Group 5 - Sound Design Document

# **Members of project**

Josh Spence  
Ryan Jarmain  
Daniel Sohler

# **Overview**

The wanted mood for ‘Ah, My 2 Legs!’ is that of a **lighthearted, simplistic** soundscape, **that isn’t cartoonish**, which represents the gameplay and its atmosphere. The game itself is about a legless individual using a slingshot to navigate themselves around a prison, with the strange concept in mind, a lighthearted soundscape compliments the games mechanical atmosphere.

The needed sound effects are going to primarily be tied to the main player controller. Which needs a accepted-launch, denied-launch, no energy, clock sound, and a finish line sound effect. Other will be detailed in the Assert List attached.  
  
Summary of High Level  
Clean and simple sounds. Light hearted and joyful. Light tone colour. Exception is the timer, which has a lower tone to create minor tension. Most sounds will occur due to the player’s direct interaction. Some audio effects will have their pitch adjusted by in game factors.

**Theme:** Directional character flinging game

**Setting:** A legless individual using a slingshot to navigate themselves around a prison

## **Mood Board**

**Example Player Sound Effects**  
- [Comical Sound Effects](https://youtu.be/GHp9OzIJwn8)  
- [Whoosh Transition SFX](https://youtu.be/cVFsoBMkpGk)  
- [Buzzer Wrong Answer](https://youtu.be/FRpq7o1mKXY)

**Example BGM Sounds**  
- [Jetpack Joyride Main theme](https://youtu.be/WyVyibeMgyo)  
- [Amiss Abyss DK Country](https://youtu.be/2YfA_jd2klQ)  
- [Able Sisters Animal Crossing](https://youtu.be/ua76q5zwv2k)

**Example UI Sound Effects**  
- [Ticking Clock](https://youtu.be/u_jz6lJoG-c)  
- [Clean Wall Clock Ticking](https://youtu.be/nZFFjn9nOwU)   
- [Clock Ticking Noises](https://youtu.be/Qgsy8BEsLzg)

**Visual Aids**



# **Design Plans**

The plan is to use scripts to manage audio, as opposed to the in-engine audio mixer or the like. The purposes of the scripts will be to edit, run and manage the sounds, with a template script dictating the function of all sounds (volume, pitch, looping ect.) Then a function will be made, which can be implanted into any other script, that will allow it to run a audio file, with the modified variables added to it (volume, pitch ect.).

# **Implementation**

* Script to define sound files and create variables
* Script to play sounds
* Script to manage what sound is played when

All as intended in the plans.

# **Initial Outcomes**

When initially implementing audio, the main issue was the pre-existing speed of some sound files. They were less than a second long, so for some scripts where they were played often (clock ticking, timer start). This made specific timing complicated initially, and adding delays based on system time would be a fix for later.

Another issue is some sound effects not activating at a time that was expected. For example, the main Background Music would play on an Audio Source in the game, rather than on start-up. This was because when trying to run the BGM normally, even with the loop function added, it wouldn’t play.

Another concern was the Collision sound effect, it would play whenever the main character would collide with a surface, even upon start-up. This led to the audio of the collision sound effect to play often, at time where it didn’t make much contextual sense. This could be remedied by some form of timer that prevents the sound from being played at a time.

# **Iteration 1**

The first issue that would be resolved is the incorrect application of the ‘Collision’ sound effect. It would play whenever the player collided, no matter how small of a nudge was before collision. This was done by creating a float which would sync its value with the deltaTime of the game upon start, then when the player collides with a surface, it checks if it is/been over 1 second. If it has, it’ll play the audio queue the next time the player collides with a surface. This prevents the audio queue from playing too many times, and becoming unwanted noise.

The clock ticking was resolved in a similar way, creating a float, defining its value by deltaTime upon start-up, waiting for its float value to be equal to/greater than 1 second, then making it tick. This ticking also disables upon the player colliding with the ‘finish line’.

The next issue was disabling the BGM when reaching the ‘finish line’, the issue here mainly was making BGM completely separate from the rest of the other audio files. But even when I merged it over, and set it up to start upon start-up, there was not much of a way I could cancel the audio file. So to circumvent this issue, I created a ‘StopPlaying’ function which was identical to the ‘play’ function, except it would cancel and reset all the variables in an audio file (volume, pitch, can loop?). This was able to stop the BGM when the player reached the ‘finish line’, producing the wanted result.