**Abstract**

Some players have been choosing to mute the game audio so that they can listen to their own music while playing the game. This creates a large disscontect between what they developers created and what the player wants.

The aim of this project to see if player would enjoy the game more and be more connected to the game if the game syncs components of the gameplay to time with the game in ways such as: Beat and Segments.

The Project consists of two main applications, a game and an Audio Analysis project. The Audio Analysis is done though several Vamp Plugins and outputs the Beat Timings, Segment Timings and which segments are predicted to be the chorus. The game application then reads in these files to determine how the game should play.

Resutsssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss

Conclusionsssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss

**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)

Games that are based around their audio/Music already exist. But they usually tend to fall under two categories.

Rhythm Games

Rhythm games such as Guitar Hero, Beatsabre and OSU have existed for a long time. These games at their core are games developed to be played to user defined music and, in some cases, to give the impression of the user making the music. They never get their audio muted due to the nature of the game, the game would become dramatically harder without audio. They all feature user created levels as well as some levels initially created by the developers. Usually come with a tool that lets the players make their levels.

The issue is that the players need to manually create the level for that song. Which not every player will be willing and/or able to do. Creates an issue where if someone wants a song that no one else has made a level for yet, they cannot play the game the way they want to play without learning how to first create a level.

Pre-set Rhythm Games

The Other category finds games that aren’t strictly trying to imitate music but use their music choice as a unique gameplay experience. This is how the game BPM: Bullets Per Minute works. It is a rhythm game in the sense that the game happens to a beat and the player is trying to do actions on the beat.

The issue is that the game uses their own self composed soundtrack for the game, while the songs are amazing it leaves a small amount of content without needing to constantly keep writing music. Which might lead to the player choosing not to play the game since they want to listen to their own music, which is not something a game developer wants.

Both Methods allow for interesting sections based on the chorus due to the ability to hardcode the timings for the game and can be designed from ground up to work with audio. As effective as this is it requires a lot of time from the person creating the levels.

2.2 Scoring on market games

Rhythm Games use a multiplier that increases each time the player hits a note on time, rewarding the player for consistent play, with the aim of achieving the highest score possible through maintaining a high uptime on a high multiplier. The game can end if the player starts performing poorly and missing lots of notes/beats.

Bullet hells usually use a life-based system. Where score is given upon killing enemies or bosses, but the player has a limited number of lives and every time they get hit they lose a life. When they run out of lives the game ends and they need to restart the game. It provides the difficulty and harsh punishment that is associated with beating a game, but also allows players to compete by getting more kills that other players so that their score is higher.

2.3 Audio Analysis Methods

2.4 Audio Plugins – Vamp

Few different methods for doing Audio analysis and a few good applications with varying applications. Vamp was choses for it being a plugin and not a full visualiser as well. While also having lots of high-profile people making the plugins which were also well documented. Vamp comes with a helpful beginner Host Application to help with learning how to work with the Vamp Plugin Framework.

Most Other methods were either visualisers as well which would require the application to be stripped down first before working with it. Or they did not provide the data that was being looked for at the correct level of detail.

2.4.1 **Plugins Used**

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

This plugin uses a version of onset detection called Complex Domain for getting the beats. Onset detection.

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.

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. 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. This project is intended to be more of a tech demo and proof on concept than an actual game for release.

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

The starting point for Audio Analysis was to work with Vamp’s SimpleHost Application. Time was then spent understanding and pulling out the parts that were Important and then changed it from a command line program and turned it into a windows console application. The first plugin to be implemented was a Beat detection plugin, beat detection would be the main part of the application therefore it was the priority. Further digging was done to find the code that writes the values out to an external .txt doc to take these values and store them in some container so that they can be worked with. With this application, consistency was seen to be a priority over accuracy. By allowing the player to choose their own song, the application has to try its best to cover as many songs as possible and provide a playable experience. To cover every song would be nearly impossible but by having something that most players could get some enjoyment out of was more important than narrowing in on a certain song or songs and have slightly more accurate section transitions for instance.

3.2.1 Beat Detection

Implementation of the Beat Detection plugin was fairly simple. Once the program worked with the test plugins given by Vamp all that was needed was to find out what the unique Identifier and types were needed to get the application to find the correct plugin in the correct .dll and also get the correct value from this plugin. All this information can be found in the documentation for the plugin given by Queen Mary University. All the values outputted initially were loosely checked using Sonic Visualiser to ensure that the application was running correctly.

To work with the beat data from the plugin a method of finding out when a beat has passed was created this would eventually be moved to the game to allow the enemies to know when they 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.

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 to change how each segment would be split up, from a hybrid of Timbral and Chromatic to purely Timbral as sections of songs tend to be more easily identified by Timbre and in early testing using Sonic Visualiser, provided more consistent results.

In the later stages of the development the minimum segment duration was revisited to try and provide better results across the board. The default segment duration was set to 4 seconds which did provide a good amount of accuracy. But this accuracy came at the cost of frequent misfires or completely incorrect timings which would take the player out of the experience more than missing some smaller sections. The new revised Duration was set to 12 seconds. This was decided after testing with several numbers. Most of the songs tested had most of its sections as two-Four bar phrases. Which at the targeted tempo of 120BPM would have 16 seconds. The reason 12 seconds was chosen instead of 16 seconds was due to not every song being 120BPM. By having it at 12 seconds it allows the plugin to have some room to find the 8 bar phrase of fasters songs. For example a song with 140BPM would need roughly 14 seconds for each of its segments which would be below the minimum timer so that major section would be skipped which might cause the plugin to misfire on the 3rd 4 bar phrase which could feel really wrong. Furthermore, songs with a slower BPM should still work fine if they are not too much slower than 120BPM. This should hopefully allow for good results within the range of ~105 to 160BPM, which covers a large portion of songs with the Optimal value being 120BPM. Of course not every song features entirely 2- 4 bar phrases with many variations being used, but this should hopefully be the most consistent choice.

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 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.

3.3.1 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

Working with the intensity and predicted segments was a large part of the merge as

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 task 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 Experiment Design

3.6 Results from first wave?

Pretty much as expected, works when it works, doesn’t when it doesn’t.

Most players reacted neutrally to the base game as it is nothing too special, but all 3 said they enjoyed it more when they could choose their own song and it worked. They all state this is because they could choose songs that they liked/wanted to listen to. However, all the participants stated that they did not prefer the game when they chose a song to actively make the audio analysis struggle. This is to be expected but is mostly linked to the audio analysis plugins rather than the game. The game tries its best to play well with unusual outputs from the Audio analysis but there is a limit to how good it can be. Also, when the players chose another song different to the others they mostly enjoyed it more. But it is a lot more mixed than the original song that should work.

None of the 3 participants noticed anything changing due to predicted choruses. This indicated that it was not impactful enough so changes were made to the code that worked with the Predicted Chorus’. After investigation there were some issues and bugs with the code causing it to not be as responsive and also causing the Chorus to be at the wrong place and/or not happen at all. This could be why the participants did not find it impactful, as I might have not even happened, at least not properly.

The players Scores across their playthroughs were rather interesting. Most participants Base Game playthrough had their lowest scores, this is to be expected for a few reasons, one of which being that it is their first playthrough and that they are still grasping the gameplay. It also could be that the songs that they chose lead them to play better. Unexpectedly 2 of the 3 participants had their best score on the song that Should not work. This could possibly be due to the game not reacting properly to the audio analysis, allowing them to spend more time damaging the boss or by having similar choruses so more double score, or even getting lucky with bosses and getting an easy seed.

**4. Results**

**5. Discussion**

**6. Evaluation/Future Work**

Main limitation the audio analysis code.

VOD their replays in person

**References**