**1. Introduction**

Video Game music has always been a method of helping players get more immersed in a game by using another sense, one that is not just visuals. The music can be used to set the mood of a character, it can strike fear into the players during dark scenes and can be actively used to build excitement in players. By setting the scene or by giving another gameplay indicator to players music usually sits behind the gameplay to help the visuals and rarely takes the mainstage.

With music being an afterthought to many in some games some players have opted for the option of muting the game music in favour of their own music while they play, creating a disconnect between the players and the intended game envisioned by the developers.

This problem forms the idea for the project. Can a game have the best of both worlds, The players choice in song and the developers vision? This project attempts to see if there would be an active engagement increase if the game were to adjust its gameplay to the music. And take the music to the forefront of the game and work alongside the visuals and not just aid in propping up the visuals.

The project works with already existing Audio Analysis Plugins to see if the data extracted from a song can be used in meaningful ways within gameplay and to then see if these ways increase the players enjoyment or a simple game.

There exists a whole another field that works on creating more and more accurate plugins but this project will be limited to what is easily accessible. Therefore the plugins themselves may become a limiting factor of the projects performance.

**2. Literature Review**

2.1 The Problem being Tackled

<https://kotaku.com/the-year-i-gained-the-courage-to-ignore-video-game-musi-5730637>

2.1 Games Designed around Audio – Examples (Bullets per Minute, Guitar Hero, OSU)

Rhythm Games

Pre-set Rhythm Games

2.2 Scoring on market games

2.3 Audio Analysis Methods

2.4 Audio Plugins – Vamp

**3. Methodology**

The project was made with the idea of having two separate programs that work together in order to work. The first program uses the vamp plugins and C++17 and will do the audio analysis on a song and then output the values using std::filesystem to a text file so that the other program can read them in. The other program is the game that is made using unity. It reads in the text files though a StreamingAssets folder and uses those values to adjust the gameplay values.

When choosing the open source game there a few criteria that was looked for. Firstly the game had to be a fairly simple 2-Dimentional Bullet Hell style game. Ideally the game would also be built in a well-known games engine and should already have audio working to make understanding the code easier.

3.1 Game Design

3.1.1 Main Mechanic

The main objective of the game to Shoot projectiles at the enemy target without getting hit by their projectiles which is similar to a traditional bullet hell. This game will have some aspects of the gameplay linked with some parts of a song provided by the user. Players are rewarded by doing damage to enemies consistently and are punished when they get hit by enemy bullets. A key difference is that this game does not use a life-based system. This is to allow players to listen to their song uninterrupted and also finish the song, with a high score system as an incentive to make the players try and not get hit. Life based systems were also ruled out due to the not pre-set difficulties. It is possible to give the game a very fast song and have it be nearly impossible to dodge every bullet on the screen.

3.1.2 Scoring

Score Is given to the player for each projectile that they hit a boss with and by picking up score cards dropped by enemies. This score is then affected by a Score multiplier that is increased by a small amount each time they earn score. But whenever the player gets hit their multiplier is reset to 1x. the multiplier can go to a max cap of 5x score. During the chorus the multiplier is double and the rate at which you earn multiplier is also doubled.

3.2 Implementation

3.2.1 Audio Analysis

Method of Audio analysis is Vamp. Started by running their SimpleHost application. Then stripped it out and changed it from a command line program and turned it into my own windows console application. After getting the initial Beat detection plugin working, needed to dig through the code that writes it out to an external .txt doc to pull the values and store them in some container. This data was then used in different ways for each Plugin.

3.2.1Beat Detection

For beat detection the data from the plugin was used to check when beats are so that the enemies can shoot. This was done by checking if there was supposed to be a beat between the last frame and the current frame by using the total run time of the application.

Queen Mary’s Bar and Beat Tracker was the Vamp plugin used for bead detection. This was choses because they are also the makers of vamp plugins so they should have well documented and easy to use plugins while also being effective.

3.2.3 Segmentation (DO SOME RESEARCH)

Segmentation used a similar method to Beat detection, checked if there was a change in segment between frames. But also outputted which segment it was “1:A, 2:B, 3:C…” which required storing of different data and for more accurate results the “properties” part of the vamp plugins was implemented, allowing the change of some pre-defines properties. In this case it was changing from hybrid To Timbral.

Queen Mary’s Segmenter was used to get the segments. This was a part of the same package as the Beat detection so it helped keep it simple and had all the same benefits.

3.2.4 Intensity (DO SOME RESEARCH)

Intensity was linked closely to segments. first all the values were stored and then the average intensity value for each segment was calculated, this gives some kind of picture of which parts were the chorus as *usually* Chorus has the largest point of intensity/Energy. This was then sorted by intensity and the highest two values were predicted to be the chorus.

BBC’s Intensity plugin was used because the BBC are a large company which could allow for a plugin with better performance and reliability. Queen Mary’s plugin package did not include an Intensity plugin so another package was also required.

3.3.1 Game

A game was found that was made using unity and had a solid basis for the framework of the game. Initial stages included understanding of code and structure. From there the firing code was swapped from a “Fire Rate” variable to a “Shot Cooldown” variable that tracks how many beats are in-between each shot. This was then expanded to have a BeatController class that the other classes use for finding out when a new beat has occurred to increment their “InternalShotCooldown” variables

Some method of spawning bosses needed to be understood so the next area worked on was revolving around spawning bosses. Due to the nature of this game being tested by participants a method of having the same boss order was required. But this provided a stale gameplay experience and also, if the song was too short, not allow the player to test all the boss fights. So a method of seeding the bosses was used. This implementation included a set order of bosses and the see describing the starting boss. Not as well seeded as it could be but is a nice quick work around.

3.3.2 Retrofitting a game

Think about how the current game structures itself. In this case it was 70% bosses 30% small enemy waves, so the initial plan of different small enemies and boss during chorus was not applicable. So allowing for exciting Chorus gameplay is a little more complicated and other steps must be taken.

Ability difficulty needs to be considered. This game was balanced around 120bpm (average pop song). This means faster songs will be harder/Almost impossible if balanced improperly (Though there will always be a song that will make it impossible to dodge all bullets). Some types of enemy shooting patterns will require a way to get ½, ¼ and possibly smaller beats. A method of calculating these needs to be decided. This project Already had a way of staggering shots where the delay would need to be calculated and applied every new beat. If this was not here, calculating the delay times and building it into the fire methods would need to be calculated. Scalable difficulty is hard as there are always going to be very extreme songs or songs where the audio plugins struggle and give awkward results. The Game needs to be built to handle these as best as possible and allow as wide of a song variety as possible.

What can be synced up? Look at the different parts of the game and decide what parts could be related to the music, and what part are good/wanted to relate to music. Could the game be split up into multiple different sections such as bosses/minions? Do powerups effect gameplay and do they need to work with the music?

3.3.4 Merging

3.3.4.1 Removing D-Sync

A key part of the game revolves around the music being in time with the beat. If the beat is slightly off it can be very noticeable so a few methods have been taken to ensure as small of a delay exists. One of these included loading in the song prior to the game starting. This is due to the music not playing until the song is loaded in by that thread. This could taske even a fraction of a second depending on the size of the file/performance of the PC which would be enough to make it noticeable and an ear sore.

3.5 Results from first wave?

3.6 Experiment Design

**4. Discussion/Evaluation/Future Work**

Main limitation the audio analysis code.