**1. Introduction**

**2. Literature Review**

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

2.2 Scoring on market games

2.3 Audio Analysis Methods

2.4 Audio Plugins – Vamp

**3. Methodology**

3.1 Game Design

3.1.1 Main Mechanic

3.1.2 Scoring

3.2 Latency issue? Not that applicable

3.3 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.

3.3.1 Beat Detection

For beat detection the data from the plugin was used to output “Fire” into the console window every time a beat passed. 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.

3.3.2 Segmentation (FIND SOME SOURCES)

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.

3.3.3 Intensity (FIND SOME SOURCES)

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

3.4 Implementation

3.4.1 Game

A game was found that was made usiong 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 exp[anded to have a BeatController class that the other classes use for finding out when a new beat has occurred to increment their “InternalShotCooldown” variables

3.4.2 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. Game needs to be built to handle these as best as possible and allow as wide of a song variety as possible. Main limitation the audio analysis code.

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?

3.4.3 Merging

3.5 Results from first wave?

3.6 Experiment Design

**4. Discussion/Evaluation**