Design

The layout of my main menu

The main menu will be start with an image with the games title “Hercules - the untold story” where the user will need to press space to continue, this will be told to the user in text at the bottom of the screen.

Once out of the game screen, the user will be sent to the pre-game menu where the user will need to either “log in” or “sign up.” These will be the two button widgets where once pressed will open the respective screen. The two screens, once all the players details are inserted into the input field, will then allow the user to choose which one of the three game files they would like to play on, alongside the option to delete and rename the save file. Once these steps are completed by the user the game will load the user’s data from the games database and finally start the game.

# The layout of the game

Once in the game, the player can control the character “Hercules” and can continue completing the game story from where they last left off (if they previously played in the same game file). If it is the first time playing in the game file, non-playable characters (NPCs) will communicate with the user on what to do and provide story context through text boxes at the bottom of the screen as seen in other retro games such as Pokémon. In addition, I will provide NPC (non-playable characters) pictures to the left of the text box so that the user can easily identify which character is speaking. Once communicated and the player is in free-play and is out of the text animation, the user’s heads up display (HUD) will be visible.

In the Game Window the user can see the character’s health bar, ammunition, and currency count at the top left corner of the screen. In the top right corner, there will be a pause button, once interacted with, the pause menu will be displayed on screen and the game will be paused. From the pause menu the user can access the Map, Progress, Character and Option screens by a Row of buttons with the respective names displayed.

The Map screen will display the game map, including the greyed out undiscovered parts of the map, the game map discovered, character’s current location, identified collectible items.

The Progress screen will display the Gods to be saved, Bosses to be defeated, special powers to be unlocked. If the user wishes, these can be clicked on to find out more about the Greek mythology behind them.

The Character Screen will display the character, his cosmetics, and the gear available to equip onto him. These include different skins, swords, and bows. Some of these items may also be clicked onto for the user to find out more about the Greek mythology behind them.

The Options screen will display the different settings that the user may change around to customise his playing experience. Such as audio volume, mutable sounds such as music and sound effects, difficulty settings, button layout, and a button to go back to the main screen.

## Stakeholder Input

The user will use the keyboard and mice as input. The user will only need the mouse to navigate the menus with the pointer and to aim Hercules’ bow. The keyboard is the main input device for movement, abilities, and interactions.

i.e., W, A, S, and D keys for movement up, left, down, and right respectively. Space bar to jump. Shift key will be the alternative option such as sprint instead of walk and special attack instead of the normal. E key is for melee attacks (sword) and the Left click for ranged attacks (bow and arrow). Q key is for interactions with NPCs and the map.

## Output

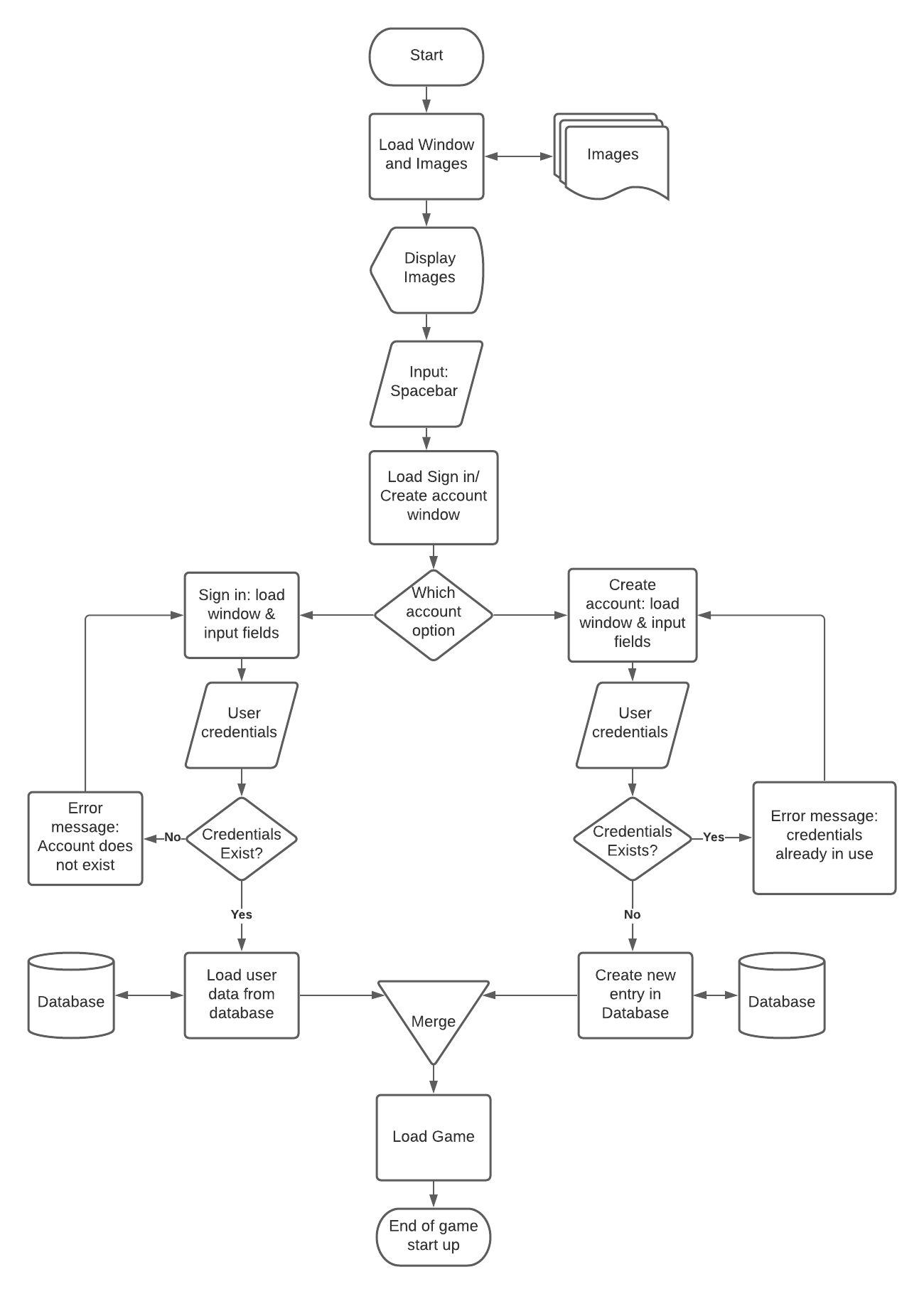
The output for the inputs outlined above are changes in the x and y coordinates of the player held in the player.location attribute; vertical momentum for the jump input held in the player.momentum attribute; movement speed for the shift key; the players interactions on screen such as attacking and dealing damage to enemies and text boxes for when interacting with NPCs.

## Usability features

Usability features are features which make playing the game a smoother and more fun experience for the user. I will include many of these features such as having a count for the user so that they know how many arrows and coins they have; a health bar to indicate if they are on low health or not. Speech bars when NPCs speak. Damage numbers when you attack enemies to show much damage you dealt to the enemy and a large health bar when fighting the bosses to indicate how close you are to defeat them. A feature I will implement which is unique from other Metroidvanias is the ability to change game difficulty mid-game which makes the game more playable for users if they are finding it too hard at a certain point. Another unique feature will be the use of a cursor in the game to aim the bow, in traditional Metroidvanias the player can only fire their weapon horizontally, vertically and diagonally at a 45-degree angle; whereas in my game, the player is able to fire their bow in all 360 degrees.

## Algorithms

I will be using Dijkstra’s Algorithm to find the path that the arrow will be travelling when the user presses the Q key. the start point will be the centre of the character and the end point will be the position of the cursor when the Q key was pressed.

This is a Flowchart of the start-up of the game. This can be implemented as a sub-routine as each shape can be executed in a single line or by calling to another function that I can create to make this sub-routine clearer. For example, another sub-routine I can create to make this simpler is a sub-routine called “isUser” that takes user credentials and returns True or False depending on if the user credentials exist or not, this would be useful as this can be used multiple times, as show in the flow chart. On the penultimate shape “Load Game”, the program should load the game files of the user that has signed in, including empty files. Each user is allowed three different save files. Alongside the option to start these save files, the user is given the option to “rename” and “delete” the given save file. Once the user has chosen a file to play on, the data is loaded from the database and the game is loaded from their most previous save location.

## Code structure

I will structure my code in multiple files, each file being a class. This makes my code very reusable and organised. From my main code, I will call out to these other scripts to create objects and so then I can manipulate them in the main code. For example:

* Tile class {dictionary of CSV values linked to each tile image}
* Map class {}
* Objects class
* Physics class
* Character class {movement, dealDamage, takeDamage}
  1. Player class {location, last save, items, coins, arrows}
  2. Enemy class {playerTrack, itemDrop}
     + The different types of enemies
* Database class – a list of functions e.g. isUser

## Database Structure

I will structure my database with a main table followed by a series of small tables. The main table will contain all every users account information including their username, password and security question, for when users forget their password, all in one record followed by a key unique to each user.

Each user will have their own table, created using their key as the table name. Each player’s table will have only three records, each record being a save file. Each record will have data saved such as the last save point, the heart, arrow, and coin counts, and inventory.

#### Account Validation

When creating a new user, firstly the program needs to check if the input fields are filled in and that the data is validated to the appropriate requirements, e.g. Having a special character in the password. Then the program needs to check if there is already a user with the same username, if there is another account cannot be created. Finally, the program can create the record in the main table with the user credentials. Then the user’s table is created using the key from the main table.

When a user is signing in, the program needs to check if the account exists using the credentials from the input fields. Once an account is accessed, the key will be returned to then access the user’s table and then save files’ names will be displayed on the screen for the user to choose which one they would like to play. When the user selects an account, they whole record associated with the name will be accessed and used to load the game from the last time the user saved.

When saving, the data is collated by all the players attributed and then used to overwrite the current record.

## Rooms

The map will be created using a 2D array which will hold a value which will be a room, for example (0,0) for the top-left room in the layout. Each value will be held in a dictionary which will relate to a CSV file, each file being a room layout. Each value CSV file will relate to a different tile, again by another dictionary. Reusable rooms will not be given coordinate values, instead names. For example, “Save” for the save rooms and “Armory” for the armory rooms. Each one will look the same and having one reusable file for each is more efficient.

## Music

I will have music playing in the background throughout the game with sound effects (sfx) with every interaction on screen. I will have three soundtracks to play, with each one playing in different scenarios. Harmonic soundtrack for the Temple (save) rooms and God rooms, an action soundtrack for when there are enemies in the room, a calm soundtrack for when enemies are in the room and finally, an intense soundtrack for boss battles.

## Key Variables

A few key variables to outline in this document are the player location which will be held as an attribute of the player object i.e. player.location, which will be used to move the player on the screen and updated as the user uses the movement input keys.

Another variable, player.momentum for when the player is jumping and will be used to alter the players y coordinate value to move them up the screen every frame; the momentum value is also decreased every frame to show a fluid jump motion. Another value (a constant of gravity) will decrease the y value of the player every frame that he is in the air and not touching a tile beneath him to make him come back to the ground.

Count variable such as health, coins, arrows will be used to keep track of how much resources the player has and if he’s allowed to do certain actions. For example, if the arrow count is zero then the player will not be able to fire an arrow until he purchases or loots more arrows.

The last save variable will be used to determine the room coordinate the player has last saved in and this will be updated in the database when the player saves to be accessed when the player wants to load their game in another session.

These are only a few but the most vital variables that I will be using in my project, so it operated and runs successfully.

## Testing Methods

I will be using an iterative development to complete this program. This allows me to decompose the game into sub problems to solve independently. Each problem will be addressed and coded in its own prototype. This will allow me to build and quickly test each prototype to check if the new code works as it should be; it also allows me to be flexible and identify new requirements for my game.

Each sub procedure, class and section of logic will be tested as they are made to ensure that they work to a good standard for the current prototype. At the end of a prototype, I will conduct a larger test on my prototype to help identify any bugs, issues or improvements for the next prototype.

At the end of development, a final test and play through of the game will be conducted by my stakeholders. The reason I will let my stakeholders test is so that they will replicate how an average user will play my game. And therefore, will be able to find n identify issues and bugs that I wouldn’t have tried to look for as some of my stakeholders do not come from a programming or gaming background.

I will be using both white-box and black-box testing for all my sub-routines to ensure that they function correctly when values are given in their operating range but also to ensure that they do not execute on malicious data outside of their input range. When testing, I will record the input, output, and if the output was expected. To help debug logical problems, I will print out values before, after and during a process to check how they are changing.

This is especially important in the subroutines that access and overwrite the database tables as I need to ensure the players data is saved properly because if false or wrong data is stored in them then the save file is no longer valid and cannot run properly and the user will need to restart the game from the start which will leave my stakeholders frustrated and make them not want to play again.

#### Checklist for all subroutines

My testing checklist will be;

Test for: (Example: Health [capped at 100])

* A normal value - (65)
* Minimum value - (1)
* Maximum value - (100)
* Value outside of the range - (0 & 101 & -1)
* Value of the wrong data type - (“Helf")

This will ensure each subroutine is tested thoroughly for correct execution for all values and that no errors arise further down the line of my development that will need me to go back to and amend the subroutine.

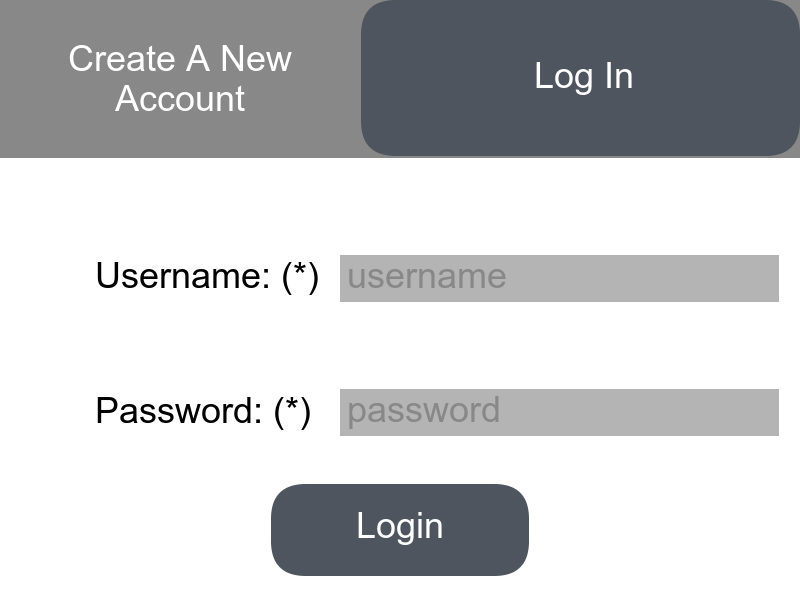
My checklist of functions to test:

* Log in verification
* Creating account verification
* Save file saving and retrieving
* Loading character in the last saved position with correct item counters
* Options menu
* Character menu
* Character movement
* Character animations
* Character collisions
* Particle collisions
* Tile collisions
* Enemy collisions
* Item collisions

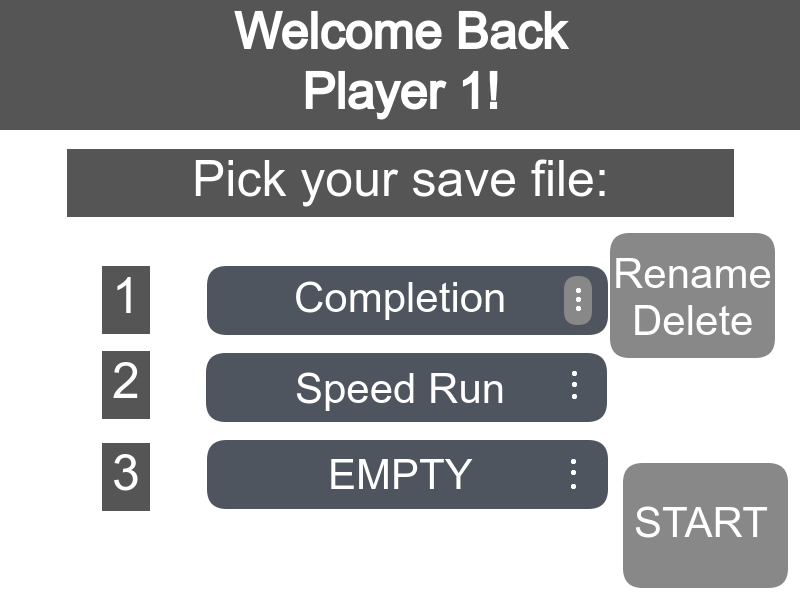
## Story board



The starting image will be a front cover with the game title. Which will prompt the user to press the space key to start the game.



The user will now need to choose one of the options “login” or “create”. Login for existing players and create for new players. The users will need to input their credentials which will need to be validated against the database until they’re let through to the next window.



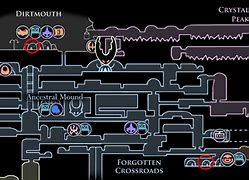
The user will need to select the save file they would like to play in and press the start button. They also have the option to rename and delete certain save files.



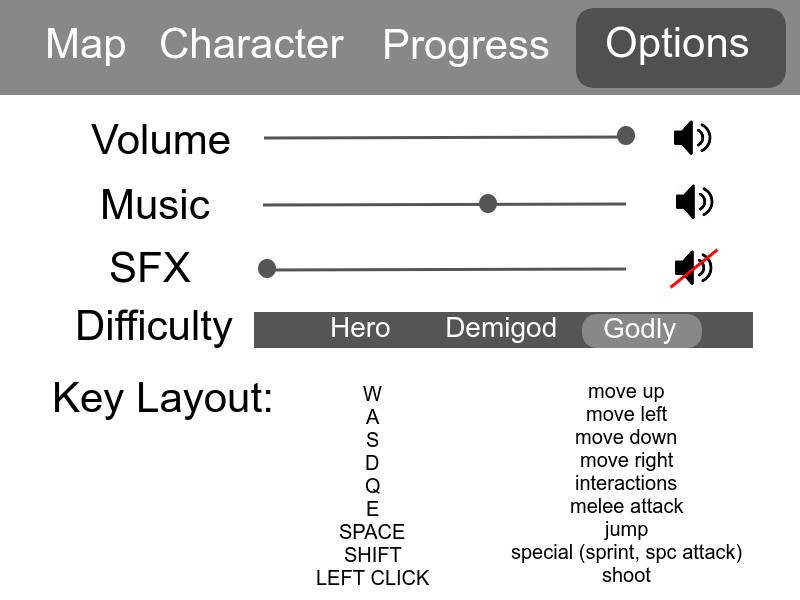
This is an image taken from google images. This is an example of how my game is going to look like, the 2D layout with tiles and character are similar components to my game.



This is from a game called Hollowknight. This is how I want the NPC interactions to look like, with a text box of what the NPC is saying to create a story.



This is from a game called Hollowknight. This is how I would like my map to look like, an interactive image which changes as the player moves from room to room and located different landmarks of the map.



This is how the menu will look like with the different tabs of Map, Character, Progress and Options. This is the options tab whew the user will be able to customise their different settings to their liking. In the menu the user will be able to customise their character and check out different gear in the character tab. They can look at the map and see where they’re at and what they’ve discovered. The user will also be able to check how much progress they’ve made in the progress tab and look at everything they’ve discovered about Greek Mythology.