemy. For each enemy’s ‘take damage’ sound, I used horse neighs, deer roars, orc grunts, or bull bellows depending on what creature the enemy was based on. After the sounds were imported to FMOD, I assigned them to a scatterer instrument in their respective event and performed the same steps I did for the player’s ‘take damage’ event. For the enemies’ ‘defeated’ sounds, I created an additional roar or cry not used in their ‘take damage’ sounds and assigned them to single instruments in their events. For several of these ‘take damage’ and ‘defeated’ sounds, I needed to go back and change the pan on them, since the original sound effects sounded too close to one side and didn’t match the pan on the rest of the sounds. Since only two enemy types used attacks that didn’t involve contact damage, only two of them needed attack sounds designed for them. For one enemy, I found a simple fire-whooshing noise and used ReaEQ to lower the higher range of frequencies, since the enemy physically throws their projectile. For the other enemy, who spits several fireballs, I combined a monster-exhale sound with a different fire-whooshing noise. I added each attack sound to a single instrument in their respective enemy’s event, then added random pitch modulation to all the sound effects for each enemy to make each instance seem unique like I did for the player.

Once all the character sounds were done, I started creating the effects that occur during gameplay but aren’t directly caused by a player or enemy. I started with the ‘door open’ effect, which involved isolating two parts of a “cell door” sound and changing the sound’s pitch to decrease over the course of the sound. I used this sound effect as a base for the ‘switch pressed’ sound, which ended up as half the ‘door open’ sound at a higher pitch. I also made a ‘key spawn’ effect using the sound of someone dropping their keys on a ceramic floor. In FMOD, each of these sounds was added to a single instrument in their respective event, like most of the game’s other one-shot sounds. Once these simpler sounds were done, I designed a ‘room transition’ sound, which used the same footstep sounds as the player’s ‘walk’ sound but with different settings in FMOD. The event still used a scatterer instrument and loop region, but instead of shuffling the clips, it played them in order which gives the event a pattern and makes it distinct from the player’s ‘walk’ event. I also added a snapshot instrument to the effect that mutes all player and enemy sound effects, preventing unwanted sounds from playing when changing rooms. Last, I created ‘victory’ and defeat’ sounds that play after the game ends, however, I later decided to group these as interface sounds since they don’t occur during the main level of the game. The victory event was created by raising the volume of the lower frequencies of an “achievement fanfare” sound using ReaEQ, and the defeat event was created by layering several “magic spell” sounds that included things like small explosions, fire crackling, and heartbeats. In FMOD, both sounds were added as single instruments to their respective events like the other one-shots.

After finishing the game’s interface sounds and sound effects, I worked on its music. The game’s title music ended up being an adventurous song that is lighter and slower. Its main dungeon music is a deeper, underground-themed song, with its boss theme being a strong and dramatic orchestral song . The victory and defeat music used similar music sources, consisting of slower, ambient-sounding music. For each music track, I split the songs into their intro and melodies, then added each song fragment to a separate track in FMOD. After deciding which parts of the song should loop, I used destination and transition markers to move between different parts of the track. Since the victory and defeat music were fairly ambient and didn’t have quick tone shifts, I used transition timelines to gradually move between different parts of the songs, while the title and dungeon music changed instantaneously. The boss music was unique in that its melody changes depending on the value of a parameter. This parameter, called “BossHealth”, records the percent of health the boss has left, with song transitioning to become more or less intense depending on how much health the boss has left.

Once I finished creating the base sounds for the game, I was able to focus on the game’s final mixing and balancing. I had been balancing the sounds somewhat as I made them, but I wanted to take the time at the end of the project to make sure that the sounds were as polished and cohesive as possible. I did this by using FMOD’s Live Update feature to test the levels of each sound group and adjust their volumes to help them fit together better. This was vital in balancing the music and sound effects, since the music would sometimes drown out certain effects. By the end, I had lowered the music group’s volume by 8.5 dB and the sound effect group’s volume by 4 dB, which allowed the groups to play together without blocking each other or being too loud. I also spent a lot of time balancing the individual sound effects with each other, since some sounds were too loud and drowned out other effects just like the music had.

After the relative volumes of the sounds were set, I worked on mastering and setting the game’s absolute volume. I started by reviewing the loudness standard for TV broadcasts in the US, which use an integrated loudness of -24.0 (+/-2) LUFS and a true peak max of -2 dB. I recorded gameplay from similar games that are similar to Azrael’s Journey, which included The Legend of Zelda (the game Azrael’s Journey is based on), Crypt of the NecroDancer (a rhythm-based roguelike), and The Binding of Issac (an exploration-focused roguelike). After measuring the loudness of each game with the Youlean Loudness Meter, I found the games’ loudness values were as follows:

|  |  |  |
| --- | --- | --- |
| Game | Integrated Loudness (LUFS) | True Peak Max (dB) |
| The Legend of Zelda | -23.7 | -10.7 |
| Crypt of the Necrodancer | -18.6 | -0.7 |
| The Binding of Isaac | -17.5 | +0.6 |

After analyzing these values, I decided that the integrated loudness of Azrael’s Journey should be around -22 to -24 LUFS, since I wanted its loudness to be close to the game it was based on, but a little louder like the more recent games. I decided that the game’s true peak max should be around -3.6 dB, which is the average value of all three games true peak maxes. I chose to go with this average because the values were very spread out among the three games and this average was relatively close to the US’s recommended true peak max of -2 dB. Once I had a target for the game’s loudness, I tested its current loudness using FMOD’s built in meter, which recorded an integrated loudness of -20.4 LUFS and a true peak max of 8 dB. This meant the integrated loudness of the game was a little higher than it should be, but the true peak max was much too high. This was confusing to me, since the 8 dB peaks were always recorded when the ‘key spawn’ or ‘pickup key’ played, despite the sounds actually being quiet compared to other sounds effects. I tried editing the sounds in both Reaper and FMOD to allow them to be both loud and clear enough without spiking the loudness meter, and was eventually able to reach a decent balance between the two. While fixing the key sounds, I also took the time to rebalance the game’s music, which was still covering certain sound effects and causing the game’s integrated loudness to be higher than I wanted. After testing the loudness again, I recorded an integrated value of -23.4 LUFS and a true peak max of -3.2 dB, which meant the game was within the target loudness ranges I had set for it and the game’s audio was finally finished.

References

acclivity. *BullBellow*. <https://freesound.org/people/acclivity/sounds/50669/>

acclivity. *Neigh1*. <https://freesound.org/people/acclivity/sounds/19494/>

acclivity. *Neigh2*. <https://freesound.org/people/acclivity/sounds/19812/>

acclivity. *Neigh3*. <https://freesound.org/people/acclivity/sounds/19813/>

Ali\_6868. *Bow Release (Bow and Arrow)*. <https://freesound.org/people/Ali_6868/sounds/384915/>

BenjaminNelan. *Wooden Click*. <https://freesound.org/people/BenjaminNelan/sounds/321083/>

BenjaminNelan. *Wooden Hover*. <https://freesound.org/people/BenjaminNelan/sounds/321082/>

Eponn. *Hit Swing Sword*. <https://freesound.org/people/Eponn/sounds/547038/>

Erokia. *Elementary Wave 8*. <https://freesound.org/people/Erokia/sounds/523222/>

Erokia. *Elementary Wave 9 (MSfxP7—1\_2 Extended (P’sESS)*. <https://freesound.org/people/Erokia/sounds/521302/>

Erokia. *Elementary Wave 11*. <https://freesound.org/people/Erokia/sounds/183881/>

Fission9. *Footsteps on Stone*. <https://freesound.org/people/Fission9/sounds/521590/>

ForTheHorde68. *FX\_Orc*. <https://freesound.org/people/ForTheHorde68/sounds/407356/>

GoodListener. *Cell Door 2*. <https://freesound.org/people/GoodListener/sounds/322451/>

IchBinChrist. *Bellowing deer—Animal—Nature—Hirsch—Röhren—Tiere—Wald*. <https://freesound.org/people/IchBinChrist/sounds/407631/>

InspectorJ. *Heartbeat, Regular, single, 01-01, LOOP*. <https://freesound.org/people/InspectorJ/sounds/485076/>

kingsrow. *BreakingVase01*. <https://freesound.org/people/kingsrow/sounds/194685/>

kingsrow. *BreakingVase02*. <https://freesound.org/people/kingsrow/sounds/194684/>

kingsrow. *BreakingVase03*. <https://freesound.org/people/kingsrow/sounds/194683/>

LiamG\_SFX. *Fireball Cast 1*. <https://freesound.org/people/LiamG_SFX/sounds/334234/>

LittleRobotSoundFactory. *Jingle\_Achievement\_00*. <https://freesound.org/people/LittleRobotSoundFactory/sounds/270404/>

LittleRobotSoundFactory. *Spell\_00*. <https://freesound.org/people/LittleRobotSoundFactory/sounds/270409/>

LittleRobotSoundFactory. *Spell\_01*. <https://freesound.org/people/LittleRobotSoundFactory/sounds/270396/>

LittleRobotSoundFactory. *Spell\_02*. <https://freesound.org/people/LittleRobotSoundFactory/sounds/270397/>

LittleRobotSoundFactory. *Spell\_03*. <https://freesound.org/people/LittleRobotSoundFactory/sounds/270394/>

LittleRobotSoundFactory. *Spell\_04*. <https://freesound.org/people/LittleRobotSoundFactory/sounds/270395/>

matthewHoldenSound. *OWI\_Bow Srting 2*. <https://freesound.org/people/matthewHoldenSound/sounds/542517/>

Migfus20. *Epic Orchestra Music*. <https://freesound.org/people/Migfus20/sounds/560449/>

Migfus20. *Epic Trailer Background Music*. <https://freesound.org/people/Migfus20/sounds/560454/>

.name. *Heartbeat loop*. <https://freesound.org/people/.name/sounds/418866/>

Nox\_Sound. *Foley\_Keys\_Drop\_Tile\_Mono\_NTG4P*. <https://freesound.org/people/Nox_Sound/sounds/550339/>

Nox\_Sound. *Object\_Coins*. <https://freesound.org/people/Nox_Sound/sounds/473578/>

qubodup. *Fireball*. <https://freesound.org/people/qubodup/sounds/442827/>

qubodup. *Whoosh*. <https://freesound.org/people/qubodup/sounds/60013/>

ryansnook. *Exhale*. <https://freesound.org/people/ryansnook/sounds/103568/>

Samulis. *Bow Release*. <https://freesound.org/people/Samulis/sounds/209399/>

ShadyDave. *Small Adventurers (music loop)*. <https://freesound.org/people/ShadyDave/sounds/471510/>

sketchygio. *Male\_Grunts*. <https://freesound.org/people/sketchygio/sounds/144907/>

soundslikewillem. *Young Bull mooing*. <https://freesound.org/people/soundslikewillem/sounds/367544/>

Stevious42. *Drinking Sip Swallow Gasp*. <https://freesound.org/people/Stevious42/sounds/259640/>

TheoTeravainen. *Underground Ambient*. <https://freesound.org/people/TheoTeravainen/sounds/510800/>