**Analysis**

Introduction

I have decided to create a single player and multiplayer Pac-man which will consist of three very different gamemodes, inspired by the original 2D maze arcade game Pac-man, which was developed and released by Namco in 1980. Pac-man is a famous and beloved game, that captured the hearts of many people of all ages. Originally the 2D maze arcade game had the Japanese title ‘Puck man’, which later changed in 1980 as the name could have been used for vandalism. The game was now known as Pac-man.



The original Pac-man in 1980

The gameplay of Pac-man is simple, yet amazingly fun. The game consists of one player (Pac-man) and a Maze with dots in each free space, and four flashing energizers in each corner. There are also fruits which appear twice every level. The dots are each worth 10 points, and the flashing energizers are worth 50 points, and depending on what level fruits can give you between 100 and 5,000 points. A level is complete once all dots on the screen (including the flashing energizers) have been eaten. The game consists of 256 levels, the last one being endless as the developers found a bug if it wasn’t to be.

Furthermore, the game consists of four enemies, which are called Ghosts, each having their own personality. The objectives for the ghost are to find Pac-man using an algorithm and eventually eat him which would cost the player a life. The player begins with three lives. However, Pac-man can possess the ability to eat the Ghosts once he eats a flashing energizer. The player will know this as the ghosts turn blue. A score of 200 points is given for eating the first ghost, 400 points for the second, 800 points for the third and 1600 points for the fourth. The time at which the ghosts are able to be eaten are limited, with the time decreasing every level. If a ghost is eaten, it will respawn in its original form.

To summarise, the overall objective of the original Pac-man game is to avoid the ghosts and obtain all the dots to gain access to the next level.

Project Scope

My main focus will be to investigate how to create a working multiplayer game, first on a local network and then if possible, on a worldwide scale. In addition, I will need to investigate how to sync player movement and the game state between clients with little latency.

The single player aspect of my game will only involve one game mode, which will also be included in the multiplayer part of the game. The game mode will be an almost identical version of the original Pac-Man, but the ghosts I will create will have different AI algorithms than the original and therefore each ghost I create will have different personalities.

The multiplayer aspect of my game will contain a different game mode. This second game mode will be a player against player game mode. This game mode will involve two players who aim to survive as long as they can and to try get to 500 points. Rather than the players killing each other, tiles such as lava will spawn randomly around the map which players have to try avoid to survive. These lava tiles will only be available for players to pass through when they turn from lava to rock tiles. Players can either win by getting to 500 points while avoiding the lava tiles, or they out survive the other player (if the other player loses all three lives).

The settings part of my game, will include various features, from checking your high-score, to changing the colour of your Pac-man, to maybe even checking where you are in the online leader board. Furthermore, the settings may include a help option and an option to change the game difficulty.

I will also be focusing on developing different algorithms for the enemy AI for the single player part of the game. I will be comparing my algorithms to various other single player Pac-man games. Alongside this I will be creating a progression system where players unlock various colours for their Pac-man and map for the single player part of the game also. I will also test it to see how my algorithm matches against other single player Pac-man games. This will check the difficulty of my game.

Below I have outlined the structure of how I will be tackling this problem to create a single-player and local multiplayer Pac-Man with not just the original Pac-Man as a game mode, but another unique second game mode and maybe more.

|  |  |
| --- | --- |
| Objectives | Duration (weeks) |
| Prototype | 4 |
| Analysis | 2 |
| Documented Design | 4 |
| Technical Solution | 12 |
| Testing | 1 |
| Evaluation | 2 |

This table of objectives and how long they will take are subject to change depending on if I think I need to remove or add extra features to my project. The duration could therefore change throughout my documentation.

Client/Supervisor

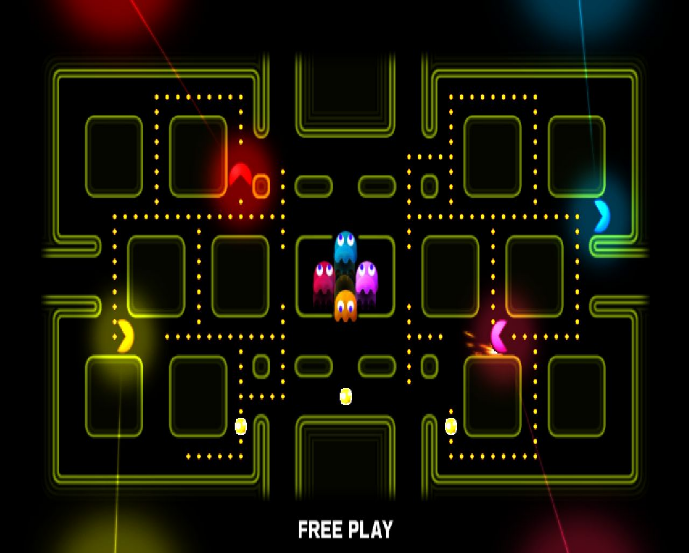
My supervisor for this project will be my computer science teacher, Mr Abbas. Me and Mr Abbas will have meetings weekly to discuss if my project is going along the right path, or if there are any alterations I need to make.

Target Audience

When I have finished creating my game, I will test it amongst friends and family spread around the UK (or even World) if possible and of all ages, to see how latency affects gameplay and to see which game mode is preferred and if there is a correlation with age. However, arcade games tend to appeal to teenage and kid audiences, so I will aim to cater to those first.

Investigation

**1.Pac-Man Battle Royale**

Pac-Man Battle Royale is the 17th arcade game (11th official game) in the in Pac-Man’s series. It was developed and released by Namco Bandai Games in January 2011 and later ported for a digital release on console and PC in 2014. The game can be played as a single player or local Co-Op. It was never released as an online multiplayer game.

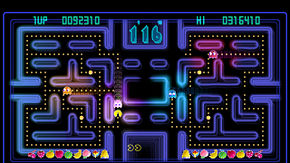
Up to four players can compete at once, choosing a game length of between three and nine rounds. Each player controls a colour Pac-Man character. The players move throughout the maze trying to avoid the four ghosts that are present, while collecting dots or powered pellets scattered around the map. If a player collects a powered pellet, their Pac-Man character grows in size for a limited time and is now able to eat ghosts and other players. If two players have powered Pac-Men and they run into each other, they will knock each other back a certain distance (but that is it). A player wins the round if all other Pac-Men have been eaten, or they have the most points once the round ends after a maximum of two minutes. Points are obtained from eating dots and powered pellets, in which they respawn whenever they have all been eaten. The player who has won the most rounds wins the game.

The game could be very inconvenient if there are four players playing locally as there may be a lack of controllers if played on the console, or lack of space on the keyboard or controller connections if played on the PC. The game could have benefited greatly by adding in an online multiplayer feature that did not require all the users to be controlling Pac-Man locally on the same device.

My solution aims to solve this problem by providing a network for players to play on wherever they are in the world, with a focus on making sure the gameplay still remains smooth and there is a reduction in latency between clients. Friends and family would be able to play wherever they want, as long as they have access to a Computer and Wi-Fi.

|  |  |
| --- | --- |
| Pros | Cons |
| * Allows players to play with each other, rather than just on their own. * There are many clear and creative animations which make the game exciting to watch. E.g. Sparkles when collecting powered pellets or a dying animation once Pac-Man has been eaten. * Music and sound effects. Lighting effects were also great. * Gameplay is smooth as the controls are not just very easy to use but have very little delay and are very responsive during the game. * The game contains different maps, which avoid repetition. Each map is created with a different colour. | * The game is not online multiplayer, and only a local Co-Op game, so players may find it very inconvenient trying to find a way to organise a game between two to four players to all play on the same device (locally). * The game does not contain many settings besides choosing the number of rounds and colour of Pac-Man. * The score for each Pac-Man is not shown on the screen during the round, so players are not aware of the score once the round ends. |

Inspired by the creative and cool animations, I will be adding some into my project. These animations will occur when Pac-Man has been eaten, obtains a power-up (powered pellet), eats a ghost or even when the Player wins the game. I will be adding a live scoring system that will show when a player(s) is playing, and it will be updating during the game. This will let the player know how they are doing. I will also be adding a more in-depth settings menu. Other than the colour of Pac-Man, I will be adding a difficulty setting to the single-player part of my project and a speed setting to the multi-player part of my project. The difficulty setting will range from easy, to medium, too hard. The speed setting will be determined by the host and will decide how fast the Ghost’s and the Pac-Men are.

**2.Pac-Man Championship Edition**

Pac-Man Championship Edition was developed and released by Namco Bandai Games in 2007 for consoles. The game was designed by the creator of the original Pac-man arcade game in 1980, and it was also the final game that he designed. The game is now ported to many devices such as android and iOS.

Like the original Pac-Man arcade game he designed, Pac-Man Championship Edition consisted of the same basic gameplay. The player navigates Pac-Man throughout a Maze, collecting dots, power pellets and fruits, all while avoiding the four ghosts. Pac-Man still dies if the ghost catches him, and the ghosts still turn blue once Pac-Man collects a powered pellet which allows him to then eat the ghosts for points.

Pac-Man Championship Edition was designed to make it a much faster paced game, decreasing the users time to decide and react as the game goes on. The game offers some major differences from the original Pac-Man. Each maze (map) is divided into two halves. Eating all dots on one side of the maze, causes a fruit to appear on the other side. Once Pac-Man eats the fruit, it causes a new maze to appear on the other side. The longer the player stays alive, the faster the game gets, increasing the speed of Pac-Man and the ghosts. If a player dies and loses a life, the game slows down. In addition, if Pac-Man eats a powered pellet, not only do the ghosts turn blue like the original Pac-Man but they go back to their original speed, until they return back to their original form.

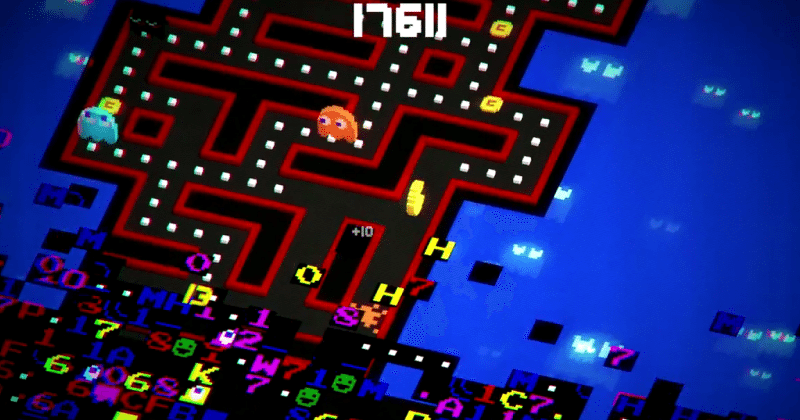
The game features six different modes. It features a Championship mode which is the basic five-minute mode. It also features two Challenge modes which affect the stage, such as by increasing the number of powered pellets or putting the maze into darkness which lets the player only see Pac-Man (themselves), the ghosts and a smaller area around them. This limits the players ability to see how the maze has changed and increases difficulty. One of the Challenge modes is ten-minutes while the other is five-minutes. There are also three Extra modes featuring different Mazes. Furthermore, the game has some great features such as an online leader board so you can see how your high score compares to theirs. The game features new sound effects while still retaining the original Pac-Man music and provides very modern graphics and smooth animations, with even particle effects. The game also provides an achievement section, where a player can track their progress and unlock various achievements. The menu of the game contains a leader board and achievement menu. The leader board menu is something which is a priority of mine to incorporate in my game.

|  |  |
| --- | --- |
| Pros | Cons |
| * Provides more than one gamemode which increases or decreases difficulty allowing the user to customise their experience. Such as the fog of war (darkness) feature. * The game provides very modern graphics which make the feel aesthetically pleasing. * Contains an online leader board. * Contains an achievement menu. * The game is available on many devices. * The game provides a help page to allow players to get familiar with the game before playing. | * The game has no Co-Op or online features beside the leader board. * You can’t customize Pac-Man in many ways. Such as the colour of Pac-Man. * Although the game provides many modes, the concept of each mode and how to play each mode is the same. * Controls are not responsive enough. This may be due to how fast-paced the game gets. |

I aim to provide my game with an online leader board system so players can compare how they are doing against their friends and other. In addition, I will be providing a help page to allow all players to get an understanding of each game mode I will be providing. My game will also feature changes to the colour of Pac-Man. Furthermore, I aim to make my controls as responsive as possible, especially with the network part of my game. This will involve me having to reduce latency between client and server and having to make sure to sync the game-state and player movement between clients. The game has also inspired me to include some interesting features such as an increase in speed as the game goes on. I will be including this in my second game mode in the multiplayer section of my game, hoping that it will add some extra intensity to the game.

I also aim to add a menu, similar to ‘Pac-Man Championship Edition’s’ menu. I will have a single-player, multiplayer, leader board and settings options in the menu and possible an achievement option.

**3.Pac-Man 256**

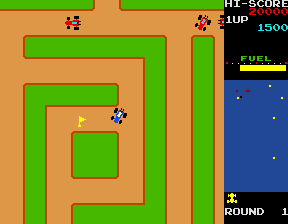
Pac-Man 256 is an endless running video game developed by Hipster Whale and 3 Sprockets and published by Bandai Namco Entertainment for iOS and android in 2015, then later ported to console and PC in 2016. The game was inspired by the original Pac-Man game’s level 256 glitch. Pac-Man 256 puts a player in control of Pac-man as he moves through an endless trying to avoid both the ghosts and the glitches trying to catch up to him, so the player must act quick. While trying to survive the player tries to obtain powered dots, like from the original Pac-Man games, or power-ups which are a new feature introduced in Pac-Man 256. There are over 20 power ups ranging from tornadoes to laser-beams. Another new feature specifically designed for this game is whenever Pac-man collects 256 dots, all ghosts in the vicinity are wiped out and the player gets a mega bonus score. Furthermore, on console and PC a new local Co-Op game mode was introduced, which allows up to 4 players to play simultaneously trying to gain as many points together as a team. If a player dies a power-up will spawn and one player will return if it is collected. In conclusion Pac-Man 256 is an exciting endless runner game rather than a level-based Pac-man game we are used to seeing.

|  |  |
| --- | --- |
| Pros | Cons |
| * Provides a local Co-Op mode, for up to 4 players. Rather than players against each other, players team up together. * Contains new features, such as powerups and an endless maze which adds a fresh new look to the Pac-Man Series. * Translates Pac-Man’s art style into 3D. * Several visual themes for the maze and player character, making the game more interactive. * The game provides amazing particle effects, especially when using one of the many power-ups. * Available on many devices, including console, mobile and PC. This allows for a variety of target audiences. * The game gives the player a feel of progression by allowing the player to obtain coins to upgrade power-ups. | * No music apart from the title screen, which feel boring for the user. * No global leader board only a friend’s leader board. * Not many games modes, so could lead to repetitiveness. * Quests are very minimal, so game may tend to become quite tedious (especially single player) rather than fun. * Although it is a local Co-Op game, the game is not an online multiplayer game, meaning the multiplayer aspect does have its limitations. |

In my game I will also be including a Co-Op, with the aim to make it local first, and hopefully to a point where anyone around the world can play, making it an online multiplayer game. In addition, I will be adding a sense of progression in my game, by being able to unlock different colour characters and possibly even a levelling system. However, unlike Pac-Man 256 I will be adding music throughout my game, and more than one game mode to make the game feel more interesting as it will minimise repetitiveness.

I will not be making my game 3D (so no 3D-art), as I want as many users to be able to access the game as possible without being limited by hardware or operating system.

**4.Rally-X**

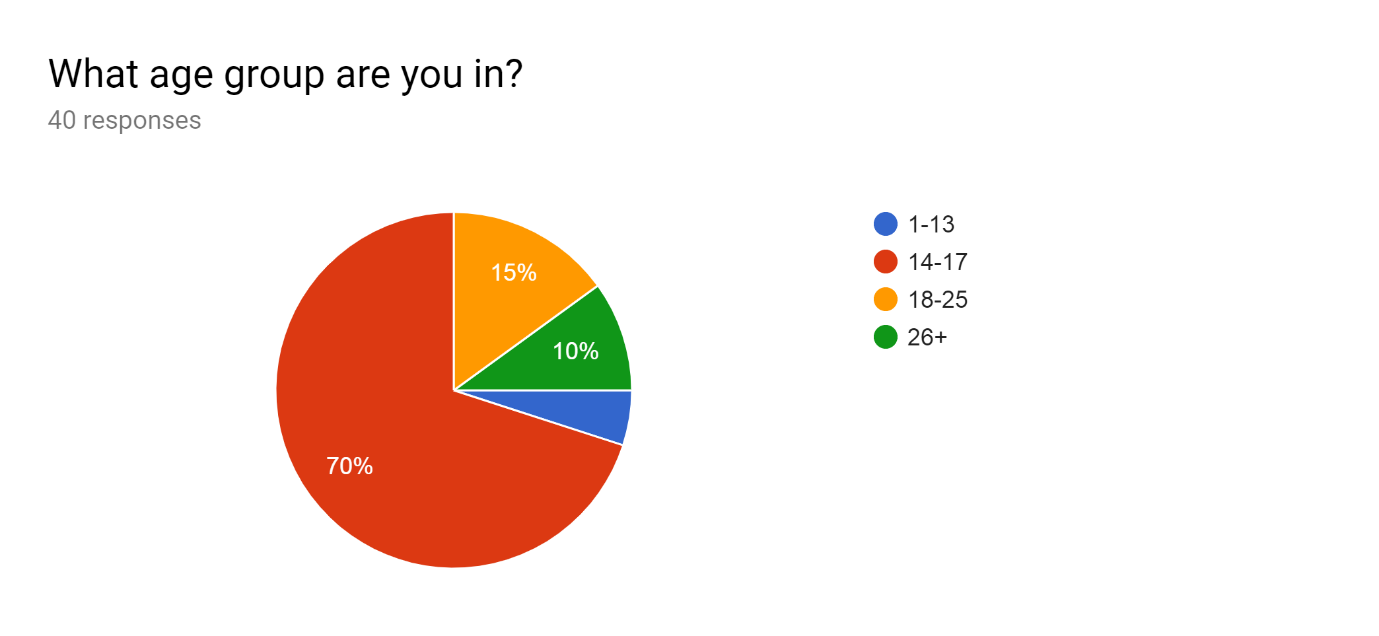
Rally-X is a maze arcade game developed and published by Namco in 1980. The player takes control of a blue race car, where the objective is to collect all ten yellow flags around the map, while avoiding the red cars. Once all flags have been obtained, the player progresses to the next round, which is of greater difficulty and the map also changes. On the right of the screen, the score, high-score, round number and the amount of fuel the car has is shown. The fuel depletes every round, and if it reaches zero before all flags are collected the players car goes extremely slow, often resulting in the player losing a life. The flags increase in value by 100 as they are collected. If the flag with the letter ‘S’ (meaning special) is next to a flag being collected, all other flags after this one earns the player double the value of what they would usually get. The player is also given a smoke screen to try stun any red enemy cars following. There also obstacles place randomly throughout the game to make the player lose a life, as the player must avoid them also. Rally-X provides exciting single player gameplay for various reasons. One of which being the idea that you never know exactly where the enemy is, although the map does give you a rough idea. Another reason being that you have to race time, as you have to try get the flags before fuel runs out, otherwise you are in a lot of trouble. However, the game can get repetitive after you have played it many times due to the fact that the map for that certain round never changes. animations and particle affects are very minimal, even the smoke screen animation is very limited. This is most likely due to the hardware limitations in 1980. Music however is played throughout the game.

|  |  |
| --- | --- |
| Pros | Cons |
| * Variety of maps * Scoring system allows the user to feel a sense of progress. * The game can provide slow or fast paced moments, making it exciting. * Music throughout game. Specific music when collecting flags or losing a life. * The game adds colour to try limit how quickly a player may decide to exit the game, by making it look more aesthetically pleasing. * Features like the fuel system adds an extra challenge as it makes the user race against time. | * Animations and particles are very limited, most likely due to hardware limitations. Although music is played throughout the game, it doesn’t seem to bring much life to the game. * Only single player and the maps for the specific round never change, so game may get boring quickly. * There are no game settings, so very little customisation by the user. * The difficulty of the game is quite hard and can’t be change. This is not a good idea as it does not attract a large children audience. |

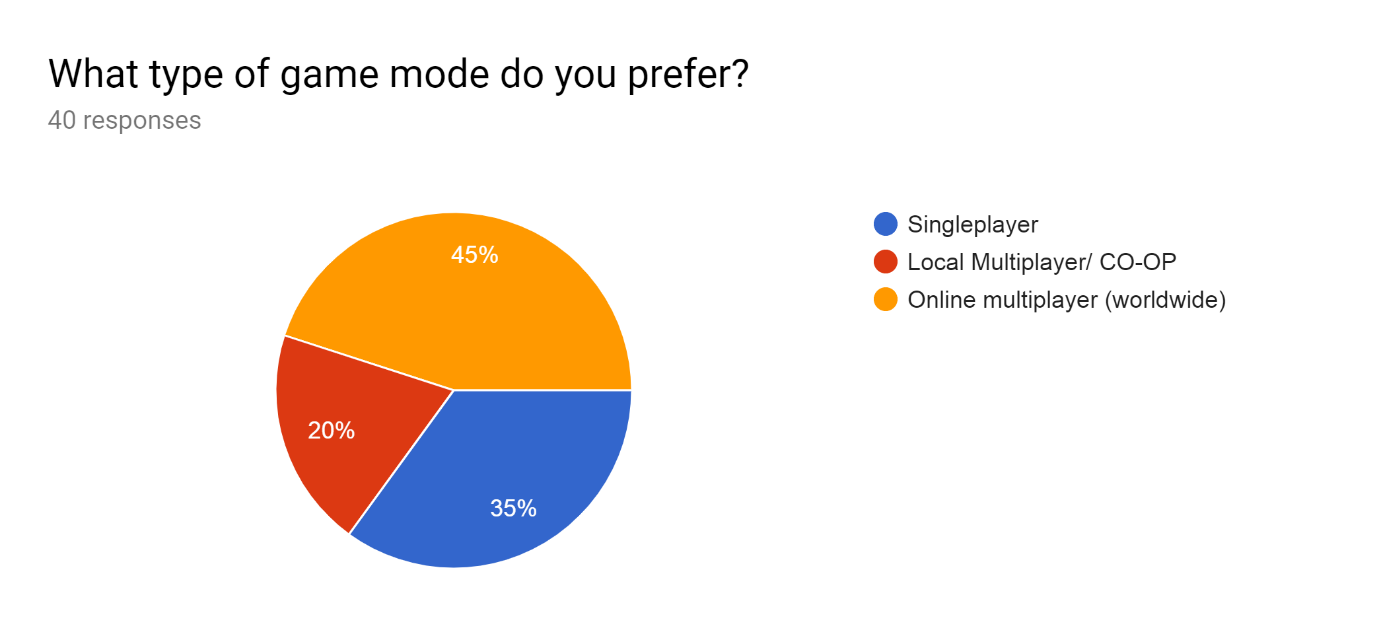
While looking at Rally-X, the two main things that I will be adding to my game is a feeling of progression and a difficulty setting. By adding a difficulty setting I will be able to appeal to a variety of people, from children who prefer easier games and those who aren’t experienced with games to those who want a very hard challenge. In addition, a difficulty setting on the Singleplayer game mode will make the game more fast-paced, so the player never gets bored and the game always feels intense. Moreover, the progression system will involve unlocking different colour schemes for Pac-Man, and for the map. A progression system will also give the user a sense of satisfaction once that unlocked something. Furthermore, particle effects and music will be added when the player collects power-ups or power ups, and also when the player eats another player or a ghost, depending on the game mode.

Questionnaire

This questionnaire provides me with the suitable information, so I can provide the best user experience.

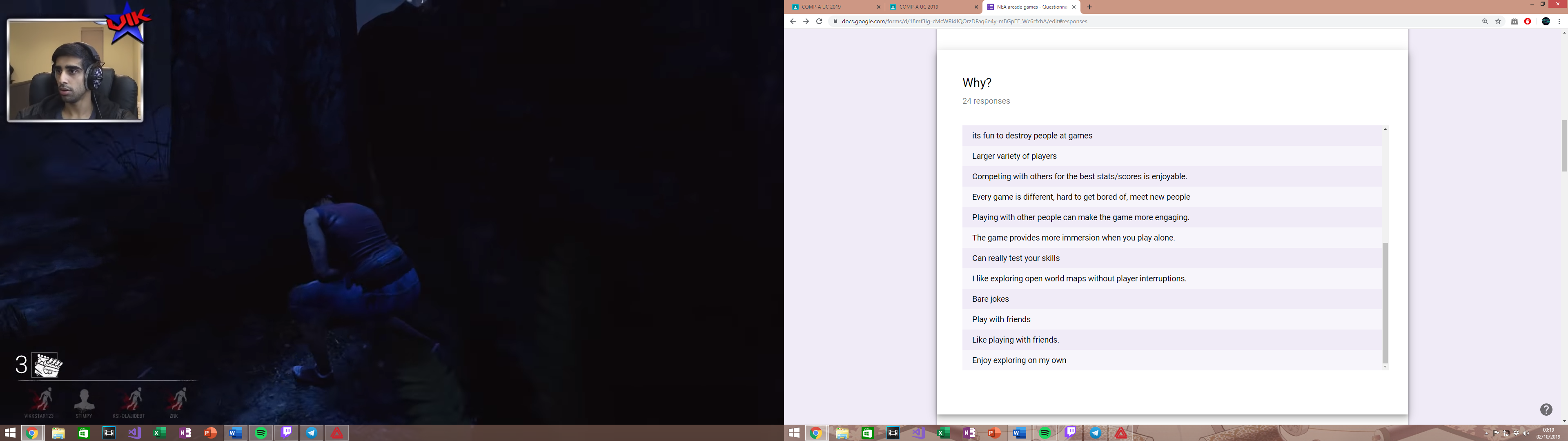
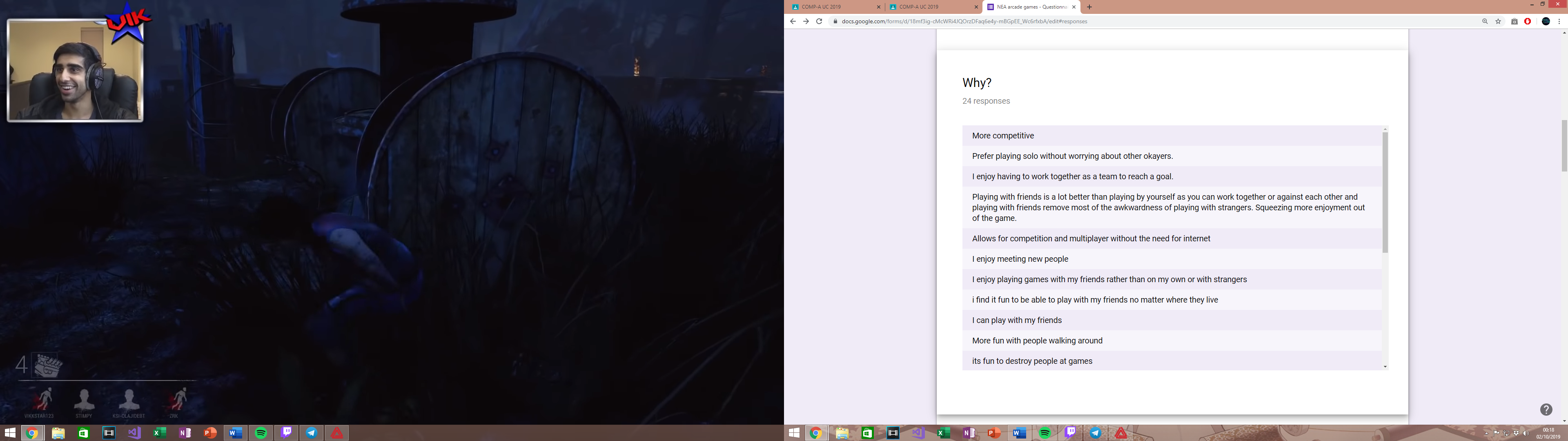


The majority group in this chart, are people aged between 14-17, and the second most are aged between 18-25. The rest of the groups only take up a small part of the chart. Therefore, the majority of my users will be teenagers, so I need to cater the game to that audience mainly. In addition, this confirms I do not need to make the game simple, as teenagers will most likely be able to pick up and understand any complex ideas the game may have.



The majority of the people that responded, chose online multiplayer, however there were still a large amount of response for both single player and local multiplayer. Therefore, the single player part of my game should still be popular amongst users. However, I will be concentrating on implementing more multiplayer features, and making the gameplay more complex than the single player part of the game. I will be implementing features such as an increase in speed for the ghosts and/or Pac-man as time goes on, in the multiplayer part of the game. First, I will be creating a local multiplayer game, where players can play along the same network, this will allow me to implement the most popular multiplayer features below (in the questionnaire). With a local multiplayer game, players will still be able to enjoy a lot of the things they enjoy with an online multiplayer game.

On the other hand, a majority of users would prefer an online multiplayer game instead, which I will be attempting once I have completed the local multiplayer and if there is enough time. When being attempted, I will have to focus on reducing latency between clients so user inputs are not delayed wherever they are in the world.

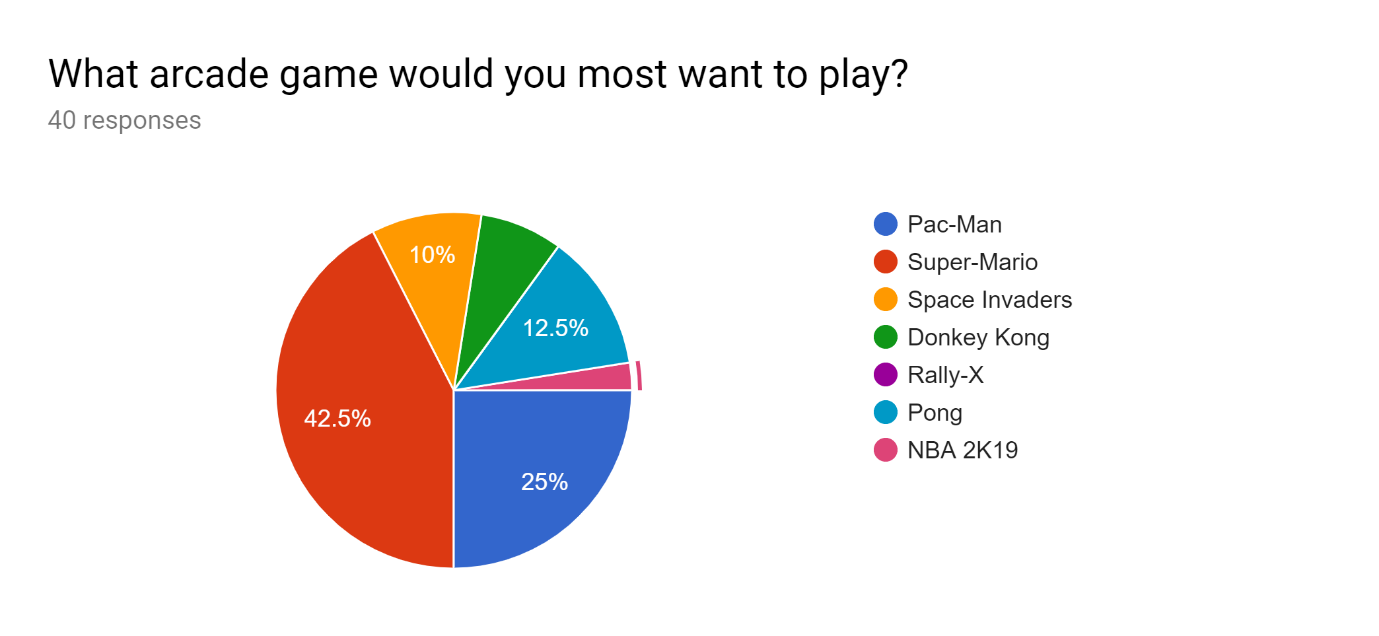


These responses are to why the user has preferred a certain game mode, from the above. This allows me to identify specifics aspects to each game mode where the user has enjoyed their experienced.

For the single player aspect of the game, users said how the game was more immerse when playing solo. With this information I will be adding specific single player features for the user, such as a customizable Pac-man, where the user can change Pac-man’s colour, and even change the colour of the map and the background. In addition, users said how they prefer playing without the interruptions of other players. With this I am to make the AI clever enough so the user gets a good and challenging experience while not having the interruptions of other users. Moreover, a user specified a reason by saying, “I like exploring open world maps without player interruptions”, which I can’t relate to my arcade game as it does not have anything to do with playing an open world game, so I will not be including aspects from an open-world game in my game.

For the local multiplayer aspect of the game, users said that playing with or against friends is much more fun, as it squeezes more enjoyment out of the game, while removing the awkwardness from meeting strangers. I will therefore aim to create an option where players can select who they want to join their game, via a lobby option or a friends list where they can invite specific users (friends). Another user responded, that they want to test their skills. This can be applied to all game modes, but especially for the Singleplayer part of my game as the difficulty setting can be adjusted. By increasing or decreasing the difficulty of my game, users will experience a different pace in the Singleplayer game mode, due to the cost function, which means the user can adjust it to cater to their skillset of arcade games.

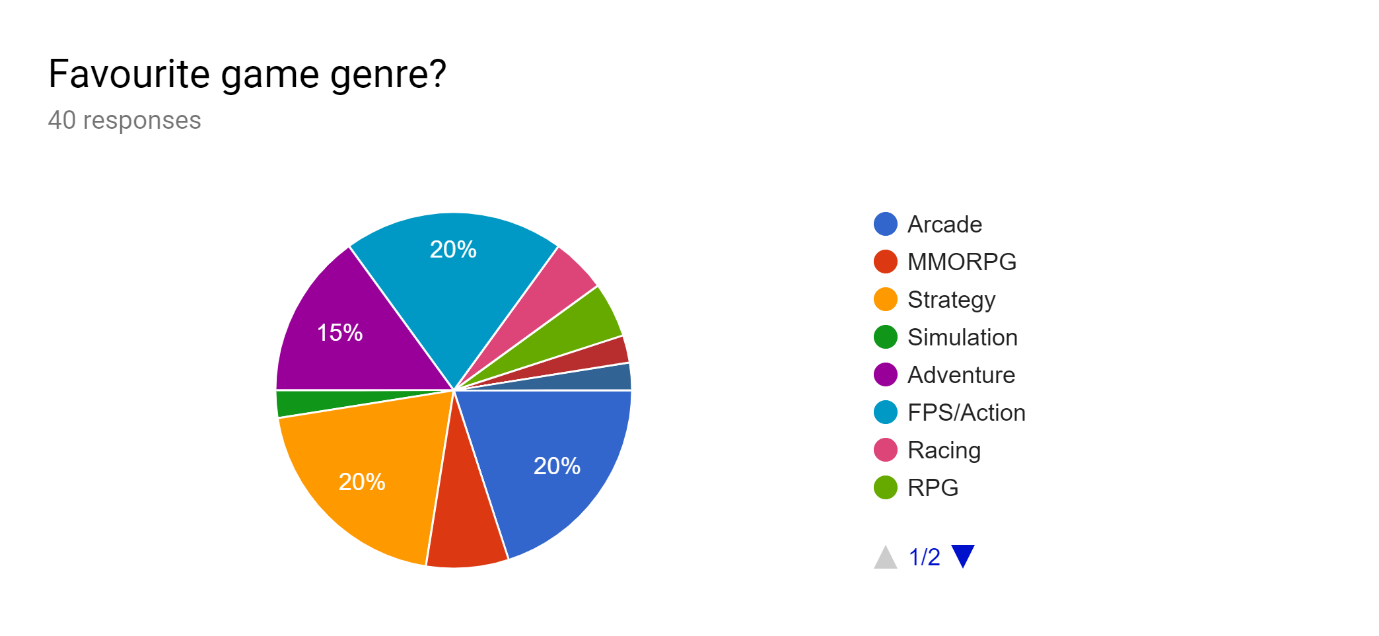
For the online multiplayer part of the game, users responded by saying, “I find it fun to be able to play with my friends no matter where they live”. If I attempt to create an online multiplayer part to the game, I will focus on reducing latency between players, to keep the game fun and fair no matter where you are in the world. Another user said they want to compete for the best states/score. I will aim to create an online leader board here, for all modes (single player, local multiplayer and possibly online multiplayer), where players can see their high score. Users also stated how they enjoy playing with a large variety of players, who each have different play styles. This is one of the main reasons I will be focusing on creating an online multiplayer version after I have finished my main focus, which is the single player and local multiplayer parts of the game.



Here users decided on which arcade game that would most want to play. I found out that the majority of users selected Super-Mario. This shows me, that I will need to aim to include various features from Super-Mario that are not included in the conventional Pac-man games. A feature that I may use are the powerups, one of which being the character enlargement one, this being the mushroom that Mario collects to grow bigger and stronger. I will use this, so whenever Pac-man obtains this power-up scattered randomly around the maze, the user increases in size and the ghosts will start avoiding him. This is actually a similar concept Pac-Man Battle Royale has, in which I have discussed above. In addition, another power-up I may use is the flower power-up which allows the user to spray fire in front of them. I might adapt this however, so Pac-man shoots a laser directly in front of killing either a ghost or an enemy Pac-man. A main part of super Mario was trying to collect three coins in three different locations during the level. I may incorporate this in my game while still maintain the authenticity of Pac-man, by spawning fruits at random locations, and giving the user only a small amount of time to collect them, and if the user collects all three, they may gain an extra life plus extra points to their score.

Furthermore, another thing I take from this chart is very few would want to play Donkey-Kong, so I may not include some features that Donkey-Kong may have. Donkey-Kong was a one of the most difficult games in its times (1980), as you had to concentrate hard to time the players jumps. This may be a reason my users would not enjoy the game, so therefore I will aim to not make my game hard but still add an option to increase difficulty if the user wants to, by adding a settings option for this.

Twenty-five percent of users said they would play Pac-man which is the second most on my chart. This lets me know that although 25% of my users would rather play Pac-man as an arcade game rather than any other, which is still a substantial amount to maintain an audience interested in a Pac-man based game, but I would want to try increase this number so I can get a larger player base. This means I have to include various of other features to try get players interested that my game (project) is unlike any Pac-Man game they have seen before. Therefore, while implementing my game, I will be taking constant feedback from various users of all ages, including my client/supervisor.



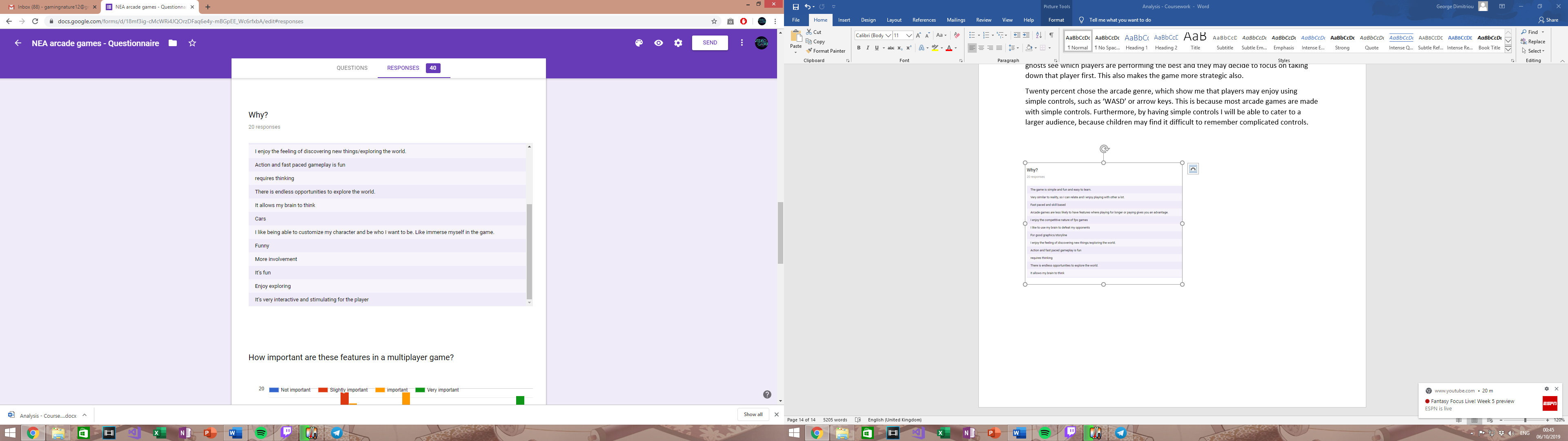
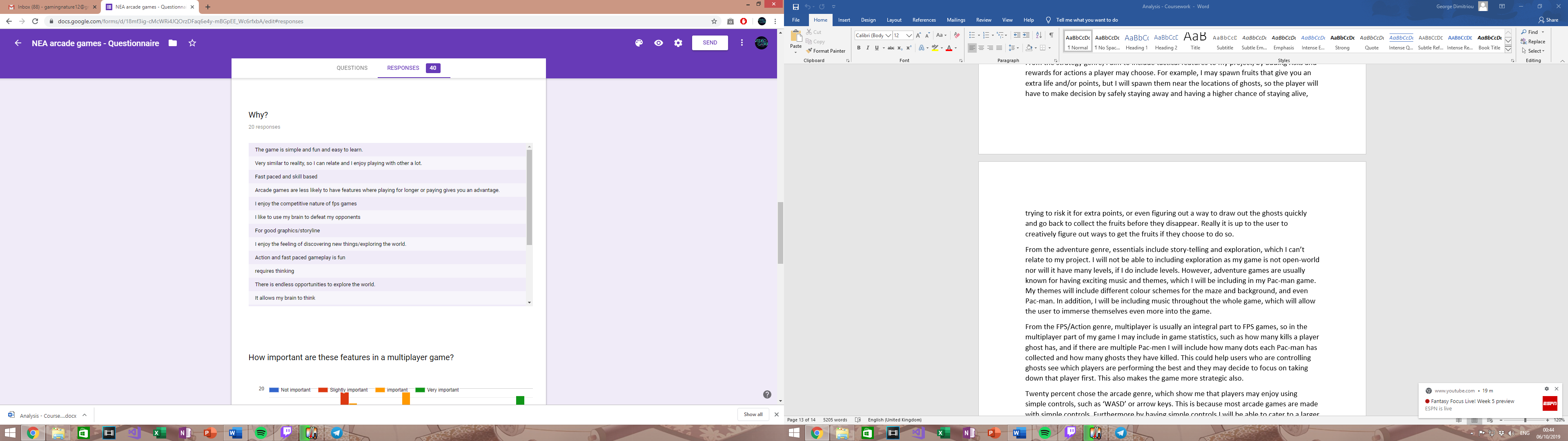
This chart showed me a variety of things, as there was no majority response. The leading responses though, were Adventure, FPS/Action, Arcade and Strategy.

From the strategy genre, I aim to include tactical features to my project, by adding risks and rewards for actions a player may choose. For example, I may spawn fruits that give you an extra life and/or points, but I will spawn them near the locations of ghosts, so the player will have to make decision by safely staying away and having a higher chance of staying alive, trying to risk it for extra points, or even figuring out a way to draw out the ghosts quickly and go back to collect the fruits before they disappear. Really it is up to the user to creatively figure out ways to get the fruits if they choose to do so.

From the adventure genre, essentials include story-telling and exploration, which I can’t relate to my project. I will not be able to including exploration as my game is not open-world nor will it have many levels, if I do include levels. However, adventure games are usually known for having exciting music and themes, which I will be including in my Pac-man game. My themes will include different colour schemes for the maze and background, and even Pac-man. In addition, I will be including music throughout the whole game, which will allow the user to immerse themselves even more into the game.

From the FPS/Action genre, multiplayer is usually an integral part to FPS games, so in the multiplayer part of my game I may include in game statistics, such as how many kills a player ghost has, and if there are multiple Pac-men I will include how many dots each Pac-man has collected and how many ghosts they have killed. This could help users who are controlling ghosts see which players are performing the best and they may decide to focus on taking down that player first. This also makes the game more strategic also.

Twenty percent chose the arcade genre, which show me that players may enjoy using simple controls, such as ‘WASD’ or arrow keys. This is because most arcade games are made with simple controls. Furthermore, by having simple controls I will be able to cater to a larger audience, because children may find it difficult to remember complicated controls.



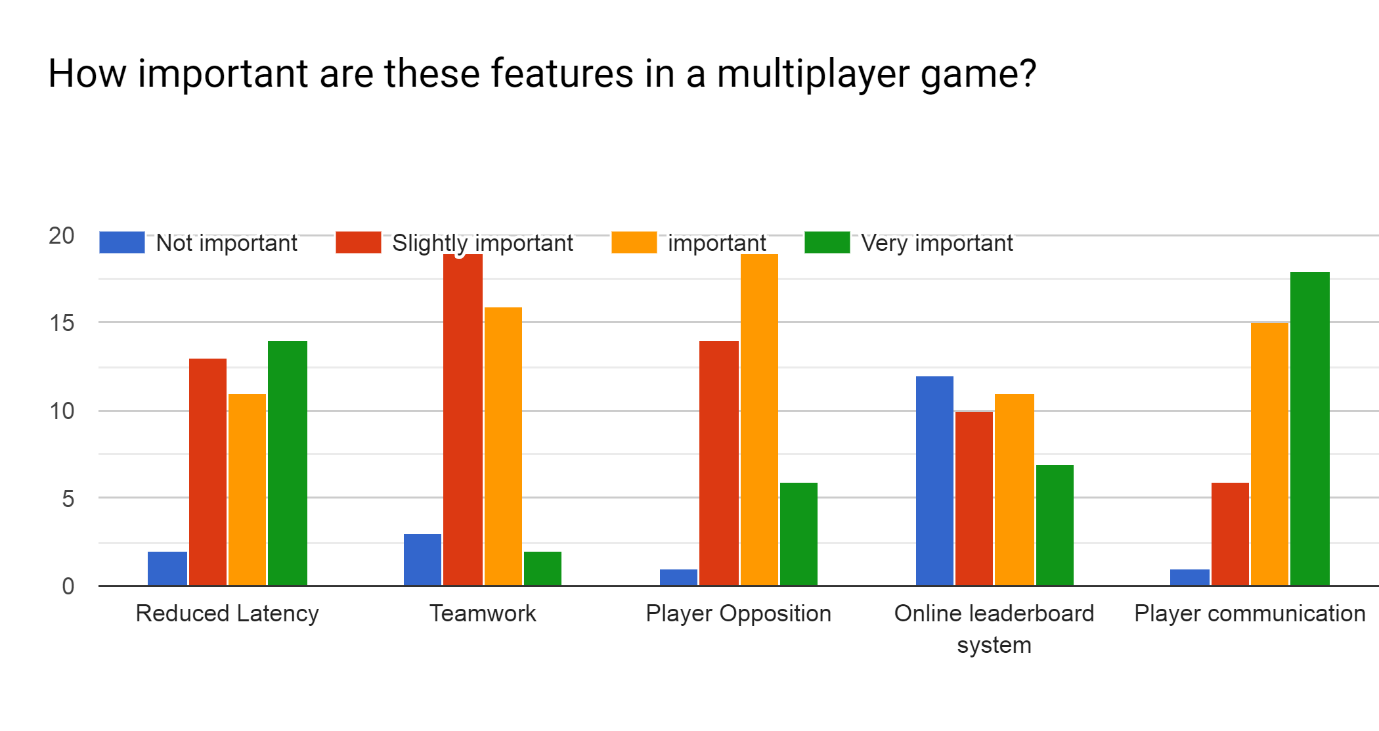
A response I received stated, ‘The game is simple and fun and easy to learn’. From this I will strive to create my games with simple controls and possibly give the user a choice between using arrow or ‘WASD’ keys.

A couple responses that users sent were, ‘fast paced and skill based’, and ‘action and fast paced gameplay is fun’. From these responses, I aim to add an element of fast gameplay to my project. In my first game mode, which is meant for Singleplayer, I will add a feature, where as time goes on and score increases, Pacman and the ghosts will increase in speed, requiring faster reactions and quicker thinking from the user. Furthermore, many responses from my users also said, ‘I like to use my brain to defeat opponents’, ‘requires thinking’ and ‘it allows my brain to think’. I believe by adding a fast gameplay feature as previously mentioned, already adds a challenging feature. In addition, the power-ups I will be adding will need to user to think how they will approach the game. The power-ups range from Pac-man being able to go invisible, to shooting lasers that kill ghosts, to even items that will give the player a disadvantage, such as slowness.

Many users seemed to enjoy game genres, where the main focus was role-play or adventure. The only feature that I will be adding that may relate to an adventure style game, is a progression system. However, anything else from those responses I can’t relate to the type of project(game) I am creating.

A response I got was ‘good graphics/storyline’. My game will not feature a storyline nor 3D graphics but I will aim to create unique animations that engage the play more. This is not a priority of mine, but it is still something I would like to achieve. Also, I believe 3D graphics in a Pac-Man game will lose its authenticity.

One user responded, ‘I like being able to customize my character and be who I want to be. Like immerse myself in the game’. Features in my game that will allow to customize Pac-man will be integrated into my progression system. For example, when a player reaches 100,00 points achieved all-time or a certain level (if I have a levelling system), they may unlock a new Pac-man colour or even a new animation. In additions, players may even receive an option to upgrade power-ups.

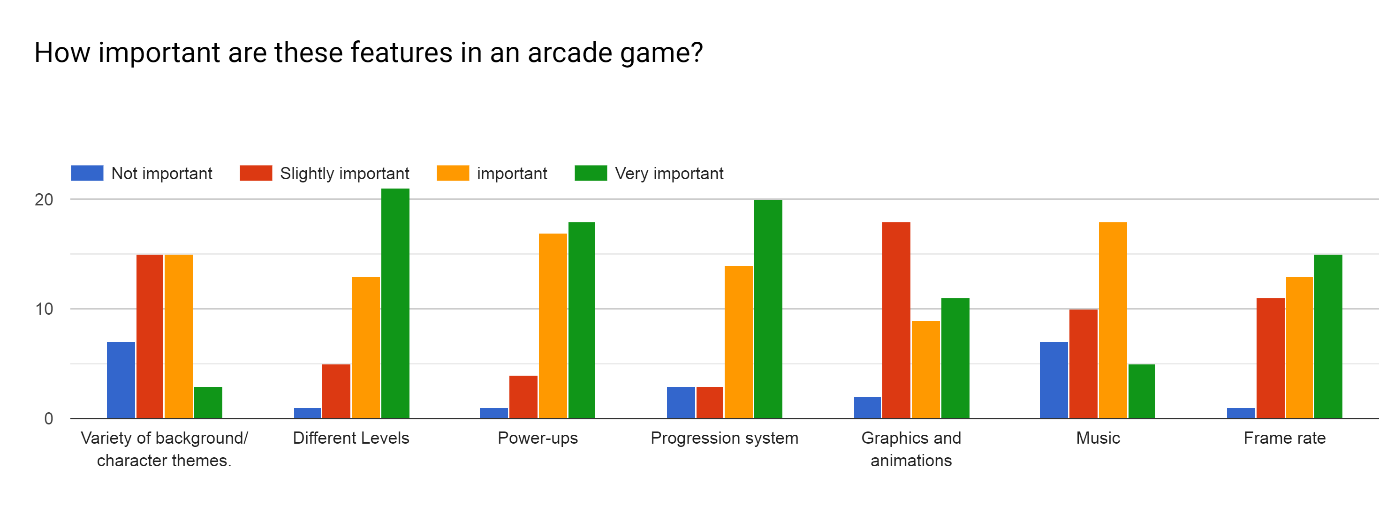


In the first section ‘Reduced Latency’, the responses were very spread out. However, almost all users saw it as somewhat important, which means when creating my multiplayer section, if and where I manage to create an online multiplayer rather than just local, I need to focus on reduced latency between clients. To reduce latency, I will have to make the client to most of the work rather than the server. In addition, I need to reduce the server sharing very big chunks of information often to all clients, as this can cause high cases of latency, which can create a bad user experience. This will be high on my priority list once if I enter the stage of creating an online multiplayer mode for my project, as all users found it somewhat important.

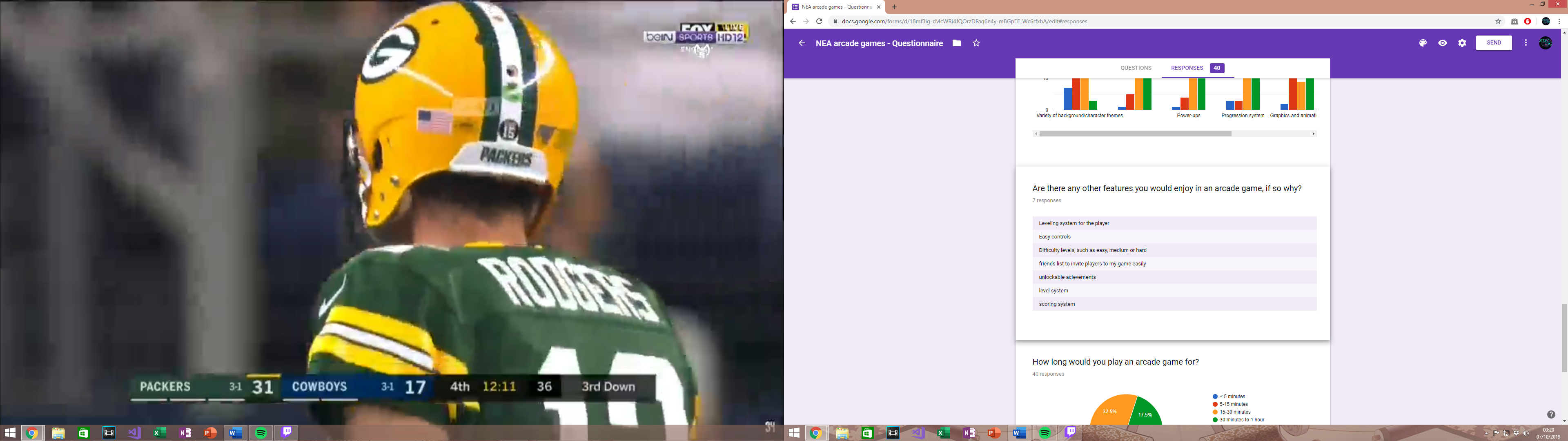
From this information, more users preferred having player opposition, than working with other players towards a specific goal, therefore I will be creating game modes that focus on player vs player rather than teamwork, due to majority of users prefer so. The second game mode where users have to avoid lava tiles, is focused on player vs player, and will have features that support it. These features will include the various powerups and a score counter to see who has collected the most pellets/dots which will make the game more intense and competitive. Players having to avoid lava tiles will also make it more intense. A power-up inspired by Pac-Man Battle Royale, will allow Pac-man to increase in size and eat the other Pac-men that are in the game. Moreover, a possible third game mode will actually include a mix of both teamwork and player opposition. This is so my game can accommodate to all users, so I must take into account those that prefer a multiplayer teamwork mode.

The online leader board system had a majority of users seeing it as not important. This means that I will not be placing it as a high priority, but I would love to implement it at some point, so users can see how they are doing against friends and others around the world, which I believe will add longevity to the game, as players will constantly want to climb the leader board. I will be able to have my users more immersed into the game and more likely to play for longer. The online leader board system will be able to be accessed via the menu.

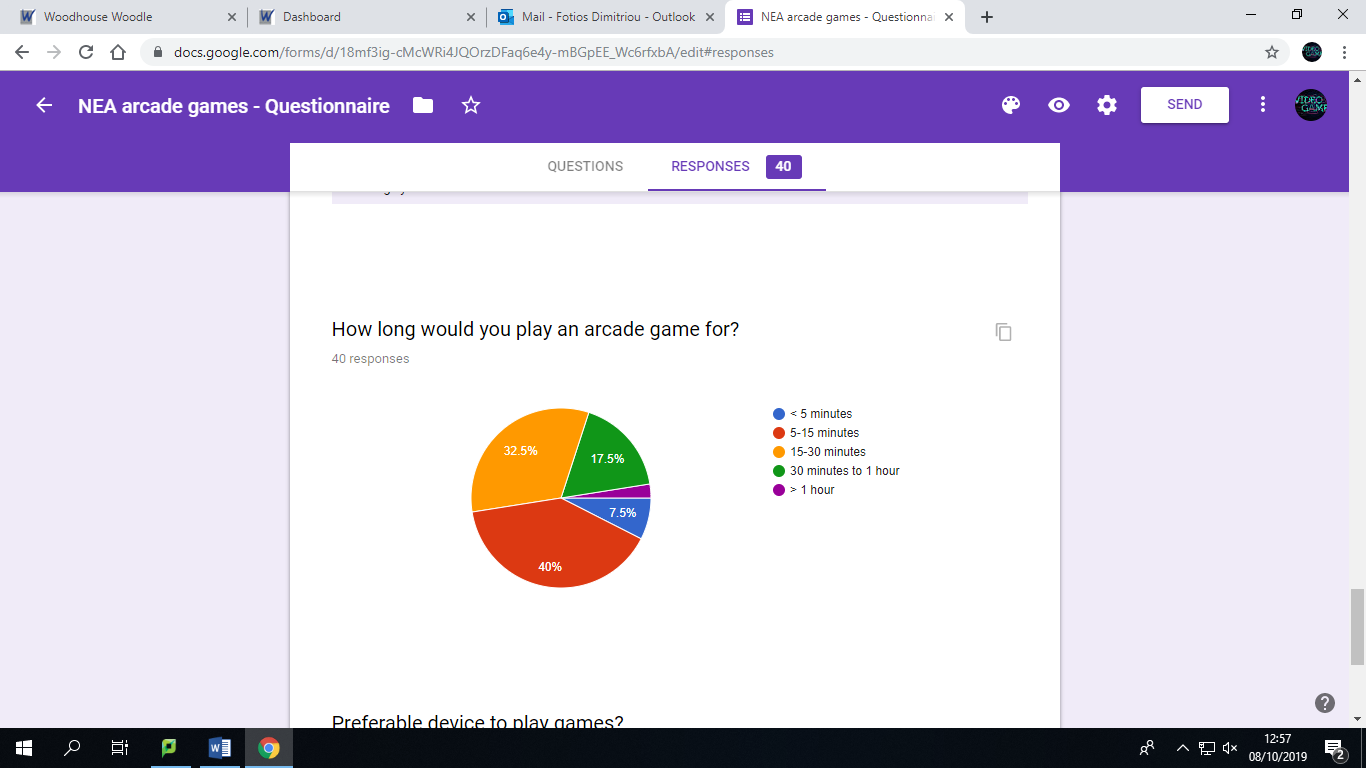
Player communication is something that users found very important in a multiplayer game. As my game will be mainly focusing on player opposition, communication will not be necessary, however when I create my third game mode, I will need to consider some form of communication as the game mode will consist of players working together. A form of communication that I believe will effective yet not too difficult, is a ping system. This is where users can ping their teammates at a point in the maze, and the ping will appear in the maze, depending on where the user pointed at.



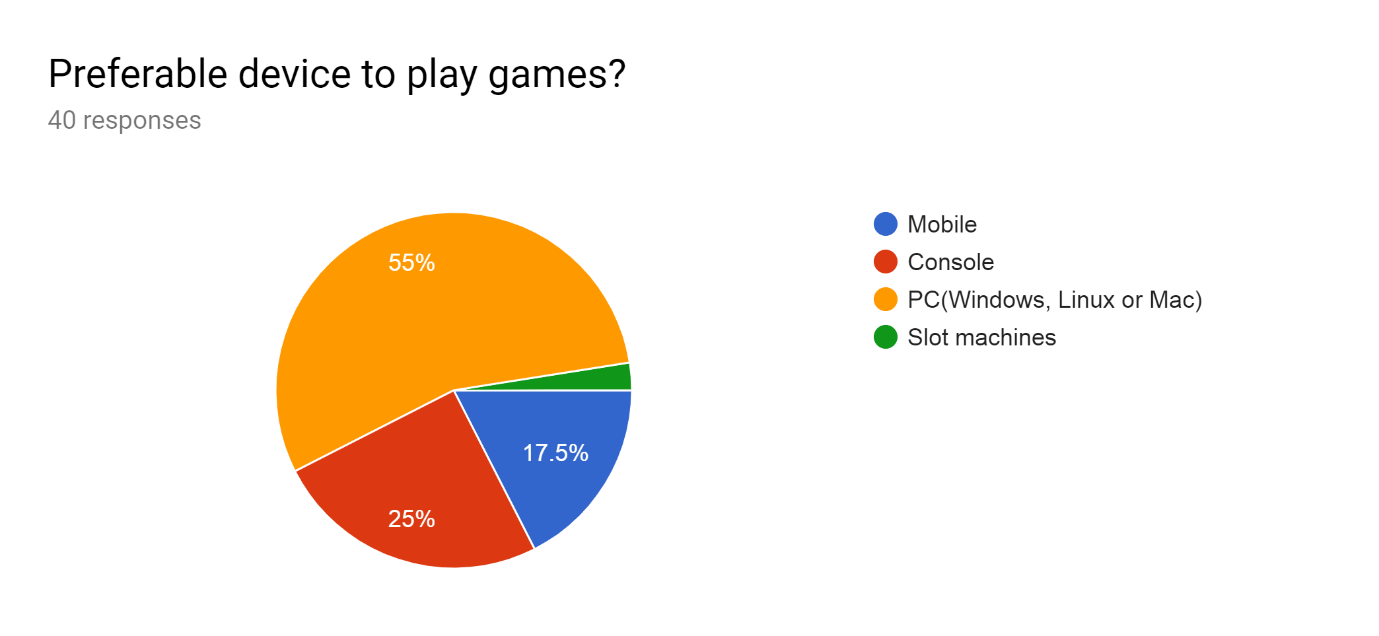
* Variety of background/character themes:
  + Majority of user found that having a variety of background and character themes was either important or slightly important, however, many found it not important. I will aim to add this into my progression system and allow players to unlock different backgrounds and themes, however it will not be a key component to my game.
  + I will hope to add a variety of colour schemes to Pac-man, as well as different styles of tiles to backgrounds, giving my game creative and new looks. Themes may include, lava(fire) themes, galaxy themes etc.
* Different Levels:
  + Most users thought that different levels are very important to an arcade game, so I have decided to make it an integral part of my project. The levels will each include different power-ups and also the pace of the game will change as gameplay will get faster through levels. In addition, my game may include a few mazes which may change throughout the levels, so players have to adjust accordingly.
  + Levels will definitely be included in the single-player part of my game. I will give the player the option to either play an endless Pac-Man game or a Pac-Man game with levels, on single-player.
  + Levels in multiplayer are not a part of my project plans, but if free time is available after I have finished my project, I will definitely take it into consideration.
* Power-ups:
  + Once again, the majority of users found it very important to have power-ups in an arcade game, which was already going to be a main focus for my project, as power ups will be included especially in the first game mode, and I will aim to include it in the second gamemode too. I aim to include somewhere between 3 to 5 power-ups.
* Progression System:
  + This is another response, where users also seemed to think it was very important to have a progression system in an arcade game. This is also an integral part of my game, as I want players to feel immersed into my game. In addition, I want my game to experience longevity, as players will continuously come back to try unlock new things and ‘progress’ through the game.
  + By having a progression system, my game will become more competitive as users will naturally want to try progress further than others (including friends).
  + Players will be able to unlock power-up upgrades and even different background/character themes for single-player.
  + A levelling system will also possibly be put in place so players have an extra go to try and reach (levelling up). This will also allow players to see each other’s experience when in a lobby.
* Graphics and Animations:
  + Most users only seemed to think graphics were slightly important for an arcade game. This could be because arcade games are not usually known for having good graphics as that is not the reason people mainly enjoy them. In addition, most arcade games also have basic animations, however some fairly new arcade games provide very unique graphics that engage the user more. For example, super smash bros and even street fighter. New Pac-Man games, especially ‘Pac-Man Championship Edition 2’ is very well-known for having amazing 3D graphics and animations, but it has received many complaints for losing Pac-Mans authenticity. This a reason to why I do not want to create 3D graphics. Another is because it is very time consuming and my main focus is making sure all aspects of the main game are functioning correctly before I introduce better graphics and animations.
  + I aim to have a tile-based graphics games. The tiles may even include depth which will make the gameplay look more impressive.
* Music:
  + Users categorized music as an important feature to add to an arcade game. I aim to add music throughout the game, from the main menu to when the user is actually in-game.
  + The music will be inspired by the original Pac-Man game, and will most likely feature the same or similar music.
* Frame-rate/FPS:
  + Majority of users categorized this feature as very important. I believe this to be very important, as I do not want users to have a bad experience while playing my game.
  + To increase FPS and allow all users with hardware and software limitations play my game, I need to focus on optimising the client side of my code. I do not want my game to contain unnecessary chunks of code that are not needed. In addition, by keeping my graphics 2D and tile based, I am not putting pressure on the user’s hardware, to keep their frame-rate as high as possible.
  + This should not be a problem to maintain high FPS for the user because arcade games are not very hardware dependent and do not require expensive machines.



* Multiple users stated that they wanted a levelling system. Since many players have wanted this feature, I am very much considering to add a levelling system. This levelling feature, will be a part of my progression system. As players level up, they may unlock various things which I have discussed above.
* A scoring system I have already considered, and will be applying it to my project, because an essential part of Pac-Man and my project is a scoring system. With a scoring system, players will be able to see how good or bad they are doing, with the addition of seeing how good or bad other users are doing (in the multiplayer section).
* Easy controls, is something I have already implemented into my prototype from my previous analysis, as I want to cater to a large audience, so children may find it difficult to play with complicated controls. Therefore, simple controls were the best solution.
* Difficulty levels, is also something I have considered as I want to cater to players with different abilities to play arcade games. Some players also prefer challenge, and some prefer to play the game chilled and easy. My difficulty levels will involve how fast paced the game is by using an inverse-logarithmic cost function, making ‘hard’ the fastest paced the game can get. It will also feature how often power-ups spawn, and how also how often bad items spawn.
* A user responded by saying that another feature they would like is a friends list so they can easily invite friends to a game. Players join multiplayer games, by browsing lobbies that players have hosted and joining the available ones. However, to make things easier I might add an invite option, where the host will have to type in the name of the user to invite them rather than a friends list. Once they invite the player, it will check for the name in the database and locate the user, the user will then be notified to join the hosts lobby and can decide if they want to or not. This will be an extra feature.



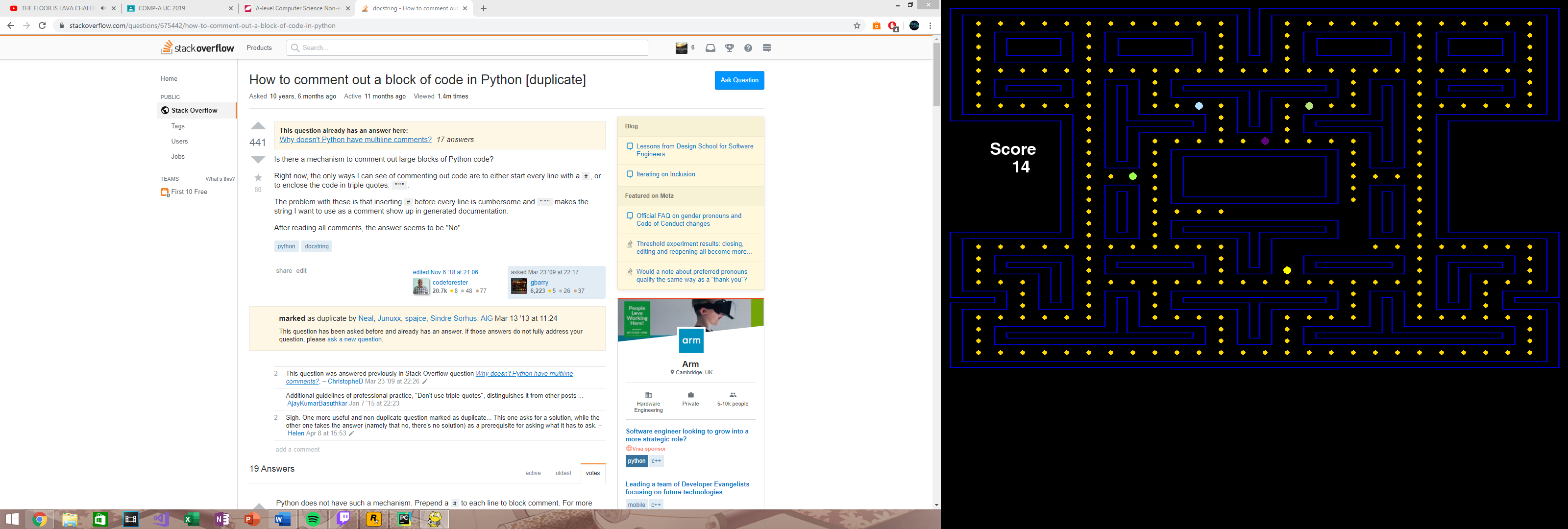
Majority of users would play an arcade between 5 to 15 minutes or 15 to 30 minutes. This shows that my gameplay should not be too long or too short. Having a difficulty level which adjusts how fast-paced the game is allows me to provide all users the opportunity to play the game for however long they want (playing on easy of course will allow you to play for a much longer time than hard difficulty).



Majority of users prefer to play games on PC, so I will focus on making my game playable on PC first, which is what I intended to do. I also considered mobile devices, but not many of my users preferred to play games on mobile, so by porting my game to mobile, it would most likely be a waste of time, which could be better spent on improving other aspects of my game.

Prototype

Over a 4 to 6 week span, I created a prototype of my game, to get an understanding of what may work, or what may need changing. The features I mainly added were, Pac-man’s basic movement using arrow keys, random enemy movement (but I have explored various algorithms that I will implement), a scoring system which gets updated when Pac-man eats the dots and collision detection for both the enemies and Pac-man. Video that explains my prototype <https://www.youtube.com/watch?v=1LXjLjGR8UE&feature=youtu.be>. GitHub for my project (only includes prototype at the moment). <https://github.com/FotiosGeorge/Pacman>



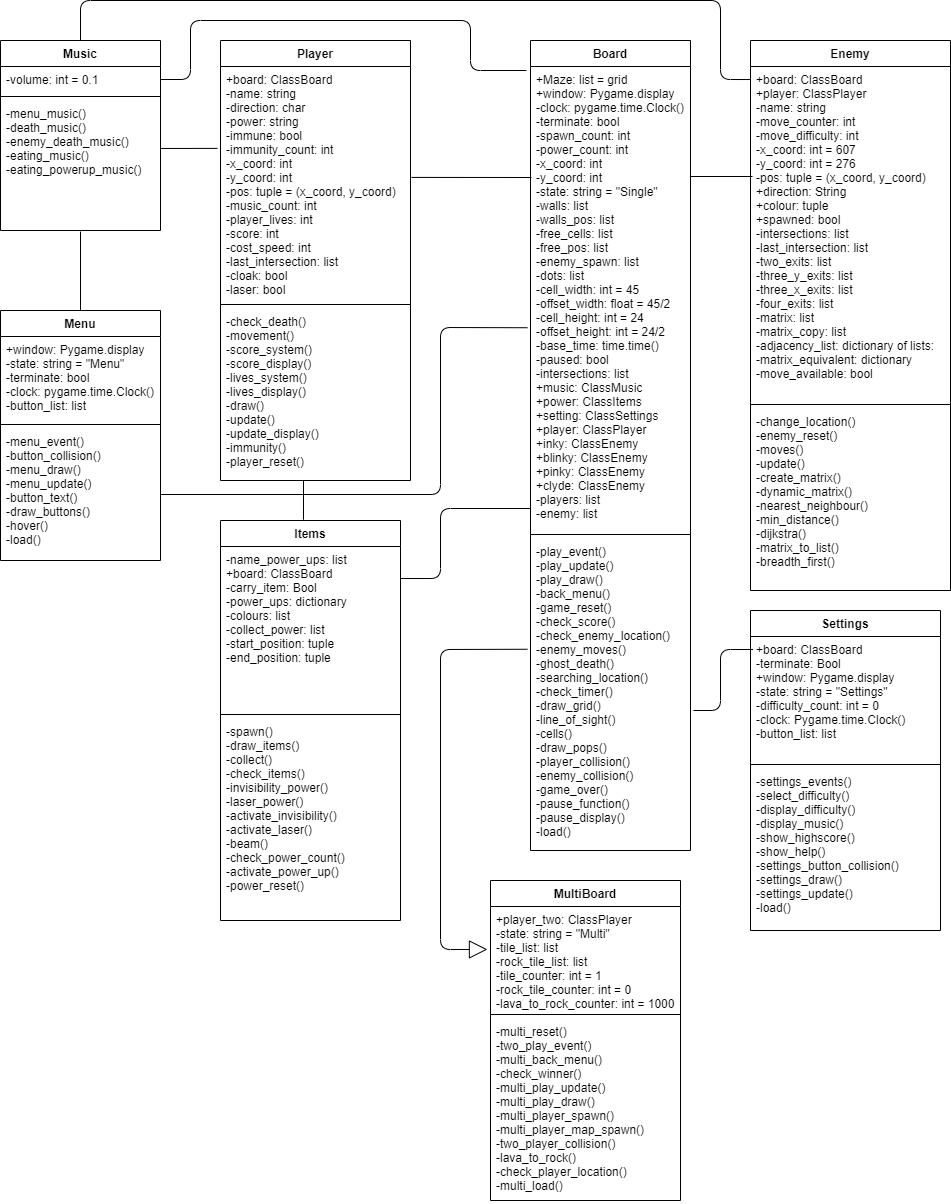
Screenshot of prototype.

Furthermore, I looked into various additional things, that I will be most likely implementing in the next step. By creating a Pac-man maze based off an image png and a grid array, I was able to understand that a better, more optimal way than what I have done so far, is by tile rendering. Tile rendering will allow me to create various levels much easier. Due to my research I also found out that a lot of arcade games are made using tiles.

I have researched the personalities of the four ghosts, “inky”, “blinky” “pinky”, “clyde”. They all have three modes in common: Chase, Scatter, or Frightened. Moreover, each ghost decides on their next move at every intersection, and each ghost has a personality which makes them slightly different to the other ghosts (therefore each one has a different algorithm). One of the ghosts will find the quickest route to Pac-man, meanwhile the other ghost tries to go to the nearest intersection where Pac-man is to try ambush him. The third ghost will move around the centre of the Maze, waiting for Pac-man to cross to the other side of the maze. Once Pac-man crosses to the other side, the ghost will gain extra speed every time, and if Pac-man is within a four-tile radius, the ghost will lock onto him. The fourth ghost will be a random ghost which only locks onto Pac-man when Pac-man has one life left.

* A new skill which I have learnt developing this prototype is how to use classes in python. I will now explain how the classes of my prototype works so far.
  + Class Board:
    - A function called “load” is called first which renders everything onto the screen. Then function “cells” is called which assigns the pixel position of the free cells, walls and dots to three different lists, so I can compare each one to the position of ghost and Pac-man.
    - A player object, and four ghost objects are created
    - A main loop is then called, which has three functions, “event”, “draw” and “update”. An event occurs, such as a user moving Pac-man, then using the other classes, Pac-man gets redrawn and so do the dots and the ghosts. This is done via the draw function, then are updated per frame via the update function.
    - This has made me realise, that as my project progresses, I will need states, otherwise all my event, draw and update functions will get kluged.
      * To solve this, I will need to create states. For example, when the game starts, the user will enter a menu, so the state has to equal ‘menu’, and when the user selects single player, the state would change to ‘playing’, and when the user selects multi player, the state would change to ‘multi\_playing’.
  + Class Player:
    - Includes position attribute.
    - Changes position of Pac-man and draws him in the new position
    - Removes dots from screen if Pac-man is in the same location
    - Manages scoring system using the scoring function, and displays it to a screen.
  + Class Enemy:
    - It includes the attributes, colour, direction, and the x and y value.
    - Changes direction and moves the ghosts’ pixels using the functions “changeLocation” and “moves”.

These are what my classes should look like when I have finished. I have mapped out class diagrams for the single-player and local multiplayer aspect of my game to give me guidance of how I will be laying out my code. It will also give me an idea on how my classes will be interacting. The relationship between my classes such as association, and inheritance.



I will be explaining the new classes I will be creating, which are shown in the class diagram above, but not yet implemented (they are not in my current prototype). All current methods and attributes are explained and shown in the YouTube video above.

* Items:
  + This class will be used for the power ups in my game.
  + The Class Items, contains various functions. These functions are used to spawn in the power ups onto the board/maze.
  + This class also contains each power up’s functionality. For example, it contains how the beam of the laser power up kills enemies. In addition, the class contains the functionality of checking if the player has collected the power up, by being in the same location as it on the board.
* Music
  + This class contains all the music for ever aspect of the game. There is main menu and settings music, music for when Pacman eats a dot and music for when either Pacman or a ghost dies. Furthermore, there is music played whenever a power up is collected, when the function eating\_powerup\_music() is called.
* Menu:
  + This class is used whenever the game is in state “Menu”.
  + This class contains all the buttons that when pressed change the game state, depending on the button. The functions draw the buttons to the screen alongside handling the events when the button(s) are clicked.
  + Also, should include the animation for hovering over a button.
* Settings:
  + In this class, the many functions (methods) will draw different settings options and handle each setting option event differently.
  + The settings options will include:
    - Changing the difficulty
    - Adjusting the music/volume
    - Observing the players current high score.
    - Observing the help section in the settings.
    - Possibly, being able to adjust the players Pacman skin/colour.
* MultiBoard:
  + This class stores the state “Multi”, when the game loop for the second game mode runs.
  + This class is used for the local multiplayer aspect of my game. It inherits all the attributes and methods from the class Board.
  + This class stores all the tiles (lava tiles, water tiles, rock tiles etc…) functionality. This includes, what events occur when the player interacts with these tiles, and when these tiles will spawn and how often they will spawn.
  + In addition, a new object, Player\_Two, is created to store everything the second player does.

**What each of the functions and attributes exactly do in detail in every class are all explained in the technical solution as part of the large amount of comments in my code.**

AI Algorithms

Adjacency Matrix Vs Adjacency List:

Adjacency Matrices and Adjacency lists are used to represents graphs. All searching algorithms I will be analysing for my AI involve traversing through a graph or tree.

An adjacency list is better used for a sparse graph and an adjacency matrix is better used for a dense graph. If a graph is sparse and I use an adjacency matrix, many matrix cells will remain unused and it will lead to a waste of memory.

An adjacency list is an array/list of nodes, and each node points to only the neighbouring nodes. The space complexity of an adjacency list is O(2n) -> O(n). However, time complexity is O(n).

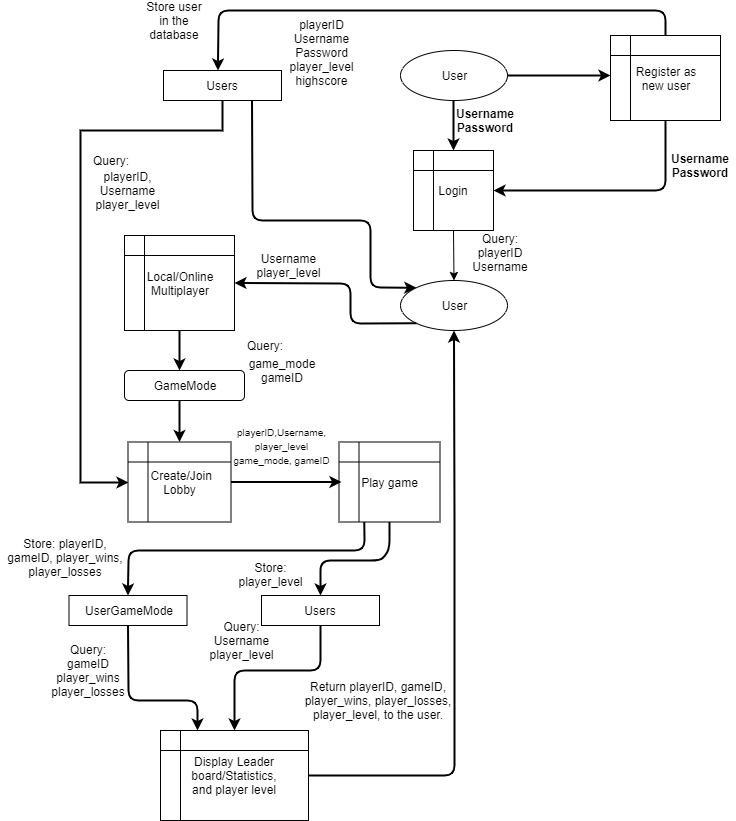
An adjacency matrix is a “n \* n” matrix to store weights of vertices between nodes. The space complexity of an adjacency matrix is O(). However, time complexity is O(1).

I will be using an adjacency matrix as my graph will be dense. The nodes in my graph will represent the intersections on the Pac-Man maze and the weights of the vertices will be distances between adjacent nodes. Furthermore, I am not worried about space complexity as my game will take up minimal space no matter what I choose, but I do care about time complexity as I want my game to run at peak performance, therefore choosing an adjacency matrix would be a better option.

I will also be using a **dynamic matrix algorithm** which adjusts the matrix (representation of the algorithms) depending on the position of other ghosts and Pacman.

* **Dijkstra:**
  + Finds the shortest paths between nodes in a graph.
  + Dijkstra traverses every single node in the graph and try finds the shortest path from source to destination. In Pac-Man, this algorithm will be used so the ghost(s) find the shortest/quickest path to Pacman.
  + Dijkstra initially sets every weight (distance) to other nodes as infinity (in the adjacency matrix). This is to imply that the nodes have no been visited. Weights are constantly being updated as adjacent nodes are being visited. Once destination node is found, the algorithm will trace backwards to store the order of nodes of the shortest path, from source to destination.
* **A\* Search:**
  + It is in an informed variation of Dijkstra
  + It is considered a “best first search” because it greedily chooses which vertex to explore next according to the value of f(v) [f(v) = h(v) + g(v)] – where h is the heuristic approach/function and g is the cost function.
  + A\* search achieves better performances by using heuristics to guide its search.
  + A heuristic approach is problem specific. For Pac-Man, my heuristic approach would be if another ghost is following close behind Pacman using the shortest path, the ghost that uses A\* search should not follow the same path behind the other ghost. Instead the ghost that uses A\* search should re-route using another open path to possibly try trap Pacman, otherwise that ghost would possibly be useless.
  + The time complexity is O. This is worse than the time complexity for Dijkstra, however the heuristic approach eliminates traversing many nodes.
* **Breath-First Search:**
  + BFS searches through a graph to find the destination node by exploring all of the neighbour nodes at the present depth before moving on to the nodes at the next depth level.
  + BFS uses a queue (First In First Out) instead of a stack to do the searching.
* **Depth-First Search:**
  + DFS searches through a graph to the find the destination node by exploring as far as possible down one branch, before back-tracking and exploring the next branch
  + DFS uses a stack for searching (First In Last Out).

For the multiplayer part of my game, I will be using “Twisted” (which is a framework and library) <https://github.com/twisted/twisted> . This is an event-driven networking engine written in Python. If I do not use “Twisted”, I will just be using pythons basic networking modules. I have not developed a prototype for the multiplayer part of my game, but I have drawn out a data flow diagram to get an idea of where the multiplayer section is heading.



* My database contains three tables:
  + Users:
    - Contains the players’ level, the playerID, username and password
    - It also contains the high score for single player.
  + GameMode:
    - Contains the different game modes and the gameID
  + UserGameMode:
    - Contains the players wins and losses per game mode and the gameID and playerID

This is the EDR:

UserGameMode

Users

GameMode

|  |
| --- |
| Users |
| playerID: int, Primary Key  Username: String  Password: String  player\_level: float  highscore: int |

|  |
| --- |
| UserGameMode |
| playerID: int, Primary Key, Foreign Key  gameID: int, Primary Key, Foreign Key  player\_wins: int  player\_losses: int |

|  |
| --- |
| GameMode |
| gameID: int, Primary Key  gamemode: String |

My database represents a many-many relationship. Therefore, I normalized my database and create a table in the middle that holds the primary key of both the Users and the GameMode, making it a composite key.

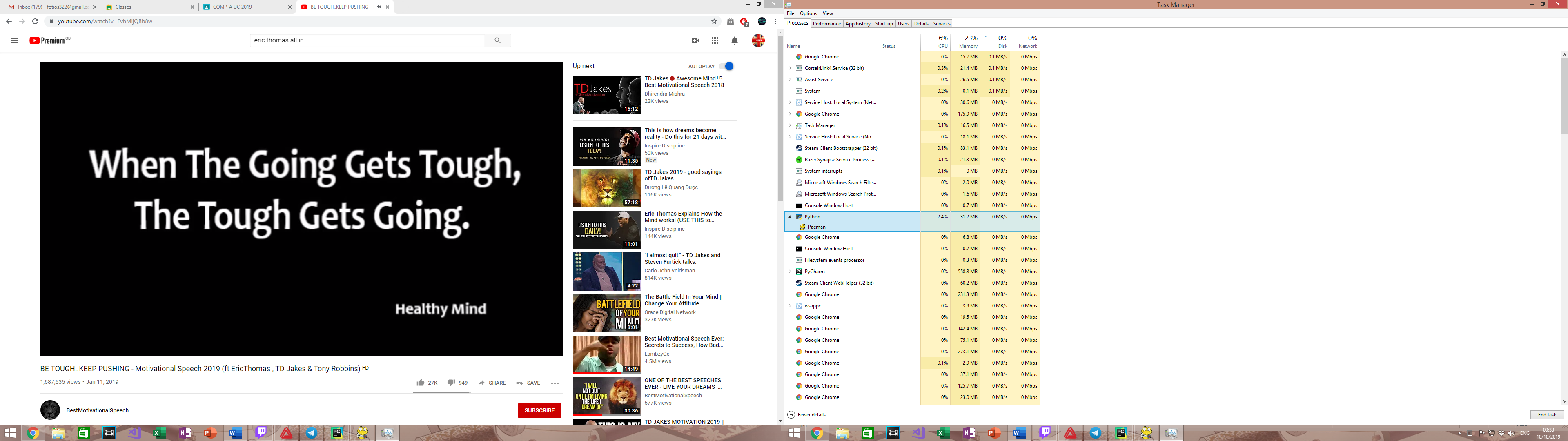
I will be running my database via MySQL database which will be running on a server that will hold all the user’s information in tables. The different entity attributes are still subject to change, but my data flow diagram has given me a rough idea of what attributes I need and when I need to store and query.

Having a GameMode table and the database in a normalized form, allows me to add in extra game modes without interrupting how the database works.

* User:
  + Contains the information of the user, such as username and player level.
* Creating/joining Lobby:
  + A player creates a room by selecting the game mode. This player is now the host and decides when to start the game and the host may decide to invite specific players. If the host does this, the database will look for a username that the host typed in to invite.
* Display leader board/statistics and player level:
  + At the end of every game the winner will be displayed on the screen, alongside the players level and the number of wins and losses they have on this game mode.

Hardware Constraints

* I want my game to be played smoothly by as many users as possible, so I will need to aim for low PC specifications.
  + CPU:
    - Low clock speed, around 1.6 GHz
    - Dual core processor
  + Graphics Card:
    - Integrated graphics card, managed by the CPU, for example Intel HD Graphics.
  + RAM:
    - I aim to make my game only require 200MB
    - PC’s with 4GB should therefore manage is, as some RAM would also be needed for the operating system, and system software.



My prototype only uses 31.2MB of RAM here.

Software Constraints

As I am only creating my project for PC, I need to make sure I satisfy all users who may have different operating systems. I will be focusing on making my game available for Windows, MacOS and Linux. This will not create complexity when programming, however I will need to compile it each time for every operating system. To do so I will need access to multiple devices with the various operating systems.

Furthermore, to test that each game is working on the different operating systems, I will ask friends and family to test the single player and the multiplayer, to see if users with different operating systems can still play with each other. In addition, I will be creating YouTube videos to show myself testing the game on each device.

* PC minimum specifications:
  + Windows:
    - Windows 7 +
  + MacOS:
    - OSX 10.6 +
  + Linux:
    - Ubuntu

Limitations

This section, I will be pointing out the features that are not essential or will not be possible.

Firstly, my project will be created using 2D graphics as pygame does not support 3D graphics. This means my animations will be in 2D as well, however if I create my game using tiles and tile rendering, I will be able to add depth in my tiles, but this will only be for visual effects. My game is therefore limited to 2D graphics as pygame does not support otherwise.

Secondly, due to time restrictions my project may only feature a local multilayer mode, but if it contains an online multiplayer mode it will have some limitations. For the online multiplayer mode, having more than four players would cause lag as the server would have to update too many clients and more players’ states at once. For a game developing company this would not be an issue, but for me, I do not obtain a larger server to support many players. I will therefore have to put a restriction on every lobby to four players. For local multiplayer, I would most likely not have this limitation.

Thirdly, a main feature of my first game mode, is that the speed of the game increases as the players score increases (every multiple of 100 the speed of the game increases using a cost function). If the online multiplayer game goes on for a long period of time, clients may experience some latency as the server will be trying to updating the clients much faster. This may cause difficulty for the user to play, but I will not be able to avoid this. If I make the second game mode local multiplayer instead of online latency will not be issue.

Lastly, creating more complex algorithms for the ghosts in my Pac-man game will be of high priority and will be my main focus for my project. If I have time, I will make my second game mode online rather than just local multiplayer. This is mainly because I am already learning many new things for this project, so by adding networking not enough time would be available.

Data Volume

I will be using a MySQL database running on a server to store data long-term. The tables will hold a record of all the matches, which will specify the what game mode it was using an ID (gameID), the users ID (userID), and if they won or not (player\_wins/player\_losses). If the user has chosen to play single-player it will update the users high-score in the database rather than players wins or players losses, as those are only for multiplayer. The high-score will be 2 bytes. So, if the user plays multiplayer (LAN) or single-player, one entry will be the same amount (specified below).

* Users table:
  + Username = 12 bytes
  + player\_level = 1 byte
* UserGameMode:
  + playerID = 4 bytes
  + gameID = 1 byte
  + players\_wins = 1 byte
  + players\_losses = 1 byte

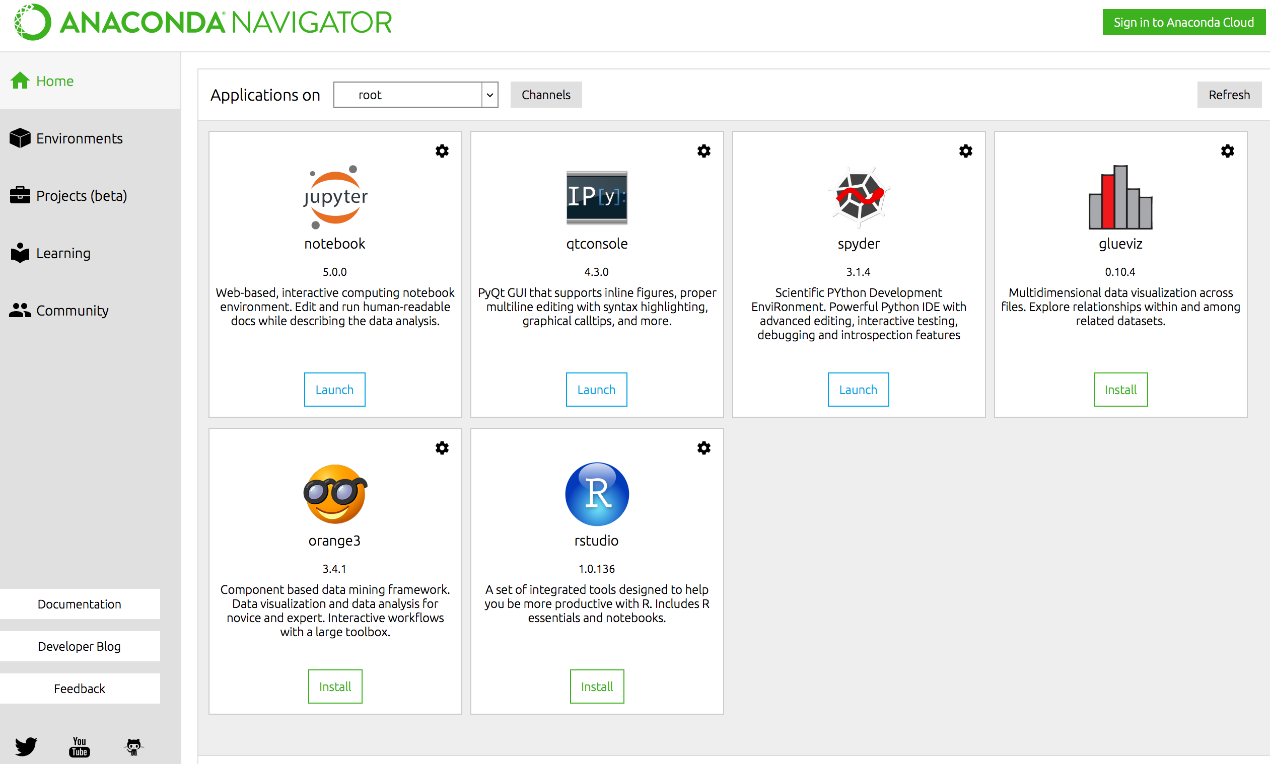
In total, 20 bytes are store in the database per game (for one player). If 1,000 games were played, that is 20,000 bytes (0.020 MB) stored per player. I also expect my game to need a maximum of 100 MB to 200 MB of RAM to run. I am therefore not concerned about data storage requirements.

General/Specific Objectives

* General Objectives:
  + Create a menu with a single player, local multiplayer, leader board and settings options.
    - When a player clicks on one of the buttons, the games state will change from ‘menu’ to ‘playing’
  + Create two game modes:
    - First game mode will be similar to the original Pac-man. The algorithms for the ghost AI will be created by me, and I will then compare them in many ways to the original Pac-man algorithms. I will be comparing how complicated they are, how smart the AI is and if my game becomes easier or more difficult with the AI I used. If this gamemode is play in local multiplayer there can be two Pac-men.
    - The second game mode will be a player vs player game mode. It will be an only local multiplayer game mode, where the player who wins has to survive the longest (the other player loses all three lives) or get to a specific score (500 points). Lava tiles will spawn temporarily to kill the players, and then eventually the lava tiles will turn into rock tiles. Players can only move over a rock tile, not lava tile.
    - Single-player will contain only the first game mode, meanwhile local multiplayer will contain the second game mode.
  + Progression System:
    - I will be creating a levelling system, where players will be able to progress through the game and unlock new things. For example, power up upgrades or new character colours.
    - Online leader board which shows high scores.
    - Users will be able to obtain high scores only through Singleplayer (so the first game mode).
  + Power-ups:
    - I will be creating between 3 to 5 power-ups.
    - These will include ones that affect the player badly or in a good way. For example, some may allow ghosts to travel through walls and some will allow Pac-man to teleport around the map or add immunity from other players.
    - An interesting power-up will be included in the multi-player part of the game that allows you to switch position with another random Pac-man user in the game.
  + Controls:
    - The game is played with the arrow keys.
    - I may give the player an option to use ‘WASD’ instead
  + Difficulty:
    - If a user clicks on single-player they will then be asked to choose a difficulty, which will not only decrease Pac-man’s lives from 3 all the way to 1, but the harder the difficulty the faster the pace of the game gets.
    - I will develop a cost function to increase the pace for Pac-man and the ghosts as time goes on. Cost function will change depending on difficulty. The cost function will be an inverse
* Specific Objectives:
  + Gameplay:
    - Assign each tile type a value, such as ‘1’, ‘0’ and ‘2’. ‘1’ being the walls, ‘0’ being the free cells and ‘2’ being the enemy spawn point.
    - Creating Pac-man and allowing him to move UP, DOWN, LEFT OR RIGHT.
    - Create a collision detection function so Pac-man can only interact with the free spaces and the enemy ghosts can only interact with free spaces once they exit out of the spawn points.
    - If both the ghosts and player are in the same location depending on what state the ghost is, the player will eat the ghost or be eaten.
    - A scoring system will also be put in place, every game each player will have their individual score being displayed on the screen.
    - Creating an algorithm/search method for each individual ghost. This will be complex as the main focus is creating the searching/pathfinding algorithms, and then the secondary focus is the local network game mode. These complex algorithms will be Dijkstra, Breadth-First Search, and my own algorithms, being, line of sight algorithm and dynamic adjacency matrix algorithm. The dynamic adjacency matrix algorithm adds complexity to my game and stops ghosts from directly following each other and taking other directions to try trap Pacman.
    - Music and animations will also occur throughout the game.
  + (LAN) Network System:
    - If the user is new, they will create a username and password which will get stored in the database. They will then be able to login with this username and password. They have to login to play single player also, as their player level carries on to multiplayer.
    - A user may decide to host a game by choosing the second game mode to play. Players will then be able to see this lobby and click on it to join.
    - A possible case may be, where I include both game modes as local multiplayer game modes, not just the second game mode.
    - The game will now start and users will play the game. Once the game is finished the players wins/losses will be updated in the database and the winner will be shown on the screen.
  + Settings:
    - Most importantly, users will be able to change the game difficulty (‘easy’, ‘medium’, or ‘hard’) in the settings.
    - Ability for users to change their Pac-man colour. Users will be able to unlock colours by levelling up.
    - Users will be able to increase/decrease the sound level (volume). Music will be played throughout the game.
    - Users will be able to unlock power upgrades for single player (possibly multiplayer).
    - Users will be able to seek help on how to the game works in the settings.
    - Users will be bale to view their Singleplayer high score also.
* Extras (if I have extra time):
  + If I have extra time, the first thing I will be considering and the most important feature I will be considering, is making my game online multiplayer not just local. The only guaranteed feature that is online is the leader board.
  + Allow the user to change controls to “WASD”.
  + Create particle affects when Pac-man eats dots.
  + I will aim to provide a third game mode where there are two teams, each consisting of a Pac-man and a ghost. The aim would be for the Pac-men to avoid the other teams’ ghost, and for the ghosts to try consume the other teams Pac-man.
  + Allow the user to invite people by username.
  + Different levels, which will mean different maps will have to be created. This would include map generation using a map/level generation algorithm. This is of low priority but it will be considered.
  + Allow a difficulty level to be chosen in multiplayer by the host.

Proposed Solution

As I will be programming in python 3.7, I will be using the Anaconda Navigator(<https://docs.anaconda.com/anaconda/navigator/>). The Anaconda Navigator provides a web-based application called Jupyter Notebook. Jupyter Notebook is interactive as it allows me to test code blocks and evaluate them. In addition, it allows me to see the input code, output code and any kind of visualization in the same document. Anaconda Navigator also me to customized my environment easily and add libraries easily.



I chose python because it provides large standard libraries which make writing code a lot simpler and easier for the user (very user friendly). This means less time is needed to code, which allows me to add in as many features as possible into my project within my given time frame. Furthermore, if I only create a local multiplayer mode, I will be using pythons basic networking modules (such as with sockets and threading), but if I decide to add on an online multiplayer mode, python will be able to provide amazing libraries and frameworks for networking, such as “Twisted” or “Asyncio”. Most other programming languages either do not have a network library or their network library is worse than Pythons.

I will be using Pygame (<https://github.com/pygame/>), which is a cross-platform library (allowing it to work on Windows, MacOS and Linux) containing modules designed for creating games. For example, it includes sound libraries which is helpful as I will be adding music throughout my game. The library handles other things for me too, such as keyboard input, game windows, rendering onto screens, drawing objects, and also calculating pixel positions. Moreover, it also allows me to update the game every frame easily. Pygame providing these features allows me to learn how to create more complicated path finding algorithms for the ghosts on the single-player part of my game (and multiplayer if there is not enough players). It also gives me more time to learn about networks locally and worldwide as I have no previous knowledge of programming networks.

If the multiplayer aspect of my game is only local, the user will be hosting the server on a LAN network using their computer. If my game expands to being online, I will be hosting an AWS server, which would cost $70.81 per month for a 15 GB RAM and 2 core CPU server. This may be too expensive so I might have to find another server provider that is cheaper.