Literature Review

Videogames in recent years have grown to be a huge proportion of the entertainment industry. There are over “37.3 million” (Newzoo, 2018) people who play games in the UK. They have many recorded benefits to using them in a healthy way while being also being a relaxing way to pass time recreationally. Each day more people are discovering the benefits that videogames can provide. Videogames are aimed at children and adults but can be played by anyone. It is estimated that in the UK “around 6% of children are disabled, compared to 16% of working age adults.” (Department for Work and Pensions, 2014) These disabilities can limit a person’s ability to perform a range of tasks in real life and this unfortunately ends up extending to videogames as well. Disabled gamers frequently find themselves in a situation where they would love to play a game but are simply unable to because of its inaccessible design.

Research has been made into identifying problems that commonly affect people who have disabilities as early as 2005, such as Bierre *et al* (2005) table titled: Common Problems for Disabled Gamers, which categorizes different problems that disabled gamers face and give one or more reasons for the cause. Further research into the topic, trying to solve the issue birthed ‘game interaction model’. After the identification of the problems a few years prior, Yuan *et al* (2011) conducted a survey which showed that a large group of people find themselves excluded from playing video games because of a disability. After these findings, the model was formulated. This model breaks down a player's interactions for the game that they are playing and by doing so, the developer is able to analyze what interactions within a game may cause issues for those with disabilities; correctly “identify(ing) how a disability affects a player’s ability to play games.” (Yuan *et al,* 2011, p.2) In hopes that with easier identification: the industry would avoid designs that cause issues in interactions for disabled people and make a gradual shift in making games more accessible for everyone.

Even after the survey and additional research, not much has changed towards making modern games more accessible. The Switch, the most recent console produced by Nintendo has been “widely criticized for not being accessible for those with physical impairments. With no action from Nintendo to make the Switch more inclusive...” Mut provides a potential reason for this stating that “making games and hardware more accessible has really only just laid its foundations firmly on the radar of many of the big game developers and manufacturers.” (Mut, 2019) and that companies are only just understanding the importance of accessibility.

Disabled gamers have now had to resort to making their own custom controllers, with companies and even charities such as AbleGamers who create “custom gaming setups for people with disabilities, including modified controllers and special assistive technology… to help people play to the best of their abilities.” (AbleGamers, 2020) This is not the solution to the problem and will further ostracize disabled people from enjoying videogames if they need a specifically made controller to play a game. The responsibility is on the companies and developers to allow these people to have the ability to enjoy their product.

Why have companies taken so long to start thinking about making their games accessible? To answer this, we must look at the process. There are two different approaches: the first takes an existing game idea and makes it accessible by tweaking or changing interactions that those with disabilities may struggle with.

The problem with this approach is that the change in the interaction can significantly alter the gameplay. Games which have inaccessible core designs are simply unable to work when this strategy is applied, for example DanceDanceRevolution’s core design is inaccessible if you have a leg impairment, or low mobility. To change this, by allowing the player to use their hands would affect the experience significantly enough so that it doesn’t resemble what DanceDanceRevolution was. This poses two problems- The interaction that was changed was the part which made the game unique to others in the genre, and then becomes almost the same as the other games it was trying to be different from or that the interaction that was changed what made the game fun, and with its removal the game isn’t enjoyable anymore.

We can avoid this problem with the second approach of making a game accessible- if we design our game to be accessible from the start, it means that the game or its players aren’t affected by it further in development. This approach is incredibly effective for creating an accessible game.

The problem with this approach is that it is much more difficult to do. Creating a game that is accessible by design but is engaging enough for the average gamer to enjoy requires each part of the game to have extra thought put into it; increasing development time and most importantly development costs. Costs are a big issue in the industry, it’s reported that: “Only 20 percent of all video games that make it to store shelves are profitable.” (CNet, 2008) Which means that companies strictly control the budgets of their games and limit their development to cater to the broadest audience possible; preventing a loss using money on smaller markets.

It is easily said though that “making games accessible could help selling more games.” (Yuan *et al,* 2011, p.16)and these companies also have more than enough money to cater for more customers. The Nintendo Switch mentioned earlier was “best-selling gaming platform of 2019." (Techspot, 2019) Nintendo have the money to easily increase the amount of funding to improve the general accessibility of the games that were released on the console, and the console itself. Even after complaints they still refuse to do so. This is seen across the industry as companies' current strategies are to make the most amount of money by catering to the broadest market possible, while minimizing costs of development where they can, restricting the audience that can play their games but maximizing profits while spending the least amount of money.

This can be shown with the success of competitive gaming and esports. “Global esports revenues will hit $1.1 billion in 2019” (Reuters, 2019), because of this- developers are now being encouraged to design competitive, “e-sport” games as they now no longer just make money from game sales. Huge global corporations and brands such as Coca-Cola are sponsoring tournament events for these games which millions of players and even non-players spectate through streaming services and even TV, creating a huge amount of revenue for the companies who own the game and host the event.

The issue here is that fast inputs, quick reactions with high inputs per minute are encouraged for these games as pro players are quoted to be making up to “10 moves per second.” (Lejacq, 2013) Accuracy of inputs and frequency of inputs are the two difficulties which disabled people suffer from the most in games and unsurprisingly, these games have inaccessible designs to them, and focus heavily on fast frequent movements, requiring quick reactions to make the game entertaining to watch as the market for spectating these games has shown to be huge.

Developers and players themselves would criticize using accessibility strategies on e-sport games, as many can simplify the complexity of the gameplay and “games with a low level of complexity don't encourage dedicated players to stick around.” (David, 2016) These dedicated players are arguably the ones who play, watch and keep the e-sport alive, so it is important to keep the competitive and complex nature of these games, otherwise the players who keep them alive wouldn’t play them.

Yuan *et al* points out that “without exactly understanding the amount of effort required for implementing the various accessibility strategies, game developers may be hesitant to invest in game accessibility.” (Yuan *et al,* 2011, p.16) This can be applied to the modern-day market. There is a great inexperience of implementing accessibility strategies across the whole industry as it wasn’t until 2019 that it was firmly on their radar.

The research has been done to display the range of accessibility strategies and how to correctly implement them. There are many strategies to improve the accessibility of a game and although many interfere directly with the gameplay, there are lot that do not. These different strategies also take varying amounts of time depending when on in development they are implemented in.

An accessibility strategy example that applies to this is text scaling- individuals with impaired vision find it difficult to read text like subtitles, menus and user interface elements that are at a size that others may not have difficulty with. Implementing this once the game has been created would be an enormous task, all UI elements would need to be redesigned for this change. However, if the strategy was implemented from the beginning, all the effected elements would be created with that in mind, and because of that, the cost of implementing it is much lower than if you were to do it further on in development. This accessibility strategy has no effect on gameplay, and only help those that have vision impairments but is never seen to be implemented in games. This technique isn’t costly to implement, if planned for and done correctly and companies think otherwise due to their previous incorrect costly implementations.

This issue comes back to e-sport games which pretend to be fair and balanced for all players are currently doing the opposite by not including baseline accessibility features, such as colorblind filters or options which are affecting the professional players who play the games for a living. Professional videogame player Sneaky “recalled the release of Tahm Kench” a character in the game League of Legends. “Before the issue was fixed... (his) gray health was indistinguishable from his regular health for some players.” (Senfeld, 2017) A serious issue for the player and others who were affected. Which shows that not all accessibility options negatively affect gameplay or competitive integrity, and that some can be implemented that give benefits to all players.

This shows that even e-sports can benefit from including accessibility strategies into them and can potentially increase the revenue for the game by making the game playable for those who had difficulties before. The more games that are made with accessibility in mind, the more streamlined the process for implementing accessibility strategies will be. If this were to be successful companies and developers will see the importance and include accessible design standards into their engines such as colorblind filters, text and UI scaling, closed captioning systems, etc. This development will cost a large amount upfront but will make money over time from making it cheaper than ever to implement these accessibility options for each new game they make.

A new wave of disabled gamers who were once unable to play games will now be able to start entering the market, and the spectrum of games that disabled gamers are currently limited by will widen. Increasing the amount of revenue that these companies could have made and allowing more people to enjoy the positive experiences that games can bring to people. Companies currently care too much about the monetary success of their games, rather than making games as an art form and caring about the success of games and their benefits that they give to people.

I hope to use my game as an example to this. I aim to use several accessibility strategies, many of which that don’t affect the gameplay to help make my game more accessible, even though the game’s genre itself has a generally inaccessible design to allow more people to enjoy a genre which many haven’t yet had the chance to enjoy.

References:

Bierre, K. Chetwynd, J. Ellis, B. Hinn, D. M. Ludi, S. and Westin, T. (2005) Game Not Over: Accessibility Issues in Video Games, *HCI International 2005: 11th International Conference on Human-Computer Interaction,* Games Accessibility Special Interest Group

Newzoo (2018) *U.K. Games Market 2018*, Newzoo, available from Newzoo, accessed 21 January 2020

Department for Work and Pensions (2014) *Disability facts and figures,* 16 January, available from <https://www.gov.uk/government/publications/disability-facts-and-figures/disability-facts-and-figures> accessed 21 January 2020

Yuan, B. Folmer, E. and Harris, F. (2011) Game accessibility: a survey, *Universal Access in the Information Society,* Vol 10, No 1, pp81-100

Mut, C. (2019) Accessibility finally matters to the game industry — but it needs to do better, *VentureBeat,* 8 October, available from <https://venturebeat.com/2019/10/08/accessibility-finally-matters-to-the-game-industry-but-it-needs-to-do-better/> accessed 21 January 2020

AbleGamers (2020) *About AbleGamers*, available from <https://ablegamers.org/about-ablegamers/> accessed 21 January 2020

Reisinger, D. (2008) Why most video games aren't profitable, *CNet,* 24 November, available from <https://www.cnet.com/news/why-most-video-games-arent-profitable/> accessed 21 January 2020

Knight, S. (2019) Nintendo's Switch is on pace to be the best-selling console of 2019, *Techspot,* 13 December, available from <https://www.techspot.com/news/83185-nintendo-switch-pace-best-selling-console-2019.html> accessed 21 January 2020

Russ, H. (2019) Global esports revenues to top $1 billion in 2019: report, *Reuters,* 12 Febuary, available from <https://uk.reuters.com/article/us-videogames-outlook/global-esports-revenues-to-top-1-billion-in-2019-report-idUKKCN1Q11XY> accessed 21 January 2020

Lejacq, Y. (2013) How fast is fast? Some pro gamers make 10 moves per second, *NBC,* 24 October, available from <https://www.nbcnews.com/technolog/how-fast-fast-some-pro-gamers-make-10-moves-second-8C11422946> accessed 21 January 2020

David, C. (2016) Why overwatch has us worried, *Grunge,* available from <https://www.grunge.com/10197/overwatch-us-worried/> accessed 21 January 2020

Senfeld, K. (2017) Video: Sneaky talks about being colorblind, *DBLTap,* 21 June, available from <https://www.dbltap.com/posts/5169202-video-sneaky-talks-about-being-colorblind> accessed 21 January 2020

Functional Requirements

I am developing and compiling my game in Unity 2019.2. This will define a set of minimum requirements that is mandatory in order to play the game. Unity’s minimum requirements are of low specification which will allow my game to be generally accessible for most people.

The minimum requirements to play the game is as follows:

Operating System: Windows 7 SP1 or higher.

CPU: Any which has SSE2 instruction set support.

GPU: Any which has Direct X 10 (shader model 4.0) support or higher.

The game is controlled using dual stick XInput and DirectInput standardized controllers. This means any player who owns an XBOX, PlayStation or any other dual stick controller which uses either API can connect it to their computer and can play the game. The game is currently in development, so a required size of the game is unknown. As it is a game which does not contain lots of levels and area-, the file size will be low, and will not reach a size any bigger than 10GB. This will require the user to have at least 10GB of space free on their system before they can play the game. These are the minimum technical requirements.

My minimum age requirement for the game is based on the ESRB ratings. Looking at their content descriptors, which they define as being content which ‘may have triggered a particular rating and/or may be of interest or concern’ only two out of all their ratings has the potential to be an issue, these would be Lyrics and Strong Lyrics. Lyrics signals that the game’s music contains profanity and mild references to drugs or sex. Whereas Strong Lyrics signals that there is explicit profanity and direct references to drugs or sex. The lyrics in the game will not be explicit enough to get a Strong Lyrics rating, which will typically give it an T rating. As such, I’d say the minimum required age to play would be 13, due to the presence of some mild profanity and references to taboo acts in the lyrics of some songs.

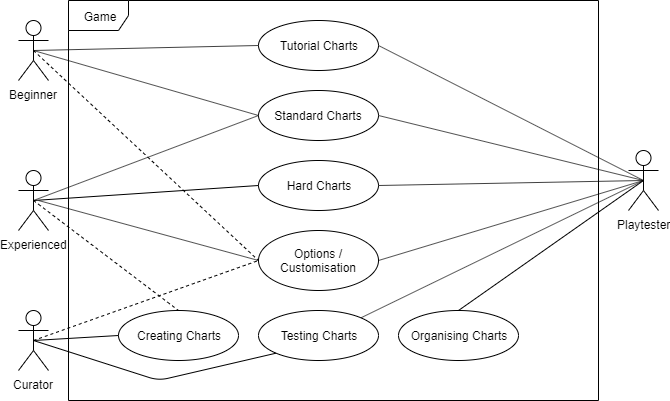
The game will have a few different user types. Beginner players, Experienced players, Curators and Play testers.

Beginner players, either new to the game or the genre will start in the introductory stage of the game, which will teach the player how to play, and prepare them for the further and harder challenges as they progress. These players will be unable to play hard modes and will typically avoid the chart editor as a good understanding of both the game itself and rhythm games is needed to do both. They may feel the need to change some of the options or customize the settings to suit the game more to their needs, but most players will only dabble once they are more experienced. After the tutorial, beginners will ease into the standard charts and as they improve, they will progress into an experienced player.

An experienced player will generally have a good understanding of the game and/or a good understanding of rhythm game principles. Many new players who have pre-existing skills in rhythm games will very quickly skip the beginner stage and become an experienced player quickly. Once a player understands what the possibilities of the game are, and how the challenges are completed. They will engage with all playing modes, including hard difficulties. Experienced players will start tweaking the settings to suit their own needs, and potentially make some customization changes to help the player in the long term or to make short-term changes that freshen up the game. Many experienced players will start to learn the level editor and continue to expand upon their skills and understanding of the game while creating their own charts to songs they want in the game. Other experienced players either haven't tried, don’t enjoy or find it too difficult to make their own levels and will just play the levels that have already been made. Some experienced players enjoy making their own charts so much that they end up becoming Curators.

Curators are players who spend the majority or even the entirety of their time playing the game making their own charts. These players are typically familiar with other games of the genre or are very experienced and engaged with your game. Curators will quickly become familiar with the chart editor and most of their time spent playing is for the purpose of play testing the charts they have created. Some are likely to use the customization options for the same purpose of experienced players, but overall less likely as experienced players as they are tweaks to directly affect gameplay.

Play testers are vetted players who will be in direct communications with the developers of the game with the responsibility of testing each part of the game and to relay any bugs, issues, fixes or improvements that can be made to the game during its development stages to the developer. As mentioned, they are likely to use all features of the game, and will be already experienced in playing video-games, this means that their game skill will be above the average player but lesser skilled play testers will be necessary to test low game skill parts of the game effectively.



Testing plan

I would like to test both the performance and functionality of my software, as my software should work in its intended way, without using an excessive amount of resources or time to do so. Using this principle of software design, I have created a plan for testing my game.

The most important of the two to test is the functionality. A piece of software could be coded in a revolutionarily efficient way, but if it didn’t behave how it was expected, the software is ultimately useless. Therefore, we must test the pure functionality of the software. This can be done in numerous ways and can take varying levels of time with variable results. We can quickly get an analysis of the software’s functionality by using black box testing. This is done by listing the events or interactions that we would like to test, and the outcome that we are expecting to happen once the interaction or event has happened. We will then run and test these interactions and assess the outcomes to our expected outcomes. This will tell us what isn’t working as intended, and we can then find out what needs to be fixed.

Black box testing isn’t enough on its own to test the functionality of our software. Alongside black box testing I will perform compatibility tests- if the software can be run on different devices or machines and still work as intended without issues. I will do this by running my game on another machine that fits the requirements with my game, and black box test the functionality based on the outcomes of the first test. This will ensure that the game is running identically on both devices with no issues and will test the functionality of our software thoroughly enough.

The performance of the software will be tested using a load test, a resource intensive part of the game will be ran and the performance of the software will be monitored to ensure that there are no issues, the same part of the game will be ran again while the device is under high load and performance will be monitored. This ensures that our game will not suffer issues for players whose devices closely match the minimum requirements, and that the game is still playable, even in the most intense sections. I will do this by creating a resource intensive test level and run it while the device is under different loads and ensure that the performance is still suitable. The results of this test will be recorded and if there is an issue, it can then be investigated.

I will regressively test any bugs that have been found, this means that if a bug has been found through an unexpected outcome in a black box test. I will re-test the unexpected outcome and ensure that not only the bug has been fixed, that no other bugs have arisen because of it. This will be done by selectively taking a fixed bug and testing its outcome to ensure its been fixed, and any additional functionality that could be affected by the bug and its fix. This will be done in a black box style fashion, recording expected outcomes and the outcomes of the test and assessing them if there is any issue.

Lastly, as it is a videogame, I would like to ensure that my game is fun and enjoyable for the person playing it, I will do this by play testing it. This non-functional type of testing is necessary for games to ensure that the player can enjoy the product, for example if earlier levels are too hard, or later levels are too easy, players will most likely not find it fun to play your game. I will do this by hand playing the game myself, and self-assessing how difficult every part of the game is, from a specific interaction to the difficulty of a level. This would be more effectively done within a team of people, as multiple levels of skill and mobility need to be considered when it comes to difficulty. Any issues with the game’s playability will be recorded, and then changed with their changes also being recorded listed with the reasoning.

Functionality Testing – Black Box and Compatibility Testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Function to be tested | Expected outcome | Test Outcome | Compatibility Outcome | Bug / Issue? |
| Does the game launch? | Game can launch on device | Game launches | Game can launch | No |
| Acceptable framerate when game is running | Framerate is good when game is running | Framerate is smooth while running the game | Framerate is lowered but still smooth while running the game | No |
| Music is synced and doesn’t fall out of sync | Music is synced and doesn’t fall out of sync | Music stays synced with the notes throughout the level | Music stays synced with the notes throughout the level | No |
| Notes can be hit using the correct buttons | Notes can be hit using the correct buttons | Each control is correctly hitting each note | Each control correctly hitting each note | No |
| Notes when missed are misses, show near when near, and hit when hit | Notes when missed are misses, show near when near, and hit when hit | Notes are correctly hit/missed based on the input | Notes still hit/missed based on the input | No |
| Score increases when notes are hit | Score increases when notes are hit | Score increases correct amount for a hit and near | Score increases the same amount for a hit and near | No |
| Health decreases when notes are missed, and increases when notes are hit | Health decreases when notes are missed, and increases when notes are hit | Health decreases when notes are missed, and health is gained when hit | Health decreases when notes are missed, and health is gained when hit | No |
| Game ends when health is depleted | Game ends when player runs out of health | Game stops when the player dies | Game stops when the player dies | No |

Load Tests

|  |  |  |
| --- | --- | --- |
| Test | Outcome | Bug / Issue? |
| Device: Normal load  Game: Normal load level | Normal framerate, no issues under standard level and standard load | No |
| Device: Normal load  Game: High load level | Fine framerate, no issues for high intensity standard level and standard load | No |
| Device: High load  Game: Normal load level | Normal framerate, no issues while game is running under high device load | No |
| Device: High load  Game: High load level | Lowered framerate under high intensity situations, still playable so no issues | No |

Regressive Tests

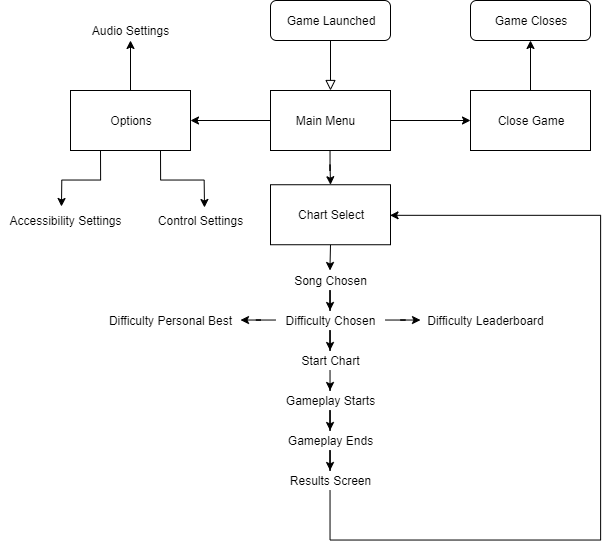
|  |  |  |  |
| --- | --- | --- | --- |
| Function to be tested | Expected outcome | Test Outcome | Bug / Issue? |

Test Conclusions and changes

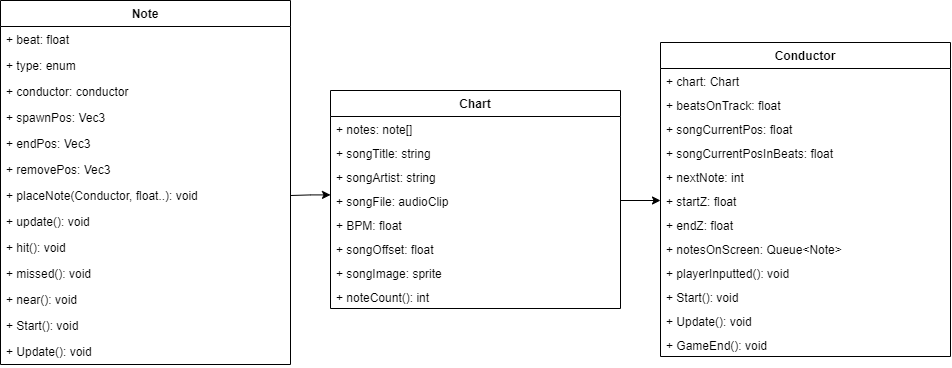
No regressive tests were needed due to no bugs being found. Play test changes included balancing of the level, reduced the number of notes nearing the beginning and increased the amount of beats the track would show to give the player more time to react to notes.

System design documentation

Gameplay Flowchart



Gameplay UML



Notes are a class for the note game objects, they have a beat that they lie on and a type to define what note type they are. When notes are spawned in the start function, they are placed with placeNote, and then their position is interpolated in the update function. Variables are used to store the positions of the notes, and there are functions which update the notes material based on the players accuracy on hitting the note.

Charts are the ‘levels’ of the game. Charts have a sequence of notes- which are timed to the song that it is attached to. This is stored in an array of notes. We have variables to store some of the basic information of the song, all of which are used in the game such as the name of the song, the audio file and the name of the artist. Other variables such as the BPM of the song are used by the conductor to keep the notes in time with the music.

This is the role of the conductor, which controls the gameplay. When a player selects a chart, the conductor will store a reference of it. This reference is then used by the conductor to play the song and spawn the notes on their correct beats. The conductor keeps a queue of all the notes that are being displayed to the player on the track, this is controlled by the player set variable beatsOnTrack, which controls how much of the song in advance the player can see. The conductor handles the players inputs, and tracks if the player has hit, missed or neared the note along with updating the players health and ending the game when it has run out. These are controlled using the labeled functions or contained within the update functions.

Implementation report (300 words max on)

I have implemented a vertical slice of the rhythm game I am creating. It is currently only playable on a standard console controller using the DirectInput or XInput API.

When the game starts, the chart starts and the song starts playing, each chart has notes which are synced to in time to the music and must me hit by the player at those times to not lose health and complete the level. Each note type on screen corresponds to a different action which needs to be performed on the controller. The track notes are currently hit by pressing the left and right triggers, and the sky notes are hit by flicking the left or right control sticks.

This action corresponding to the note needs to be hit in time to the music, and the way each note is visually represented gives an indication to that timing. The player has a health bar and if notes are hit at the right time the players health goes up. If notes are missed however, the health decreases. If the player loses all their health the game ends. The goal of the game is to survive the level without dying. Currently there is not an end to the level, however there is scoring, which players can use to try and get the best score possible.

Hitting the note at the right time will increase the player’s score, ‘nearing’ a note by getting close to hitting it on time will slightly increase your score, but do not make you gain any health.