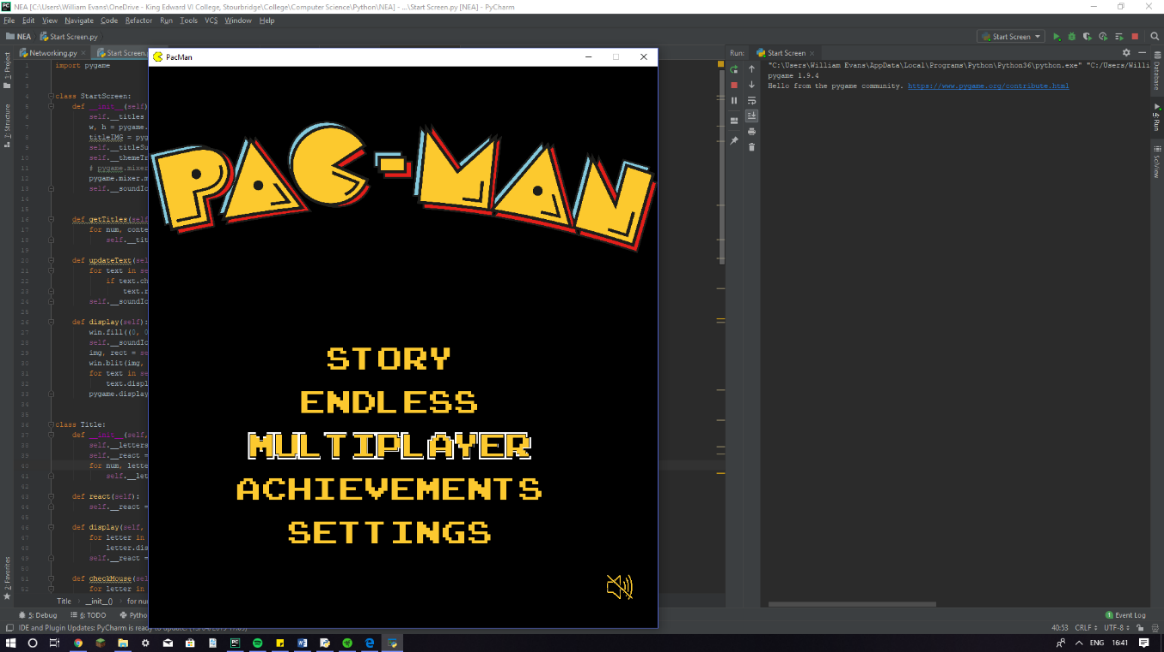
### Graphical User Interface

Arguably the most important aspect of my game is its graphical representation. It is what the end user will interact with and forms the basis of what the game actually is. For the game itself I will mostly use a sprite sheet, however I will also need to construct a GUI, using my own basic framework to form the various menu screens. This will include the following classes:

* Live word - This will control any large title like words that can react to a hovering mouse and mouse clicks.
* Icon - This will control any interactive pictures (icons) in my game.

These are the classes I have prototyped so far, however I feel I will need the following also:

* Word - This will be used for any basic words that do not need to be react to any input.
* Box - This class would simply describe a box that could respond to mouse input.
* Buttons - This could be used in the future to begin a game or ready up in a multiplayer match and would use the programming technique of aggregation by combining the word and box class.
* Input box - Again this class would be an example of aggregation (by combining the box and word class) and could be used for sign-up and login forms.
* Sliders - This class could be used for settings requiring a semi-continuous input. i.e. volume. Again, we could use aggregation here by incorporating the box and word class.
* Scrolling word - This class would most likely use the box and word class and would work by slowly revealing whatever sentence needs to be output to the screen.

These classes will be used to together to quickly build and efficiently run a graphical menu system. They can also be used in aspects of the core Pac-Man gameplay.

### Tutorial Mode

### The tutorial mode will be used to teach new player’s (especially the younger generation) how to not just play the game but also exploit its key mechanics. Following are a list of levels that I feel make sense at this time.

### The tutorial mode will be used to teach new player’s (especially the younger generation) how to not just play the game but also exploit its key mechanics. Following are a list of levels that I feel make sense at this time.

|  |  |
| --- | --- |
| **Level 1** | A tutorial on how to use the arrow keys to move Pac-Man and collect all the pellets in order to win a game. |
| **Level 2** | This level will introduce Clyde. |
| **Level 3** | This level will consist of Pinky. |
| **Level 4** | This level will introduce Blinky. |
| **Level 5** | I believe level 5 will be a good point to introduce power pellets. |
| **Level 6** | Level 6 will introduce inky and Blinky (as inky requires Blinky to function). |
| **Level 7** | This will be the final tutorial-like level that will feature all of the ghosts and allow the player to use all of the knowledge they have learnt/been given on the previous levels. |

This will begin with a cut-scene of Pac-Man, being chased around before eating a power pellet and then chasing after the ghosts. This will serve part of the tutorial (which is the general purpose of the Story mode). There will be 6 levels (taken from Pac-Man and the following 4 Ms. Pac-Man levels). The first level will consist of just Clyde mainly teaching the player how to navigate the maze using either WASD or the arrow keys and how to progress to the next level. The game won’t tell the user exactly how they AI of Clyde works but it will give subtle hints. The second level will then feature Pinky, the third: Blinky and the fourth: Inky. The fifth and sixth levels will then be progressively more difficult (similar to the later levels on the endless game mode). On the 7th and 8th levels the player will play as the ghosts and will be tasked with hunting down Pac-Man. This will serve as a kind of tutorial for the online multiplayer game.

### Endless Mode (4)

Endless mode will be most akin to the original Pac-Man game and will feature a high score system unlike the Story Mode. All 4 ghosts will be present in the game and their behaviour will change based on 3 variables:

* The current time in the level
* The actual level the player is on
* The number of pellets remaining

This table shows the first two bullet points pretty well and it’s this table that I will base my game on as to not lose the mechanical beauty of the original game.

There is also a change to the frightened mode. When Pac-Man eats a power pellet the ghosts change turn around and enter frightened mode. As the levels progress the amount of time spent in this mode decreases until (at level 13) they no longer enter frightened mode when Pac-Man eats a power pellet; they simply change direction and continue to chase Pac-Man.

The number of pellets remaining changes Blinky’s behaviour. When the number falls below 20 Blinky changes behaviour and becomes what is commonly known as ‘Elroy’. Elroy has increased speed (to the match the player) and will continue to chase the player even when in scatter mode.

When the number of pellets drops below 10 Blinky will gain another speed boost so that he is now noticeably faster than Pac-Man.

These trends will increase for an infinite number of levels as long as the player still has lives left.

### Multiplayer

Multiplayer is without a doubt the most important game mode within my project. While easily being the most difficult to implement feature, it will certainly improve the replay-ability of my game. As no other version of Pac-Man has yet contained networking, it will add to the uniqueness of my project.

When a user clicks on the multiplayer heading on the start screen, they will be treated with two options: ‘Create game’ and ‘Join game’. Creating the game will display the user’s local IPv4 address and allow other users to connect via the ‘Join game’ section, by entering the IP into the GameID box. Each lobby will have 5 slots available. The host will start as Pac-Man and each subsequent client that joins will be assigned a ghost. Of course if not all slots are filled up when the host starts the game, the remaining slots will be filled by AI.

Due to the limitation in data transfer with python sockets, all the sprites will be handled locally, including any interactions with other characters. The server (host player’s computer) will simply receive keyboard inputs, validate the inputs and calculate a new position (x, y) for the sprite which can then be sent to all players. This trading of information will occur 120 times a second (twice the frame rate) to minimise stutter. Early prototypes have deemed this method attainable, especially over my original method of pickling all objects and sending them to all clients.

When the user clicks on multiplayer they will be given the choice between local play (2 player) and online play 2 – 5 players. If they select local multiplayer they will be given a list of variables that they can change. These will include:

* Game mode (classic, co-op, 1 v 1 etc)
* Pac-Man speed
* Pac-Man boost (On/Off)
* Pac-Man boost (power)
* Number of ghosts
* Ghost speed

This mode will run with one player using WASD keys to move and the other player using the arrow keys with the spacebar being used to boost Pac-Man or in the case of co-op the b and m keys respectively.

If the user chooses online multiplayer instead they will be greeted with another splash screen with two options: ‘create game’ or ‘join game’ q. Creating a game will cause the server script to be run which will fetch the users IP. The user can then give this to their friends who can use it to join the game. This mode will support up to two people per computer and up to 10 users in total (5 of which will be spectators). The options will be to the person who created the game as in the local multiplayer section with the addition of being able to assign people characters and remove them from the match etc.

There will also be another option for joining a match which is online matchmaking. This will use a central server which holds a database of player accounts. When a request is made to the server, the server will match add the player to a host (which will be the first player to request) and redirect future requests to that host until either the game is full or the host has started the game. Kicking, and adjusting modes and settings will not be available in matches made by the server.

The core gameplay of the online multiplayer version will be points-based i.e. first to 5000 points etc. There will be one player starting as Pac-Man and 4 players playing as the ghosts. Pac-Man must collect all the pellets to earn points. He can also eat ghosts (using power pellets) to gain points. If he is caught, the ghost that caught him will become the new Pac-Man (and gain points) and another round will begin. Ghosts have a restricted view, they will only be able to see a 3-tile radius around them (this increased to 5 if they eat a fruit).

### Map Creator (7)

This mode will use a similar algorithm to the one later in the analysis. It will allow the player to click on the screen to paint walls and power pellets. Providing the map complies with the original rules for a Pac-Man level (which will be clearly displayed to the user) the level will be saved to the user’s online account (in a database connected to the server).

### Sound effects (10)

My game will include all the original sound effects such as:

* Intro music
* Pac-Man mouth opening
* Pac-Man munch
* Eating fruit
* Eating ghost
* Extra man
* Intermission
* Death

It will also include music for the start screen which will be a remix of the previous sound effects, in order to add another level to the start screen.

### Database (11)

Both the client and the central server will have a database. The client database will operate as follows:

After a classic or endless game ends the user will be prompted to enter a name (or simply skip with the spacebar) irrespective of the score they achieved. This prompt will also tell the user if they got a high score (Top ten) or not. This is just a local high score database and will store only the username, the level they got to, the time spent on each level, amount of fruit collected and of course the score.

The central database will have a relational structure so that it can link user accounts to their high scores/games. Unlike the local database, the server will only store the total score and level reached by the player as to save space. If the user is connected to the internet, they can view global leader boards as well as look at their own local records. The central server database will also contain a table for the user’s account details, high scores, matches (records all the players in each match and who won) and achievements.

### Achievements(12)

Achievements will be a great way to add replayability to the game (if trying to beat your own high score isn’t already enough). There will be a function for each achievement that is called after every frame to check its requirements (so it is imperative that it be extremely efficient).

Achievements will be (but not limited to):

* Eat your first power pellet
* Eat all the ghosts with one power pellet
* Trick Pinky by sending her down another path (hidden)
* Play 10 endless games
* Play with friends
* Spectate a whole match
* Eat 100 ghosts
* Get to level 5 without losing any lives
* Eat fruit

Locked achievements will can be viewed in the achievement section of the game unless they are ‘hidden’ in which case the achievement description will only be visible once the achievement has been unlocked.

When the user gets one of these, a pop up will appear in the corner of the screen and it will be added to the achievements section of the game. You will also be able to check up on friends’ achievements in this section.