

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Contents

Problem Identification.....	4
Problem Definition	4
Project Brief.....	5
Stakeholders.....	6
Justification of computational approach	6
Why my project is suited to a computational solution.....	6
Clear initial situation.....	7
Clear goal.....	7
Clear inputs and outputs	8
Clearly defined logic	8
Initial Research	8
Research of similar systems and ideas	9
Mini Militia – War.io by Appsomniacs LLC	9
Stick Warfare: Blood Strike by Team Modernator, MinGyu Choi	10
Warforce – Online 2D Shooter by Megatu	12
Research of the algorithms used.....	14
Research of the GUI used.....	16
Stakeholder interview	17
Questions	17
Stakeholder interview transcript	18
Limitations.....	21
Essential Features.....	22
Success Criteria	25
Data Structure Diagram	27
Class Diagram.....	28
Designing main menu screen/ GUI	29
Stakeholder interview	29
Summary of findings	30
Designing player inputs.....	32
Stakeholder interview	32
Summary of findings	33
Designing in game GUI.....	33

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Stakeholder interview	33
Summary of findings	34
Designing in game interactions/ collisions	35
Stakeholder interview	35
Summary of findings	35
Conclusion from all interviews.....	35
Justification of sprint 1.....	36
Success Criteria	37
Design.....	37
Pseudo code.....	39
Validation table.....	39
Variables and data structures.....	40
Test Data during iterative development.....	47
Copyright.....	48
Appendix	48
Aims for this sprint.....	76
Success Criteria	77
Justification of sprint 2.....	78
Pseudo Code	79
Python Development	79
Program Code	79
Areas of complexity	79
GUI	83
During iterative development.....	83
Stakeholder	85
Evaluation	85
Next sprint.....	87
Copyright.....	87
Appendix	87
Aims for this sprint.....	156
Success Criteria	157
Justification of sprint 3.....	157
Pseudo Code	158
Python Development	158

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Program Code	158
Areas of complexity	158
During iterative development.....	160
Stakeholders	161
Evaluation	161
Next Sprint	162
Copyright.....	163
Appendix	163
Aims for this sprint.....	214
Success Criteria	216
Justification of Sprint 4	216
Pseudo Code	218
Python Development	218
Program Code	218
Areas of complexity	218
During iterative development.....	222
Stakeholders	223
Evaluation	224
Next sprint.....	225
Copyright.....	225
Appendix	225
Aims for this sprint.....	260
Success Criteria	262
Justification for Sprint 5	262
Pseudo Code	263
Python Development	263
Program Code	263
Areas of complexity	263
During iterative development.....	265
Stakeholder Questions.....	266
Evaluation	267
Next sprint.....	268
Copyright.....	269
Appendix	269

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Aims for this sprint.....	295
Success Criteria	295
Justification of Sprint 6	296
Pseudo Code	297
Python Development	297
Program Code	297
Areas of complexity	297
During iterative development.....	300
Stakeholder Questions.....	301
Evaluation	301
Copyright.....	302
Appendix	303
Post-Development Testing.....	324
Stakeholder feedback	328
Evaluation of User Requirements	331
Evaluation of Usability Features	335
Potential improvements	337
Appendix	338

Analysis

Problem Identification

Problem Definition

For my A Level Coursework, I will be creating a 2D, top-down, multiplayer shooter game. My game Bullet Assault will be designed to be simple, free, and enjoyable to play so that it suits a broad range of different people. Each client can connect to an encrypted server. To ensure the reliability and safety of the server and clients connected, all the data will be encrypted to avoid any malicious external threats, ensuring that my program still is secure and scalable to show that my program can accommodate for larger-scale operations. I will be using asymmetric encryption to send unique, encrypted symmetric keys from the client to the server. The symmetric key will then be used to encrypt and decrypt private player data for each client.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

In my game Bullet Assault, there will be no player limit. It will be a “last man standing” type of game that will figure out the ultimate winner. When users run a client instance, they will have three options in the main menu: to join a game, to configure user settings like control settings, or to quit the main menu. Every client will be able to seamlessly leave and join the game without interrupting any other players. Players will be able to join in the given join period before the game starts. After that join period, players will not be allowed to join that game until the next join phase starts. If any player tries to leave the game while it is running, that player will be quickly removed from the server without causing any errors or faults to the clients or the server. In addition, when a player clicks on the play button, they will have to manually enter the IP address of the server that they intend to join. This ensures that my code is universal so that any user can run a client script and join the server that they intend to join without having to have any prior knowledge about python. Players will also have the choice to enter a name so that other players can distinguish between other players. Names will not be unique because every user already has a unique ID.

While players are playing the game, they will have a health bar and will have different weapons to try and kill the other players in the game. Once a players health bar is less than or equal to zero, they will become a ghost. Ghosts will be able to see other players that are still alive but will not be able to be seen by any players. There will be different obstacles throughout the map to try and create an immersive experience for players. Furthermore, clients will have centred camera movement. This will allow the player to never move off the screen which also means the map can be larger which aids in creating a diverse game for all users to play. In addition to having a larger map, I will also add an invisible boarder which has the purpose of ensuring that no players can escape the map.

Project Brief

Firstly, to create a server, I will be using the built-in sockets and threading modules in Python to create a server that sends and receives data. My server will be a LAN server, which means it will only allow clients to connect if they are on the network. In addition to my server, I will be using the RSA and PyCryptodome modules in Python. These are external modules, which means that anybody running the server or client scripts must have those modules installed in Python. I will use RSA to create a public and private key on the server script. After that, the server can send over the public key to each client. When the client receives the public key, I will use PyCryptodome to create a symmetric key and use the asymmetric public key to encrypt the symmetric key and send that back to the server. This way, the server and clients are both encrypted. This ensures robustness and protects all users from having any data stolen or intercepted.

For my main menu, I will be using my buttons module, which uses Pygame to create the unique text and buttons for my game, Bullet Assault. This will be used for the text box for users to enter their name and a text box to enter the IP address, in addition to the other buttons used for my main menu. Furthermore, Pygame will be used for the client graphics when they play my game. I will be using Pygame because it is a very advanced GUI (Graphical User Interface) module in Python, but mainly because it is where I have most of my knowledge and experience when it comes to game design. When players want to join the game, I will first check that the IP address entered exists and that the server is in the join phase. When a player leaves, I will check to see if we receive nothing from the client for that thread. Then I will remove the corresponding player ID from the list of players. In this way, when players leave, they do so seamlessly without interrupting the game at all.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Furthermore, I will use a player class to draw and move my player. I will also add a health value to my player class and decrease that value when a player is hit. If a player's health reaches 0, then they will die and become a ghost. I intend to hide the dead player from all the other players and reduce that player's opacity so that they look slightly ghost-like. To create my camera, I will use another separate class that will be used to draw all the sprites left, right, up, or down when the player moves, while keeping the player in the centre of the screen. Finally, I will use the external module Pygame to create custom unique graphics for my game, Bullet Assault. I will use Pygame to create my map, custom objects, and a border. I will check if a player hits a border object, and if they do, I will stop the player's movement to ensure that they stay within the map borders.

Stakeholders

The stakeholders for my game, Bullet Assault, will range from people who enjoy video games competitively to those who simply enjoy playing video games casually. My goal is to create a game that is playable for anyone, regardless of their skill level, and that people can enjoy. My stakeholders will include my father, Roland Pike, and my brother, James Pike. They have varying levels of experience with video games but will be testing my game and providing me with feedback on features and ideas I can incorporate into my game. Roland Pike has a background in Chemical Engineering, which involves advanced mathematics and a degree of programming. James Pike is a Year 10 student who regularly plays a variety of video games. Together, they will play my game to help me fine-tune the gameplay and difficulty when adjustments are needed. Additionally, I will ask them various questions when I gather research for my game. Furthermore, I intend to meet their needs and desires when it comes to a simple, engaging multiplayer game, and I believe my game, Bullet Assault, will accomplish that.

Justification of computational approach

Why my project is suited to a computational solution.

My project idea is suitable for computational methods to find a solution for several reasons. The solution will be software-based and will utilise the programming language Python to operate an encrypted server with clients. The involvement of a computer is essential to address the problem and create an environment with multiple clients connected to a single server. There is no alternative solution to my problem that wouldn't require a computer.

The central aspect of my solution is distinct yet interconnected: the server script and the client script. These are two separate scripts that can run on different computers. The server script serves as the backbone, responsible for creating and removing players, as well as sending and receiving player data while supporting real-time connections with clients. In contrast, the client script focuses on rendering players in the user interface, transmitting and receiving player data, and enabling real-time player movement. Real-time interaction is fundamental for my project, as players expect seamless, instantaneous feedback and responsiveness. Achieving this is only possible through a computational method capable of processing rapid data transmission.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

For these reasons, I believe that using a computational approach is the right choice to solve my problem, meeting the needs and desires of my stakeholders while also creating a project with potential for scalability and innovation.

Clear initial situation

When the user loads up the client script, the main menu screen will be launched. The user will be presented with three different buttons: 'Start,' 'Options,' and 'Quit.' Upon selecting 'Start' and entering the correct server IP address and their name into the respective text boxes, the player will enter the game but will be in a waiting period for 5 minutes to allow other players to join. This time allows eager players to join the session, enhancing the multiplayer experience. Once the countdown expires, and there are at least two players, players will then spawn randomly around a meticulously well-designed map. Each player's camera perspective will be a character-fixed camera, allowing for a larger map with a variety of buildings and objects. This design of the map can be more engaging to players and will help enhance a sense of exploration.

To support accuracy in gameplay mechanics, I will use a grid reference system. Each square of 64 by 64 pixels will be a distinct block. This is what will be used for the invisible barrier blocks, adding strategic depth to player movement and interactions. Players will then have an unlimited amount of time to fight each other and to determine one player as the victor.

Clear goal

In my Problem Identification, I already have a well-defined plan for the functionality of my game. Additionally, I will be discussing in detail with my stakeholders, Roland Pike, and James Pike, to gather further in-depth details. This will help me establish specific targets to work towards during the development process. Through these discussions, I hope to define specific and achievable goals to guide the development period. This way, I can align my project with my stakeholders' priorities.

The basic features for my game are:

- A main menu screen with three buttons.
- A text box to enter the server IP address.
- Loading zone for players to wait for other players to join.
- A character centred camera.
- A large map with grid zones.
- Invisible barriers to encase players within the map.
- A map with a variety of objects.
- Each Player has a health bar.
- Dead players become invisible Ghosts.
- Players can shoot and attack other players until one is left.
- Players can leave seamlessly without affecting the game.
- Encrypted data transfers from clients to the server.
- Users will be able to choose a name when they join the game.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Clear inputs and outputs

My game, Bullet Assault, will be a multiplayer shooter game, as described in my Problem Identification. There will be a main menu screen where a user can start the game, configure different options, and quit the game. When a user clicks on the options menu, they will be greeted with a second graphical menu. The user will be able to change the audio settings and the control settings to adjust the settings to their preferences. Once a user decides to click on the start button, they will be greeted with another menu system. This graphical menu will have a text box to enter the IP Address of the server and a button called "Connect" to join the server with that IP Address. There will also be another text box for the user to enter a name for when they join the game.

During my game, the background, various images, positions of each sprite, and the number of custom objects on the map will all be displayed on the screen in real-time. Each player sprite will be controlled using either the WASD keys or the arrow keys. In addition, each sprite will be able to shoot with the space button, the left-control button, or the right-control button. The actions of each sprite will be displayed to all users. When all the other players have died and become ghosts, there should be one player still alive. After the player has won, all the players should go back into a waiting zone for five minutes until the next game starts.

Clearly defined logic

The program will need to be able to encrypt and decrypt messages sent by the server and client script. The server script will have to either generate a public and a private key or retrieve a public and private key. This is more efficient, as creating public and private keys is slow and time-consuming. The server will send over the public key first to each client. Once each client has received the public key, they will send back the name of the player encrypted with the symmetric key and the encrypted symmetric key. The server will then be able to send over an encrypted dictionary with all the player's personal data. Once each player has their own dictionary, the server can now start sending the dictionary with all the data of all players to each client. This is so that each client can draw the corresponding players on the screen. The server will then receive an updated version of the player's personal dictionary for each client. This can then be used to update the player's dictionary in the server. This cycle repeats while there is at least one client connected to the server.

When a player joins a waiting zone, the main menu will pass through the user's personal options for control settings and audio settings. This is done so that while the player is in the game, they can correctly use the settings that they changed to suit them. When each player is in the game, I will use my graphical interface, Pygame, to create a background with different objects, but also a defined border. This border will ensure players cannot move if they collide with it. This is done to trap them in an enclosed environment, which encourages players to come closer together in a confined area, enhancing competitiveness in my game. Finally, when the game ends after one player is victorious, all the players will go back into a waiting area for five minutes until the next game starts.

Initial Research

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Research of similar systems and ideas

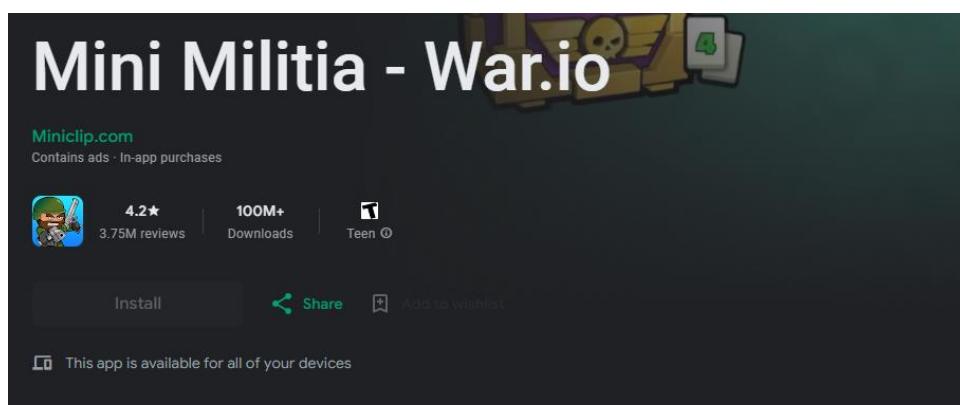
Below, I am going to research three different projects that are 2D multiplayer shooter games. I will explain what they are about, how their project is similar, and highlight some positives and negatives for each game.

Mini Militia – War.io by Appsomniacs LLC

Description

Mini Militia is a 2D online multiplayer game that allows up to 6 players to connect. It offers fast-paced combat in a doodle-themed world, enabling players to connect with friends or online opponents. Additionally, Mini Militia also allows players to play offline. The game features various maps, weapons, and customisation options. Mini Militia has a star rating of 4.2/5, which suggests that players quite enjoy playing the game.

Positives



for both a multiplayer and offline experience. This is quite good considering my game will not be able to have an offline mode. In addition, Mini Militia is known for the ability to connect with friends locally or engage in battles with up to 6 players. This is good in making an engaging and competitive game. Furthermore, Mini Militia has simple controls, this is particularly good to make a game fun when it's easy for players to learn how to play quickly. The game allows users to move, aim and shoot with touch controls making it accessible for a large audience. Mini Militia allows players to customise their character with different skins and weapons. This adds variety to the gameplay and keeps it interesting. Moreover, Mini Militia allows for many diverse maps. This is exceptionally good for creating a visually pleasing experience for players. Although, Mini Militia is known for its multiplayer game, their offline mode is incredibly good because it allows players to practice with AI

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

opponents which can help improve their skills before challenging real players. Finally, Mini Milita was last updated on September 28, 2023. This is good because regular updates help fix bugs and add updated content which overall improves the gameplay experience and help keep the game relevant and enjoyable.

Negatives

Firstly, Mini Milita features simple 2D graphics with a doodle-style aesthetic. Although this can be exciting for some players, it may not appeal to those looking for more realistic or visually stunning games. In addition, the use of In-App Purchases for skins, weapons and power-ups could be seen as controversial as many players find the microtransactions a bit intrusive or pay-to-win in nature. Furthermore, there single-player content is limited. While the game offers an offline mode with some AI, its primary focus is on the multiplayer mode. Some players may prefer single-player experiences, but Mini Milita may not be as engaging and satisfying in that regard.

Conclusion

Overall, Mini Milita is a fast-paced game built for those interested in multiplayer experiences. Although, Mini Milita has a vast number of positives, there are still some negatives that I will not be including. Firstly, my game will not have an In-App purchase options. This is because my game will be a free to play game for anyone. Mini Milita do have an offline line experience, but I will not be adding one to my game because I am primarily focused on creating a multiplayer game. In addition, an offline experience would not be interesting as there would be many limitations. Moreover, I will have some level of customisation in my game, but it will not be as advanced as Mini Milita's since my game is not as heavily focused on character customisations as I will be focusing on user customisations via the settings tab in my game. In all in, my game will not have as many features as Mini Milita, but I will focus on improving multiplayer connection, graphics, and user enjoyment. Finally, there will be no In-App purchase options in my game which is a positive for users because it is completely free and fair for all players.

Stick Warfare: Blood Strike by Team Modernator, MinGyu Choi

Description

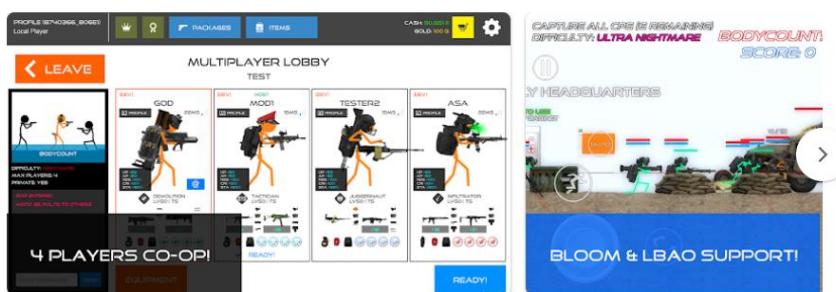
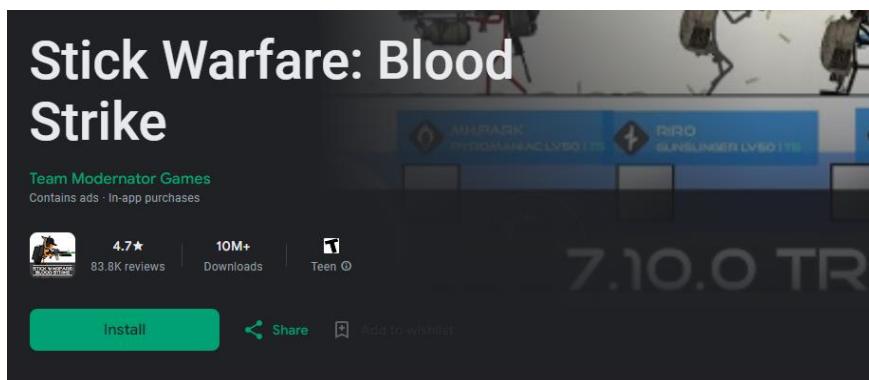
Stick Warfare: Blood Strike is a 2D, online multiplayer game. It allows up to 4 players to play with each other to fight online opponents. Stick Warfare also allows players to unlock over 180+ weapons in addition to unlocking and upgrading new skills. This game also has an in-game currency which has two forms, Cash and Gold. This allows players to buy different skins and weapons in the game. It is described as the "most fast-paced Stickman shooter." Stick Warfare has a star rating of 4.7/5. This suggests that many players really enjoy this game, in addition it seems this game gets regular updates because it was last updated on 21 August 2023. Finally, Stick Warfare has a progression for players as they can start off with no Cash and must fight to upgrade their weapons and skills to get better and to enjoy it with friends.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Positives

Firstly, the gameplay in Stick Warfare is exceptionally good. This is because of new ways to discover methods to beat different modes and a way to still improve. Furthermore, the progression in the game is very balanced. In addition, many players might appeal to a more simplistic Stickman-style aesthetics. Moreover, Stick Warfare does an exceptionally decent job with their multiplayer experience making it very engaging its player base. When playing a game, some players may like the colourful GUI. For example, they display the body-count and score in the middle of the screen in bright bold colours. To add on to my last point, the GUI menu system is slick and unique. They also have a choice to level up rewards and collect new weapons. This is particularly good for a video game experience as players want to feel a level of progression. In addition, there is over 180+ different weapons for players to choose from when it comes to progression and a way to increase skills like combat. Finally, this game allows for players to progress through diverse levels. A level system is particularly good for a game because it allows players to feel like their skills in the game are getting better and they can take on a harder level. This game overall, has lots of positives that make it very desirable for different players to play and enjoy.

Negatives



The first negative of this game would be their graphics. For many players, they may not be interested in a stickman-style aesthetics game. Since this game is a stickman-style game, I would be inclined to say that this game has been designed for a specific audience rather than for a larger audience. In addition, the graphics are quite simple. For example, there are no maps, it's a 2D white background with a line as the floor. This style of gameplay may be very enticing to some players, but it may be a negative to those who are looking for a more game with complex and detailed graphics. Another negative for Stick Warfare is the use of Ads and In-App purchases. For many players, this is a negative because microtransactions can be seen as intrusive because of the nature of many Pay-To-Win games. In addition, one review said: "Gold is hard and unintuitive to get without watching hundreds of adds."

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

You are forced to use gold for late game upgrades and it's very annoying to get." Furthermore, another review talked about connection issues. To add to my last point, another review said that the game crashed. This is extremely negative for Stick Warfare because it causes players to quit playing and play another game where they will have better connectivity. Moreover, some users also say that progression is difficult at times, especially when trying to receive Gold. Overall, there are some negatives to Stick Warfare, but in conclusion, I would say that many of their issues can be fixed or are user preference issues.

Conclusion

Finally, Stick Warfare will have some features that I will not be able to include. I will not be able to include a level progression to the degree that Stick Warfare successfully achieves. Furthermore, I believe that my graphics will be more advanced than Stick Warfare's graphics because my gameplay style is slightly different. Stick Warfare is a 2D game where players move left and right across the screen, while my game will be a 2D top-down game allowing for players to move more freely. Furthermore, my graphics will be more detailed and advanced than Stick Warfare's because my game is a slightly distinctive style of game. In addition, my game will have no In-App purchases. This is great for players because it avoids the negative view of microtransactions. Stick Warfare's progression, upgrades and customisation will be better than my game in those areas. This is because adding progression, upgrades and unique customisation can be difficult and isn't a main priority of mine. Although, I will try to add some customisation, I will not be able to have over 180+ weapons like Stick Warfare. To sum up what I have said, I hope to build on some of the negatives Stick Warfare face like connectivity, graphics, and In-App purchases so that I can create a fair and fun to play game for my stakeholders, while also knowing that I will not be able to add all the positive features of Stick Warfare.

Warforce – Online 2D Shooter by Megatu

Description

Warforce is an online, 2D, top-down, multiplayer, cross-platform, tactical shooter game. It is a free to play game which requires users that play to create an account. Warforce allows players to unlock new ranks, weapons, and tactical supports. In addition, players can connect with friends online and can chat, join their game, and fight together. Warforce also has several unique features. This includes two different maps and five different game modes. These game modes are Death Match, Team, Domination, Last Man Standing and Launch. Warforce was last updated on June 17, 2019. This may be correlated to their star rating of 2.9/5 which could suggest many players have dissatisfaction towards the game.

Positives

Firstly, this game allows is a cross-platform multiplayer game. This is particularly good because it can encourage a larger diverse audience with a range of different devices being able to connect to the game. Moreover, Warforce allows players a degree of progression when it comes to ranks and weapons. This is good for players because progression keeps a player base engaged and ensuring the game stays relevant. Furthermore, Warforce has a range of five different game modes. This is extremely good when it comes to exciting players because the game offers unique game modes that

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

may not be found anywhere else. In addition, their graphics may be unique and interesting to several of their players. Their simplistic game graphics may be interesting to some players that don't want to play a game that is particularly detailed. Finally, Warforce seem to have a unique leaderboard system. This is great for competitiveness in a game. In addition, it will help encourage players to come back to try achieving a greater high score than last time. Overall, this game has several positives and may be engaging to many players due to the range of distinctive features that Warforce has.

Negatives

The first issue with Warforce is their graphics. Many users may get the impression that the graphical interface is quite basic and does not have much detail. Some users may be more inclined to a graphical interface that is detailed and has more realistic and unique graphics that are aesthetically pleasing. Furthermore, this game is not supported by newer android operating systems. This is a major design flaw because it means that anybody with a newer software cannot play their game regardless of if their game is cross-platform. To add onto my last point, several reviews said that the "screen glitches out." Furthermore, other reviewers have seemed to say that there are various bugs and glitches. This is unbelievably bad for Warforce because if a game is unplayable for its player base, then the game will lose popularity and as a result this game has a low rating. In addition, this game does have In-App purchases and advertisements. I have already mentioned that In-App purchases are bad for general player satisfaction and enjoyment since they can be Pay-To-Win and can be seen as intrusive. Although In-App purchases can be decent in some games, many reviewers seem to have the same problem when it comes to adverts crashing their game. Overall, this game has quite a few negatives compared to the other games, maybe because it was last updated over four years ago. Furthermore, the overwhelming negatives support its low star rating of 2.9/5. Compared to the other games, this game is the worst. I believe that my game will be able to avoid many of the negatives while keeping it fun and enjoyable for my stakeholders.

Conclusion

To sum up everything above, this game has some particularly good positives when it comes to customisation like, multiple game modes, many weapons and cross-platform multiplayer. I will not be able to add some of the features in Warforce like mass customisation, but I will be able to have a degree of simple customisation for users to make it interesting and fun to play. Furthermore, I will not be able to add any ranks or a broad range of different weapons to my game. This is because my game will be designed to be a simple multiplayer game. Although, I will try to add some level of different weapons when it comes to the gameplay. Moreover, the graphics for Warforce are quite basic and dull. When I come to create my game, I will design my game with the intentions of having an interesting and aesthetically pleasing map for players. This is good because an interesting map allows for players to be creative and adventurous. However, when it comes to Warforce's five game modes, I will not be able to compete with that. This is partly because I am designing my game to solely be a last man standing type of game. Nevertheless, when I design and create my game, I will ensure that the connectivity between multiple clients and the server is strong, ensuring that no users disconnect randomly. In addition to that, having no In-App purchases or advertisement will help improve the connectivity as advertisements have been shown to cause a degree of lag to users. In conclusion, I will not be able to build a game with all the positive features that are shown in Warforce, but I will be able to build a game with a degree of the features seen in Warforce, however

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

when I create my game Bullet Assault I will avoid most of the negatives ensuring my game can have an optimised experience for users and interesting and unique for my stakeholders to play.

Research of the algorithms used.

Below I will talk about the different algorithms used to support encryption, client-server connections and the structure of the games. I will talk about each game I covered above, while also talking about general encryption, client-server connection and other structures of different 2D, multiplayer games.

Encryption

Firstly, there is not much information about the types of encryptions used in any of the game. There is no way to actually find out whether one encryption method is used or another method is used. In addition, it is tricky to suspect which programming language was used. Even though I can't find anything about the encryption on those games, I believe that they will all use similar encryption methods to each other because they are all client-server games with a multiplayer environment. The first method of encryption that would be used is called Transport Layer Security (TLS)/Secure Sockets Layer (SSL). This is commonly used to encrypt communication between the client and the server. This method of encryption ensures that data transmitted from the player's device to the server is difficult to be intercepted or tampered with. This method of encryption is especially important when dealing with sensitive information like passwords. Another type of encryption that may be used by all three games is End-to-End Encryption. This may be used for game chats or player-to-player communication. This ensures that the intended client receives and decrypts the message. Another method is data encryption. This is what I will be using in my game as there are two types of data encryption. There is XOR encryption which is basic and unsecure and there is AES (Advanced Encryption Standard). AES is very secure and is what I will use for my game. This type of encryption can be used to encrypt anything from player data to server data. Another method of encryption that all three games may use is authentication and authorisation. This is exceptionally good when verifying the identities of players. Furthermore, this is particularly good at protecting user accounts. However, my game will not have any user accounts, but I will use advanced encryption to protect data from any malicious entities. All three games will use a form of anti-cheat measures to protect their games from hackers either trying to change the client script or trying to hack the server script. Moreover, with the use of network protocols like UDP (User Datagram Protocol), server-side security such as RSA (Rivest-Shamir-Adleman encryption with public and private key) and various means like security system configurations, intrusions and unauthorised access and attacks become extremely small because of complicated encryption. Finally, I will not be able to use all the methods of encryption here, neither will I be able to use all the methods that advanced games like Mini Militia, Stick Warfare and Warforce use. Although, I will be using RSA encryption to share an AES symmetric key with a public key and a private key to set up a connection between the server and client.

Client-server connection

Knowledge about the type of client-server connection across all the three games is not common knowledge, however I will talk generally about general client-server relationships and connections. In addition, I will talk about the architecture of an online multiplayer game when it comes to clients the server. The first component of the client-server relationship is the client. The client runs the game software on their device. It is responsible for rendering the game image on the screen,

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

managing player input, and sending data or commands to the server. All three of these games will use a client-side software to provide a software and a method to talk to the server. Next is the server component. This is a powerful computer responsible for managing player data, handling communications between different clients and ensuring a stable connection between each client. There may be several servers used in Mini Milita, Stick Warfare and Warforce. For example, a game logic server might be used to handle game mechanics such as simulations, AI, and game rules. Furthermore, all three games would need some sort of authentication server to authenticate different user accounts. In addition, all three games have a degree of a leaderboard. This would require a specific database server in order to store player profiles and other important data. Overall, the server can be split into several different servers each used to store or process different data, however they are all interconnected in order to connect to clients. Moreover, the server will have to use a degree of threading in order to have a thread for each client. This can be used to talk to multiple clients at once whilst updating the player data in the server in real-time. In addition, many online multiplayer games will use network protocols like UDP or TCP/IP. This is to help reduce the latency and helps create a more reliable server. To add onto my last point about latency, many clients will use client-side prediction to reduce perceived latency. This can be used to successfully reduce latency providing a smoother gaming experience. Although I will not be able to implement perceived latency detection, I will be able to use some form of data serialisation. The server will use data serialisation to reformat data from a chat message or a string to formats like bytes, JSON, or binary. This is done to increase network efficiency and to reduce potential errors. All in all, for my game I will only have one server with multiple connected clients. I will not be able to have any network protocols or any other servers in addition to not having latency predictions. Furthermore, I will have a degree of security like RSA and AES encryption to help protect my server and clients from potential external threats. Finally, although I was not able to find details about the three games, I believe that each of those games would have elements that I have described and explained above in their game. However, I will be using a degree of data serialisation and a decent level of security in my game so that it has the potential to be scalable so that I can create a secure game for my stakeholders to play.

Structure of the game

Firstly, all three of the games have a varying difference when it comes to gameplay and the structure of their games, but I will be able to describe how each game works and how my game will differ. Mini Milita like the other games is a 2D multiplayer game for mobile devices. The distinct features of Mini Milita are 2D graphics, a top-down perspective, fast-paced gameplay. The other games feature 2D graphics too but have vary when it comes to camera perspectives. In addition to that, Mini Milita has a main focus on Multiplayer. This is because online gameplay is centred around multiplayer combat, and its point of sale (POS) is being able to play a doodle-style game with friends online. Mini Milita will have In-App purchases and options for unique weapons and customisation. However, Mini Milita is quite a unique style of game because it features a doodle-like graphics whilst also being a combat game with unique game modes like Team Deathmatch, Free for All, and other type of game modes. My game will feature a 2D top-down graphics style, but it will only have one game mode which will be a free for all type of game mode. Like stated in the description of Stick Warfare, it is a 2D stickman-style game. It has quite basic graphics because of the nature of stickman-style games. Although Stick Warfare is similar to Mini Milita, Stick Warfare seems to focus more on the multiplayer game mode and is a much faster paced game. In addition, Stick Warfare seems to have

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

more content and overall seems to be a more developed game when it comes to game structure. Moreover, Stick Warfare have multiple game modes. For example, Body count, Gun Game, Onslaught and Zombie Invasion just to name a few. Lastly, Warforce is graphically similar to Mini Milita because of similar top-down camera position. My game will have similar camera position because the player will be in the middle of the screen at all times as the user sees a top-down camera angle of what the player does. In addition, Warforce has a range of different game modes. For example, Death Match, Team, Domination, Last Man Standing and Launch. Warforce has five different game modes which is particularly good. Furthermore, my game will only feature one game mode out of Warforce's five. My game will be solely a Last Man Standing game. Although, this does not mean that players can't choose to team up against other players, however there will always be one winner in my game. In conclusion, all three of the games feature similar game modes. Many of the game modes are the same and some are the same as my game, but I will not be able to have a number of different game modes. Finally, I will ensure that my game that uses the last man standing game mode will have the positives of these other three games although not being as advanced but also aiming to avoid many negatives of these three games.

Research of the GUI used.

The GUI that was used in each game is unknown. However, all the games used some sort of graphical interface. Even though I do not know the programming languages used, popular languages used could be Python, C#, C++, or Java. In these languages they have many different modules that could be used to create a graphical interface. Below I will talk about popular different graphical libraries used by programmers to create games like Mini Milita, Stick Warfare and Warforce. Firstly, Python has a variety of different GUI modules to choose from. There is Pygame, which is what I will use for my game because Pygame is incredibly good at allowing users to create 2D applications. Although it is not the most complicated GUI there are other Graphical User Interfaces like Kivy. It is an open-source library used when developing multitouch applications like these mobile games. Lastly, Python offers PyQt and PyGTK as Graphical User Interfaces. They are respectively used to create quite sophisticated desktop games. For my game I will not be using those python libraries because they are too complicated for my simple multiplayer game. In addition to Python C# is a great programming language to use for programmers creating app store games. This is because C# is compatible with Unity and Unity is used to create many different games. Adding on to my last point, Unity an application that is used with C# to create GUI games. Unity is one of the most robust and flexible development interfaces for beginners to use. In addition, C# also has Windows Forms. This is great for developers that create games specifically for windows devices. However, this would not be great for developers looking to create a game for a larger audience. Although, C++ is an extremely popular and fast development programming language. This is because it is portable between a variety of different operating systems. In addition, C++ gets directly written into machine code without an intermediary translation needed at runtime. Also, C++ has high frame rates and responsiveness which is essential when creating a game. C++ GUI libraries include SFML (Simple and Fast Multimedia Library) and Qt. SFML library supplies a simple interface for creating 2D games, while having built-in GUI components. Qt on the other hand is a framework that offers a variety of different tools for developers when it comes to cross-platform applications and interfaces. Lastly, Java is quite complex programming language but has been used to program many popular games such as Minecraft. There are several modules in Java for GUI but here are three. Java Swing is a GUI

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

tool that is used for creating desktop interfaces. In addition, JavaFX is a framework for Java that is particularly suited for creating desktop and mobile interfaces. In addition, this GUI supports 2D and 3D applications unlike most of the GUI modules mentioned above. Lastly, LibGDX while not being a framework, it is a Java game development framework used in creating cross-platform games. Moreover, it adds basic tools for creating game interfaces and is often used in combination with Java GUI libraries. Although Mini Militia, Stick Warfare and Warforce may use one of the GUI modules used above, my game will use Python. This is because Python is much simpler and the module Pygame is quite easy to understand, and it is easily manipulated to create an enticing game for my stakeholders.

Stakeholder interview

I will be interviewing my stakeholders so that I can identify the specific details that they would like to have in my game. I will base the questions from my research of the three games and try to establish an idea of what they would like to see from my game. I will ask my stakeholders the same questions to see if their answer vary and to see their opinions on my research.

Questions

Question	Reason
Would having an aesthetically pleasing user interface be necessary to you?	To determine how much effort should be focused on producing a user interface that's looks good.
Are you interested in multiplayer and single-player modes?	To see whether my stakeholders want a different game mode.
Do you think that five minutes is too long to be in the loading zone waiting for other players to join?	To determine if the waiting zone needs to be increased or decreased.
Is customisation important to you when configuring different user settings like audio or controls?	To see if users want the ability to have some level of user customisation when it comes to different settings.
How do you feel about player customisation. For example, different weapons or skins in the game?	To see if users want customisation when playing the game.
What do you think about the standard of encryption that will be used during my game?	To figure out whether my stakeholders think my level of encryption is strong enough for the game.
Do you think that my game should have multiple game modes like in the games I researched?	To see if my stakeholders are interested in a game mode other than last man standing.
What are your opinions on a 2D top-down game with a player centred camera position.	To gather an understand of what my stakeholders think about the style of my game.
Would there be any game features that you would like to see in my game?	To determine whether game features are necessary.
What are your opinions on no In-App purchase options and a completely free to play game?	To determine their views on In-App purchases and what they think about free to play multiplayer games.

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

Do you think there should be a leaderboard system or is it not necessary for my game?	To see if storing player data on a leaderboard is of interest to them.
Do you think that my game should be a LAN (local area network) multiplayer game, or it should be hosted online?	To determine whether my stakeholders want my game to be locally hosted on a private network or to be public hosted online.
Would you like anything to happen at the end of the game for the winning player?	To determine what should happen when a player wins the game.
Do you like that dead players become invisible ghosts when they die? Would you suggest that something else happens?	To gather an understand of what my stakeholders think about when a player becomes a ghost if they die while the game is running.
What do you think about manually entering the IP Address of the server in the main menu to join the game? Would you like a different system?	To determine how players would feel about needing to enter the specific IP Address each time they want to join a server.
Would you be interested in game testing and playing my game for fun?	To see if they would like to test my game and give feedback to me.
What are your opinions on giving players the ability to choose custom names for themselves when they join a game?	To see if my stakeholders like the ability to choose a custom name for when they play the game.
What do you think about giving each player a unique ID so that players can seamlessly leave without interrupting the game play?	To figure out if my stakeholders would like my game to be as bullet proof as possible when players leave the game.
Would you like to have user accounts in my game?	To see if my stakeholders would like to have accounts.
What do you like about having a game map with invisible borders and unique objects players can collide with throughout the map?	To see what they think about the map with the objects as an aesthetic. Also, what my stakeholders think about a border stopping players from escaping the map.
Is there anything from my research that you particularly like and would like to see in my game?	To see if my research has features that my stakeholders would like to see in my game.

Stakeholder interview transcript

Q1: Would having an aesthetically pleasing user interface be necessary to you?

Roland: *Very important. Absolutely.*

James: Yes.

Q2: Are you interested in multiplayer and single-player modes?

Roland: *Multiplayer would be more interesting as there's nothing to do in a single-player game.*

James: Yes.

Q3: Do you think that five minutes is too long to be in the loading zone waiting for other players to join?

Roland: *Generally, 5 minutes is too long. Normally, a minute is fine.*

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

James: Yes.

Q4: Is customisation important to you when configuring different user settings like audio or controls?

Roland: *Yes. I would use them. It's nice to have.*

James: Yes.

Q5: How do you feel about player customisation, such as different weapons or skins in the game?

Roland: *Weapons, yes. Skins, no.*

James: *It's not a major consideration.*

Q6: What do you think about the standard of encryption that will be used during my game?

Roland: *I'm not really bothered about encryption.*

James: Yes, it's fine.

Q7: Do you think that my game should have multiple game modes like in the games I researched?

Roland: *It would make the game more playable.*

James: *No, I think one is fine.*

Q8: What are your opinions on a 2D top-down game with a player-centred camera position?

Roland: *It's fine. It's simple but effective.*

James: *It's great.*

Q9: Would there be any game features that you would like to see in my game?

Roland: *The ability to shoot.*

James: *Kill count on the screen.*

Q10: What are your opinions on no In-App purchase options and a completely free-to-play game?

Roland: *I never buy in-app purchases, so I want it to be completely free.*

James: *A free game is better.*

Q11: Do you think there should be a leaderboard system, or is it not necessary for my game?

Roland: *I like leaderboards. They are good.*

James: *A leaderboard is good for competition between different players.*

Q12: Do you think that my game should be a LAN (local area network) multiplayer game, or it should be hosted online?

Roland: *Online would offer more scope if it was on the Internet.*

James: *Probably LAN.*

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Q13: Would you like anything to happen at the end of the game for the winning player?

Roland: *Only to see where they are in the ranking. Quickest kill shots fired. Metrics.*

James: *A congratulations graphic. A little celebration graphic.*

Q14: Do you like that dead players become invisible ghosts when they die? Would you suggest that something else happens?

Roland: *If you want to see how the game ends.*

James: *Yes, the players can do that.*

Q15: What do you think about manually entering the IP Address of the server in the main menu to join the game? Would you like a different system?

Roland: *It might make it less accessible to most people.*

James: *Definitely. Far too laborious.*

Q16: Would you be interested in game testing and playing my game for fun?

Roland: *Yes. I'd love to be a tester.*

James: *Yes, I love trying new games.*

Q17: What are your opinions on giving players the ability to choose custom names for themselves when they join a game?

Roland: *Not that important to me. How else do you identify who you are.*

James: *Pretty normal. Choosing your own game name.*

Q18: What do you think about giving each player a unique ID so that players can seamlessly leave without interrupting the gameplay?

Roland: *Yes, it is necessary. Anything crashing the game needs to be avoided. User input should not be able to crash the game.*

James: *Sounds like a good idea.*

Q19: Would you like to have user accounts in my game?

Roland: *I would be in favour of having accounts.*

James: *Sure.*

Q20: What do you like about having a game map with invisible borders and unique objects players can collide with throughout the map?

Roland: *Obstacles make the game interesting and required.*

James: *It makes it interesting and requires skill and concentration.*

Q21: Is there anything from my research that you particularly like and would like to see in my game?

Roland: *A shield would be nice and the ability to teleport and become invisible.*

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

James: *Bonus features or surprise wins. Chance to win unexpectedly. Random chance to find an upgrade ability in the game.*

Limitations

Firstly, my project will not be able to have any user accounts. This is because user authentication is extremely complicated and time consuming. In addition, it is not necessary for my game to function as users do not need an account to play my game. Another limitation to my project is that I will not be able to have any AI controlled enemies. This is partly because it would be exceedingly difficult and time consuming to implement this in the server and so that the enemies' line up with the player. Furthermore, I would have to send each client data about the other players and enemies. Moreover, it would be challenging to design an AI that can target different players within a certain radius.

Another limitation to my project is that there will only be one game mode. Coding multiple game modes is complicated to implement, particularly due to the time and effort that is needed to set up a game system where users can choose what game mode they would like to play. This is why there will only be one game mode as the structure needed to be created for a last man standing type of game is much easier to develop. Another limitation for my project will be a limited level of customisation. I will be able to add some level of customisation like the settings and player names, but I will not be able to have a large level of customisation like seen in those three games. In addition, I won't be able to have many or any custom player skins and on top of that I will not be able to add many different weapons players can choose from. This is partly due to the complexity of sending and receiving a large amount of data from the clients and sending specific data to the clients. To add some more detail, the level of complexity would be extremely high, and it would require a large amount of time to complete those customisation additions. Nevertheless, another limitation will be player progression. There will be no In-App purchase options. As I have said previously, I will not have any player accounts which also means there will be no player currency. Furthermore, players will not be able to buy anything to help increase their skills. Maybe players in the game could upgrade some of their stats and weapons, but when the game restarts, players would have lost all their progression. A further limitation would be a lack of a single-player mode. There will be no single player mode because I do not have any AI controlled enemies. On top of that, creating an enemy AI with several different enemies would not be a priority of mine because I am mainly focused on the multiplayer side. Another game mode would only add time and difficulty when trying to program and design all the features that my stakeholders want. Another clear limitation when comparing my project to the games Mini Militia, Stick Warfare and Warforce would be that my project will not have as many features. Firstly, I will only have one map. This map will not be the largest map, but it will have some detail. In addition, do not have the ability to design multiple maps and having a range of different unique objects. Although, I will have some unique objects, I will not be able to compete with the level of customisation and design when it comes to different and unique map that these other games have. Unlike some of the games I covered, my game will only be able to be played on a computer. This is because I am using a keyboard as an input device. In addition, Pygame only works on a computer because touch screen devices will not be supported. The last limitation will be the level of security I will be able to use via encryption. I will not be able to use any network protocols like UDP or TCP/IP protocols to add a level of protection to my server and clients. This is partly due to the complexity, but mainly because I do not have the knowledge or skills to implement those protocols or to implement a higher level of encryption to

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

supply more security to my game. In conclusion, there are several additions that I cannot add to my game due to time, difficulty implementing and the level of knowledge that would be required to effectively design and create all these features in my game.

Essential Features

After talking with my stakeholders. I have been able to gather some reasonable information to write all the essential features that I will need for my game. Below I will talk in depth about the system I will include from talking to my stakeholders, how they will work and how my design has been affected because of my stakeholders wants. In addition to that I will be talking about all the features my stakeholders want in my game and I will also be any good features I have seen throughout my research that I would like to include.

The user interface

Firstly, both of my stakeholders wanted to have an aesthetically pleasing user interface. When I come to design my game, I will ensure that I have an aesthetically pleasing map. In addition, when I create my main menu system, I will use my personal buttons module I created in Pygame. I will use this to create a unique and an elegant interface for my users. Furthermore, when designing my main menu interface, I will design some buttons for user settings. Another point to add would be custom player names. My stakeholders were interested in having names when playing the game. This is because names can help identify different players. When I come to design the game, names will only be cosmetic, which means that different players can have the same name because it's a player's ID which identifies who they are in the players dictionary when the server is running. Another interesting point to mention is that my stakeholders were against a text box to enter the IP Address to join a game. They both agreed that it makes the game more laborious and less accessible for more people. I plan to combat this by explaining how a LAN client would join a game. In addition, when I come to design my game, I will ensure that my stakeholders are not confused on how to join a game by entering the IP Address. This will ensure that all my stakeholders and players that decide to play my game understand how to find out the IP Address of a server and how to enter the correct IP Address into the text box to join a game. Both my stakeholders wanted customisation when it came to different settings and think it is important and useful to have. Moreover, when I create my buttons, I will ensure that when a user hovers over the button, the mouse will change, and the button will slowly change colour. I will design my main menu in this way so that I produce a very aesthetic user interface that meets their needs. Lastly, I will make sure that my user interface is simple enough for my stakeholders to traverse, but also unique and interesting to my stakeholders.

Game graphics

Another element both my stakeholders wanted was some decent graphics in the game. My game already plans for some graphics in the game. For example, the map, boarders, and varying objects throughout the map. Both my stakeholders wanted obstacles throughout the map. With this design, it creates an engaging, interesting yet adventurous map for my stakeholders to explore when they play the game. Another interesting point to make is that my stakeholders were not interested in customisation in the game. They both said skins are not needed, however one of my stakeholders added that custom weapons would be good to add in my game. When I come to design the graphics, I will keep in mind what my stakeholders feedback to my questions so that when I create the graphics, they are aesthetically pleasing but also particularly fun to play.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Game layout

For my game layout, I am creating a 2D top-down game view. In addition to the camera view being player centred. Both my stakeholders liked the idea of camera view. This is because it allows the player to explore the map. In addition, it also allows me to create a larger map as players can explore the map in detail. Finally, when I create my game, I will ensure that I can make a visually stunning map whilst also being able to integrate my camera design and top-down view so that it looks natural for the players to explore the map.

Game Mechanics

Firstly, my game will be a multiplayer game. As for some of the reasons stated in my limitations paragraph, a single-player mode and a multiplayer mode is complicated to design at best. In addition, my stakeholders had mixed opinions on whether they wanted a multiplayer mode and a single-player mode. However, I will only focus on the multiplayer side because my stakeholders said that multiplayer is more interesting, and it should be what I focus on most. Furthermore, when I asked my stakeholders on whether they wanted multiple game modes, they said that one is fine, but it makes a game more playable if there are multiple game modes. From this, I will focus solely on my Last Man Standing game mode. This is partly because my design would be that as I do not need to create any enemy AI to interact with other game modes. In addition, both my stakeholders like the ideas of accounts and leaderboards. Although, I disagreed with this in my limitations paragraph, I will be able to record some data about the game. Players won't see a leaderboard system, and neither will they have accounts for the reasons stated in my limitation paragraph above, but I will add a small leaderboard comparing the players from that round. For example, kill count, fastest kill, a ranking of where each player died on a leaderboard which would show up to each player when a player wins the game. After the players finish the game and start the waiting zone period, the data from the leaderboard would have been cleared. To add another point, I talked to my stakeholders about the five minutes waiting zone and they both said that five minutes was too long. This is not an issue when it comes to the design because I will be able to reduce that time to one minute which means that when the first player joins the game, they will have one minute for other players to join. After that, the game will start with as many players that joined in the one-minute period. When the game has started players that try join will not be allowed to join until the game is over. Also, when different players die, they will become invisible ghosts. This means that they will be able to see what happens throughout the rest of the game, but no one will be able to see them. Furthermore, both my stakeholders said that it would be good to see how the game would end by allowing players to become ghosts. Moreover, when I come to design this, I will ensure that each player has a visibility state so that I can change the visibility state to ghost. This would mean that they would not be drawn for any other player apart from themselves. In addition, I will make sure that dead players look slightly different to regular alive players ensuring there is some unique diversity.

Game features

When asking my stakeholders about different game features I could add into my game. They wanted the ability to be able to shoot other players with different weapons. When I come to design and create my game, I will ensure that I have this hard coded into the game because the ability to shoot other players in the game is vital to have a functioning game. Moreover, one of my stakeholders wanted to some information that was displayed on the screen about the game. For example, along with a UI menu system, there could be a kill counter to let players know how many kills they have. In

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

addition, I will have to add some health to each player so that players can get shot and lose health. When I come to programming this, when players have no health left, I will be able to turn them into a ghost. Another addition that one of my stakeholders requested was to have a leaderboard ranking at the end of each round. Although, I already mentioned that above, another suggestion was to have a little congratulations graphic to give the player that won a small celebration to show that they won the game. This would be simple enough to add into my design so that when I program, I will easily be able to add this celebration into the game for a few seconds before starting the new loading zone for new players to join. Lastly, my stakeholders wanted me to add some special additions and features in my game. They suggested that I add the ability to teleport, become invisible, have a shield, and have a random chance to find something in the game or to upgrade an ability. I believe that these are incredibly good suggestions. However, adding them may be tricky since I would have to design and program that last into my game. Although, I will keep in mind that my stakeholders want these abilities, and I will try to add some sort of special power ups that could include some of those abilities.

Server Functions

In addition to talking to my stakeholders about my game, I also talked to them about the server and what they like about the server. Firstly, I asked my stakeholders their view on the level of encryption I have decided to use for my game. Both my stakeholders agreed that the level of encryption was good for the game I was producing. In addition, they were not particularly bothered about encryption since there are no user accounts. This means that there is no threat to data being stolen. However, I will create my encryption to use asymmetric and symmetric encryption because asymmetric encryption is great for sharing a symmetric key between clients and the server. In addition, I asked my stakeholders on their views on unique player IDs so that players can leave seamlessly, and they wanted me to add a way for players to leave the server without affecting the game play of the other players and the server. Furthermore, both my stakeholders believed that it is an innovative idea to have some sort of ID system so that each player can be identified in the server. In this way, I would be able to successfully incorporate an efficient design for my server. Lastly, my stakeholders were conflicted between a game that was locally hosted on a private network (local area network) or between a server that was hosted online on the internet. Although for my game, I will primarily focus on hosting my game privately because my game is supposed to be simple and accessible for those that only want to play casually at home.

Game details

Finally, both my stakeholders were interested in testing my game. This is particularly good for me because I will be able to get feedback from my stakeholders as soon as I create a design for my game and when I program my project, I will be able to get latest ideas and feedback on what my stakeholders think about the project so far. I also asked my stakeholders on what they thought about my research about In-App purchase options and a free-to-play game, and they both agreed that a free game is better and that they do not like In-App purchases. Furthermore, I will not have any In-App purchase options because they are complicated and controversial when it comes to microtransactions, but mainly because my game will be designed with the intentions of creating a simple, 2D multiplayer game that is free for all players to play.

In conclusion, there are several features that I will be adding into my game to make it more enjoyable and engaging for my stakeholders to play. Finally, I have been able to identify several

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

features from the interviews that I will be adding. Overall, I will add these features into my design so that they can be implemented correctly when I come to program my game.

Success Criteria

Below I will describe the user requirements and system requirements I will need for my game bullet assault. I will describe in detail what I want and what I will create. In addition, I will describe specific information about the system requirements so that my game is able to run with all the libraries installed. Furthermore, I will sum up my Problem Identification and my Essential features in my user requirements below so that my game has all the features below that my stakeholders requested and wanted in my game.

User Requirements

1. The user must be able to interact with an aesthetically pleasing graphical user interface when either playing the game or navigating the main menu system. I will use Pygame to achieve this as well as my personal Pygame function module to create the buttons.
2. The main menu system must have an option to quit the game. Another option to configure settings like controls and audio. Finally, an option to start the game by entering the IP Address of the server. In addition, players should be able to enter their name into another text box and have custom names in the game. The main menu system must also be able to connect players to the server when they choose to join a game.
3. When the server creates the player, they must have a unique ID so that they can be identified in the server. If a player leaves, that corresponding ID should be deleted from the player list ensuring players no longer show up on other clients' screen. Furthermore, players should leave seamlessly without affecting the server or any clients.
4. The game must be a locally hosted multiplayer game that must feature the last man standing game mode. In addition, there should be no player limit allowing as many players as possible to join during the allocate time before the game starts. Furthermore, the game should not start the counter until at least one player has joined and should not start until there are at least two players. Finally, the game should end when there is only one left alive.
5. When players' health bar reaches zero or below, they should become invisible. They should not be able to be seen by other players but should be able to see other players. Their character skin should change to a ghost when they die and should change back when the game is over when the waiting period starts.
6. The players must be able to take damage from other players. The health bar should decrease when a player is hit. Furthermore, the server must be able to update that to all the other players who are playing the game.
7. The game must include a variety of different weapons for players to use to eliminate other players. The server must be able to register that a player has been damaged and update a player's health to all players.
8. The map must be aesthetically pleasing and feature some custom objects. In addition, there must be an invisible barrier which prevent players from escaping the map.
9. The server must use asymmetric encryption to share the public key with each client. Clients should then encrypt their symmetric key and share that with the server to establish an encrypted connection. The server and clients should then encrypt and decrypt player data

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

with the symmetric ensuring data security. Furthermore, no external threats should be able to do anything with the encrypted data.

10. The game should be a 2D top-down game with a player centred camera view. In addition, the game should feature some camera movement so that the player will never move off the screen allowing for a larger, more immersive map.
11. The server must be robust and include an efficient server system that can handle multiple simultaneous connections whilst ensuring a smooth and lag free experience for players.
12. The game must have an in-game user interface that displays essential information such as health, kill count and different abilities. Furthermore, players must be able to use power ups and other abilities.
13. Players must be able to receive a small celebratory animation when a player wins the game. This must be a few seconds long and allow for any player statistics to be displayed on screen for that round before players join the waiting zone again.
14. Players must be able to change their controls in the settings. Players must also be able to use either the WASD keys or the arrow keys to move around. In addition, players should be able to use the space key or left/right control key to attack other players.
15. The game should have a variety of unique and visually pleasing weapons. Each should have its own characteristics and attributes, offering players a diverse gameplay experience.

System Requirements

- Windows 10 or newer
- Python 3.12 installed with path. Anything under 3.10 will not work.
- Python installed with additionally libraries (Pygame, RSA and Pycryptodome) – pip will be needed install those modules.
- At least two or three computers all connected to the same network with the firewall enabled for python that allows the clients to connect to the server (when you run the server script a firewall notification will pop up – make sure you click allow otherwise only that computer will be able to connect the server). The port 5555 or any port must be created as an inbound and outbound rule one for TCP, one for UDP and allow connection in order for clients to connect over the LAN.
- One computer running the server script.
- The other computers running the client script (the computer running the server script can also connect to the server using the client script).

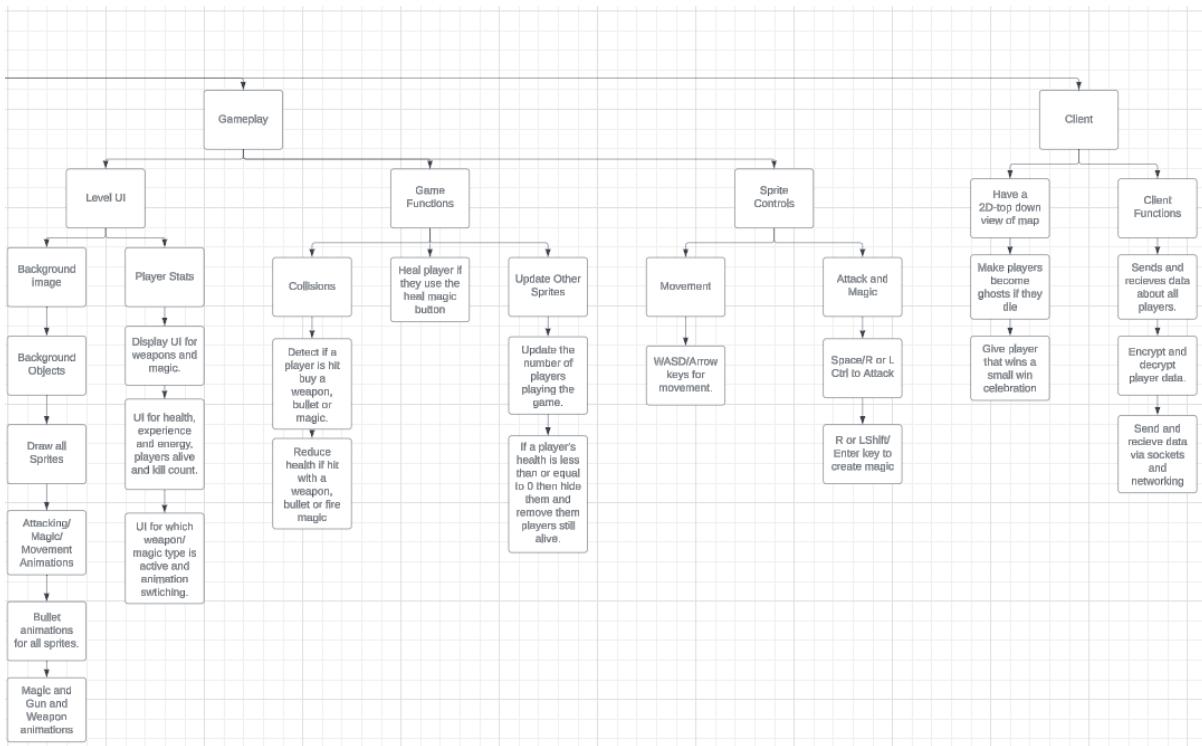
Bullet Assault Design

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

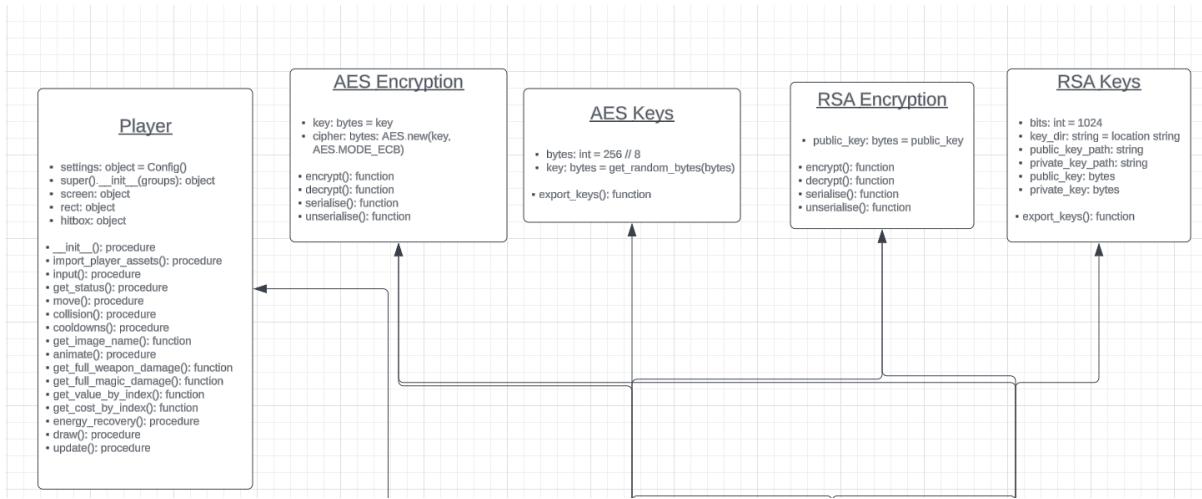
Data Structure Diagram



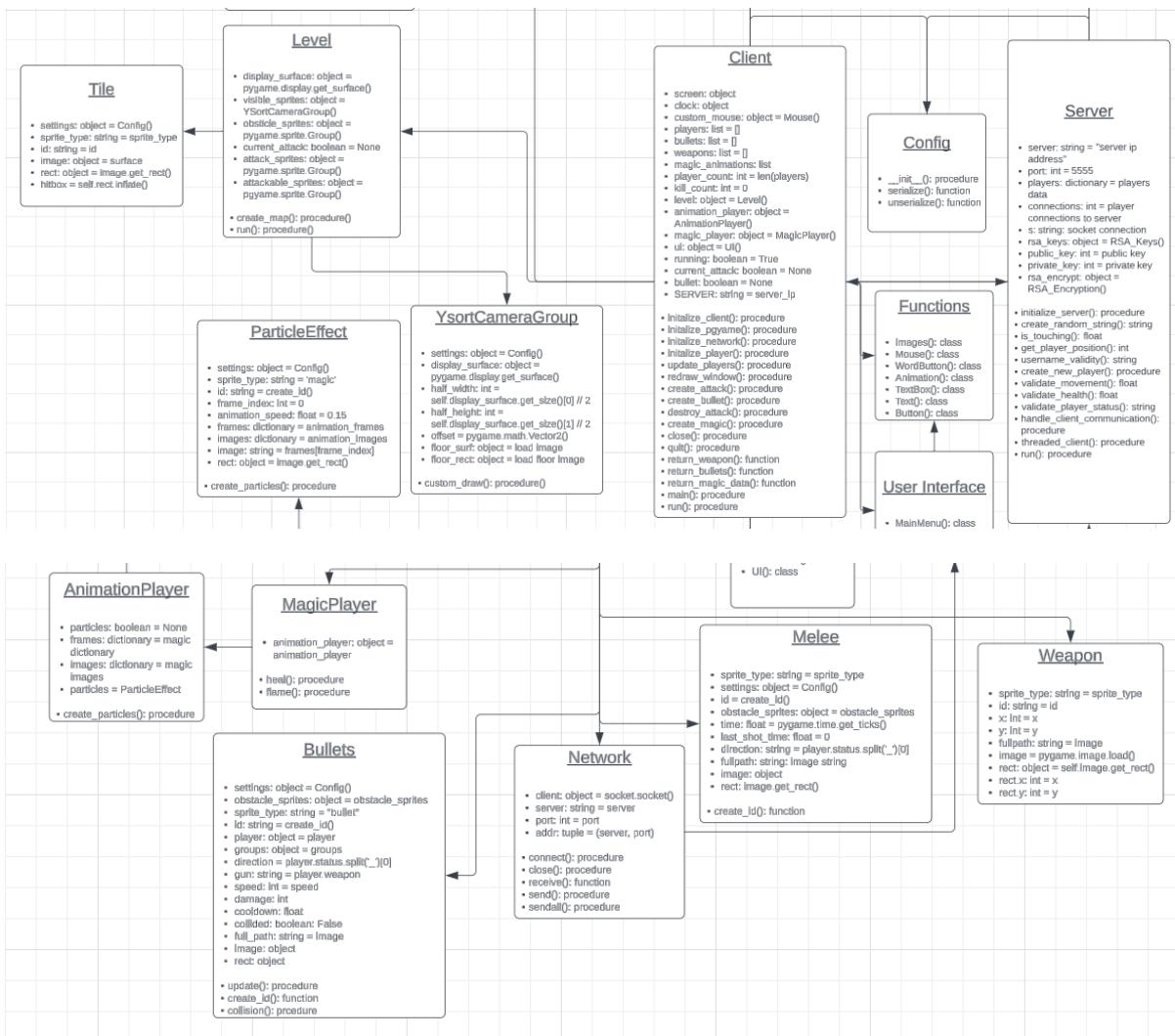
Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335



Class Diagram



Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335



Designing main menu screen/ GUI

Stakeholder interview

I already have a plan of what I intend to do for the main menu screen and how I want to make the GUI however, I need to check with my stakeholders that what I intend to do is good and what changes I could make when designing. I am going to produce the following questions that explain what I want to implement and what the current design is as well as improvements and small adjustments that each stakeholder wants.

Question No.	Question	Roland	James
1	I am thinking of implementing a background of that game. I want it to be moving. What are your thoughts on this?	Moving backgrounds are dynamic and interesting. It would be cool if the background could be an image of the game map.	I like the idea of a moving background, but I just think that it should move slowly to create a soothing effect.
2	The game will include a mouse with a hovering and clicking effect.	That sounds like an innovative idea.	It would be good if the mouse could be red and

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

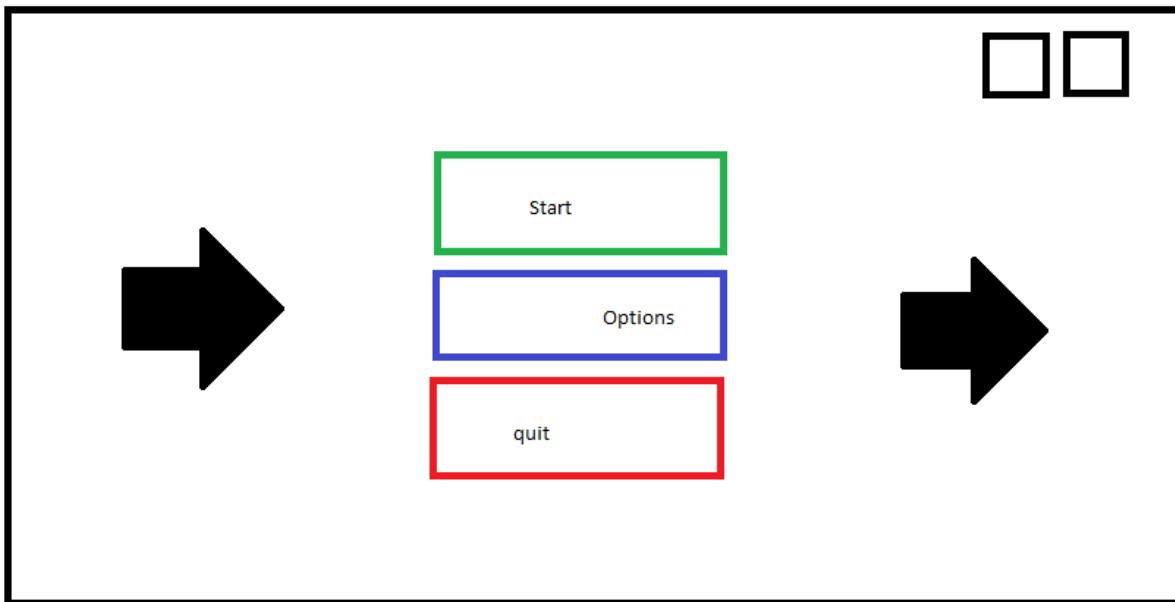
	Would you want any additions to this?		grey when hovered to create an advantageous effect.
3	In the top hand corner, I am considering putting a YouTube and a GitHub button. Do you have any comments about this?	Yep. That is fine.	That looks good.
4	In the design, I plan make three different buttons: quit, options and start. Do you want any unique graphics for that?	Distinct colours for each button would look good.	Would the buttons be able to slowly change colour to create an aesthetic effect.
5	For the options button, I plan to have a GUI menu system pop up. Would there be anything you would like to configure?	It would be nice to configure the controls for the character.	I would like to be able to change the audio like the sound and music.
6	As for the start button, I plan to have another GUI menu where you can configure your name and IP. Do you have any comments about that?	This looks good but it would be nice but if it is intuitive.	Yep. That seems fine.
7	Do you have any further comments or ideas about the start menu?	That seems fine to me. The only other thing I would like to see is the background constantly moving.	I like the design. I would like the GUI designs to look minimalistic with a darker colour.

Summary of findings

After completing the research and asking my stakeholders, I have found their preferences and ideas for each element of the main menu GUI. Firstly, my stakeholders enjoy the moving background, but from what they have said, I will make the background image a picture of the game background. I will also have the background moving slowly to create a soothing effect. The next thing my stakeholders wanted was to make the custom mouse red. This will help make the mouse more visible. Another thing my stakeholders liked was having the YouTube and GitHub buttons. My stakeholders liked the three buttons, but from talking with them, I have decided to make the colours invert slowly and smoothly to create a cool animation effect. In addition, from seeing the responses from my stakeholders, they wanted to be able to change their key binds or character control and to be able to change their volume settings and settings to do with audio. I will create the options menu system in mind with the ideas from my stakeholders. As for my start menu GUI interface, I am going to continue to develop that as intended, however I will keep in mind that they want the IP address to be intuitive and not too complicated. Finally, from my last question, my goal was to find out anything else that my stakeholders wanted to let me know about. As stated above, I will develop the background to be constantly moving as well as ensuring that my GUI menus are minimalistic.

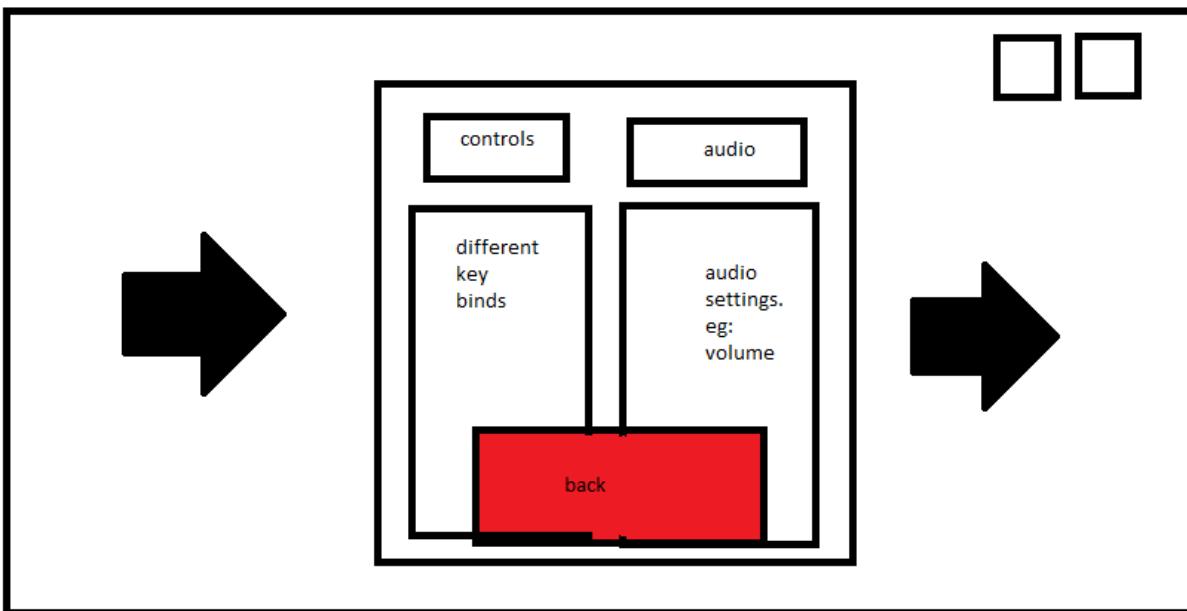
Design of main menu GUI

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335



The two large arrows represent the way the background will be moving, and the two boxes represent the YouTube and GitHub buttons. This is a simplistic diagram, but the game menu will still follow the design above.

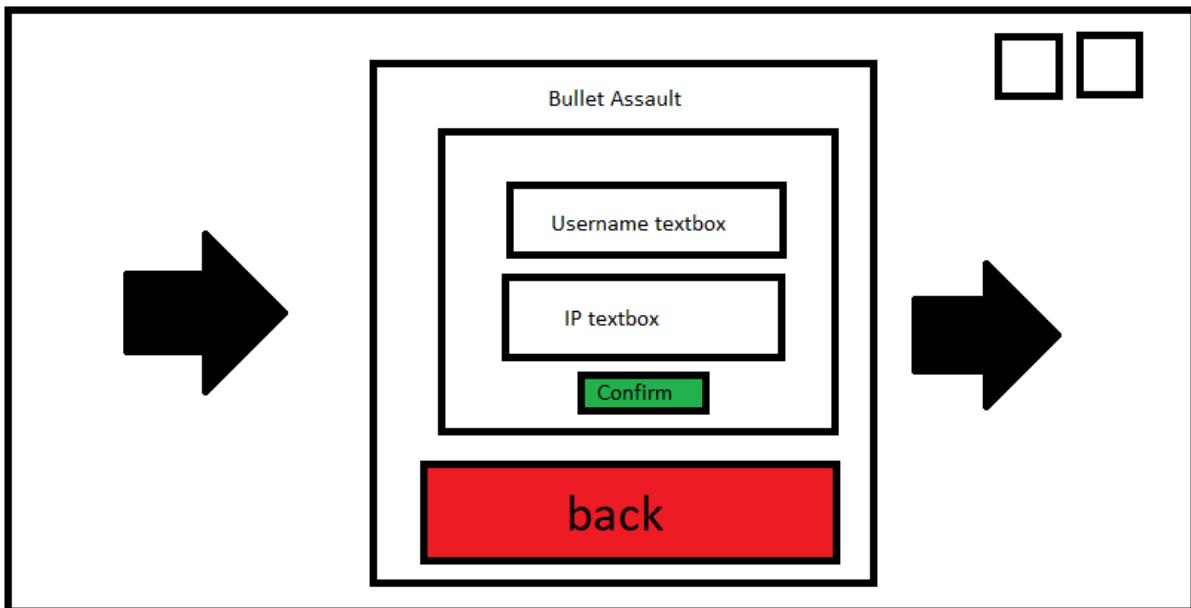
Design of Options GUI



Again, the arrows represent the background moving, however this is the options GUI menu instead. The back button allows users to go back to the main menu. As seen previously, the YouTube and GitHub buttons are also there.

Design of Start GUI

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335



The start GUI will be like the options GUI menu, but it will have textboxes and text to describe what each textbox does. As seen in the other two images, the background is still moving, and the YouTube and GitHub buttons are there.

Designing player inputs

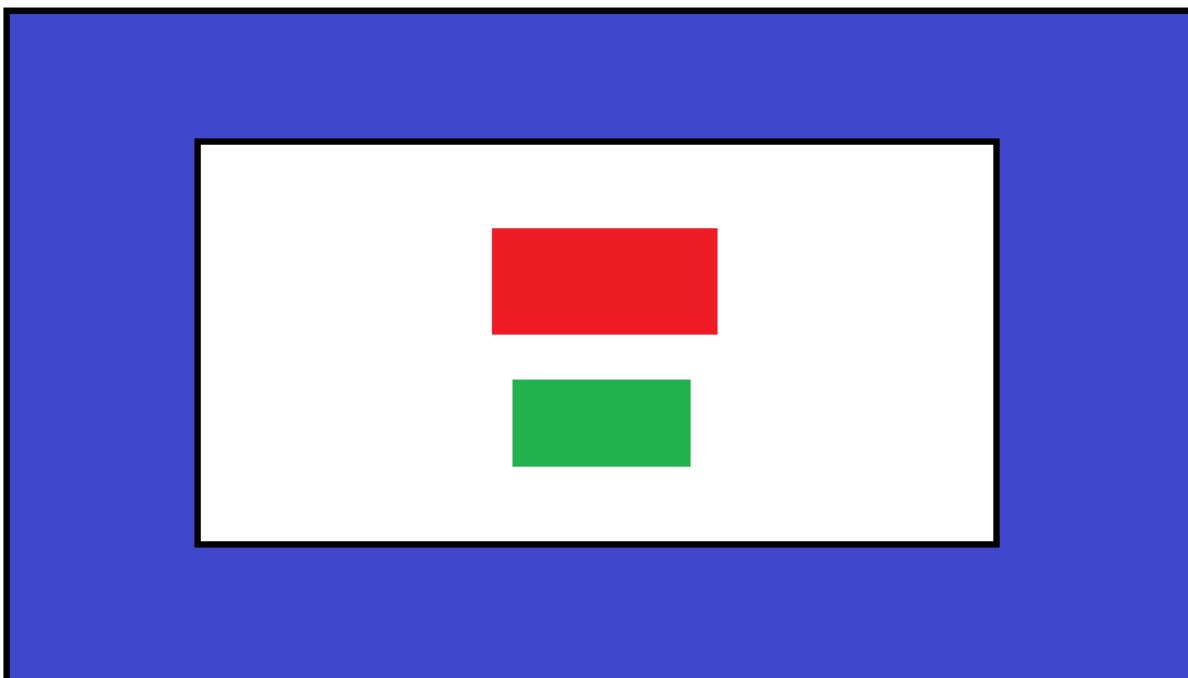
Stakeholder interview

Question No.	Question	Roland	James
1	As my game allows the user to control a player, I am considering what controls should be used for the player. Are three different input functions enough?	For movement and attacking and magic that seems fine. It would also be nice to press escape for a menu GUI.	Three input functions for the player are enough.
2	Would you want custom key binds for movement?	Yes, this would be particularly useful in the game.	I would like the ability to change my key binds.
3	Is it good if movement is with the WASD or Arrow keys?	Personally, I would use the Arrow keys.	I would prefer to use the WASD keys over the arrow keys.
4	What do you think of having an attack button?	I think that would be good for the game to have.	That would be good as you can attack players.
5	What do you think of a key for magic?	That seems like a clever idea. It can create diversity in the game.	I like that idea.
6	Is there anything else that you think should be an input or any other ideas?	The ability to leave the game and have a menu pop up.	A way to exit the game would be good.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Summary of findings

From talking to my stakeholders, they agree that three inputs are enough for the player regarding the movement and two methods for attacking. Another thing to mention is that my stakeholders would want the ability to customise their key binds. This is a decent quality of life feature as it allows players to have more control and customisability when they play as they can change their custom settings. To add on to this, point my stakeholders had different views on whether they would use the WASD keys or the Arrow keys. This therefore is a reason to have key binds in the game as I want to create a game that is inclusive to the needs of both my stakeholders not just what is easiest to do. In addition, from asking my stakeholders about the magic attack key and the regular attack key, they both agree that having two attacks is good. This is because having multiple separate ways to attack is better than one as it makes the game more diverse and interesting to players as there is more content in the game. Finally, the only other input I will include from talking to my stakeholders is having a way to leave the game. This will be the escape key but there will be a way of having a menu screen pop up when it is pressed.



In this diagram, the blue represents the game, and the red represents the leave button, and the green represents the continue button. The buttons might not be in this exact order but there will be a menu screen when leaving the game.

Designing in game GUI

Stakeholder interview

Question No.	Question	Roland	James
1	In the game I am thinking of having several player animations. Would this be beneficial for the game?	This would be a great feature in the game.	I like this would be a cool addition to have in the game.
2	In the game I am considering using a background image. Would you want to have other	Absolutely. A good background image with different objects would	I would want objects, trees, and grass on the game map.

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

	objects, trees, and grass as well?	be good for a diverse map.	
3	I am also considering implementing animations for the magic abilities and graphics for the guns, weapons, and bullets. Do you think this would be good?	I believe that animations for the magic attacks and GUI for the other attacks would be a good feature.	Yes, I think that animations will add to the overall game satisfaction for players.
4	In addition, I want to develop an in-game UI. Would this be good?	I think an in-game UI would be good for players.	I like that idea.
5	In addition, I am thinking of having multiple weapons, therefore would it be an innovative idea to be able to switch through the weapons?	That would be good to switch between different weapons.	I think that is a good idea.
6	I am considering having dead players looking different from alive players to distinguish them from alive players. What are your views on this?	I personally think that is a good idea. I think dead players should look slightly grey to distinguish them.	To me that seems like a cool idea. It is not necessary but is good for the game overall.
7	When the game ends, I am contemplating having a game over animation. What are your opinions on this?	That seems like a good idea. It's good to display visually the game has ended.	I like the animation at the end. It is nice to end with a little animation.
8	What are your ideas of having a leaderboard screen at the end of the game?	It is not a top feature in mind, but it is good for more competitive players.	It seems like a clever idea. It is interesting comparing players stats.

Summary of findings

From discussing with my stakeholders about the GUI of the game I discovered that they are very much interested in several different elements when it comes to GUI in the game. Firstly, from what my stakeholders told me, they are interested in player animations. This is the case as animations are more visually stimulating and cleaner to the eye. It also helps to increase a range of variety into the game. Adding on to the last point, I will also add in the animations for the magic abilities and unique graphics for the weapons, guns, and bullets. This is because my stakeholders agreed with me and said they do want some animations for the magic attacks. Another thing to raise is that my stakeholders did want a graphically pleasing map with aesthetically pleasing objects, trees and grass. Another thing I found out from interviewing my stakeholders was that an in-game UI would be of value. This could be used for tracking each players kill count, or the number of players left in the game. Furthermore, it could track what weapon the player is currently using as well as their health just to mention a few possibilities with an in-game UI. Building on from that last point, question 5 asked whether being able to switch weapons would be useful as well as the UI to inform the user which weapon, they are currently holding. This is a feature that I will be implementing since a UI is needed if I am going to have multiple weapons. From asking my stakeholders about whether dead players should look different, I found that dead players should look greyer. They should look as if they are dead and clearly not in the game. This would be important as players need to be distinguished based on if they are alive or not. Leading on to the last two points about the GUI when

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

the game ends, based on the responses of my stakeholders, I will include a leaderboard for all players but mainly for those players that are more competitive and want to compare stats. I will also include the end game animation, which from the responses of my stakeholders, would be good to show the players that the game is over as well as a smooth transition displaying to players that the game is over, and a new game will start. In conclusion, my stakeholders do want the GUI features I asked them about. My game will be quite heavy on GUI as the information used to communicate to players is graphical as it is a game.

Designing in game interactions/ collisions

Stakeholder interview

Question No.	Question	Roland	James
1	Do you think it would be a good idea to use collisions to stop players from leaving the map?	Yes, that is a clever idea to stop players from escaping the game.	Collisions is an effective way to stop players from escaping the game map.
2	What are your views on detecting when the player is hit on their own client?	This seems like a good method to use.	Yes, to me that seems good.
3	What are your thoughts on having a hitbox for the player that is smaller than their character size?	I think that is good. It is good to give the player some leeway when it comes to collisions from other players.	Yes. That seems fine. I believe that a slightly smaller hitbox works.

Summary of findings

Overall, from asking my stakeholders about what their views are on collisions restricting players from leaving the game map, I have concluded that they agree that that is a good method to stop players from leaving the map. Furthermore, my stakeholders agree that detecting player collisions when it comes to the weapons, guns, bullets, and magic attacks should be done on the client. Lastly my stakeholders agreed that having the player hitbox slightly smaller is good as it allows players to be given more leeway when it comes to collisions as it must hit their character to deal damage otherwise the player isn't damaged. This should also stop players from visually not being hit but still being damaged.

Conclusion from all interviews

From all the stakeholder interviews I did above, my main goal was to see what their views were on how I would develop my game as well as what was going to be implemented. The overall goal was to split my project that will be designed into multiple separate sections that I will design. Asking my stakeholders their views and opinions on each section is useful as it ensures that the game is tailored to their specific needs rather than me creating a game that is not what they would have wanted or keeping them out of the loop when it comes to the design and implementation process. Finally, using their feedback is useful to me as I know how to create a game that they would want to play, and it means I can effectively make a better game with their input and make a game tailored to their wants and needs when it comes to a multiplayer game.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Sprint 1

Justification of sprint 1

Aims and justification for sprint 1.

For sprint 1, I am going to start to develop the encrypted server-client connection along with the ability for clients to connect to the server. To add onto my last point, I will implement number 11 from my User Requirements list so that the server is robust allowing for players to seamlessly join and leave at any time. I will also add some basic collisions and player movement with a camera object that centres the player in the middle of the screen. Furthermore, I will sort through all the sprites and draw the sprites with the highest y-axis coordinate value first to create an illusion of 3D depth.

Client-Server Connection

Firstly, for my Client-Server connection, I will create an encryption module which generates a public and private key. The server will then send over the public key to the client. Once the client receives the public key, it will create a symmetric key and send it over to the server to then receive the encrypted player dictionary. This way a secure connection is created between each client and the server. In addition, the server will be checking to receive data from clients. If the server receives no data from a client, then that connection is terminated leaving for the client to seamlessly leave without affecting other players. Furthermore, the server will be able to add new clients to the game without disrupting the connections of any existing players. The reason I am developing the encrypted Client-Server connection first is because this is the hardest part to successfully implement due to encryption and connection clients through the firewall being complicated. Furthermore, I am using asymmetric encryption to exchange a symmetric key. This can be time consuming and difficult to do because multiple exchanges from the server and client must be done successfully executed smoothly, but also ensuring existing connections are not affected.

Creating Players

After the Client-Server connection is established, I will focus on creating a player to be controlled with the WASD keys. When each client loads up, they will be able to see their own player but also the character of other clients connected. Players will be able to move in different directions (up, down, left, and right). If any player hits a barrier object, they will not be able to move past the barrier. The main reason for creating the player sprite first will be to start off with a basic player for later in development. This is useful because already having a player sprite gives me an idea of how I can add the rest of my User Requirements to my game later. In addition, adding player animations is easier if I have an existing player and player class to manipulate. Furthermore, having a simplified version of my player will help create an idea of how implement the game mechanics in future and how I can adapt my player.

Player Collisions

For player collisions, I will create some basic barrier blocks. These barrier blocks will be visible for players to see as they are being used to create a simple map. The player will be able to collide with these barrier blocks. When the player hits a barrier block, they will be able to move past the barrier block. They will be able to move away from the barrier as the purpose of the barriers is to demonstrate a collision detection system whilst also creating the mechanics for the invisible barrier blocks in further sprints. The reason creating barrier blocks is necessary in sprint 1 is it will keep

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

players from escaping from the map and because the map is infinite, there must be a system in place to limit players to an enclosed area.

Player Centred Camera

Finally, the player centred camera will be used for each client. Moreover, the camera object will render all the sprites based on their descending y-axis values. This will be used to create some 3D like depth in the game. The player centred camera will find the centre coordinate position from the player and use that to generate an offset position vector. Using the position vector, the camera can then find the offset vector for the map, barrier object and other players based on the offset position of the player. This is necessary to implement early into the game because exploring a large map without a player centred camera is limiting to a smaller map. Furthermore, a player centred camera will make development easier because the system used to create the camera effect can be easily updated for later sprints.

Success Criteria

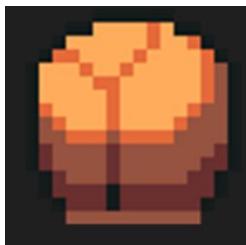
1. Create a simple, encrypted Client-Server connection using asymmetric encryption to exchange a symmetric key. Allow the server to send over the public key and to decrypt the symmetric key with the private key.
2. Allow clients to seamlessly connect and disconnect without interrupting the server or disrupting currently connected clients. Clients should not experience any lag when leaving or joining and other clients should not be affected whether a client joins or leaves the game.
3. Create a simple player that can be controlled with the WASD keys. Player movement should stop when they hit a barrier. All clients should be able to see all the other players moving on their screen.
4. Create visible barrier blocks. They should stop players from moving past the barrier. Barriers should not move and enclose the players in the map. Players should be able to move away from the barriers.
5. There should be a camera object that will draw the player in the centre of the screen. The camera object should also draw the floor, other players and the barrier blocks relative to the player.
6. The camera should also draw all the sprites in descending y-axis value. A sprite with a lower y-axis value will be drawn first to create the illusion of depth.

Design

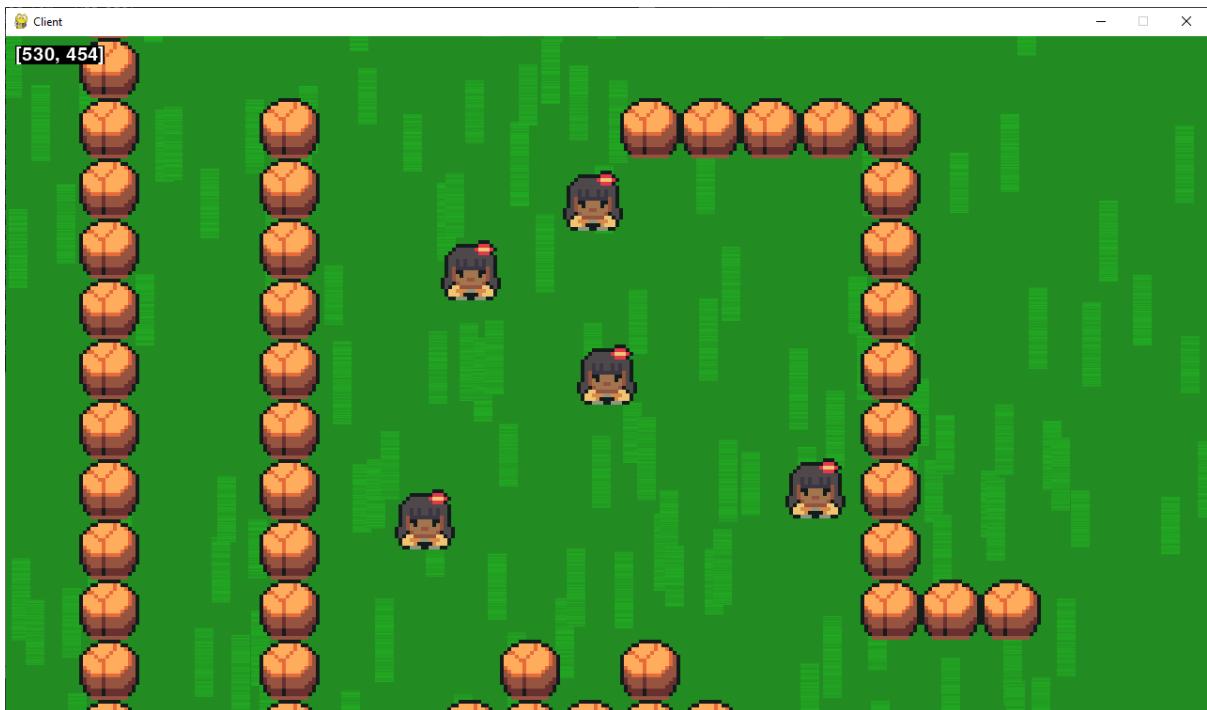


Design of GUI

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335



For my design of character in sprint one, I am only going to use a simple 2D character. For sprint 1, this will be the only image for the player. They will move in four different directions (up, down, left, and right). For the barriers, I will use this 64 by 64-pixel rock. In the final sprint the barriers will be invisible however, for sprint 1, the barrier will be this rock sprite to demonstrate some basic barrier blocks. My game will have a simple background image that is only temporary. My game will have the rock barrier blocks placed around the map to create an enclosed environment for the player. This is what my game would look like for 5 players. For this temporary version, this design will show each players' location for each client. This design will only be simple only to show the possibility of a server-client connection that allow different clients to interact with objects.



Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

Pseudo code

Pseudo code and program code is in the appendix.

Validation table

Test Case	Input	Expected Output	Expected Behaviour
Case 1	Server encrypted connection	Encrypted connection established	Secure connection between server and client
Case 2	Seamless client-server connection	No disruptions on client join/leave	Stable connections unaffected by joins/leaves
Case 3	Basic player movement	Player halts at barriers	WASD control with collision detection
Case 4	Barrier collision	Stops player movement at barriers	Effective barrier collision detection
Case 5	Client disconnection	Server connection termination	Graceful handling of client exits
Case 6	Robust server handling	Existing clients unaffected	Smooth handling of multiple connections
Case 7	Camera centres player	Player remains centred	Dynamic player-centred camera
Case 8	Depth illusion rendering	Sprites ordered by y-axis	Depth illusion through proper sprite rendering
Case 9	Encryption key transmission	Successful public key exchange	Proper key transmission between server and client
Case 10	Symmetric key exchange	Successful encrypted data transfer	Asymmetric encryption for key exchange
Case 11	Disconnected client handling	Connection closure without impact	Proper disconnected client handling
Case 12	Boundary restriction	Limits player movement within map	Effective boundaries to confine players
Case 13	Camera adjusts with player	Renders objects around player	Dynamic camera relative to player's position
Case 14	Multiple client addition	No disruption to existing clients	Seamless multiple client additions
Case 15	Barrier block creation	Blocks restrict player movement	Effective barriers to enclose players

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

Case 16	Encrypted player data transmission	Decryption and reception by client	Successful encrypted player data transmission
Case 17	Synchronized player movement	Consistent movement for all clients	Uniform display of player movement
Case 18	Player collision detection	Avoids overlapping among players	Effective collision prevention
Case 19	Firewall bypass connection	Successful connection through firewall	Handling connections despite firewall
Case 20	Network error resilience	Server maintains connections	Robust handling of network instability
Case 21	Dynamic camera positioning	Proper rendering around player	Camera adjusts view with player
Case 22	Client reconnection	Server accepts reconnection	Successful client reconnection
Case 23	Multiple player collision	Effective collision detection	Accurate collision among multiple players
Case 24	Centred view despite movement	Consistent player-centred view	Maintains centred view despite movement
Case 25	Efficient multiple connections	No server crashes or slowdowns	Handling high volume of client connections

Variables and data structures

```

def initialize_server(self):
    self.server = self.SERVER
    self.port = self.PORT
    self.players = {}
    self.connections = 0
    self.s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    self.rsa_keys = RSA_Keys(self.ENCRYPTION_DATA_SIZE)
    self.public_key, self.private_key = self.rsa_keys.export_keys()
    self.rsa_encrypt = RSA_Encryption(self.public_key)

    try:
        self.s.bind((self.server, self.port))
    except socket.error as e:
        logger.error(str(e))
    self.s.listen()
    logger.info("Waiting for connections, Server Started")           The first
  
```

objective of my game is to establish a secure encrypted connection between the client and server. The server creates either creates a new public or private key or finds the public key and private key stored in file. The image below initialises the server by creating the necessary variables and constants like the server and port.

After the server is initialised, it waits for a client to connect here. The code below is running all the time and accepts new client joining. When they do join each client has to exchange keys.

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

def run(self):
    while True:
        conn, addr = self.s.accept()
        self.connections += 1
        logger.info(f"Connected to: {addr}")
        logger.info(f"There are a total of {self.connections} connections!")

        start_new_thread(self.threaded_client, (conn,))
  
```

The next section of code is used to establish an encrypted connection. The new player is created with a unique id string to identify the player. The server sends the public key over to the client then it can receive the symmetric key. This key exchange is necessary to send over the encrypted player. These two functions are responsible for sending and receiving the keys. The function "initialize_network" receives the public first from the server. The client is then able to send the encrypted symmetric key back to the server. In this way, the client and server both have the encryption keys. The client then receives the player dictionary and can get a player class object based off the data.

```

def threaded_client(self, conn):
    try:
        # Create Player
        new_player, key_string = self.create_new_player()
        self.players[key_string] = new_player

        # Send Public Key
        key_to_send = self.serialize(self.public_key)
        data_to_send = self.serialize({'public_key': key_to_send}) #, 'player': encrypted_player_data})
        conn.send(data_to_send)
        logger.info(f"Sending Public Key: {self.public_key}")

        # Get AES Key
        aes_key_dict = self.rsa_encrypt.decrypt(self.unserialize(conn.recv(self.ENCRYPTION_DATA_SIZE)), self.private_key)
        aes_key = aes_key_dict['aes_key']
        logger.info(f"Received AES Key: {aes_key}")

        # Create AES Encryption
        aes_encryption = AES_Encryption(aes_key)
        player_dict_send = {'player':new_player}
        encrypted_player = aes_encryption.encrypt(self.serialize(player_dict_send))
        conn.send(encrypted_player)
        logger.info(f"Sending Player dict to client: {new_player}")

        self.handle_client_communication(conn, key_string, aes_encryption)
    except:
        logger.error("An Error Occurred trying to setup Client-Server connection.")

def initialize_network(self):
    # Setup Network
    self.network = Network()
    self.network.connect()

    # Get Public Key
    public_key_dict = self.unserialize(self.network.receive(self.ENCRYPTION_DATA_SIZE))
    self.public_key = self.unserialize(public_key_dict['public_key'])
    self.rsa_encrypt = RSA_Encryption(self.public_key)
    logger.info(f"Received Public Key: {self.public_key}")

    # Setup and send AES Encryption
    self.aes_key = AES_Keys(self.BITS)
    key = self.aes_key.export_key()
    key_dict = {'aes_key':key}
    encrypted_key_dict = self.serialize(self.rsa_encrypt.encrypt(key_dict))
    self.network.send(encrypted_key_dict)
    logger.info(f"Sending AES KEY: {key}")

    # Receive Player
    self.encryption = AES_Encryption(key)
    data = self.unserialize(self.encryption.decrypt(self.network.receive(self.ENCRYPTION_DATA_SIZE)))
    player_info = data['player']
    self.initialize_player(player_info)
    logger.info(f"Received player dict: {player_info}")

def initialize_player(self, player_info):
    self.players = []
    self.player_x = player_info['x']
    self.player_y = player_info['y']
    self.player_image = player_info['image']
    self.id = player_info['id']
    self.player = Player([self.player_x, self.player_y], self.player_image, self.level.visible_sprites, self.level.obstacle_sprites, self.id)
  
```

When the client is connected to the sever. The server will receive the dictionary for that client. Each client is connected to a server thread. There can be many server threads that all run in the

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

background for each client. However, each server thread can update the main server variables. For example, the players dictionary. This is vital because although each server thread cannot directly talk to each other they are able to update the players dictionary in real time for all clients. In addition, each client will update their own dictionary and as they send it back to the server, their sever thread will only update their client's data. This intern removes any server delay and stops other threads from editing another client's dictionary. Furthermore, the server is then able to send the dictionary of all the players to each client. This allows each client to draw and render the other players on their screen.

```

def handle_client_communication(self, conn, key_string, aes_encryption):
    running = True
    while running:
        try:
            data = aes_encryption.decrypt(self.unserialize(conn.recv(self.DATA_SIZE)))
            self.players[key_string] = data

            if not data:
                logger.info(f"Player {key_string} disconnected.")
                running = False
            else:
                reply = self.players
                encrypted_reply = self.serialize(aes_encryption.encrypt(reply))

            conn.sendall(encrypted_reply)
        except:
            logger.info(f"Player {key_string} lost connection.")
            running = False

    logger.info(f"Connection Closed for Player {key_string}.")
    del self.players[key_string]
    self.connections -= 1
    conn.close()

def main(self):
    while self.run:
        player_dict = {'x': self.player.rect.x, 'y': self.player.rect.y, 'image': self.player_image, 'id': self.id}
        player_encrypted_dict = self.serialize(self.encryption.encrypt(player_dict))
        self.network.send(player_encrypted_dict)
        all_players_dict = self.encryption.decrypt(self.unserialize(self.network.receive(self.DATA_SIZE)))

        logger.debug("ALL DICT", all_players_dict)
        logger.debug("Sending player data:", player_dict)
        logger.debug("Received players dictionary:", all_players_dict)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                self.close()
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_ESCAPE:
                    self.close()
            self.redraw_window(all_players_dict)

```

The code above the what the client does. Once receiving the dictionary containing all the players, it then creates a player object for the other player can renders them on the screen. The function “update_players” first must update the existing players to ensure that no players are duplicated. After that, players that have left the game must be removed from the dictionary and must be removed from the screen. Finally, any player that does not exist is added to the players dictionary and drawn onto the screen. Using this method, we can avoid duplicating players and thus avoid lag as we are reducing the number of player objects being drawn at once.

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

def update_players(self, player_dict):
  # Update existing player instances and remove players that are not in player_dict
  players_to_remove = []

  for player in self.players:
    if player.id in player_dict:
      player_data = player_dict[player.id]
      player.rect.x = player_data['x']
      player.rect.y = player_data['y']
    else:
      logger.debug(f"Removing player instance with id: {player.id}")
      players_to_remove.append(player)

  for player in players_to_remove:
    self.players.remove(player)
    self.level.visible_sprites.remove(player)

  # Create new player instances for players not already in self.players
  for player_data in player_dict.values():
    player_ids = [player.id for player in self.players]
    if player_data['id'] not in player_ids:
      logger.debug(f"Creating new player instance with id: {player_data['id']}")
      new_player = Player(
        [player_data['x'], player_data['y']],
        player_data['image'],
        self.level.visible_sprites,
        self.level.obstacle_sprites,
        player_data['id'])
    )
    self.players.append(new_player)

def redraw_window(self, all_players_dict):
  self.update_players(all_players_dict)
  self.win.fill(self.BG_COLOR)
  self.level.run(self.player)
  debug([self.player.rect.x, self.player.rect.y])
  pygame.display.update()
  self.clock.tick(self.FPS)

```

Another thing to mention is how the players are being drawn on the screen. In level.py, there is a class called “YSortCameraGroup” which is responsible for drawing all the sprites, objects, and the map relative to the player. In the function “custom_draw”, the offset is found for the x-axis and y-axis. This will give the location where the player is in the middle of the map. This offset is then able to position the other players, the objects, and the map relative to the offset of the player. We then find the vector position of the player by subtracting half of the width of the screen from the offset. In addition to drawing everything relative to the location of the player, each sprite is drawn in order from highest y-axis value to the lowest y-axis value. This is used the “sorted” function and puts all the sprites in a list to be drawn in order.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
class YSortCameraGroup(pygame.sprite.Group):
    def __init__(self):

        # general setup
        super().__init__()
        self.display_surface = pygame.display.get_surface()
        self.half_width = self.display_surface.get_size()[0] // 2
        self.half_height = self.display_surface.get_size()[1] // 2
        self.offset = pygame.math.Vector2()

    # Floor
    self.floor_surf = pygame.image.load('../Graphics/ground.png').convert()
    self.floor_rect = self.floor_surf.get_rect(topleft = (-600,-600)) # in order to not see the white

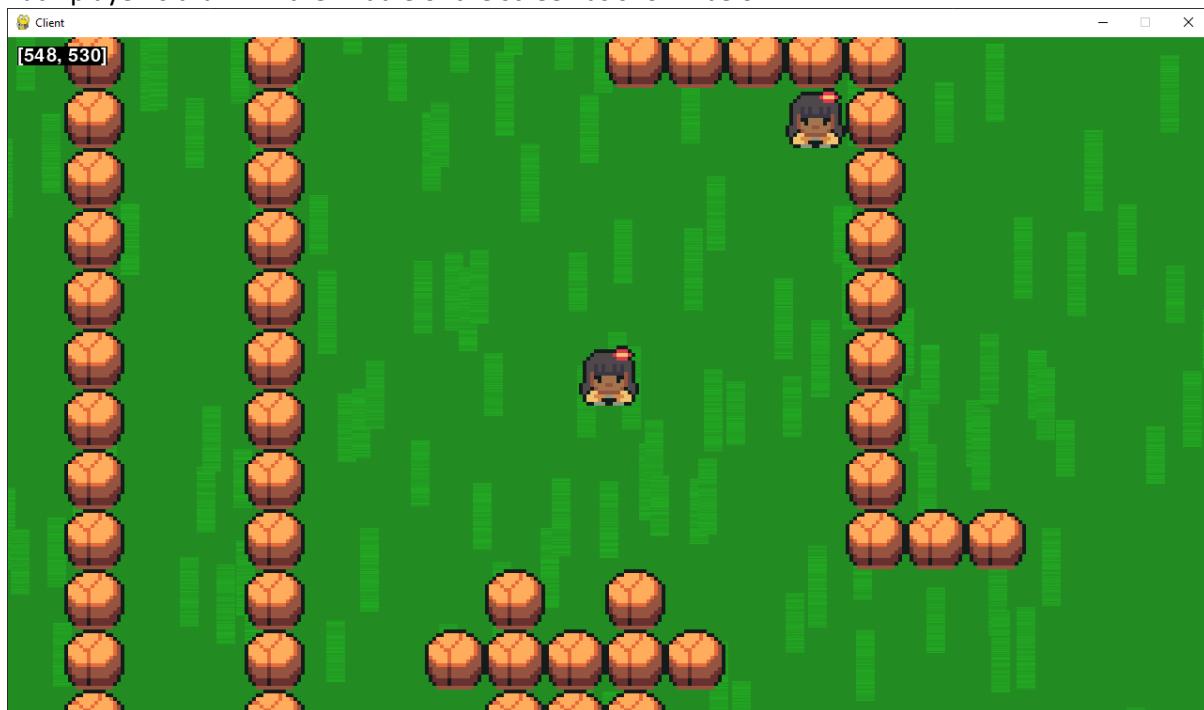
    def custom_draw(self, player):

        # getting the offset
        self.offset.x = player.rect.centerx - self.half_width
        self.offset.y = player.rect.centery - self.half_height

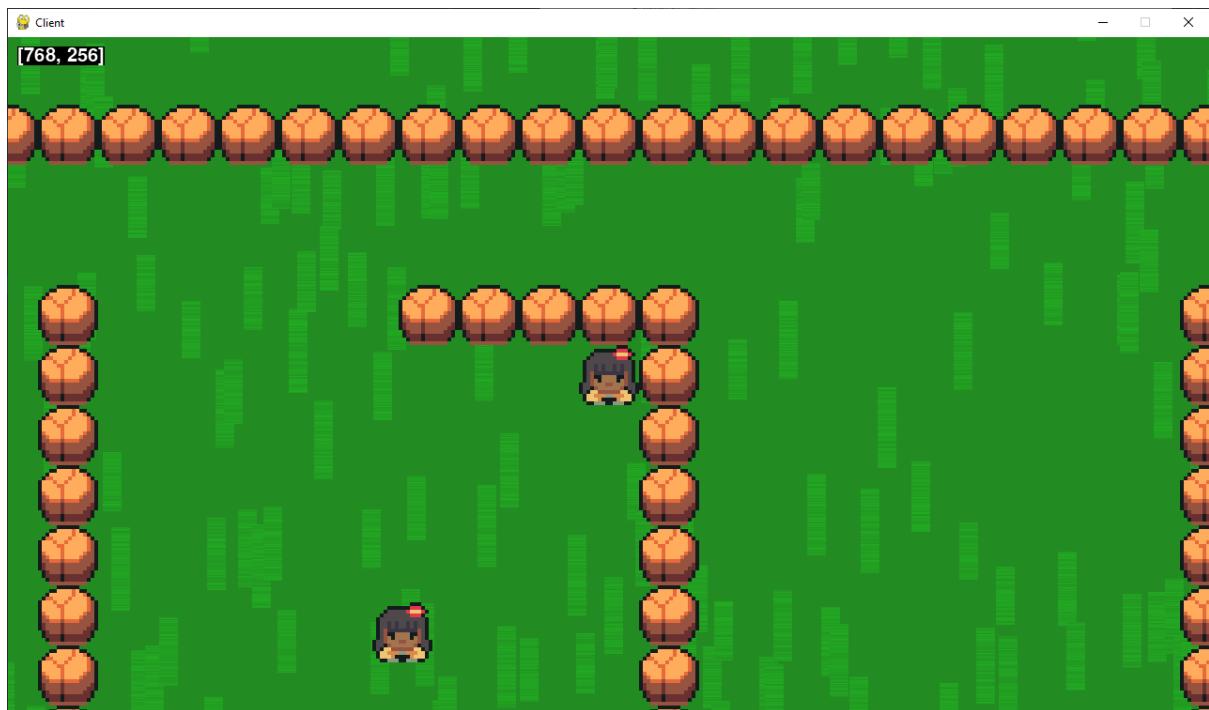
        # drawing the floor
        floor_offset_pos = self.floor_rect.topleft - self.offset
        self.display_surface.blit(self.floor_surf,floor_offset_pos)

        # for sprite in self.sprites():
        for sprite in sorted(self.sprites(),key = lambda sprite: sprite.rect.centery):
            offset_pos = sprite.rect.topleft - self.offset
            self.display_surface.blit(sprite.image,offset_pos)
```

Each player is drawn in the middle of the screen as shown below.



Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335



Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

def move(self, speed):
    if self.direction.magnitude() != 0:
        self.direction = self.direction.normalize()

    self.rect.x += self.direction.x * speed
    self.collision('horizontal')
    self.rect.y += self.direction.y * speed
    self.collision('vertical')

def collision(self, direction):
    if direction == 'horizontal':
        for sprite in self.obstacle_sprites:
            if sprite.rect.colliderect(self.rect):
                if self.direction.x > 0: # moving right
                    self.rect.right = sprite.rect.left
                if self.direction.x < 0: # moving left
                    self.rect.left = sprite.rect.right

    if direction == 'vertical':
        for sprite in self.obstacle_sprites:
            if sprite.rect.colliderect(self.rect):
                if self.direction.y > 0: # moving down
                    self.rect.bottom = sprite.rect.top
                if self.direction.y < 0: # moving up
                    self.rect.top = sprite.rect.bottom

def update(self):
    self.input()
    self.move(self.speed)
def input(self):
    keys = pygame.key.get_pressed()

    if keys[pygame.K_LEFT] or keys[pygame.K_a]:
        self.direction.x = -1
    elif keys[pygame.K_RIGHT] or keys[pygame.K_d]:
        self.direction.x = 1
    else:
        self.direction.x = 0

    if keys[pygame.K_UP] or keys[pygame.K_w]:
        self.direction.y = -1
    elif keys[pygame.K_DOWN] or keys[pygame.K_s]:
        self.direction.y = 1
    else:
        self.direction.y = 0
  
```

Each player is drawn in the middle of the screen. Above I have shown an example of two players. One player is trying to move past an object. As stated in my User Requirements and Justification of Sprint 1, no players will be allowed to move past an object because the function of the objects are to encase players in a limited map. When a player decides to move, we check whether they have collided with an object. The first thing to mention is that a collision is detected between the objects in "self.obstacle_sprites". This detects any collisions between an obstacle and the player. If there is a collision, then the player position is fixed, and they cannot move unless they move away from the obstacle. When the player does decide to move, I have implemented a method to detect whether the WASD keys are being pressed. When the keys are pressed the direction of the y-axis or x-axis is changed. This intern with

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

multiplying the players x-coordinate or y-coordinate by the speed moves the player. Lastly, when the client sends and receives data from the server, there is a separate module used to send and receive the packets.

```

class Network(Config):
    def __init__(self):
        Config.__init__(self)
        self.client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.server = self.SERVER
        self.port = self.PORT
        self.addr = (self.server, self.port)

    def connect(self):
        try:
            self.client.connect(self.addr)
        except socket.error as Error:
            logger.error(f"Socket failed trying to connect: {Error}")

    def receive(self, data_size):
        try:
            return self.client.recv(data_size)
        except socket.error as Error:
            logger.error(f"Socket failed trying to receive: {Error}")

    def send(self, data):
        try:
            self.client.send(data)
        except socket.error as Error:
            logger.error(f"Socket failed trying to send: {Error}")

    def sendall(self, data):
        try:
            self.client.sendall(data)
        except socket.error as Error:
            logger.error(f"Socket failed trying to sendall: {Error}")

```

The network module is used by the client script. The client script connects to the server. The functions “receive”, “send” and “sendall” are then all used to transmit data.

Test Data during iterative development

Test Data Description	Test Scenarios
Client-Server Connection	
Simulate 3 client connections and disconnections	Connect 3 clients, disconnect 2, verify others are still connected and disconnected clients' sprites are removed.
Create new public and private key.	Server creates a new public and private key and successfully establishes an encrypted connection
Send encrypted messages between client and server	Test various encrypted message exchanges, validate decryption
Player Creation and Movement	
Validate player movement with WASD inputs	Move player in all directions, ensure smooth movement

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

Test player collisions with barrier blocks	Ensure players cannot pass through barrier blocks
Player-Centred Camera	
Check player-centred camera functionality	Verify camera centres on player, tracks movement smoothly
Test with 5 simultaneous client movements	Simulate 5 clients moving at the same time, observe camera handling
Test handling abrupt disconnections	Disconnect a player abruptly, check camera adjustment

Copyright

Something very important to note is parts of my code has been taken and used from tech with Tim. Tech with Tim created a simple multiplayer game and I have used that architecture to create my game. I have also use elements from clear code. For example, the player class from clear code was used to create my player. My game will use many elements of clear code game, but I do believe that they are distinctly different as I am intending the create a multiplayer game with collisions different to clear codes game. I will be using his graphics, music, and sound effects in his game to create my project.

Appendix

Pseudo code:

Client:

```

import pygame, sys
from network import Network
from settings import Config
from level import Level
from player import Player
from logger import *
from encryption import *
from debug import debug

```

```

class Client inherits Config
  private player
  private id
  private player_image
  private player_y
  private player_x
  private players
  private encryption
  private aes_key
  private rsa_encrypt
  private public_key
  private network
  private run
  private level
  private clock
  private win

  public procedure new()
    Config.__init__()

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

pygame.init()
initialize_pygame()
initialize_network()
endprocedure

public procedure initialize_pygame()
  win = new pygame.display.set_mode((WIDTH, HEIGHT))
  pygame.display.set_caption("Client")
  clock = new pygame.time.Clock()
  level = new Level()
  run = True
endprocedure

public procedure initialize_network()
  // Setup Network
  network = new Network()
  network.connect()
  // Get Public Key
  public_key_dict = new unserialize(network.receive(ENCRYPTION_DATA_SIZE))
  public_key = new unserialize(public_key_dict['public_key'])
  rsa_encrypt = new RSA_Encryption(public_key)
  logger.info(f'Received Public Key: {public_key}')
  // Setup and send AES Encryption
  aes_key = new AES_Keys(BITS)
  key = new aes_key.export_key()
  key_dict = {'aes_key':key}
  encrypted_key_dict = new serialize(rsa_encrypt.encrypt(key_dict))
  network.send(encrypted_key_dict)
  logger.info(f'Sending AES KEY: {key}')
  // Receive Player
  encryption = new AES_Encryption(key)
  data = new unserialize(encryption.decrypt(network.receive(ENCRYPTION_DATA_SIZE)))
  player_info = data['player']
  initialize_player(player_info)
  logger.info(f'Received player dict: {player_info}')
endprocedure

public procedure initialize_player(player_info)
  players = []
  player_x = player_info['x']
  player_y = player_info['y']
  player_image = player_info['image']
  id = player_info['id']
  player = new Player([player_x, player_y], player_image, level.visible_sprites, level.obstacle_sprites, id)
endprocedure

public procedure update_players(player_dict)
  // Update existing player instances and remove players that are not in player_dict
  players_to_remove = []
  for player in players
    if player.id in player_dict then
      player_data = player_dict[player.id]
      player.rect.x = player_data['x']
      player.rect.y = player_data['y']
    else
      logger.debug(f'Removing player instance with id: {player.id}')
      players_to_remove.append(player)
    endif
  next player
  for player in players_to_remove

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

players.remove(player)
level.visible_sprites.remove(player)
next player
// Create new player instances for players not already in players
for player_data in player_dict.values()
  player_ids = [player.id for player in players]
  if player_data['id'] NOT in player_ids then
    logger.debug(f"Creating new player instance with id: {player_data['id']}")
    new_player = Player(
      [player_data['x'], player_data['y']],
      player_data['image'],
      level.visible_sprites,
      level.obstacle_sprites,
      player_data['id']
    )
    players.append(new_player)
  endif
next player_data
endprocedure

public procedure redraw_window(all_players_dict)
  update_players(all_players_dict)
  win.fill(BG_COLOR)
  level.run(player)
  debug([player.rect.x, player.rect.y])
  pygame.display.update()
  clock.tick(FPS)
endprocedure

public procedure close()
  pygame.quit()
  sys.exit()
endprocedure

public procedure main()
  while run
    player_dict = {'x': player.rect.x, 'y': player.rect.y, 'image': player_image, 'id': id}
    player_encrypted_dict = new serialize(encryption.encrypt(player_dict))
    network.send(player_encrypted_dict)
    all_players_dict = new encryption.decrypt(unserialize(network.receive(DATA_SIZE)))
    logger.debug("ALL DICT", all_players_dict)
    logger.debug("Sending player data:", player_dict)
    logger.debug("Received players dictionary:", all_players_dict)
    for event in pygame.event.get()
      if event.type == pygame.QUIT then
        close()
      endif
      if event.type == pygame.KEYDOWN then
        if event.key == pygame.K_ESCAPE then
          close()
        endif
      endif
    next event
    redraw_window(all_players_dict)
  endwhile
endprocedure

if __name__ == "__main__":
  client = new Client()
  client.main()

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

endif

Debug:

```
import pygame
from logger import *

pygame.init()
font = new pygame.font.Font(None,30)
logger.debug("RealDL Encryption Code.")
procedure debug(info,y = new 10, x = new 10)
endprocedure

display_surface = new pygame.display.get_surface()
debug_surf = new font.render(str(info),True,'White')
debug_rect = new debug_surf.get_rect(topleft = new (x,y))
pygame.draw.rect(display_surface,'Black',debug_rect)
display_surface.blit(debug_surf,debug_rect)
```

Encryption:

```
import rsa, pickle, os
from logger import *
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
from Crypto.Random import get_random_bytes

logger.info('RealDL Encryption Module : RSA')
class RSA_Keys
    private public_key,
    private private_key
    private public_key
    private bits

    public procedure new(bits)
        bits = bits
        key_dir = new "../Keys" if os.path.exists("../Keys") else "Keys"
        endif
        public_key_path = new os.path.join(key_dir, "public.pem")
        private_key_path = new os.path.join(key_dir, "private.pem")
        try
            with open(public_key_path, "rb") as f:
                public_key = new rsa.PublicKey.load_pkcs1(f.read())
            with open(private_key_path, "rb") as f:
                private_key = new rsa.PrivateKey.load_pkcs1(f.read())
        except FileNotFoundError
            public_key, private_key = new rsa.newkeys(bits)
            with open(public_key_path, "wb") as f:
                f.writeLine(public_key.save_pkcs1("PEM"))
            with open(private_key_path, "wb") as f:
                f.writeLine(private_key.save_pkcs1("PEM"))
        endtry
    endprocedure

    function export_keys()
        return public_key, private_key
    endfunction

class RSA_Encryption
    private public_key

    public procedure new(public_key)
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
try
    public_key = public_key
except ValueError as e
    logging.error(f"Error: {e}")
endtry
endprocedure

function encrypt(message)
    try
        return rsa.encrypt(serialise(message), public_key)
    except ValueError as e
        logging.error(f"Error: {e}")
    endtry
endfunction

function decrypt(encrypted_message, private_key)
    try
        encoded_message = new rsa.decrypt(encrypted_message, private_key)
        return unserialise(encoded_message)
    except ValueError as e
        logging.error(f"Error: {e}")
    endtry
endfunction

function serialise(data)
    try
        return pickle.dumps(data)
    except ValueError as e
        logging.error(f"Error: {e}")
    endtry
endfunction

function unserialise(data)
    try
        return pickle.loads(data)
    except ValueError as e
        logging.error(f"Error: {e}")
    endtry
endfunction

class AES_Keys
    private key
    private bytes

    public procedure new(bits)
        bytes = bits DIV 8
        key = new get_random_bytes(bytes)
    endprocedure

    function export_key()
        return key
    endfunction

class AES_Encryption
    private cipher
    private key

    public procedure new(key)
        try
            key = key
        
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

cipher = new AES.new(key, AES.MODE_ECB)
except ValueError as e
  logging.error(f"Error: {e}")
endtry
endprocedure

function encrypt(message)
try
  message_bytes = new serialise(message)
  padded_message = new pad(message_bytes, AES.block_size)
  return cipher.encrypt(padded_message)
except ValueError as e
  logging.error(f"Error: {e}")
endtry
endfunction

function decrypt(encrypted_message)
try
  decrypted_padded_message = new cipher.decrypt(encrypted_message)
  decrypted_unpadded_bytes_message = new unpad(decrypted_padded_message, AES.block_size)
  return unserialise(decrypted_unpadded_bytes_message)
except ValueError as e
  logging.error(f"Error: {e}")
endtry
endfunction

function serialise(data)
try
  return pickle.dumps(data)
except ValueError as e
  logging.error(f"Error: {e}")
endtry
endfunction

function unserialise(data)
try
  return pickle.loads(data)
except ValueError as e
  logging.error(f"Error: {e}")
endtry
endfunction
  
```

Level:

```

import pygame
from settings import *
from tile import Tile
from logger import *

logger.info("RealDL Level Code.")

class Level inherits Config
  private obstacle_sprites
  private visible_sprites
  private display_surface

  public procedure new()
    Config.__init__()
    // get the display surface
    display_surface = new pygame.display.get_surface()
    // sprite group setup
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

visible_sprites = new YSortCameraGroup()
obstacle_sprites = new pygame.sprite.Group()
// sprite setup
create_map()
endprocedure

public procedure create_map()
for row_index, row in enumerate(WORLD_MAP)
  for col_index, col in enumerate(row)
    x = col_index * TILESIZE
    y = row_index * TILESIZE
    if col == 'x' then
      Tile((x,y),[visible_sprites,obstacle_sprites],'Graphics/rock.png',TILE_ID)
    endif
    next col_index, col
  next row_index, row
endprocedure

public procedure run(player)
// update and draw the game
visible_sprites.custom_draw(player)
visible_sprites.update()
endprocedure

class YSortCameraGroup inherits pygame.sprite.Group
private for
private offset.y
private offset.x
private floor_rect
private floor_surf
private offset
private half_height
private half_width
private display_surface

public procedure new()
// general setup
super.new()
display_surface = new pygame.display.get_surface()
half_width = new display_surface.get_size()[0] DIV 2
half_height = new display_surface.get_size()[1] DIV 2
offset = new pygame.math.Vector2()
// Floor
try
  floor_surf = new pygame.image.load('Graphics/ground.png').convert()
except
  floor_surf = new pygame.image.load('../Graphics/ground.png').convert()
endtry
floor_rect = new floor_surf.get_rect(topleft = new (-600,-600)) // in order to not see the white
endprocedure

public procedure custom_draw(player)
// getting the offset
offset.x = player.rect.centerx - half_width
offset.y = player.rect.centery - half_height
// drawing the floor
floor_offset_pos = floor_rect.topleft - offset
display_surface.blit(floor_surf,floor_offset_pos)
// for sprite in sprites():
for sprite in sorted(sprites(),key = new lambda sprite sprite.rect.centery)

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

offset_pos = sprite.rect.topleft - offset
display_surface.blit(sprite.image,offset_pos)
next sprite
endprocedure
  
```

Logger:

```

import logging

// Configure logging
logging.basicConfig(level= new logging.DEBUG) // Set the base logging level
// Create a logger instance
logger = new logging.getLogger(__name__)
logger.propagate = False // Disable propagation to parent logger
// Create a formatter for the log messages

class ColoredFormatter inherits logging.Formatter
  COLORS = {
    logging.DEBUG: "\033[1;34m", // Blue for DEBUG
    logging.INFO: "\033[1;29m", // Grey for INFO
    logging.WARNING: "\033[1;33m", // Yellow for WARNING
    logging.ERROR: "\033[1;91m", // Orange for ERROR
    logging.CRITICAL: "\033[1;31m" // Red for CRITICAL
  }
  function format(record)
    log_color = new COLORS.get(record.levelno, "\033[0m") // Default to no color
    log_level = record.levelname
    timestamp = new formatTime(record, datefmt)
    message = log_color + log_level + "\033[0m" + " - " + record.msg
    return f"{timestamp} - {message}"
  endfunction

// Create a handler for console output with the colored formatter
colored_console_handler = new logging.StreamHandler()
colored_console_handler.setFormatter(ColoredFormatter())
// Add the colored handler to the logger
logger.addHandler(colored_console_handler)
logger.info("RealDL Logger Code.")
  
```

Network:

```

import socket
from settings import Config
from logger import *

logger.debug("RealDL Network Code.")

class Network inherits Config
  private addr
  private port
  private server
  private client

  public procedure new()
    Config.__init__()
    client = new socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server = SERVER
    port = PORT
    addr = new (server, port)
  endprocedure

  public procedure connect()
  
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
try
    client.connect(addr)
except socket.error as Error
    logger.error(f"Socket failed trying to connect {Error}")
endtry
endprocedure

function receive(data_size)
    try
        return client.recv(data_size)
    except socket.error as Error
        logger.error(f"Socket failed trying to receive {Error}")
    endtry
endfunction

public procedure send(data)
    try
        client.send(data)
    except socket.error as Error
        logger.error(f"Socket failed trying to send {Error}")
    endtry
endprocedure

public procedure sendall(data)
    try
        client.sendall(data)
    except socket.error as Error
        logger.error(f"Socket failed trying to sendall {Error}")
    endtry
endprocedure
```

Player:

```
import pygame
from logger import *

logger.debug("RealDL Player Code.")

class Player inherits pygame.sprite.Sprite
    private rect.top
    private rect.bottom
    private rect.left
    private rect.right
    private rect.y
    private rect.x
    private direction.y
    private direction.x
    private obstacle_sprites
    private id
    private speed
    private direction
    private rect
    private image

    public procedure new(pos, image, groups, obstacle_sprites, id)
        // Setting up player
        try
            super.new(groups)
            try
                image = new pygame.image.load(image).convert_alpha()
            except
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
image = new pygame.image.load(f"../{image}").convert_alpha()
endtry
rect = new image.get_rect(topleft = new pos)
direction = new pygame.math.Vector2()
speed = 5
id = id
obstacle_sprites = obstacle_sprites
except FileExistsError as FileNotFoundError
    logger.error(f"Failed to create Player Class: {FileNotFoundError}")
endtry
endprocedure

public procedure input()
    keys = new pygame.key.get_pressed()
    if keys[pygame.K_LEFT] OR keys[pygame.K_a] then
        direction.x = -1
    endif
    elseif keys[pygame.K_RIGHT] OR keys[pygame.K_d] then
        direction.x = 1
    else
        direction.x = 0
    endif
    if keys[pygame.K_UP] OR keys[pygame.K_w] then
        direction.y = -1
    endif
    elseif keys[pygame.K_DOWN] OR keys[pygame.K_s] then
        direction.y = 1
    else
        direction.y = 0
    endif
endprocedure

public procedure move(speed)
    if direction.magnitude() != new 0 then
        direction = new direction.normalize()
    endif
    rect.x += direction.x * speed
    collision('horizontal')
    rect.y += direction.y * speed
    collision('vertical')
endprocedure

public procedure collision(direction)
    if direction == 'horizontal' then
        for sprite in obstacle_sprites
            if sprite.rect.colliderect(rect) then
                if direction.x > 0 then // moving right
                    rect.right = sprite.rect.left
                endif
                if direction.x < 0 then // moving left
                    rect.left = sprite.rect.right
                endif
            endif
        next sprite
    endif
    if direction == 'vertical' then
        for sprite in obstacle_sprites
            if sprite.rect.colliderect(rect) then
                if direction.y > 0 then // moving down
                    rect.bottom = sprite.rect.top
                endif
            endif
        next sprite
    endif
endprocedure
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

      endif
      if direction.y < 0 then // moving up
        rect.top = sprite.rect.bottom
      endif
    endif
  next sprite
endif
endprocedure

public procedure update()
  input()
  move(speed)
endprocedure

```

Server:

```

from _thread import *
from random import randint, choice
import string, math, socket
from settings import Config
from encryption import *
from logger import *

logger.debug("RealDL Server Code.")

class Server inherits Config
  private rsa_encrypt
  private public_key,
  private rsa_keys
  private s
  private connections
  private players
  private port
  private server

  public procedure new()
    try
      Config.__init__()
      initialize_server()
    except ValueError as Error
      logger.error(f"Couldn't initialize server: {Error}")
    endtry
  endprocedure

  public procedure initialize_server()
    server = SERVER
    port = PORT
    players = {}
    connections = 0
    s = new socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    rsa_keys = new RSA_Keys(ENCRYPTION_DATA_SIZE)
    public_key, private_key = new rsa_keys.export_keys()
    rsa_encrypt = new RSA_Encryption(public_key)
    try
      s.bind((server, port))
    except socket.error as e
      logger.error(str(e))
    endtry
    s.listen()
    logger.info("Waiting for connections, Server Started")
  endprocedure

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

function create_random_string()
  characters = string.ascii_letters + string.digits
  return ".join(choice(characters) for _ in range(ID_STRING_LENGTH))
endfunction

function is_touching(x, y, other_x, other_y, threshold)
  distance = new math.sqrt((x + SQUARE_SIZE / 2 - other_x) ** 2 + (y + SQUARE_SIZE / 2 - other_y) ** 2)
  sum_half_widths = new sum_half_heights = new (SQUARE_SIZE + threshold) / 2
  return distance <= new math.sqrt(sum_half_widths ** 2 + sum_half_heights ** 2)
endfunction

function get_player_position()
  x, y = new randint(256, 763), randint(256, 634)
  while any(is_touching(x, y, p['x'], p['y'], SQUARE_SIZE) for p in players.values())
    x, y = new randint(256, 763), randint(256, 634)
  endwhile
  return x, y
endfunction

function create_new_player()
  key_string = new create_random_string()
  player_x, player_y = new get_player_position()
  return {
    "x": player_x,
    "y": player_y,
    "image": "./Graphics/player.png",
    "id": key_string
  }, key_string
endfunction

public procedure handle_client_communication(conn, key_string, aes_encryption)
  running = True
  while running
    try
      data = new aes_encryption.decrypt(unserialize(conn.recv(DATA_SIZE)))
      players[key_string] = data
      if NOT data then
        logger.info(f"Player {key_string} disconnected.")
        running = False
      else
        reply = players
        encrypted_reply = new serialize(aes_encryption.encrypt(reply))
      endif
      conn.sendall(encrypted_reply)
    except
      logger.info(f"Player {key_string} lost connection.")
      running = False
    endtry
  endwhile
  logger.info(f"Connection Closed for Player {key_string}.")
  del players[key_string]
  connections -= 1
  conn.close()
endprocedure

public procedure threaded_client(conn)
  try
    // Create Player
    new_player, key_string = new create_new_player()
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

players[key_string] = new_player
// Send Public Key
key_to_send = new serialize(public_key)
data_to_send = new serialize({'public_key': key_to_send}) //, 'player': encrypted_player_data})
conn.send(data_to_send)
logger.info(f"Sending Public Key: {public_key}")
// Get AES Key
aes_key_dict = new rsa_encrypt.decrypt(unserialize(conn.recv(ENCRYPTION_DATA_SIZE)),private_key)
aes_key = aes_key_dict['aes_key']
logger.info(f"Received AES Key: {aes_key}")
// Create AES Encryption
aes_encryption = new AES_Encryption(aes_key)
player_dict_send = {'player':new_player}
encrypted_player = new aes_encryption.encrypt(serialize(player_dict_send))
conn.send(encrypted_player)
logger.info(f"Sending Player dict to client: {new_player}")
handle_client_communication(conn, key_string, aes_encryption)
except
  logger.error("An Error Occurred trying to setup Client-Server connection.")
endtry
endprocedure

public procedure run()
  while True
    conn, addr = new s.accept()
    connections += 1
    logger.info(f"Connected to: {addr}")
    logger.info(f"There are a total of {connections} connections!")
    start_new_thread(threaded_client, (conn,))
  endwhile
endprocedure

if __name__ == "__main__":
  server = new Server()
  server.run()
endif
  
```

Settings:

```

from socket import gethostname, gethostbyname
from logger import *
import pickle

logger.debug("RealDL Settings Code.")

class Config
  
```

```

    private WORLD_MAP
    private TILE_ID
    private TILESIZE
    private ENCRYPTION_DATA_SIZE
    private SMALL_DATA
    private DATA_SIZE
    private BG_COLOR
    private ID_STRING_LENGTH
    private PORT
    private SERVER
    private HOST_NAME
    private FPS
    private BITS
    private SQUARE_SIZE
    private WIDTH
  
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
function unserialize(data)
    try
        return pickle.loads(data)
    except pickle.UnpicklingError as Error
        logger.error(f"Failed to unpickle data: {Error}")
    endtry
endfunction

// This class is used for the client and the server.
// This means that if you are the server then you can keep this bit of code.
// But if you are not the server, but still on the same computer (assuming you have the same ipv4) then you will be fine.
// You will need to change this if you are running a Client instance from another computer.
```

Tile:

```
import pygame
from settings import *
from logger import *

logger.debug("RealDL Tile Code.")

class Tile inherits pygame.sprite.Sprite
    private id
    private rect
    private image
    public procedure new(pos,groups,image,id)
        try
            super.new(groups)
        try
            image = new pygame.image.load(image).convert_alpha()
        except
            image = new pygame.image.load(f"../{image}").convert_alpha()
        endtry
        rect = new image.get_rect(topleft = new pos)
        id = id
    except FileExistsError as FileNotFoundError
        logger.error(f"Failed to create Tile Class: {FileNotFoundError}")
    endtry
endprocedure
endclass
```

Program code:

Client:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
import pygame, sys
from network import Network
from settings import Config
from level import Level
from player import Player
from logger import *
from encryption import *
from debug import debug

class Client(Config):
    def __init__(self):
        Config.__init__(self)
        pygame.init()
        self.initialize_pygame()
        self.initialize_network()

    def initialize_pygame(self):
        self.win = pygame.display.set_mode((self.WIDTH, self.HEIGHT))
        pygame.display.set_caption("Client")
        self.clock = pygame.time.Clock()
        self.level = Level()
        self.run = True

    def initialize_network(self):
        # Setup Network
        self.network = Network()
        self.network.connect()

        # Get Public Key
        public_key_dict = self.unserialize(self.network.receive(self.ENCRYPTION_DATA_SIZE))
        self.public_key = self.unserialize(public_key_dict['public_key'])
        self.rsa_encrypt = RSA_Encryption(self.public_key)
        logger.info(f"Received Public Key: {self.public_key}")

        # Setup and send AES Encryption
        self.aes_key = AES_Keys(self.BITS)
        key = self.aes_key.export_key()
        key_dict = {'aes_key':key}
        encrypted_key_dict = self.serialize(self.rsa_encrypt.encrypt(key_dict))
        self.network.send(encrypted_key_dict)
        logger.info(f"Sending AES KEY: {key}")

        # Receive Player
        self.encryption = AES_Encryption(key)
        data = self.unserialize(self.encryption.decrypt(self.network.receive(self.ENCRYPTION_DATA_SIZE)))
        player_info = data['player']
        self.initialize_player(player_info)
        logger.info(f"Received player dict: {player_info}")

    def initialize_player(self, player_info):
        self.players = []
        self.player_x = player_info['x']
        self.player_y = player_info['y']
        self.player_image = player_info['image']
        self.id = player_info['id']
        self.player = Player([self.player_x, self.player_y], self.player_image, self.level.visible_sprites, self.level.obstacle_sprites, self.id)
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
def update_players(self, player_dict):
    # Update existing player instances and remove players that are not in player_dict
    players_to_remove = []

    for player in self.players:
        if player.id in player_dict:
            player_data = player_dict[player.id]
            player.rect.x = player_data['x']
            player.rect.y = player_data['y']
        else:
            logger.debug(f"Removing player instance with id: {player.id}")
            players_to_remove.append(player)

    for player in players_to_remove:
        self.players.remove(player)
        self.level.visible_sprites.remove(player)

    # Create new player instances for players not already in self.players
    for player_data in player_dict.values():
        player_ids = [player.id for player in self.players]
        if player_data['id'] not in player_ids:
            logger.debug(f"Creating new player instance with id: {player_data['id']}")
            new_player = Player(
                [player_data['x'], player_data['y']],
                player_data['image'],
                self.level.visible_sprites,
                self.level.obstacle_sprites,
                player_data['id']
            )
            self.players.append(new_player)

def redraw_window(self, all_players_dict):
    self.update_players(all_players_dict)
    self.win.fill(self.BG_COLOR)
    self.level.run(self.player)
    debug([self.player.rect.x, self.player.rect.y])
    pygame.display.update()
    self.clock.tick(self.FPS)

def close(self):
    pygame.quit()
    sys.exit()
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
def main(self):
    while self.run:
        player_dict = {'x': self.player.rect.x, 'y': self.player.rect.y, 'image': self.player_image, 'id': self.id}
        player_encrypted_dict = self.serialize(self.encryption.encrypt(player_dict))
        self.network.send(player_encrypted_dict)
        all_players_dict = self.encryption.decrypt(self.unserialize(self.network.receive(self.DATA_SIZE)))

        logger.debug("ALL DICT", all_players_dict)
        logger.debug("Sending player data:", player_dict)
        logger.debug("Received players dictionary:", all_players_dict)

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                self.close()
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_ESCAPE:
                    self.close()
            self.redraw_window(all_players_dict)

if __name__ == "__main__":
    client = Client()
    client.main()
```

Debug:

```
import pygame
from logger import *
pygame.init()
font = pygame.font.Font(None,30)

logger.debug("RealDL Encryption Code.")

def debug(info,y = 10, x = 10):
    display_surface = pygame.display.get_surface()
    debug_surf = font.render(str(info),True,'White')
    debug_rect = debug_surf.get_rect(topleft = (x,y))
    pygame.draw.rect(display_surface,'Black',debug_rect)
    display_surface.blit(debug_surf,debug_rect)
```

Encryption:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
import rsa, pickle, os
from logger import *
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
from Crypto.Random import get_random_bytes

logger.info('RealDL Encryption Module : RSA')

class RSA_Keys:
    def __init__(self, bits):
        self.bits = bits
        key_dir = "../Keys" if os.path.exists("../Keys") else "Keys"
        public_key_path = os.path.join(key_dir, "public.pem")
        private_key_path = os.path.join(key_dir, "private.pem")

        try:
            with open(public_key_path, "rb") as f:
                self.public_key = rsa.PublicKey.load_pkcs1(f.read())

            with open(private_key_path, "rb") as f:
                self.private_key = rsa.PrivateKey.load_pkcs1(f.read())
        except FileNotFoundError:
            self.public_key, self.private_key = rsa.newkeys(self.bits)
            with open(public_key_path, "wb") as f:
                f.write(self.public_key.save_pkcs1("PEM"))

            with open(private_key_path, "wb") as f:
                f.write(self.private_key.save_pkcs1("PEM"))

    def export_keys(self):
        return self.public_key, self.private_key

class RSA_Encryption:
    def __init__(self, public_key):
        try:
            self.public_key = public_key
        except ValueError as e:
            logging.error(f"Error: {e}")

    def encrypt(self, message):
        try:
            return rsa.encrypt(self.serialise(message), self.public_key)
        except ValueError as e:
            logging.error(f"Error: {e}")

    def decrypt(self, encrypted_message, private_key):
        try:
            encoded_message = rsa.decrypt(encrypted_message, private_key)
            return self.unserialise(encoded_message)
        except ValueError as e:
            logging.error(f"Error: {e}")

    def serialise(self, data):
        try:
            return pickle.dumps(data)
        except ValueError as e:
            logging.error(f"Error: {e}")
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
def unserialise(self, data):
    try:
        return pickle.loads(data)
    except ValueError as e:
        logging.error(f"Error: {e}")

class AES_Keys:
    def __init__(self, bits):
        self.bytes = bits // 8
        self.key = get_random_bytes(self.bytes)

    def export_key(self):
        return self.key

class AES_Encryption:
    def __init__(self, key):
        try:
            self.key = key
            self.cipher = AES.new(self.key, AES.MODE_ECB)
        except ValueError as e:
            logging.error(f"Error: {e}")

    def encrypt(self, message):
        try:
            message_bytes = self.serialise(message)
            padded_message = pad(message_bytes, AES.block_size)
            return self.cipher.encrypt(padded_message)
        except ValueError as e:
            logging.error(f"Error: {e}")

    def decrypt(self, encrypted_message):
        try:
            decrypted_padded_message = self.cipher.decrypt(encrypted_message)
            decrypted_unpadded_bytes_message = unpad(decrypted_padded_message, AES.block_size)
            return self.unserialise(decrypted_unpadded_bytes_message)
        except ValueError as e:
            logging.error(f"Error: {e}")

    def serialise(self, data):
        try:
            return pickle.dumps(data)
        except ValueError as e:
            logging.error(f"Error: {e}")

    def unserialise(self, data):
        try:
            return pickle.loads(data)
        except ValueError as e:
            logging.error(f"Error: {e}")
```

Level:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
import pygame
from settings import *
from tile import Tile
from logger import *

logger.info("RealDL Level Code.")

class Level(Config):
    def __init__(self):
        Config.__init__(self)

        # get the display surface
        self.display_surface = pygame.display.get_surface()

        # sprite group setup
        self.visible_sprites = YSortCameraGroup()
        self.obstacle_sprites = pygame.sprite.Group()

        # sprite setup
        self.create_map()

    def create_map(self):
        for row_index, row in enumerate(self.WORLD_MAP):
            for col_index, col in enumerate(row):
                x = col_index * self.TILESIZE
                y = row_index * self.TILESIZE
                if col == 'x':
                    Tile((x,y),[self.visible_sprites,self.obstacle_sprites], 'Graphics/rock.png',self.TILE_ID)

    def run(self, player):
        # update and draw the game
        self.visible_sprites.custom_draw(player)
        self.visible_sprites.update()

class YSortCameraGroup(pygame.sprite.Group):
    def __init__(self):

        # general setup
        super().__init__()
        self.display_surface = pygame.display.get_surface()
        self.half_width = self.display_surface.get_size()[0] // 2
        self.half_height = self.display_surface.get_size()[1] // 2
        self.offset = pygame.math.Vector2()

        # Floor
        try:
            self.floor_surf = pygame.image.load('Graphics/ground.png').convert()
        except:
            self.floor_surf = pygame.image.load('../Graphics/ground.png').convert()
        self.floor_rect = self.floor_surf.get_rect(topleft = (-600,-600)) # in order to not see the white
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

def custom_draw(self, player):

    # getting the offset
    self.offset.x = player.rect.centerx - self.half_width
    self.offset.y = player.rect.centery - self.half_height

    # drawing the floor
    floor_offset_pos = self.floor_rect.topleft - self.offset
    self.display_surface.blit(self.floor_surf,floor_offset_pos)

    # for sprite in self.sprites():
    for sprite in sorted(self.sprites(),key = lambda sprite: sprite.rect.centery):
        offset_pos = sprite.rect.topleft - self.offset
        self.display_surface.blit(sprite.image,offset_pos)
  
```

Logger:

```

import logging

# Configure logging
logging.basicConfig(level=logging.DEBUG) # Set the base logging level

# Create a logger instance
logger = logging.getLogger(__name__)
logger.propagate = False # Disable propagation to parent logger

# Create a formatter for the log messages
class ColoredFormatter(logging.Formatter):
    COLORS = {
        logging.DEBUG: "\033[1;34m",      # Blue for DEBUG
        logging.INFO: "\033[1;29m",       # Grey for INFO
        logging.WARNING: "\033[1;33m",    # Yellow for WARNING
        logging.ERROR: "\033[1;91m",      # Orange for ERROR
        logging.CRITICAL: "\033[1;31m"   # Red for CRITICAL
    }

    def format(self, record):
        log_color = self.COLORS.get(record.levelno, "\033[0m") # Default to no color
        log_level = record.levelname
        timestamp = self.formatTime(record, self.datefmt)
        message = log_color + log_level + "\033[0m" + " - " + record.msg
        return f"{timestamp} - {message}"

# Create a handler for console output with the colored formatter
colored_console_handler = logging.StreamHandler()
colored_console_handler.setFormatter(ColoredFormatter())

# Add the colored handler to the logger
logger.addHandler(colored_console_handler)

logger.info("RealDL Logger Code.")
  
```

Network:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
import socket
from settings import Config
from logger import *

logger.debug("RealDL Network Code.")

class Network(Config):
    def __init__(self):
        Config.__init__(self)
        self.client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.server = self.SERVER
        self.port = self.PORT
        self.addr = (self.server, self.port)

    def connect(self):
        try:
            self.client.connect(self.addr)
        except socket.error as Error:
            logger.error(f"Socket failed trying to connect: {Error}")

    def receive(self, data_size):
        try:
            return self.client.recv(data_size)
        except socket.error as Error:
            logger.error(f"Socket failed trying to receive: {Error}")

    def send(self, data):
        try:
            self.client.send(data)
        except socket.error as Error:
            logger.error(f"Socket failed trying to send: {Error}")

    def sendall(self, data):
        try:
            self.client.sendall(data)
        except socket.error as Error:
            logger.error(f"Socket failed trying to sendall: {Error}")
```

Player:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
import pygame
from logger import *

logger.debug("RealDL Player Code.")

class Player(pygame.sprite.Sprite):
    def __init__(self, pos, image, groups, obstacle_sprites, id):
        # Setting up player
        try:
            super().__init__(groups)
            try:
                self.image = pygame.image.load(image).convert_alpha()
            except:
                self.image = pygame.image.load(f"../{image}").convert_alpha()
            self.rect = self.image.get_rect(topleft = pos)
            self.direction = pygame.math.Vector2()
            self.speed = 5
            self.id = id

            self.obstacle_sprites = obstacle_sprites
        except FileNotFoundError as FileNotFoundError:
            logger.error(f"Failed to create Player Class: {FileNotFoundError}")

    def input(self):
        keys = pygame.key.get_pressed()

        if keys[pygame.K_LEFT] or keys[pygame.K_a]:
            self.direction.x = -1
        elif keys[pygame.K_RIGHT] or keys[pygame.K_d]:
            self.direction.x = 1
        else:
            self.direction.x = 0

        if keys[pygame.K_UP] or keys[pygame.K_w]:
            self.direction.y = -1
        elif keys[pygame.K_DOWN] or keys[pygame.K_s]:
            self.direction.y = 1
        else:
            self.direction.y = 0

    def move(self,speed):
        if self.direction.magnitude() != 0:
            self.direction = self.direction.normalize()

        self.rect.x += self.direction.x * speed
        self.collision('horizontal')
        self.rect.y += self.direction.y * speed
        self.collision('vertical')
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

def collision(self,direction):
    if direction == 'horizontal':
        for sprite in self.obstacle_sprites:
            if sprite.rect.colliderect(self.rect):
                if self.direction.x > 0: # moving right
                    self.rect.right = sprite.rect.left
                if self.direction.x < 0: # moving left
                    self.rect.left = sprite.rect.right

    if direction == 'vertical':
        for sprite in self.obstacle_sprites:
            if sprite.rect.colliderect(self.rect):
                if self.direction.y > 0: # moving down
                    self.rect.bottom = sprite.rect.top
                if self.direction.y < 0: # moving up
                    self.rect.top = sprite.rect.bottom

def update(self):
    self.input()
    self.move(self.speed)
  
```

Server:

```

from _thread import *
from random import randint, choice
import string, math, socket
from settings import Config
from encryption import *
from logger import *

logger.debug("RealDL Server Code.")

class Server(Config):
    def __init__(self):
        try:
            Config.__init__(self)
            self.initialize_server()
        except ValueError as Error:
            logger.error(f"Couldn't initialize server: {Error}")

    def initialize_server(self):
        self.server = self.SERVER
        self.port = self.PORT
        self.players = {}
        self.connections = 0
        self.s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.rsa_keys = RSA_Keys(self.ENCRYPTION_DATA_SIZE)
        self.public_key, self.private_key = self.rsa_keys.export_keys()
        self.rsa_encrypt = RSA_Encryption(self.public_key)

        try:
            self.s.bind((self.server, self.port))
        except socket.error as e:
            logger.error(str(e))
        self.s.listen()
        logger.info("Waiting for connections, Server Started")

    def create_random_string(self):
        characters = string.ascii_letters + string.digits
        return ''.join(choice(characters) for _ in range(self.ID_STRING_LENGTH))

    def is_touching(self, x, y, other_x, other_y, threshold):
        distance = math.sqrt((x + self.SQUARE_SIZE / 2 - other_x) ** 2 + (y + self.SQUARE_SIZE / 2 - other_y) ** 2)
        sum_half_widths = sum_half_heights = (self.SQUARE_SIZE + threshold) / 2
        return distance <= math.sqrt(sum_half_widths ** 2 + sum_half_heights ** 2)

    def get_player_position(self):
        x, y = randint(256, 763), randint(256, 634)
        while any(self.is_touching(x, y, p['x'], p['y'], self.SQUARE_SIZE) for p in self.players.values()):
            x, y = randint(256, 763), randint(256, 634)
        return x, y
  
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
def create_new_player(self):
    key_string = self.create_random_string()
    player_x, player_y = self.get_player_position()
    return {
        "x": player_x,
        "y": player_y,
        "image": "./Graphics/player.png",
        "id": key_string
    }, key_string

def handle_client_communication(self, conn, key_string, aes_encryption):
    running = True
    while running:
        try:
            data = aes_encryption.decrypt(self.deserialize(conn.recv(self.DATA_SIZE)))
            self.players[key_string] = data

            if not data:
                logger.info(f"Player {key_string} disconnected.")
                running = False
            else:
                reply = self.players
                encrypted_reply = self.serialize(aes_encryption.encrypt(reply))

                conn.sendall(encrypted_reply)
        except:
            logger.info(f"Player {key_string} lost connection.")
            running = False

    logger.info(f"Connection Closed for Player {key_string}.")
    del self.players[key_string]
    self.connections -= 1
    conn.close()
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
def threaded_client(self, conn):
    try:
        # Create Player
        new_player, key_string = self.create_new_player()
        self.players[key_string] = new_player

        # Send Public Key
        key_to_send = self.serialize(self.public_key)
        data_to_send = self.serialize({'public_key': key_to_send}) #, 'player': encrypted_player_data})
        conn.send(data_to_send)
        logger.info(f"Sending Public Key: {self.public_key}")

        # Get AES Key
        aes_key_dict = self.rsa_encrypt.decrypt(self.unserialize(conn.recv(self.ENCRYPTION_DATA_SIZE)), self.private_key)
        aes_key = aes_key_dict['aes_key']
        logger.info(f"Received AES Key: {aes_key}")

        # Create AES Encryption
        aes_encryption = AES_Encryption(aes_key)
        player_dict_send = {'player':new_player}
        encrypted_player = aes_encryption.encrypt(self.serialize(player_dict_send))
        conn.send(encrypted_player)
        logger.info(f"Sending Player dict to client: {new_player}")

        self.handle_client_communication(conn, key_string, aes_encryption)
    except:
        logger.error("An Error Occurred trying to setup Client-Server connection.")

def run(self):
    while True:
        conn, addr = self.s.accept()
        self.connections += 1
        logger.info(f"Connected to: {addr}")
        logger.info(f"There are a total of {self.connections} connections!")

        start_new_thread(self.threaded_client, (conn,))

if __name__ == "__main__":
    server = Server()
    server.run()
```

Settings:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Tile:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
import pygame
from settings import *
from logger import *

logger.debug("RealDL Tile Code.")

class Tile(pygame.sprite.Sprite):
    def __init__(self, pos, groups, image, id):
        try:
            super().__init__(groups)
            try:
                self.image = pygame.image.load(image).convert_alpha()
            except:
                self.image = pygame.image.load(f"../{image}").convert_alpha()
            self.rect = self.image.get_rect(topleft = pos)
            self.id = id
        except FileNotFoundError as FileError:
            logger.error(f"Failed to create Tile Class: {FileError}")


```

Test proofs:

See the MP4 videos.

Sprint 2

Aims for this sprint.

Adding on my sprint 1, I will be including all the requirements and functionality that I designed whilst also adding new requirements and features in this new sprint.

User Interface

In this sprint, I aim to develop a visually pleasing user interface, including as described in my analysis, three unique buttons. There will be a button to start the game, a button to configure settings and an option to quit the game. In addition, I will have two extra buttons that will be for my YouTube and my GitHub where my project files will be stored. When a user hovers over those two buttons, the opacity of the buttons will be reduced and there will be text that say “YouTube” or “GitHub” under each button. Furthermore, I will add a custom mouse that will change when hovering on buttons. When a button is hovered on it will change colour. The colours will invert and smoothly transition. When each button is click, they will respectively go my GitHub page and my YouTube page on the browser.

For my settings button, I will implement a simple user interface. It will include two options: controls and audio settings. The control settings will allow users to select the keys they want to use, and the audio settings will control the volume of the music and sounds as well as if the user wants sound and music in the game.

For the start button, when clicked, will bring up another user interface. It will have a button to go back to the main menu, a button to connect and 2 text boxes. One for the username and one for the Server IP Address. In this sprint only the username of the player will be in the game. There will be no music/sounds and the controls keys will not change the keys used to control the character yet. The text box for Server IP Address will try connecting to the server with the inputted IP address. If the IP address is invalid the user will go back to the main menu, however if the user can successfully connect to the server, they will load their character into the game.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Finally for the main menu, I will have a moving background image of my game map. I will use a portion of the game map (1920 pixels by 1080 pixels). The image will be blurred so that the buttons are highlighted which will create an effective illusion. If the player clicks on start or options, the background image will still be moving to the right.

Player Movement

I also want to add some movement animations to the player. Each player will have a different animation when they move left, right, up, or down. In addition, I will have different animations for the players attacking and when the player is idle. When I animate the player there will be a cooldown period for the attacking movement, the player will not be able to move for a brief period. The player animations will cycle through 4 different animations for the walking. This will be seen by all players.

Game Map

For this sprint, I will add the game map in the game. I will keep the temporary rock barriers; however, I will increase the area from 20 by 20 to 30 by 30. I will still include the collisions from sprint 1 and I will print the players coordinates on the screen in the top left for debugging purposes. Finally, the mouse will still be on the screen when the player is playing the game.

Leaving the game

Lastly, when the player decides to leave the game, all they must press is escape. This will bring them back to the main menu screen where they can either join a different server or they can quit by pressing escape again or by clicking on the quit button.

Success Criteria

1. **Visually Pleasing Interface:** Develop a user interface with three unique buttons (Start, Settings, Quit) and additional buttons for YouTube and GitHub, ensuring a visually appealing design.
2. **Interactive Button Features:** Implement hover effects on YouTube and GitHub buttons, including reduced opacity and text display on hover, along with a custom cursor and colour transition on click.
3. **Settings Interface:** Create a simple settings user interface allowing users to customise controls and manage audio settings (volume, sound/music toggles).
4. **Start Menu Interface:** Develop a start menu with options to go back, connect, and input username/Server IP Address, initiating connection with valid IP and loading the player's character.
5. **Dynamic Main Menu:** Design a moving background image of the game map for the main menu, employing a blurred effect to highlight buttons and maintaining movement even when options are selected.
6. **Player Movement Animations:** Implement different animations for player movement in all directions (left, right, up, down), including attack and idle animations with a cooldown period for attack movements.
7. **Multi-Animation Cycle:** Set up a cycling system of four animations for player walking, ensuring synchronization and visibility to all players.
8. **Game Map Integration:** Introduce the game map into the interface, expanding the area, maintaining previous collisions, displaying player coordinates for debugging, and keeping the mouse visible during gameplay.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

9. **User Exit Functionality:** Enable easy exit from the game by pressing the 'Escape' key, offering options to return to the main menu, join another server, or quit the game entirely.

Justification of sprint 2

User Interface

The reason I am developing a menu interface and different button options for my game is because my stakeholders wanted the ability to change their settings in the game. They wanted the ability to customise their game settings. Another reason for developing my user interface is to create an aesthetically pleasing user interface as described in my user requirements. Furthermore, point two from my user requirements in my analysis states there must be a main menu system with buttons for settings, to quit the game and to start the game.

When I create my three buttons, I will use a custom python script that uses Pygame to create a button. When each button is hovered, I will invert the colours as described in my Aims for sprint 2. How I will do this is by slowly changing the colour in steps by changing the RGB value by a set increment until it reaches the inverted RGB value. When it comes to creating the mouse, I will find the coordinates of the mouse using Pygame and hide the cursor and replace the cursor with a custom image that follows the coordinates of the original cursor. When a user clicks on the GitHub or YouTube buttons, I will slowly reduce the image opacity by a small amount each time until it reaches the desired opacity. This is also how the text under those buttons will work but in reverse. The opacity of the text will seamlessly increase from no opacity to full opacity in a few seconds.

When a user clicks on those buttons, I will use python's inbuilt module called webbrowser that will be able to open the websites in a web browser. For my quit button, the application will close using Pygame's inbuild function that closes the Pygame windows. As well as that, I will close the python application with the sys module. However, my options button, to create the interface, I will require giving that page an ID. This ID if ran will show up the options menu. I will use my Pygame functions module to create the user interface. In addition, the options menu has two subpages. These will run under the page ID for settings. Each subpage will have a small window inside the settings interface and will either display the controls or audio settings. Lastly for my start button I will have another page ID for that. Unlike the settings page, I will require two text boxes which I will have to create. Using my custom Pygame functions script I will be able to make text boxes in python where users will be able to input text into. The text boxes will require the user to click on them to be able to type in. They will also change colour to show the user that the text box is active. Furthermore, I will be able to take the text from the username and IP address text box and connect to the IP address entered and if connected, send the username to the server. For my moving background, that will be active regardless of what page the user is currently on (whether it is the start page or the settings page). The moving background will draw two images of the background. The first image will be displaying the background image with the x-axis value being increased by one each time. The second image will be displaying the background image at the same x-axis value as the first image; however, the x-axis value will be reduced by the width of the background image. This creates a moving effect by always increasing the x-axis value by one.

Player Movement

I will be adding player walking animations into this sprite because my stakeholders wanted a more graphically pleasing interface. Furthermore, player animations can make the game seem more engaging and satisfying as my stakeholders did want a graphically pleasing game.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

My animations for the player will run every 0.15 seconds. I will have an animation function in my player class. This function will get the list of animations from the players status. Then I will find the current frame to display on the screen which will be done by increasing the frame index by the animation speed of 0.15 seconds. The image will be then found by finding the position of the image in the animations list. When each client sends the server their data, they will retrieve their current animation image. When this is received by all the clients, a separate player class is created for the other players where we can update the image of the players on other clients.

Game Map

As described in my analysis, my game is a last man standing game. This sort of game requires a map of some sort. I have decided to implement a basic version of my map in sprint two. My stakeholders did want a visually pleasing game and my map will be an aesthetic map.

In my settings script, I will increase world map list to 30 by 30. This is to make the map temporarily larger as in the next sprint I will remove the visible barriers and I will opt for invisible barrier blocks. For the debug screen I will retrieve the coordinates from the player then draw a simple black rectangle around the x-axis and y-axis position. Finally for the game map I will draw the image with Pygames built in 'blit' function.

Leaving the game

The players need to be able to leave the game and disconnect. One of my user requirements is to have seamless connections and disconnection to and from the server. This will be done by detecting whether the escape button has been pressed. If it has, the connection between the server will be terminated and the client will go back to the main menu screen.

Pseudo Code

The Pseudo Code is in the Appendix.

Python Development

Program Code

Code is in the appendix. (I will email the Appendix to you separately).

Areas of complexity

Menu Board

The first area of complexity that I faced when creating my program code was creating the Functions.py script for my menu. Creating the buttons themselves were not too difficult but changing the colours from the first colour to the second colour was challenging. As shown below, I had to calculate each new colour for every frame. I also needed a way to keep count of the multiplier so that the colour will change.

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

def change_color(self, original_color, new_color, step_multiplier=0):
    # Calculate color change per frame
    colors = []
    new_color_list = []
    for rgb1, rgb2 in zip(original_color, new_color):
        difference = rgb2 - rgb1
        colors.append(difference)

    color_step = [difference / self.num_frames for difference in colors]

    for og_color_rgb, new_color_rgb, step in zip(original_color, new_color, color_step):
        if og_color_rgb > new_color_rgb:
            if int(og_color_rgb + step_multiplier*step) < new_color_rgb:
                new_color_list.append(new_color_rgb)
            else:
                new_color_list.append(int(og_color_rgb + step_multiplier*step))
        else:
            if int(og_color_rgb + step_multiplier*step) > new_color_rgb:
                new_color_list.append(new_color_rgb)
            else:
                new_color_list.append(int(og_color_rgb + step_multiplier*step))

    return tuple(new_color_list)
  
```

The first difficult area was finding the colour step for each RGB value and putting that into a list so that the colour would change accordingly. However, the for loop below seemed to be most challenging. I had to use the zip function to find a new colour. The code uses a linear progression as the colour changes smoothly. Once we have found a new set of RGB values we return it as a tuple. The second difficulty when programming was calculating the change on opacity for the buttons and text.

```

def change_opacity(self, original_opacity, new_opacity, step_multiplier=0):
    # Calculate opacity change per frame
    new_opacity_return = None
    difference = new_opacity - original_opacity
    opacity_step = difference / self.num_frames
    if original_opacity > new_opacity:
        if int(original_opacity + opacity_step*step_multiplier) < new_opacity:
            new_opacity_return = new_opacity
        else:
            new_opacity_return = int(original_opacity + opacity_step*step_multiplier)
    else:
        if int(original_opacity + opacity_step*step_multiplier) > new_opacity:
            new_opacity_return = new_opacity
        else:
            new_opacity_return = int(original_opacity + opacity_step*step_multiplier)

    return new_opacity_return
  
```

The problem was trying to find a method to linearly increment the opacity change. The aim was to create a step value to increase the opacity by a fixed increase each time then return the new

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

opacity, however, this could only be used for images and text because button which use rectangles in Pygame do not have the ability to change to opacity.

The next difficult element was creating the mechanisms for the volume bar. I had to create 5 different elements for this. I first had to have the volume text. In addition, I needed an outline, a bar for the volume bar, a button for the button slider, and finally a number for the volume. The first tricky part was ensuring that the cursor was in the centre of the button, and that the slider would not move off the screen.

```

def volume_settings(self):
    # Get the mouse and sets the volume to where the mouse is.
    mouse_pos = pygame.mouse.get_pos()
    if mouse_pos[0] - self.volume_button.width/3 > self.volume_line.x:
        if mouse_pos[0] + (self.volume_button.width/3)*2 < self.volume_line.x + self.volume_line.width:
            self.volume_button.x = mouse_pos[0] - self.volume_button.width/3
            self.volume = int(self.volume_ratio * (mouse_pos[0] - self.min_vol_x))
    
```

As shown above, the code checks that the button is within the allowed area (within the volume bar). The next difficult part was determining the volume. I had to do that by finding the volume ratio. I need the minimum and maximum x-axis coordinates.

```

# Volume
self.min_vol_x = self.volume_line.x + self.volume_button.width/3 + 1
self.max_vol_x = self.volume_line.x + self.volume_line.width - (self.volume_button.width*2)/3 - 1
self.volume_ratio = 100 / (self.max_vol_x - self.min_vol_x)
self.volume = int(self.volume_ratio * ((self.volume_button.x + self.volume_button.width/4) - self.min_vol_x))
self.sound_change = 0
self.music_change = 0
    
```

I divided it by 100 as it was a ratio. Then using the volume ratio, I was then able to calculate the total distance between the mouse and the minimum x-axis value and multiplied that by the ratio to get a number between 0 and 100 rounded to an integer. Other than that, the UI creation was not too difficult, other than creating a unique and aesthetically pleasing UI for my stakeholders.

Finally, the hardest part of the UI creation was combining the main menu user interface with the game itself without having any lag or limitations. What I did was develop a dictionary contained all the relevant information about the player which would be passed into the client.

```

def start_game(self):
    # Creates the dictionary to start the game.
    # sets game loop to false
    self.starting_dict = {
        "settings": {
            "control": {
                "movement": self.keys,
                "offense": self.attack_keys
            },
            "audio": {
                "volume": self.volume,
                "sound": True if self.sound_change == 1 else False,
                "music": True if self.music_change == 1 else False
            }
        },
        "start": [
            {"username": self.name_text_box.return_text() or "Player",
             "server_ip": self.ip_text_box.return_text() or "192.168.0.223"}
        ]
    }
    self.loop = False
    
```

The code above would break from the main menu loop and try connecting to the server.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
def run(self):
    while self.loop:
        events = pygame.event.get()
        for event in events:
            # checks if game has been closed.
            if event.type == pygame.QUIT:
                self.close()
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_ESCAPE:
                    self.close()

        # Update the display and frame rate
        self.redraw_window(events)
        pygame.display.update()
        self.clock.tick(self.FPS)
    if not self.loop:
        return self.starting_dict
```

The code would then receive the relevant dictionary and pass it into the client.

This would be a continuous loop that when the player disconnects the main menu screen is active again. In the client, when they disconnect, we would close the network then we would repeat this while True loop at the beginning.

Another difficult element was loading the animation images.

The code above when executed, would find the image file, and load the images surfaces and image names into one list. This was difficult because I needed to use the os walk and path functions in order to find the files inside the main folder.

As shown above, I would use that function to load the images surfaces and file locations in two separate lists. When the player was in the game, I would have to update their image with animations.

I would use this function above to find the image and animation surface based on the players status (e.g.: left, right, up, or down. Also, whether the player was idle or attacking. I created a frame index that was used to find the image and image_name that was to be used. The reason why I have an image name was in order to send over the player dictionary with an image. This is because sockets don't allow class objects or pygame surfaces to be sent over as it's an object that cannot be serialised or sent over. This is a limitation for my project which will be why I used another method (finding the image name) to send to the server. This is why I used a custom function in my player that return this image name.

That's why when I send over the player_dict, I get the image name, which is a string to send over to the server. This successfully allows me to draw each player on the screen.

This is where I faced one of the hardest elements with my code. I had to find a method where I could check if any players in the players list were no longer playing. I then had to either update those

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

existing players or delete any players who were found not in the players' dictionary. The players that were not in player list but were in the players dictionary were created. This was very tricky because I had to develop a system that could update players, then remove them then create new players. Updating and removing players was not tricky, but checking which players needed to be created was complicated. This is because we needed create a list of ids of players in self.players. Then we needed to iterate through each player ID until we found that a player did not exist. When a player was found to not exist in self.players, we created a new player. Then we would be able to update this player. In this way, I have created an efficient method that only creates each player once rather than creating players multiple times, which can lead to a visual error and can cause lag.

GUI

Main Menu Screen

The main menu screen as stated in my design has a moving background with three buttons including two additional buttons for my YouTube and GitHub.

Options Screen (Control Settings)

The options screen shows the ability to change user's settings. The background is still moving, and the two buttons are still there at the top.

Options Screen (Audio Settings)

The audio screen as stated in my design successfully shows the ability to customise the volume and sound/music settings.

Start Menu

The start menu shows that a username and a Server IP Address can be entered into a text box. When I press connect the user will be able to play in the server.

Game Screen

If we use the username "Player's Username" above it will display the same username in the game. The images above successfully show that my game is working, and I can have multiple users on screen.

During iterative development

Formal Test Table

Test Number	Link to User Requirement	Test Data	Expected Result	Actual Result
Test 2.1	1 - Visually Pleasing Interface	Inspect GUI design of buttons	Buttons visually distinct, labelled correctly, visually appealing	Yes, the game is visually pleasing. My stakeholders agree. Look at the GUI images for proof.
Test 2.2	2 - Interactive Button Features	Validate hover effects and button interaction	YouTube & GitHub buttons exhibit hover effects: reduced opacity, text display, colour and have	Yes, this works as expected. The buttons change colour and have

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

			custom cursor, colour transition on click	functionality. Look at Fig 1 for proof.
Test 2.3	3 - Settings Interface	Test functionality of the settings UI	Users can customise controls and manage audio settings (volume, sound/music toggles)	User can customise their settings however these settings are not implemented into the game yet. Look at GUI images for proof.
Test 2.4	4 - Start Menu Interface	Check options and connection initiation	Start menu allows going back, connecting, entering username/IP; connects with valid IP and loads player character	Yes, these connections work. The player can connect with the server IP. Look at GUI images for proof.
Test 2.5	5 - Dynamic Main Menu	Inspect dynamic main menu design	Background image shows game map with a blurred effect; maintains movement even when options are selected	Yes, this effect successfully works. The background image is blurred and moves to the right. Look at Fig 2 for proof.
Test 2.6	6 - Player Movement Animations	Verify player movement animations	Different animations for all directions (left, right, up, down), including attack and idle animations with cooldown for attacks	Yes, player animations are successful. Players can be seen with different walk animations by different players. Look at Fig 3 for proof.
Test 2.7	7 - Multi-Animation Cycle	Test synchronization of walking animations	Cycling system exhibits four animations for player walking, synchronized and visible to all players	Yes, four animations are successfully cycled when the player is walking, in addition other players can see the animations in real time too. Look at Fig 4 for proof.
Test 2.8	8 - Game Map Integration	Inspect map integration and functionality	Game map integrated, expanded area with maintained collisions, player coordinates displayed for debugging, visible mouse during gameplay	The game map has successfully been implemented, however there is no functionality yet, but there is some debugging, collisions with the rocks. Look at GUI images for proof of map.
Test 2.9	9 - User Exit Functionality	Check functionality for easy game exit	Pressing 'Escape' key offers options to	Yes, users can disconnect,

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

			return to main menu, join another server, or quit game	reconnect and or connect to another server. They can log out and go to the main menu screen and they can close the application down as expected. Look at Fig 5 for proof.
--	--	--	--	---

Stakeholder

Question Number	Question	Stakeholder 1 (Roland) response	Stakeholder 2 (James) response
1	What do you think of the visual effects?	The visual effects were good. I like the button effects.	Yep. They look great.
2	What do you think of the customisation of the user interface?	It seems incredibly good to me. I like the pretty design.	I like it. The moving background looks good.
3	Do you like the walk animations of the player?	Yes. They look amazing. The animations are great.	Yes, they look particularly good.
4	Do you like the ability to customise settings?	Yes. That is especially useful to have in the game.	Yes, I like customisable settings.
5	Do you like the start menu interface?	Yes, the moving background with the buttons make the menu look great.	I like the interface.
6	Do you like that the player goes back to the menu screen when the player disconnects?	Yes. That is a good feature to have in games.	Yes. It is good to be able to reconnect.
7	Do you like the new game map in the game?	Yes, the new map looks particularly good.	The map looks pretty and aesthetic.

Evaluation

Table of complete Success Criteria

Success Criteria	Successfully implemented
Visually Pleasing Interface: Develop a user interface with three unique buttons (Start, Settings, Quit) and additional buttons for YouTube and GitHub, ensuring a visually appealing design.	Yes.
Interactive Button Features: Implement hover effects on YouTube and GitHub buttons, including reduced opacity and text display on hover, along	Yes.

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

with a custom cursor and colour transition on click.	
Settings Interface: Create a simple settings user interface allowing users to customise controls and manage audio settings (volume, sound/music toggles).	Yes.
Start Menu Interface: Develop a start menu with options to go back, connect, and input username/Server IP Address, initiating connection with valid IP and loading the player's character.	Yes.
Dynamic Main Menu: Design a moving background image of the game map for the main menu, employing a blurred effect to highlight buttons and maintaining movement even when options are selected.	Yes.
Player Movement Animations: Implement different animations for player movement in all directions (left, right, up, down), including attack and idle animations with a cooldown period for attack movements.	Yes.
Multi-Animation Cycle: Set up a cycling system of four animations for player walking, ensuring synchronization and visibility to all players.	Yes.
Game Map Integration: Introduce the game map into the interface, expanding the area, maintaining previous collisions, displaying player coordinates for debugging, and keeping the mouse visible during gameplay.	Yes.
User Exit Functionality: Enable easy exit from the game by pressing the 'Escape' key, offering options to return to the main menu, join another server, or quit the game entirely.	Yes.

Evaluation of Sprint 2

Based on my success criteria, I have successfully completed implemented all the success criteria of this sprint to my game. From my stakeholder interviews above and my proof from testing, it is clear to suggest that I have been able to develop a successful sprint 2. My stakeholders make it clear to me that I have been able to meet their needs and that what I have implemented was what they wanted and what they wanted to see.

User Interface

Firstly, I have implemented a unique and aesthetically pleasing user interface in my game menu. This has been proven from my stakeholder comments and testing above. I used my own functions module that successfully used Pygame to create unique buttons. I have been able to create a semi functional user interface, however a user's key binds and audio settings have not yet been implemented into the game.

Player Animations

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

This sprint I have been able to implement some player animations into my game. These animations are for the player walking. My stakeholders agree that the animations are good and that that is what they have wanted. Overall, I have managed to implement everything from my success criteria and my testing proves that my game works, as well as my stakeholders agreeing that my programming for this sprint meets their requirements.

Next sprint

Next sprint I want to start developing more of the map. I also want to tweak the way that the clients can rejoin the game without requiring a while loop which could break and be ineffective. Furthermore, I want to add a way to log out that bring up a screen to leave the game when the 'Escape' key is pressed. In addition to changing the main menu rejoin mechanisms slightly, I would also like to change to UI of the username to be more see through. Furthermore, I want to add a user interface into the game itself where players can switch weapons. I also want to add some trees, grass, invisible barriers, and some unique objects to the game and removing the rocks barriers as they were only temporary. I will not add any weapons, or magic next sprint however, I will add the UI for that in the game.

Copyright

This sprint I have again used several elements from Clear Code's YouTube channel. I have firstly used his map. I have also used code from his player animations which are the same as I am using the same player animations as he did. I also have used elements from the player like rendering the player and the animations. I have also copied his level.py to create my camera object. The files like tile, UI and support are all copied or inspired by clear code. As my game is linked to his game but in the end, it is different. I am using his classes and files, but I am using them to create my own game.

Appendix

Pseudo Code:

Client:

```
public procedure update_players(player_dict)
    // Update existing player instances and remove players that are not in player_dict
    try
        players_to_remove = []
        for player in players
            if player.id in player_dict then
                player_data = player_dict[player.id]
                player.rect.x = player_data['x']
                player.rect.y = player_data['y']
                try player.image = new pygame.image.load(player_data['image']).convert_alpha()
                except player.image = new pygame.image.load(f"../{player_data['image']}").convert_alpha()
            else
                endtry
                logger.debug(f"Removing player instance with id: {player.id}")
                players_to_remove.append(player)
        endif
    next player
    for player in players_to_remove
        players.remove(player)
        level.visible_sprites.remove(player)
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

next player
// Create new player instances for players not already in players
for player_data in player_dict.values()
  player_ids = [player.id for player in players]
  if player_data['id'] NOT in player_ids then
    logger.debug(f'Creating new player instance with id: {player_data['id']}')
    new_player = Player(
      [player_data['x'], player_data['y']],
      player_data['image'],
      level.visible_sprites,
      level.obstacle_sprites,
      player_data['username'],
      player_data['id']
    )
    players.append(new_player)
  endif
next player_data
except
  logger.error("Player disconnected.")
  close()
endtry
endprocedure

public procedure redraw_window(all_players_dict)
  try
    update_players(all_players_dict)
    level.run(player)
    debug([player.rect.x, player.rect.y])
    ui.draw_menu()
    if ui.draw_ui then player.paused = True
    else player.paused = False
    endif
    pygame.display.update()
    clock.tick(FPS)
  except
    logger.error("Player has left the game.")
    close()
  endtry
endprocedure

public procedure main()
  while run
    for event in pygame.event.get()
      if event.type == pygame.QUIT then
        close()
      endif
      if event.type == pygame.KEYDOWN then
        if event.key == pygame.K_ESCAPE then
          ui.draw_ui = NOT ui.draw_ui
        endif
      endif
    next event
  try
    // Get player data.
  
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
player_dict = new {'x': player.rect.x, 'y': player.rect.y, 'image': player.get_image_name(),  
'username':username, 'id': id}  
player_encrypted_dict = new serialize(encryption.encrypt(player_dict))  
network.send(player_encrypted_dict)  
all_players_dict = new encryption.decrypt(unserialize(network.receive(DATA_SIZE)))  
logger.debug(f"Players Dictionary: {all_players_dict}")  
logger.debug(f"Sending player data: {player_dict}")  
logger.debug(f"Received players dictionary: {all_players_dict}")  
redraw_window(all_players_dict)  
except  
    logger.error("Failed to send over player to server.")  
    close()  
endtry  
endwhile  
endprocedure  
  
class MainMenu  
private custom_mouse.mode  
private youtube_button.draw((27,31,35),None,"https://www.youtube.com/watch?v= new dQw4w9WgXcQ")  
private starting_dict  
private volume_button.x  
private music_box.color  
private sound_box.color  
private music_change  
private sound_change  
private volume  
private volume_ratio  
private max_vol_x  
private min_vol_x  
private keys  
private attack_keys  
private settings_screen  
private main_menu_pages  
private name_box_text  
private name_text_box  
private server_ip_text  
private join_btn  
private join_boarder  
private ip_text_box  
private join_message  
private bullet_assault  
private start_back  
private start_board  
private audio_box  
private music_box  
private sound_box  
private music_text  
private sound_text  
private audio_text  
private volume_indicator  
private volume_button  
private volume_line  
private volume_bar  
private volume_text  
private box_audio
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
private attack_btn
private attack_text
private movement
private key_binds
private box_control
private underline2
private underline
private audio_settings
private control_settings
private back
private options_board
private current_x
private quit_button
private options_button
private start_button
private custom_mouse
private youtube_button
private github_button
private github_name
private youtube_name
private sunset_image
private icon
private base_button_height
private base_button_width
private thickness
private curve
private big_text_size
private base_text_size
private text_height
private button_padding
private image_padding
private image_height
private image_width
private BASE_BUTTON_HEIGHT
private BASE_BUTTON_WIDTH
private THICKNESS
private CURVE
private BUTTON_PADDING
private IMAGE_PADDING
private IMAGE_HEIGHT
private IMAGE_WIDTH
private TEXT_HEIGHT
private BASE_TEXT_SIZE
private BIG_TEXT_SIZE
private screen
private height_ratio
private width_ratio
private DEFAULT_HEIGHT
private DEFAULT_WIDTH
private screen_height
private screen_width
private loop
private FPS
private clock
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

public procedure new()
  // Pygame/ Game setup
  info = new pygame.display.Info()
  pygame.display.set_caption('Bullet Assault')
  clock = new pygame.time.Clock()
  FPS = 60
  loop = True
  screen_width = info.current_w
  screen_height = info.current_h
  DEFAULT_WIDTH = 1920
  DEFAULT_HEIGHT = 1080
  width_ratio = screen_width / DEFAULT_WIDTH
  height_ratio = screen_height / DEFAULT_HEIGHT
  screen = new pygame.display.set_mode((screen_width, screen_height))
  // Constants setup
  BIG_TEXT_SIZE = 50
  BASE_TEXT_SIZE = 15
  TEXT_HEIGHT = 20
  IMAGE_WIDTH = 64
  IMAGE_HEIGHT = 64
  IMAGE_PADDING = 20
  BUTTON_PADDING = 85
  CURVE = 10
  THICKNESS = 2
  BASE_BUTTON_WIDTH = 250
  BASE_BUTTON_HEIGHT = 70
  // Variable setup
  image_width = new int(IMAGE_WIDTH*width_ratio)
  image_height = new int(IMAGE_HEIGHT*height_ratio)
  image_padding = new int(IMAGE_PADDING*width_ratio)
  button_padding = new int(BUTTON_PADDING*height_ratio)
  text_height = new int(TEXT_HEIGHT*height_ratio)
  base_text_size = new int(BASE_TEXT_SIZE*height_ratio)
  big_text_size = new int(BIG_TEXT_SIZE*height_ratio)
  curve = new int(CURVE*height_ratio)
  thickness = new int(THICKNESS*height_ratio)
  base_button_width = new int(BASE_BUTTON_WIDTH*width_ratio)
  base_button_height = new int(BASE_BUTTON_HEIGHT*height_ratio)
  // Images
  icon = new Images("Graphics/MainMenu/General/icon.png")
  sunset_image = new Images("Graphics/MainMenu/General/bg.png")
  icon.display_icon()
  // Setting up Objects for home
  youtube_name = new Text(text_height, "Fonts/Orbitron-Medium.ttf", (27,31,35), None, None, None,
  base_text_size)
  github_name = new Text(text_height, "Fonts/Orbitron-Medium.ttf", (27,31,35), None, None, None,
  base_text_size)
  github_button = new Button((27,31,35), (27,31,35), screen_width-(image_width/2)-
  image_padding,(image_width/2)+image_padding, "Fonts/Orbitron-Regular.ttf", (27,31,35), (27,31,35),
  image_width, image_height,'Image', None, big_text_size, curve, 'Graphics/MainMenu/Buttons/github.png')
  youtube_button = new Button((27,31,35), (27,31,35), screen_width-(image_width*2)-
  (image_padding/4),(image_width/2)+image_padding, "Fonts/Orbitron-Regular.ttf", (27,31,35), (27,31,35),
  image_width, image_height,'Image', None, big_text_size, curve, 'Graphics/MainMenu/Buttons/youtube.png')
  custom_mouse = new Mouse("Graphics/MainMenu/Mouse/mouse1.png",
  "Graphics/MainMenu/Mouse/mouse2.png", "Graphics/MainMenu/Mouse/mouse3.png")

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
start_button = new Button((39, 174, 96), (240,240,240), screen_width/2, (screen_height/2)-button_padding,  
"Fonts/Orbitron-Medium.ttf", (240,240,240), (39, 174, 96), base_button_width,  
base_button_height,'Start','Rectangle', None, int(big_text_size/1.3), curve)  
options_button = new Button((39, 96, 174), (240,240,240), screen_width/2, screen_height/2, "Fonts/Orbitron-  
Medium.ttf", (240,240,240), (39, 96, 174), base_button_width, base_button_height,'Options','Rectangle', None,  
int(big_text_size/1.3), curve)  
quit_button = new Button((174, 39, 96), (240,240,240), screen_width/2, (screen_height/2)+button_padding,  
"Fonts/Orbitron-Medium.ttf", (240,240,240), (174, 39, 96), base_button_width,  
base_button_height,'Quit','Rectangle', None, int(big_text_size/1.3), curve)  
current_x = 0  
// Options buttons  
options_board = new Button((27,31,35), (27,31,35), screen_width/2, screen_height/2, "Fonts/Orbitron-  
Regular.ttf", (27,31,35), (27,31,35), screen_width*0.7, screen_height*0.7,'Rectangle', None, big_text_size,  
int(curve*1.5))  
back = new Button((174, 39, 96), (27,31,35), screen_width/2, screen_height*0.78, "Fonts/Orbitron-  
Medium.ttf", (27,31,35), (174, 39, 96), base_button_width, base_button_height,'Back','Rectangle', None,  
int(big_text_size/1.3), curve)  
control_settings = new Text(text_height, "Fonts/Orbitron-Bold.ttf", (240,240,240), None, None, None,  
int(base_text_size*2))  
audio_settings = new Text(text_height, "Fonts/Orbitron-Bold.ttf", (240,240,240), None, None, None,  
int(base_text_size*2))  
underline = new Button((27,31,35), (240,240,240), (screen_width/2)*0.70,  
(screen_height/2)*0.46,"Fonts/Orbitron-Regular.ttf", (240,240,240), (240,240,240), 275*width_ratio,  
3.5*height_ratio,'Rectangle', None, big_text_size, curve)  
underline2 = new Button((27,31,35), (240,240,240), (screen_width/2)*1.3,  
(screen_height/2)*0.46,"Fonts/Orbitron-Regular.ttf", (240,240,240), (240,240,240), 245*width_ratio,  
3.5*height_ratio,'Rectangle', None, big_text_size, curve)  
// Control Settings  
box_control = new Button((27,31,35), (27,31,35), (screen_width/2)*0.70, (screen_height/2)*0.96,  
"Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), base_button_width*2,  
base_button_height*7,'Rectangle', None, big_text_size, curve)  
key_binds = new Button((27,31,35), (27,31,35), (screen_width/2)*0.70, (screen_height/2)*0.685,  
"Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), base_button_width,  
base_button_height,'WASD','Rectangle', None, int(big_text_size/1.5), curve)  
movement = new Text(text_height, "Fonts/Orbitron-Medium.ttf", (240,240,240), None, None, None,  
int(base_text_size*2))  
attack_text = new Text(text_height, "Fonts/Orbitron-Medium.ttf", (240,240,240), None, None, None,  
int(base_text_size*2))  
attack_btn = new Button((27,31,35), (27,31,35), (screen_width/2)*0.70, (screen_height/2)*0.935,  
"Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), base_button_width,  
base_button_height,'Space','Rectangle', None, int(big_text_size/1.5), curve)  
// Audio Settings  
box_audio = new Button((27,31,35), (27,31,35), (screen_width/2)*1.3, (screen_height/2)*0.96, "Fonts/Orbitron-  
Regular.ttf", (240,240,240), (136,173,227), base_button_width*2, base_button_height*7,'Rectangle', None,  
big_text_size, curve)  
volume_text = new Text(text_height, "Fonts/Orbitron-Medium.ttf", (240,240,240), None, None, None,  
int(base_text_size*2))  
volume_bar = new Button((27,31,35), (27,31,35), (screen_width/2)*1.3, (screen_height/2)*0.685,  
"Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), base_button_width*1.5,  
base_button_height*1,'Rectangle', None, big_text_size, curve)  
volume_line = new Button((27,31,35), (27,31,35), (screen_width/2)*1.25, (screen_height/2)*0.685,  
"Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), base_button_width,  
base_button_height*0.36,'Rectangle', None, big_text_size, curve)
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
volume_button = new Button((27,31,35), (51,96,158), (screen_width/2)*1.3, (screen_height/2)*0.685,  
"Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), base_button_height*0.3,  
base_button_height*0.3,'Rectangle', None, big_text_size, int(curve/2))  
volume_indicator = new Text(text_height, "Fonts/Orbitron-Regular.ttf", (240,240,240), None, None, None,  
int(base_text_size*2))  
audio_text = new Text(text_height, "Fonts/Orbitron-Medium.ttf", (240,240,240), None, None, None,  
int(base_text_size*2))  
sound_text = new Text(text_height, "Fonts/Orbitron-Medium.ttf", (240,240,240), None, None, None,  
int(base_text_size*2))  
music_text = new Text(text_height, "Fonts/Orbitron-Medium.ttf", (240,240,240), None, None, None,  
int(base_text_size*2))  
sound_box = new Button((27,31,35), (39,174,96), (screen_width/2)*1.45, (screen_height/2)*0.97,  
"Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), base_button_height*0.4,  
base_button_height*0.4,'Rectangle', None, big_text_size, curve)  
music_box = new Button((27,31,35), (39,174,96), (screen_width/2)*1.45, (screen_height/2)*1.08,  
"Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), base_button_height*0.4,  
base_button_height*0.4,'Rectangle', None, big_text_size, curve)  
audio_box = new Button((27,31,35), (27,31,35), (screen_width/2)*1.3, (screen_height/2)*1.03, "Fonts/Orbitron-  
Regular.ttf", (240,240,240), (136,173,227), base_button_width*1.5, base_button_height*2.5,'Rectangle', None,  
big_text_size, curve)  
// Start  
start_board = new Button((27,31,35), (27,31,35), screen_width/2, screen_height/2, "Fonts/Orbitron-  
Regular.ttf", (27,31,35), (27,31,35), screen_width*0.7, screen_height*0.7,'Rectangle', None, big_text_size,  
int(curve*1.5))  
start_back = new Button((174, 39, 96), (27,31,35), screen_width/2, screen_height*0.78, "Fonts/Orbitron-  
Medium.ttf", (27,31,35), (174, 39, 96), base_button_width, base_button_height,'Back','Rectangle', None,  
int(big_text_size/1.3), curve)  
bullet_assault = new Text(text_height, "Fonts/Orbitron-Bold.ttf", (240,240,240), None, None, None,  
int(base_text_size*3.5))  
join_message = new Text(text_height, "Fonts/Orbitron-Regular.ttf", (240,240,240), None, None, None,  
int(base_text_size*1.7))  
ip_text_box = new TextBox(base_button_width*2.5, base_button_height*1.5, (screen_width/2),  
(screen_height/2)*0.99, (240,240,240), (200,200,200),"Fonts/Orbitron-  
Regular.ttf",curve,thickness,base_button_width*2.5)  
join_boarder = new Button((27,31,35), (27,31,35), (screen_width/2), (screen_height/2), "Fonts/Orbitron-  
Regular.ttf", (240,240,240), (136,173,227), base_button_width*3, base_button_height*6,'Rectangle', None,  
big_text_size, curve)  
join_btn = new Button((39, 174, 96), (27,31,35), (screen_width/2), (screen_height/2)*1.445-button_padding,  
"Fonts/Orbitron-Medium.ttf", (27,31,35), (39, 174, 96), base_button_width,  
base_button_height,'Connect','Rectangle', None, int(big_text_size/1.3), curve)  
server_ip_text = new Text(text_height, "Fonts/Orbitron-Regular.ttf", (240,240,240), None, None, None,  
int(base_text_size*1.7))  
name_text_box = new TextBox(base_button_width*2.5, base_button_height*1.5, (screen_width/2),  
(screen_height/2)*0.70, (240,240,240), (200,200,200),"Fonts/Orbitron-  
Regular.ttf",curve,thickness,base_button_width*2.5)  
name_box_text = new Text(text_height, "Fonts/Orbitron-Regular.ttf", (240,240,240), None, None, None,  
int(base_text_size*1.7))  
// Settings  
main_menu_pages = "home"  
settings_screen = "control"  
attack_keys = "Space"  
keys = "WASD"  
// Volume  
min_vol_x = volume_line.x + volume_button.width/3 + 1  
max_vol_x = new volume_line.x + volume_line.width - (volume_button.width*2)/3 - 1
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
volume_ratio = new 100 / (max_vol_x - min_vol_x)
volume = new int(volume_ratio * ((volume_button.x + volume_button.width/4) - min_vol_x))
sound_change = 0
music_change = 0
endprocedure

public procedure close()
loop = False
pygame.quit()
sys.exit()
endprocedure

public procedure options()
main_menu_pages = "settings"
endprocedure

public procedure home()
main_menu_pages = "home"
endprocedure

public procedure start_option()
main_menu_pages = "start"
endprocedure

public procedure draw_moving_background()
// Draw the moving image on the screen
sunset_image.draw(current_x, 0)
sunset_image.draw((current_x - sunset_image.rect.width), 0)
current_x += 1
// If the image has moved beyond its width, reset it to 0
if current_x >= sunset_image.rect.width then
    current_x = 0
endif
endprocedure

public procedure change_keys()
if keys == 'WASD' then
    keys = 'Arrow Keys'
else
    keys = 'WASD'
endif
endprocedure

public procedure control_settings_change()
settings_screen = "control"
endprocedure

public procedure audio_settings_change()
settings_screen = "audio"
endprocedure

public procedure change_attack()
// Changes the key binds.
if attack_keys == "Space" then
    attack_keys = "L-Ctrl"
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
endif
elseif attack_keys == "L-Ctrl" then
    attack_keys = "R-Ctrl"
else
    attack_keys = "Space"
endif
endprocedure

public procedure sound_box_color()
// Changes the sound box color
sound_change += 1
if sound_change == 1 then
    sound_box.color = new (39,174,96)
else
    sound_change = 0
    sound_box.color = new (27,31,35)
endif
endprocedure

public procedure music_box_color()
// Changes the music box check color.
music_change += 1
if music_change == 1 then
    music_box.color = new (39,174,96)
else
    music_change = 0
    music_box.color = new (27,31,35)
endif
endprocedure

public procedure volume_settings()
// Get the mouse and sets the volume to where the mouse is.
mouse_pos = new pygame.mouse.get_pos()
if mouse_pos[0] - volume_button.width/3 > volume_line.x then
    if mouse_pos[0] + (volume_button.width/3)*2 < volume_line.x + volume_line.width then
        volume_button.x = mouse_pos[0] - volume_button.width/3
        volume = new int(volume_ratio * (mouse_pos[0] - min_vol_x))
    endif
endif
endprocedure

function start_game()
    starting_dict = {
        "settings": {
            "control": {
                "movement": keys,
                "offense": attack_keys
            },
            "audio": {
                "volume": volume,
                "sound" then True if sound_change == 1 else False,
                "music" then True if music_change == 1 else False
            }
        }
    }
}
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

},
"start": {
  "username": name_text_box.return_text() OR "Player",
  "server_ip": ip_text_box.return_text() OR "Server IP"
}
}
loop = False
// Check the values
//
for category, subcategories in starting_dict.items()
  logger.info(f"{category}:")
  for subcategory, values in subcategories.items()
    logger.info(f" {subcategory}:")
    if isinstance(values, dict) then
      for key, value in values.items()
        logger.info(f"  {key}: {value}")
    else
      next key, value
      logger.info(f"  {values}")
    endif
  next subcategory, values
next category, subcategories
//
endfunction

public procedure draw_objects(events)
  // Draw the buttons and check for hover
  if main_menu_pages == "home" then
    github_button.draw((27,31,35),None,"https://github.com/TheRealDL1/Simple-Client-Server")
    youtube_button.draw((27,31,35),None,"https://www.youtube.com/watch?v= new dQw4w9WgXcQ")
    start_button.draw((27,31,35),start_option)
    options_button.draw((27,31,35),options)
    quit_button.draw((27,31,35),close,None,True)
    start_button_hover = new start_button.is_hovered()
    options_button_hover = new options_button.is_hovered()
    quit_button_hover = new quit_button.is_hovered()
    github_button_hover = new github_button.is_hovered()
    youtube_button_hover = new youtube_button.is_hovered()
    start_button_click = new start_button.is_clicking()
    options_button_click = new options_button.is_clicking()
    quit_button_click = new quit_button.is_clicking()
    github_button_click = new github_button.is_clicking()
    youtube_button_click = new youtube_button.is_clicking()
  endif
endprocedure

// Check if mouse is hovering over buttons
if start_button_hover OR options_button_hover OR quit_button_hover OR github_button_hover OR
youtube_button_hover then
  // Draw hover text.
  if github_button_hover then github_name.draw("draw","Github", screen_width-(image_width/2)-
image_padding, (image_width/2)+image_padding*3.1)
  else github_name.draw("undraw","Github", screen_width-(image_width/2)-image_padding,
(image_width/2)+image_padding*3.1)

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

  endif
  if youtube_button_hover then youtube_name.draw("draw","Youtube", screen_width-(image_width*2)-
(image_padding/4), (image_width/2)+image_padding*3.1)
    else youtube_name.draw("undraw","Youtube", screen_width-(image_width*2)-(image_padding/4),
(image_width/2)+image_padding*3.1)
  endif
  if NOT start_button_click AND NOT options_button_click AND NOT quit_button_click AND NOT
github_button_click AND NOT youtube_button_click then
    custom_mouse.mode = 1
  else
    custom_mouse.mode = 2
  else
  endif
  custom_mouse.mode = 0
  github_name.draw("undraw","Github", screen_width-(image_width/2)-image_padding,
(image_width/2)+image_padding*3.1)
  youtube_name.draw("undraw","Youtube", screen_width-(image_width*2)-(image_padding/4),
(image_width/2)+image_padding*3.1)
  endif
if main_menu_pages == "settings" then
  options_board.draw((27,31,35),None, None, False)
  back.draw((240,240,240),home)
  control_settings.draw("draw","Control Settings", (screen_width/2)*0.70,
(screen_height/2)*0.42,control_settings_change)
  audio_settings.draw("draw","Audio Settings", (screen_width/2)*1.3,
(screen_height/2)*0.42,audio_settings_change)
  github_button.draw((27,31,35),None,"https://github.com/TheRealDL1/Simple-Client-Server")
  youtube_button.draw((27,31,35),None,"https://www.youtube.com/@dominicpike")
  back_hover = new back.is_hovered()
  github_button_hover = new github_button.is_hovered()
  youtube_button_hover = new youtube_button.is_hovered()
  key_binds_hover = new control_settings.is_hovered()
  audio_hover = new audio_settings.is_hovered()
  key_hover = new key_binds.is_hovered()
  attack_hover = new attack_btn.is_hovered()
  volume_btn_hover = new volume_button.is_hovered()
  sound_box_hover = new sound_box.is_hovered()
  music_box_hover = new music_box.is_hovered()
  back_click = new back.is_clicking()
  github_button_click = new github_button.is_clicking()
  youtube_button_click = new youtube_button.is_clicking()
  key_binds_click = new control_settings.is_clicking()
  audio_click = new audio_settings.is_clicking()
  key_click = new key_binds.is_clicking()
  attack_click = new attack_btn.is_clicking()
  volume_btn_click = new volume_button.is_clicking()
  sound_box_click = new sound_box.is_clicking()
  music_box_click = new music_box.is_clicking()
  if back_hover OR github_button_hover OR youtube_button_hover OR key_binds_hover OR audio_hover OR
key_hover OR attack_hover OR volume_btn_hover OR sound_box_hover OR music_box_hover then
    if github_button_hover then github_name.draw("draw","Github", screen_width-(image_width/2)-
image_padding, (image_width/2)+image_padding*3.1)
      else github_name.draw("undraw","Github", screen_width-(image_width/2)-image_padding,
(image_width/2)+image_padding*3.1)
    endif
  endif

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

if youtube_button_hover then youtube_name.draw("draw","Youtube", screen_width-(image_width*2)-(image_padding/4), (image_width/2)+image_padding*3.1)
else youtube_name.draw("undraw","Youtube", screen_width-(image_width*2)-(image_padding/4), (image_width/2)+image_padding*3.1)
endif
if NOT back_click AND NOT github_button_click AND NOT youtube_button_click AND NOT key_binds_click AND NOT audio_click AND NOT key_click AND NOT attack_click AND NOT volume_btn_click AND NOT sound_box_click AND NOT music_box_click then
  custom_mouse.mode = 1
else
  if volume_btn_click then
    volume_settings()
  endif
  custom_mouse.mode = 2
endif
custom_mouse.mode = 0
github_name.draw("undraw","Github", screen_width-(image_width/2)-image_padding, (image_width/2)+image_padding*3.1)
youtube_name.draw("undraw","Youtube", screen_width-(image_width*2)-(image_padding/4), (image_width/2)+image_padding*3.1)
endif
if settings_screen == "control" then
  if audio_hover then underline2.draw(None,None,None,True,None,None,None,"draw")
  else underline2.draw(None,None,None,True,None,None,None,"undraw")
  endif
  underline.draw(None,None,None,True,None,None,None,"draw")
  box_control.draw((240,240,240))
  movement.draw("draw","Movement", (screen_width/2)*0.70, (screen_height/2)*0.57)
  key_binds.draw((240,240,240),change_keys, None, True, keys)
  attack_text.draw("draw","Offense", (screen_width/2)*0.70, (screen_height/2)*0.82)
  attack_btn.draw((240,240,240),change_attack, None, True, attack_keys)
endif
elseif settings_screen == "audio" then
  if key_binds_hover then underline.draw(None,None,None,True,None,None,None,"draw")
  else underline.draw(None,None,None,True,None,None,None,"undraw")
  endif
  underline2.draw(None,None,None,True,None,None,None,"draw")
  box_audio.draw((240,240,240))
  volume_text.draw("draw","Volume", (screen_width/2)*1.3, (screen_height/2)*0.57)
  volume_bar.draw((240,240,240))
  volume_line.draw((240,240,240))
  volume_button.draw((240,240,240))
  volume_indicator.draw("draw",str(volume), (screen_width/2)*1.435, (screen_height/2)*0.685)
  audio_text.draw("draw","Audio", (screen_width/2)*1.3, (screen_height/2)*0.82)
  audio_box.draw((240,240,240))
  sound_text.draw("draw","Sound", (screen_width/2)*1.185, (screen_height/2)*0.97)
  music_text.draw("draw","Music", (screen_width/2)*1.18, (screen_height/2)*1.08)
  sound_box.draw((240,240,240),sound_box_color)
  music_box.draw((240,240,240),music_box_color)
endif
endif
if main_menu_pages == "start" then
  options_board.draw((27,31,35),None, None, False)
  start_back.draw((240,240,240),home)

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
github_button.draw((27,31,35),None,"https://github.com/TheRealDL1/Simple-Client-Server")
youtube_button.draw((27,31,35),None,"https://www.youtube.com/@dominicpike")
join_boarder.draw((240,240,240))
bullet_assault.draw("draw","Bullet Assault", (screen_width/2), (screen_height/2)*0.43)
server_ip_text.draw("draw","Server IP Address", (screen_width/2), (screen_height/2)*0.955)
message = "Enter the Server's IP Address that you want to join!"
join_message.draw("draw",message, (screen_width/2), (screen_height/2)*0.52)
ip_text_box.draw()
ip_text_box.updateText(events)
ip_text_box.update()
join_btn.draw((240,240,240),start_game)
// Name Text Box
name_box_text.draw("draw","Username", (screen_width/2), (screen_height/2)*0.665)
name_text_box.draw()
name_text_box.updateText(events)
name_text_box.update()
github_button_hover = new github_button.is_hovered()
youtube_button_hover = new youtube_button.is_hovered()
start_back_hover = new start_back.is_hovered()
join_hover = new join_btn.is_hovered()
text_box_hover = new ip_text_box.is_hovered()
name_hover = new name_text_box.is_hovered()
github_button_click = new github_button.is_clicking()
youtube_button_click = new youtube_button.is_clicking()
start_back_click = new start_back.is_clicking()
join_click = new join_btn.is_clicking()
text_box_click = new ip_text_box.is_clicking()
name_click = new name_text_box.is_clicking()
if start_back_hover OR github_button_hover OR youtube_button_hover OR join_hover OR text_box_hover
OR name_hover then
    // Draw hover text.
    if github_button_hover then github_name.draw("draw","Github", screen_width-(image_width/2)-
image_padding, (image_width/2)+image_padding*3.1)
        else github_name.draw("undraw","Github", screen_width-(image_width/2)-image_padding,
(image_width/2)+image_padding*3.1)
    endif
    if youtube_button_hover then youtube_name.draw("draw","Youtube", screen_width-(image_width*2)-
(image_padding/4), (image_width/2)+image_padding*3.1)
        else youtube_name.draw("undraw","Youtube", screen_width-(image_width*2)-(image_padding/4),
(image_width/2)+image_padding*3.1)
    endif
    if NOT start_back_click AND NOT github_button_click AND NOT youtube_button_click AND NOT
join_click AND NOT text_box_click AND NOT name_click then
        custom_mouse.mode = 1
    else
        custom_mouse.mode = 2
    else
        endif
        custom_mouse.mode = 0
        github_name.draw("undraw","Github", screen_width-(image_width/2)-image_padding,
(image_width/2)+image_padding*3.1)
        youtube_name.draw("undraw","Youtube", screen_width-(image_width*2)-(image_padding/4),
(image_width/2)+image_padding*3.1)
    endif
endif
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
// Draw the mouse
custom_mouse.draw()
public procedure redraw_window(events)
    draw_moving_background()
    draw_objects(events)
endprocedure

function run()
    while loop
        events = new pygame.event.get()
        for event in events
            if event.type == pygame.QUIT then
                close()
                endif
            if event.type == pygame.KEYDOWN then
                if event.key == pygame.K_ESCAPE then
                    close()
                    endif
                endif
            next event
        // Update the display and frame rate
        redraw_window(events)
        pygame.display.update()
        clock.tick(FPS)
    endwhile
    if NOT loop then
        return starting_dict
    endif
endfunction

if __name__ == "__main__":
    while True
        game = new MainMenu()
        user_dict = new game.run()
        client = new Client(user_dict)
        client.main()
    endwhile
endif
```

Functions:

```
import pygame
import webbrowser
from Scripts.logger import *
```

logger.info("Pygame Functions 0.0.1 (Python 3.11.0)\nI am NOT affiliated with pygame.
https://github.com/TheRealDL1/pygame_functions")

```
class Images inherits object
    private rect
    private load_image
    private screen
    private image

    public procedure new(image)
        image = image
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
screen = new pygame.display.get_surface()
load_image = new load_image_from_file()
rect = new load_image.get_rect()
endprocedure

function load_image_from_file()
try
    return pygame.image.load(image).convert_alpha()
except
    return pygame.image.load(f"../{image}").convert_alpha()
endtry
endfunction

public procedure display_icon()
    pygame.display.set_icon(load_image)
endprocedure

public procedure draw(x= new 0, y= new 0)
    screen.blit(load_image, (x,y))
endprocedure

public procedure resize(width, height)
    load_image = new pygame.transform.scale(load_image, (width, height))
endprocedure

class Mouse inherits object
private mouse_mode3
private mouse_mode2
private mouse_mode1
private mode
private screen

public procedure new(mouse_image1, mouse_image2, mouse_image3)
    pygame.mouse.set_visible(False)
    screen = new pygame.display.get_surface()
    mode = 0
    mouse_mode1 = new Images(mouse_image1)
    mouse_mode2 = new Images(mouse_image2)
    mouse_mode3 = new Images(mouse_image3)
endprocedure

public procedure draw()
    // Get the current mouse position
    mouse_x, mouse_y = new pygame.mouse.get_pos()
    // Draw a green square at the mouse cursor position
    if mode == 0 then
        mouse_mode1.draw(mouse_x , mouse_y)
    endif
    elseif mode == 1 then
        mouse_mode2.draw(mouse_x , mouse_y)
    else
        mouse_mode3.draw(mouse_x , mouse_y)
    endif
endprocedure
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
class WordButton inherits object
    private release
    private click
    private hover
    private large_font
    private base_font
    private largefont
    private basefont
    private largeSize
    private textSize
    private text
    private y
    private x
    private color2
    private color

    public procedure new(x, y, text, color1, color2, basefont, largefont, textSize= new 30,
        //Sets the values for button//
        color = color1
        color2 = color2
        x = x
        y = y
        text = text
        textSize = textSize
        largeSize = new round(textSize * 1.25)
        basefont = basefont
        largefont = largefont
        try
            base_font = new pygame.font.Font(basefont, textSize)
            large_font = new pygame.font.Font(largefont, largeSize)
        except
            base_font = new pygame.font.Font(f"../{basefont}", textSize)
            large_font = new pygame.font.Font(f"../{largefont}", largeSize)
        endtry
        hover = False
        click = False
        release = False
    endprocedure

    public procedure displayText(win, newText= new None, action= new None, link= new None)
        if newText != None then
            text = newText
        endif
        mouse = new pygame.mouse.get_pos()
        click = new pygame.mouse.get_pressed()
        // Render text with both fonts
        text = new large_font.render(text, 1, color2)
        text2 = new base_font.render(text, 1, color)
        // Check if mouse is over the text
        if x + text.get_width()/2 > mouse[0] > x - text.get_width()/2 AND y + text.get_height()/2 > mouse[1] > y - text.get_height()/2 then
            hover = True
            // Render text with larger font and lighter color
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

text = new large_font.render(text, 1, color2)
// Center text vertically as well as horizontally
win.blit(text, (x - text.get_width() / 2, y - text.get_height() / 2))
// Check for mouse click and run action if specified
if click[0] == 1 then
  click = True
endif
if click == True AND NOT click[0] then
  release = True
endif
if release then
  if action != None then
    action()
  endif
  if link != None then
    webbrowser.open(link)
  endif
  release = False
  click = False
else
  endif
hover = False
click = False
release = False
// Render text with base font and color
text2 = new base_font.render(text, 1, color)
// Center text vertically as well as horizontally
win.blit(text2, (x - text2.get_width() / 2, y - text2.get_height() / 2))
endif
endprocedure
  
```

```

class Animation inherits object
  private monopoly_rect
  private screen_center_y
  private screen_center_x
  private scale_speed
  private scale_max
  private scale_min
  private scale_direction
  private image
  private scale

  public procedure new(image, scale, scale_direction, scale_min, scale_max, scale_speed, screen_y, screen_x)
    scale = scale //1.0
    image = image
    scale_direction = scale_direction // -1.35 // Start by zooming out
    scale_min = scale_min //0.9
    scale_max = scale_max //1.2
    scale_speed = scale_speed //0.01 // The rate at which the scale changes
    screen_center_x = screen_x
    screen_center_y = new screen_y //324//win.get_height() // 2
    monopoly_rect = new image.get_rect(center= new(screen_center_x, screen_center_y))
  endprocedure
  
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
public procedure animation(win)
    scale += scale_direction * scale_speed
    if scale <= scale_min then
        if scale_direction < 0 then
            scale_direction = scale_direction * -1
        else
            scale_direction = scale_direction * 1 // Start zooming in
        endif
    endif
    elseif scale >= scale_max then
        if scale_direction > 0 then
            scale_direction = scale_direction * -1
        else
            scale_direction = scale_direction * 1 // Start zooming out
        endif
    endif
    // Scale the image and get its rect
    scaled_monopoly = new pygame.transform.rotozoom(image, 0, scale)
    scaled_monopoly_rect = new scaled_monopoly.get_rect(center= new(screen_center_x, screen_center_y))
    // Blit the scaled image to the screen and update the display
    win.blit(scaled_monopoly, scaled_monopoly_rect)
endprocedure
```

```
class TextBox inherits object
    private show_cursor
    private click
    private hover
    private screen
    private thickness
    private curve
    private maxTextWidth
    private cursor
    private show_cursOR
    private last_update
    private color
    private color_passive
    private color_active
    private y
    private ogWidth
    private x
    private height
    private width
    private active
    private base_font
    private textfont
    private textSize
    private text

    public procedure new(width, height, x, y, color1, color2, textfont, curve, thickness, maxTextWidth, text_size=
new 50)
        text = ""
        textSize = text_size
        textfont = textfont
        try
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
base_font = new pygame.font.Font(textfont, textSize)
except
    base_font = new pygame.font.Font(f"../{textfont}", textSize)
endtry
active = False
width = width
height = height
x = x - width / 2
ogWidth = width
y = y
color_active = new pygame.Color(color1)
color_passive = new pygame.Color(color2)
color = color_passive
last_update = 0 // time of last update
show_cursOR = False // whether to show cursOR OR not
cursor = 0
maxTextWidth = maxTextWidth
curve = curve
thickness = thickness
screen = new pygame.display.get_surface()
hover = False
click = False
endprocedure

public procedure draw()
    area = [x,y,width,height]
    pygame.draw.rect(screen, color, area, thickness, curve)
endprocedure

function is_hovered()
    mouse = new pygame.mouse.get_pos()
    return x + width > mouse[0] > x AND y + height > mouse[1] > y
endfunction

function is_clicking()
    mouse = new pygame.mouse.get_pos()
    click = new pygame.mouse.get_pressed()
    return x + width > mouse[0] > x AND y + height > mouse[1] > y AND click[0]
endfunction

public procedure checkTextBox()
    mouse = new pygame.mouse.get_pos()
    click = new pygame.mouse.get_pressed()
    if x + width > mouse[0] > x AND y + height > mouse[1] > y then
        hover = True
        if click[0] then click = True
        endif
        if click AND NOT click[0] then click = False
        endif
        if active then active = False
        else active = True
    else
        endif
        hover = False
        active = False
    endif
endprocedure
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
click = False
endif
if active then
    color = color_active
else
    color = color_passive
endif
endprocedure

public procedure update(color= new (240,240,240),x_pos= new None, y_pos= new None)
    middleOfX = new screen.get_width() DIV 2
    if x_pos then x = x_pos
    endif
    if y_pos then y = y_pos
    endif
    if text then
        surface_area = new base_font.render(text, True, color)
        text_width = new surface_area.get_width() + 20
        if ogWidth > width then
            width = ogWidth
            x = middleOfX - width DIV 2
        endif
        if text_width > width then
            width = text_width
            x = middleOfX - width DIV 2
        endif
        elseif width > text_width then
            if ogWidth < width then
                width = text_width
                x = middleOfX - width DIV 2
            endif
        endif
        screen.blit(surface_area, (x + 5, y + (height DIV 2 - surface_area.get_height() DIV 2)))
    else
        width = ogWidth
        x = middleOfX - width DIV 2
        surface_area = new base_font.render(text, True, color)
        screen.blit(surface_area, (x + 5, y + (height DIV 2 - surface_area.get_height() DIV 2)))
    endif
    time_since_last_update = new pygame.time.get_ticks() - last_update
    if active AND time_since_last_update > 500 then
        show_cursor = NOT show_cursor
        last_update = new pygame.time.get_ticks()
    endif
    if show_cursor AND active == True then
        cursor_width = 2
        cursor_pos = new base_font.size(text[:cursor])[0] - 5
        text_to_show = text[:cursor] + '|' + text[cursor:]
        cursor_pos += new base_font.size(' ')[0] * 0.8
        surface_area = new base_font.render(text_to_show, True, color)
        cursor_area = new pygame.Surface((cursor_width, surface_area.get_height()))
        cursor_area.fill(color)
        screen.blit(cursor_area, (x + cursor_pos, y + (height DIV 2 - cursor_area.get_height() DIV 2)), (0, 0,
cursor_width, cursor_area.get_height()))
    // else
```

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

//surface_area = new base_font.render(text, True, color)
//screen.blit(surface_area, (x + 5, y + (height // 2 - surface_area.get_height() // 2)))
endif
endprocedure

public procedure updateText(events, color= new (0,0,0), function= new None)
    surface_area = new base_font.render(text, True, color)
    text_width = new surface_area.get_width() + 20
    for event in events
        if event.type == pygame.MOUSEBUTTONDOWN then
            checkTextBox()
        endif
        if event.type == pygame.KEYDOWN then
            if active == True then
                if event.key == pygame.K_BACKSPACE then
                    if cursor > 0 then
                        text = text[:cursor-1] + text[cursor:]
                        cursor -= 1
                    endif
                endif
                elseif event.key == pygame.K_DELETE then
                    text = text[:cursor] + text[cursor+1:]
                endif
                elseif event.key == pygame.K_RETURN then
                    if function != None then
                        function()
                    endif
                endif
                elseif event.key == pygame.K_LEFT then
                    if cursor > 0 then
                        cursor -= 1
                    endif
                endif
                elseif event.key == pygame.K_RIGHT then
                    if cursor < text.length
                        cursor += 1
                    else
                        endif
                    if text_width < maxTextWidth AND event.key is NOT pygame.K_TAB then
                        new_text = text[:cursor] + event.unicode + text[cursor:]
                        if new_text.length > len(text)
                            text = new_text
                            cursor += 1
                        endif
                    endif
                endif
            endif
        endif
    next event
endprocedure

function return_text()
    return text
endfunction

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
class Text inherits object
    private opacity_color
    private opacity_color2
    private opacity_color1
    private opacityb
    private opacitya
    private time2
    private time
    private num_frames
    private animation_speed
    private animation_time
    private stop_start_animation
    private release
    private click
    private hover
    private base_font
    private screen
    private color
    private textSize
    private height
    private baseFont
    private y
    private x
    private text

    public procedure new(height, baseFont, color= new (0,0,0), x= new None, y= new None, text= new None,
textSize= new 32)
        text = text
        x = x
        y = y
        baseFont = baseFont
        height = height
        textSize = textSize
        color = color
        screen = new pygame.display.get_surface()
        try
            base_font = new pygame.font.Font(baseFont, textSize)
        except
            base_font = new pygame.font.Font(f"../{baseFont}", textSize)
        endtry
        hover = False
        click = False
        release = False
        // Animation
        stop_start_animation = True
        animation_time = 150
        animation_speed = 13
        num_frames = 60
        time = 0
        time2 = 0
        // Opacity
        opacitya = 0
        opacityb = 255
        opacity_color1 = 0
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
opacity_color2 = 0
opacity_color = 0
endprocedure

public procedure draw(mode= new"draw",new_text= newNone, new_x= newNone, new_y= newNone,
function= new None, link= newNone)
    if new_text then text = new_text
    endif
    if new_x then x = new_x
    endif
    if new_y then y = new_y
    endif
    if mode == "normal" then
        surface_area = new base_font.render(text, 1, color)
        screen.blit(surface_area, (x - surface_area.get_width() / 2, y - surface_area.get_height() / 2))
    endif
    if mode == "draw" then
        stop_start_animation = False
        time = new pygame.time.get_ticks()
        step = new (time - time2) / (animation_time / animation_speed)
        opacity_color1 = new change_opacity(opacity_color2, opacityb, step)
        opacity_color = opacity_color1
    endif
    if mode == "undraw" then
        time2 = new pygame.time.get_ticks()
        step = new (time2 - time) / (animation_time / animation_speed)
        opacity_color2 = new change_opacity(opacity_color1, opacitya, step)
        opacity_color = opacity_color2
    endif
    if NOT stop_start_animation then
        surface_area = new base_font.render(text, 1, color)
        surface_area.set_alpha(opacity_color)
        screen.blit(surface_area, (x - surface_area.get_width() / 2, y - surface_area.get_height() / 2))
    endif
    if is_hovered() then hover = new True
    endif
    if is_clicking() then click = new True
    endif
    if click AND NOT is_clicking() then release = new True
    endif
    if release then
        if function then function()
        endif
        if link then webbrowser.open(link)
        endif
        release = False
        click = False
    endif
    if NOT is_hovered then
        hover = False
        click = False
        release = False
    endif
endprocedure
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
function change_opacity(original_opacity, new_opacity, step_multiplier= new 0)
    new_opacity_return = None
    difference = new_opacity - original_opacity
    endif
    opacity_step = difference / num_frames
    endif
    if original_opacity > new_opacity then
        if int(original_opacity + opacity_step*step_multiplier) < new_opacity then
            new_opacity_return = new_opacity
        else
            new_opacity_return = new int(original_opacity + opacity_step*step_multiplier)
        else
            new_opacity_return = new int(original_opacity + opacity_step*step_multiplier)
        endif
    endif
    return new_opacity_return
endfunction

function is_hovered()
    mouse_pos = new pygame.mouse.get_pos()
    text_rect = new base_font.render(text, 1, color).get_rect()
    text_rect.center = new (x, y)
    return text_rect.collidepoint(mouse_pos)
endfunction

function is_clicking()
    mouse_pos = new pygame.mouse.get_pos()
    mouse_buttons = new pygame.mouse.get_pressed()
    text_rect = new base_font.render(text, 1, color).get_rect()
    text_rect.center = new (x, y)
    return text_rect.collidepoint(mouse_pos) AND mouse_buttons[0] = new= new 1
endfunction

class Button inherits object
    private base_font
    private basefont
    private opacity2
    private opacity1
    private opacityb
    private opacitya
    private button_trans_color2
    private text_trans_color2
    private button_trans_color
    private text_trans_color
    private text_color2
    private text_color
    private color2
    private color
    private time2
    private time
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
private num_frames
private animation_speed
private animation_time
private stop_start_animation
private release
private click
private hover
private curve
private textSize
private radius
private buttonType
private text
private image
private height
private width
private y
private x
private screen

//A class for all buttons//
public procedure new(color, color2, x, y, basefont, text_color= new (0,0,0), text_color2= new (0,0,0), width= new
None, height= new None, text= new "",buttonType= new 'Rectangle', radius= new None, textSize= new 30,
curve= new 10, image= new "")
//Sets the values for button//
screen = new pygame.display.get_surface()
try
    x = x - width/2
    y = y - height/2
    width = width
    height = height
except
    x = x
    y = y
endif
if image != "" then
    image = new Images(image)
    if height AND width then
        image.resize(width,height)
    endif
endif
text = text
buttonType = buttonType
radius = radius
textSize = textSize
curve = curve
//Mouse clicking
hover = False
click = False
release = False
// Animation
stop_start_animation = True
animation_time = 150
animation_speed = 13
num_frames = 60
time = 0
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

time2 = 0
// Colors
color = color
color2 = color2
text_color = text_color
text_color2 = text_color2
text_trans_color = text_color
button_trans_color = color
text_trans_color2 = text_color
button_trans_color2 = color
// Opacity
opacitya = 255
opacityb = 180
opacity1 = 255
opacity2 = 180
// Fonts
basefont = basefont
try
  base_font = new pygame.font.Font(basefont, textSize)
except
  base_font = new pygame.font.Font(f"../{basefont}", textSize)
endtry
endprocedure

public procedure draw(outline= new None,action= new None, link= new None, colorChange= new True,
newText= new None, newX= new None, newY= new None, mode= new "normal")
  //New X, Y and Text values//
  if newText then text = newText
  endif
  if newX then x = newX
  endif
  if newY then y = newY
  endif
  //Draws the button. Variable for mouse detection//
  mouse = new pygame.mouse.get_pos()
  click = new pygame.mouse.get_pressed()
  if buttonType == "Rectangle" then
    //If it is a rectangle it will draw it here//
    if mode == "normal" then
      if outline then
        //draws an outline//
        pygame.draw.rect(screen, outline, (x-2,y-2,width+4,height+4),0,curve)
      endif
      if x+width > mouse[0] > x AND y+height > mouse[1] > y AND colorChange then
        hover = True
        stop_start_animation = False
        time2 = new pygame.time.get_ticks()
        step = new (time2 - time) / (animation_time / animation_speed)
        //Draws a lighter version of the image//
        button_trans_color = new change_color(button_trans_color2, color2, step)
        pygame.draw.rect(screen, button_trans_color, (x,y,width,height), 0, curve)
      if text != " "
        text_trans_color = new change_color(text_trans_color2, text_color2, step)
        text = new base_font.render(text, 1, text_trans_color)
        screen.blit(text, (x + (width/2 - text.get_width()/2), y + (height/2 - text.get_height()/2)))
      endif
    endif
  endif
endprocedure
  
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
endif
// Clicking the Button
if click[0] == 1 then
    click = True
endif
if click == True AND NOT click[0] then
    release = True
endif
if release then
    //If there is an action it is run//
    if action then
        action()
    endif
    if link then
        webbrowser.open(link)
    endif
    release = False
    click = False
else
endif
time = new pygame.time.get_ticks()
hover = False
click = False
release = False
step = new (time - time2) / (animation_time / animation_speed)
//A darker version of image when the player isn't hovering over//
if NOT stop_start_animation then
    button_trans_color2 = new change_color(button_trans_color, color, step)
    pygame.draw.rect(screen, button_trans_color2, (x,y,width,height),0, curve)
else
    pygame.draw.rect(screen, color, (x,y,width,height),0, curve)
endif
if text != " then
    //Text is blit here//
    if NOT stop_start_animation then
        text_trans_color2 = new change_color(text_trans_color, text_color, step)
        text = new base_font.render(text, 1, text_trans_color2)
    else
        text = new base_font.render(text, 1, text_color)
    endif
    screen.blit(text, (x + (width/2 - text.get_width()/2), y + (height/2 - text.get_height()/2)))
else
endif
if mode == "draw" then
    time2 = new pygame.time.get_ticks()
    step = new (time2 - time) / (animation_time / animation_speed)
    //Draws a lighter version of the image//
    button_trans_color = new change_color(button_trans_color2, color2, step)
    pygame.draw.rect(screen, button_trans_color, (x,y,width,height), 0, curve)
endif
if mode == "undraw" then
    time = new pygame.time.get_ticks()
    step = new (time - time2) / (animation_time / animation_speed)
    //A darker version of image when the player isn't hovering over//
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
button_trans_color2 = new change_color(button_trans_color, color, step)
pygame.draw.rect(screen, button_trans_color2, (x,y,width,height),0, curve)
endif
endif
endif
if buttonType == "Circle" then
    //If it is a circle it will draw it here//
    if outline then
        //draws an outline//
        pygame.draw.circle(screen, outline, (x, y), radius+2.5,0)
    endif
    //Trig for area//
    differenceInX = mouse[0] - x
    endif
    differenceInY = mouse[1] - y
    endif
    difference = new ( differenceInX**2 + differenceInY**2 )**0.5
endif
//Info on the circle//
if difference <= radius then
    hover = True
    stop_start_animation = False
    time2 = new pygame.time.get_ticks()
    step = new (time2 - time) / (animation_time / animation_speed)
    button_trans_color = new change_color(button_trans_color2, color2, step)
    //Draws a lighter version of the image//
    pygame.draw.circle(screen, button_trans_color, (x,y),radius,0)
if text != " then
    //Text is blit here//
    text_trans_color = new change_color(text_trans_color2, text_color2, step)
    text = new base_font.render(text, 1, text_trans_color)
    screen.blit(text, (x - (text.get_width()/2), y - (text.get_height()/2)))
endif
// Clicking the Button
if click[0] == 1 then
    click = True
endif
if click == True AND NOT click[0] then
    release = True
endif
if release then
    //If there is an action it is run//
    if action != None then
        action()
    endif
    if link != None then
        webbrowser.open(link)
    endif
    release = False
    click = False
else
endif
time = new pygame.time.get_ticks()
step = new (time - time2) / (animation_time / animation_speed)
button_trans_color2 = new change_color(button_trans_color, color, step)
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
hover = False
click = False
release = False
//A darker version of image when the player isn't hovering over//
pygame.draw.circle(screen, button_trans_color2, (x,y),radius,0)
if text != " " then
    //Text is blit here//
    text_trans_color2 = new change_color(text_trans_color, text_color, step)
    text = new base_font.render(text, 1, text_trans_color2)
    screen.blit(text, (x - (text.get_width()/2), y - (text.get_height()/2)))
endif
endif
endif
if buttonType == "Image" then
    if outline then
        //draws an outline//
        pygame.draw.rect(screen, outline, (x-3,y-3,width+6,height+6),0,curve)
    endif
    image.draw(x,y)
    if x+width > mouse[0] > x AND y+height > mouse[1] > y then
        time2 = new pygame.time.get_ticks()
        step = new (time2 - time) / (animation_time / animation_speed)
        opacity2 = new change_opacity(opacity1, opacityb, step)
        image.load_image.set_alpha(opacity2)
        hover = True
        // Clicking the Button
        if click[0] == 1 then
            click = True
        endif
        if click == True AND NOT click[0] then
            release = True
        endif
        if release then
            //If there is an action it is run//
            if action != None then
                action()
            endif
            if link != None then
                webbrowser.open(link)
            endif
            release = False
            click = False
        endif
    else
        endif
    time = new pygame.time.get_ticks()
    hover = False
    click = False
    release = False
    step = new (time - time2) / (animation_time / animation_speed)
    opacity1 = new change_opacity(opacity2, opacitya, step)
    image.load_image.set_alpha(opacity1)
endif
endif
endprocedure
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
function is_hovered()
    return hover
endfunction

function is_clicking()
    return click
endfunction

function change_color(original_color, new_color, step_multiplier= new 0)
    colors = []
    new_color_list = []
    for rgb1, rgb2 in zip(original_color, new_color)
        difference = rgb2 - rgb1
    endif
    colors.append(difference)
    endif
    next rgb1, rgb2
    color_step = [difference / num_frames for difference in colors]
    endif
    for og_color_rgb, new_color_rgb, step in zip(original_color, new_color, color_step)
        if og_color_rgb > new_color_rgb then
            if int(og_color_rgb + step_multiplier*step) < new_color_rgb then
                new_color_list.append(new_color_rgb)
            else
                new_color_list.append(int(og_color_rgb + step_multiplier*step))
            endif
        else
            if int(og_color_rgb + step_multiplier*step) > new_color_rgb then
                new_color_list.append(new_color_rgb)
            else
                new_color_list.append(int(og_color_rgb + step_multiplier*step))
            endif
        endif
        next og_color_rgb, new_color_rgb, step
    return tuple(new_color_list)
endfunction

function change_opacity(original_opacity, new_opacity, step_multiplier= new 0)
    new_opacity_return = None
    difference = new_opacity - original_opacity
    endif
    opacity_step = difference / num_frames
    endif
    if original_opacity > new_opacity then
        if int(original_opacity + opacity_step*step_multiplier) < new_opacity then
            new_opacity_return = new_opacity
        else
            new_opacity_return = new int(original_opacity + opacity_step*step_multiplier)
        endif
    else
        if int(original_opacity + opacity_step*step_multiplier) > new_opacity then
            new_opacity_return = new_opacity
        else
            new_opacity_return = new int(original_opacity + opacity_step*step_multiplier)
        endif
    endif
endfunction
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
endif
return new_opacity_return
endfunction
```

Player:

```
import pygame
from Scripts.logger import *
from Scripts.support import import_folder
logger.debug("RealDL Player Code.")

class Player inherits pygame.sprite.Sprite
    private image
    private animations
    private obstacle_sprites
    private sprite_type
    private font
    private textSize
    private basefont
    private id
    private username
    private paused
    private attack_time
    private attack_cooldown
    private attacking
    private speed
    private direction
    private animation_speed
    private frame_index
    private status
    private key_binds
    private offense
    private movement
    private hitbox
    private rect
    private image_name
    private except

    public procedure new(pos, image, groups, obstacle_sprites, username, id, movement= new "WASD", offense=
new "Space")
        // Setting up player
        try
            super.new(groups)
            screen = new pygame.display.get_surface()
            try image = new pygame.image.load(image).convert_alpha()
            except image = new pygame.image.load(f"../{image}").convert_alpha()
        endtry
        image_name = f"../{image}"
        rect = new image.get_rect(topleft = new pos)
        hitbox = new rect.inflate(0,-26)
        movement = movement
        offense = offense
        key_binds = {
            'move_up' then pygame.K_w if movement == "WASD" else pygame.K_UP,
            'move_down' then pygame.K_s if movement == "WASD" else pygame.K_DOWN,
            'move_left' then pygame.K_a if movement == "WASD" else pygame.K_LEFT,
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

'move_right' then pygame.K_d if movement == "WASD" else pygame.K_RIGHT,
'attack' then pygame.K_SPACE if offense == "Space" else pygame.K_LCTRL if offense == "L-Ctrl" else
pygame.K_RCTRL
}
// graphics setup
import_player_assets()
status = 'down'
frame_index = 0
animation_speed = 0.15
// Movement
direction = new pygame.math.Vector2()
speed = 5
attacking = False
attack_cooldown = 400
attack_time = None
paused = False
//Other stats
username = username
id = id
basefont = "Fonts/Orbitron-Medium.ttf"
textSize = 20
try
  font = new pygame.font.Font(basefont, textSize)
except
  font = new pygame.font.Font(f"../{basefont}", textSize)
endtry
sprite_type = "player"
obstacle_sprites = obstacle_sprites
except FileExistsError as FileNotFoundError
  logger.error(f"Failed to create Player Class: {FileNotFoundError}")
endtry
endprocedure

public procedure import_player_assets()
  character_path = 'Graphics/Game/player/'
  animations = {'up': [], 'down': [], 'left': [], 'right': []}
endprocedure

  'right_idle':[],'left_idle':[],'up_idle':[],'down_idle':[],
  'right_attack':[],'left_attack':[],'up_attack':[],'down_attack':[])
animation_images = {'up': [], 'down': [], 'left': [], 'right': [],
  'right_idle':[],'left_idle':[],'up_idle':[],'down_idle':[],
  'right_attack':[],'left_attack':[],'up_attack':[],'down_attack':[])
for animation in animations.keys()
  full_path = character_path + animation
  animations[animation],animation_images[animation] = new import_folder(full_path)

next animation
public procedure input()
  keys = new pygame.key.get_pressed()
  if NOT paused then
    if keys[key_binds['move_up']] then
      direction.y = -1
      status = 'up'
    endif
  endif

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
elseif keys[key_binds['move_down']] then
    direction.y = 1
    status = 'down'
else
    direction.y = 0
endif
if keys[key_binds['move_right']] then
    direction.x = 1
    status = 'right'
endif
elseif keys[key_binds['move_left']] then
    direction.x = -1
    status = 'left'
else
    direction.x = 0
endif
if keys[key_binds['attack']] then
    attacking = True
    attack_time = new pygame.time.get_ticks()
    logger.info('attack')
endif
endif
endprocedure
```

```
public procedure move(speed)
if NOT paused then
    if direction.magnitude() != new 0 then
        direction = new direction.normalize()
    endif
    hitbox.x += direction.x * speed
    collision('horizontal')
    hitbox.y += direction.y * speed
    collision('vertical')
    rect.center = hitbox.center
endif
endprocedure
```

```
public procedure cooldowns()
    current_time = new pygame.time.get_ticks()
    if attacking then
        if current_time - attack_time >= attack_cooldown then
            attacking = False
        endif
    endif
endprocedure
```

```
function get_image_name()
    image_to_return = image_name
    if "../" in image_to_return then
        image_to_return = new image_name.replace("../","")
    endif
    return image_to_return
endfunction
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

public procedure animate()
  if NOT paused then
    animation = animations[status]
    image = animation_images[status]
    // loop over the frame index
    frame_index += animation_speed
    if frame_index >= animation.length
      frame_index = 0
    endif
    // set the image
    image = new animation[int(frame_index)]
    image_name = new image[int(frame_index)]
    rect = new image.get_rect(center = new hitbox.center)
  endif
endprocedure

public procedure draw(offset_pos)
  // Draw the player.
  text_surface = new font.render(username, True, (240,240,240))
  text_rect = new text_surface.get_rect()
  username_pos = new (offset_pos[0] + rect.width DIV 2 - text_surface.get_width() DIV 2, offset_pos[1] - 33)
  // Define the dimensions and position of the black box
  box_width = text_rect.width + 6 // Adjust the width as needed
  box_height = text_rect.height + 6 // Adjust the height as needed
  box_pos = new (username_pos[0]-3, username_pos[1]-3) // Adjust the position as needed
  // Draw the black box
  pygame.draw.rect(screen, (200,200,200), ((box_pos[0]-2,box_pos[1]-2), (box_width+4, box_height+4)),0,10)
  pygame.draw.rect(screen, (27,31,35), (box_pos, (box_width, box_height)),0,10)
  screen.blit(image, offset_pos)
  screen.blit(text_surface, username_pos)
endprocedure

public procedure update()
  input()
  cooldowns()
  get_status()
  animate()
  move(speed)
endprocedure
  
```

Support:

```

from csv import reader
from os import walk, path
import pygame
function import_csv_layout(path)
  terrain_map = []
  with open(path) as level_map:
    layout = new reader(level_map, delimiter = new ',')
    for row in layout
      terrain_map.append(list(row))
  next row
  return terrain_map
endfunction

function import_folder(folder_path)
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

surface_list = []
image_list = []
send_back = False
while NOT send_back
  for root, dirs, img_files in walk(folder_path)
    for image in img_files
      full_path = new path.join(root, image)
      try
        image_surf = new pygame.image.load(full_path).convert_alpha()
      except pygame.error
        alt_path = new path.join('..', folder_path, image)
        image_surf = new pygame.image.load(alt_path).convert_alpha()
      endtry
      surface_list.append(image_surf)
      image_list.append(full_path)
    next image
  next root, dirs, img_files
  if surface_list AND image_list then
    send_back = True
  else
    if NOT folder_path.startswith('..') then
      folder_path = '..' + folder_path
    else
      send_back = True
    endif
  endif
endwhile
return surface_list, image_list
endfunction
  
```

UI:

```

from Scripts.functions import *
from Scripts.logger import *
import pygame

logger.debug("RealDL UI Code.")
class UI
  private custom_mouse.mode
  private continue_btn
  private bullet_assault
  private quit
  private info2
  private info
  private join_boarder
  private border
  private base_button_height
  private base_button_width
  private thickness
  private curve
  private big_text_size
  private base_text_size
  private text_height
  private button_padding
  private image_padding
  
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
private image_height
private image_width
private BASE_BUTTON_HEIGHT
private BASE_BUTTON_WIDTH
private THICKNESS
private CURVE
private BUTTON_PADDING
private IMAGE_PADDING
private IMAGE_HEIGHT
private IMAGE_WIDTH
private TEXT_HEIGHT
private BASE_TEXT_SIZE
private BIG_TEXT_SIZE
private quit_function
private draw_ui
private custom_mouse
private height_ratio
private width_ratio
private DEFAULT_HEIGHT
private DEFAULT_WIDTH
private screen_height
private screen_width
private screen

public procedure new(custom_mouse, quit_function)
    // Setup Pygame Variables
    screen = new pygame.display.get_surface()
    info = new pygame.display.Info()
    screen_width = info.current_w
    screen_height = info.current_h
    DEFAULT_WIDTH = 1920
    DEFAULT_HEIGHT = 1080
    width_ratio = screen_width / DEFAULT_WIDTH
    height_ratio = screen_height / DEFAULT_HEIGHT
    custom_mouse = custom_mouse
    draw_ui = False
    quit_function = quit_function
    // Constants setup
    BIG_TEXT_SIZE = 50
    BASE_TEXT_SIZE = 15
    TEXT_HEIGHT = 20
    IMAGE_WIDTH = 64
    IMAGE_HEIGHT = 64
    IMAGE_PADDING = 20
    BUTTON_PADDING = 85
    CURVE = 10
    THICKNESS = 2
    BASE_BUTTON_WIDTH = 250
    BASE_BUTTON_HEIGHT = 70
    // Variable setup
    image_width = new int(IMAGE_WIDTH*width_ratio)
    image_height = new int(IMAGE_HEIGHT*height_ratio)
    image_padding = new int(IMAGE_PADDING*width_ratio)
    button_padding = new int(BUTTON_PADDING*height_ratio)
    text_height = new int(TEXT_HEIGHT*height_ratio)
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
base_text_size = new int(BASE_TEXT_SIZE*height_ratio)
big_text_size = new int(BIG_TEXT_SIZE*height_ratio)
curve = new int(CURVE*height_ratio)
thickness = new int(THICKNESS*height_ratio)
base_button_width = new int(BASE_BUTTON_WIDTH*width_ratio)
base_button_height = new int(BASE_BUTTON_HEIGHT*height_ratio)
// UI Board
border = new Button((27,31,35), (27,31,35), screen_width/2, screen_height/2, "Fonts/Orbitron-Regular.ttf",
(27,31,35), (27,31,35), screen_width*0.7, screen_height*0.7,'Rectangle', None, big_text_size, int(curve*1.5))
join_boarder = new Button((27,31,35), (27,31,35), (screen_width/2), (screen_height/2), "Fonts/Orbitron-
Regular.ttf", (240,240,240), (136,173,227), base_button_width*3, base_button_height*6,'Rectangle', None,
big_text_size, curve)
info = new Text(text_height, "Fonts/Orbitron-Regular.ttf", (240,240,240), None, None, None,
int(base_text_size*1.7))
info2 = new Text(text_height, "Fonts/Orbitron-Regular.ttf", (240,240,240), None, None, None,
int(base_text_size*1.7))
quit = new Button((174, 39, 96), (27,31,35), screen_width/2, screen_height*0.78, "Fonts/Orbitron-Medium.ttf",
(27,31,35), (174, 39, 96), base_button_width, base_button_height,'Quit','Rectangle', None, int(big_text_size/1.3),
curve)
bullet_assault = new Text(text_height, "Fonts/Orbitron-Bold.ttf", (240,240,240), None, None, None,
int(base_text_size*3.5))
continue_btn = new Button((39, 174, 96), (27,31,35), (screen_width/2), (screen_height/2)*1.445-
button_padding, "Fonts/Orbitron-Medium.ttf", (27,31,35), (39, 174, 96), base_button_width,
base_button_height,'Continue','Rectangle', None, int(big_text_size/1.3), curve)
endprocedure

public procedure stop_drawing()
draw_ui = False
endprocedure

public procedure draw_menu()
if draw_ui then
    border.draw((27,31,35),None, None, False)
    join_boarder.draw((240,240,240))
    info.draw("draw","Game Paused: You have entered the main menu.",(screen_width/2),
(screen_height/2)*0.665)
    info2.draw("draw","Movement is currently disabled.",(screen_width/2), (screen_height/2)*0.72)
    quit.draw((240,240,240),quit_function)
    continue_btn.draw((240,240,240),stop_drawing)
    bullet_assault.draw("draw","Bullet Assault", (screen_width/2), (screen_height/2)*0.43)
    start_back_hover = new quit.is_hovered()
    join_hover = new continue_btn.is_hovered()
    start_back_click = new quit.is_clicking()
    join_click = new continue_btn.is_clicking()
endif
endprocedure

if start_back_hover OR join_hover then
    if NOT start_back_click AND NOT join_click then
        custom_mouse.mode = 1
    else
        custom_mouse.mode = 2
    else
        endif
endif
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

    custom_mouse.mode = 0
endif
// Draw the mouse
custom_mouse.draw()
  
```

Python Code:

Client:

```

def update_players(self, player_dict):
    # Update existing player instances and remove players that are not in player_dict
    try:
        players_to_remove = []

        for player in self.players:
            if player.id in player_dict:
                player_data = player_dict[player.id]
                player.rect.x = player_data['x']
                player.rect.y = player_data['y']
                try: player.image = pygame.image.load(player_data['image']).convert_alpha()
                except: player.image = pygame.image.load(f"../{player_data['image']}").convert_alpha()
            else:
                logger.debug(f"Removing player instance with id: {player.id}")
                players_to_remove.append(player)

        for player in players_to_remove:
            self.players.remove(player)
            self.level.visible_sprites.remove(player)

        # Create new player instances for players not already in self.players
        for player_data in player_dict.values():
            player_ids = [player.id for player in self.players]
            if player_data['id'] not in player_ids:
                logger.debug(f"Creating new player instance with id: {player_data['id']}")
                new_player = Player(
                    [player_data['x'], player_data['y']],
                    player_data['image'],
                    self.level.visible_sprites,
                    self.level.obstacle_sprites,
                    player_data['username'],
                    player_data['id']
                )
                self.players.append(new_player)
    except:
        logger.error("Player disconnected.")
        self.close()
  
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
def redraw_window(self, all_players_dict):
    try:
        self.update_players(all_players_dict)
        self.level.run(self.player)
        debug([self.player.rect.x, self.player.rect.y])
        self.ui.draw_menu()
        if self.ui.draw_ui: self.player.paused = True
        else: self.player.paused = False
        pygame.display.update()
        self.clock.tick(self.FPS)
    except:
        logger.error("Player has left the game.")
        self.close()

def close(self):
    self.network.close()
    self.run = False
    self.player = None
    self.players = None

def main(self):
    while self.run:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                self.close()
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_ESCAPE:
                    self.ui.draw_ui = not self.ui.draw_ui

        try:
            # Get player data.
            player_dict = {'x': self.player.rect.x, 'y': self.player.rect.y, 'image': self.player.get_image_name(), 'username': self.username, 'id': self.id}
            player_encrypted_dict = self.serialize(self.encryption.encrypt(player_dict))
            self.network.send(player_encrypted_dict)
            all_players_dict = self.encryption.decrypt(self.unserialize(self.network.receive(self.DATA_SIZE)))

            logger.debug(f"Players Dictionary: {all_players_dict}")
            logger.debug(f"Sending player data: {player_dict}")
            logger.debug(f"Received players dictionary: {all_players_dict}")

            self.redraw_window(all_players_dict)
        except:
            logger.error("Failed to send over player to server.")
            self.close()
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
class MainMenu:  
    def __init__(self):  
        # Pygame/ Game setup  
        info = pygame.display.Info()  
        pygame.display.set_caption('Bullet Assault')  
        self.clock = pygame.time.Clock()  
        self.FPS = 60  
        self.loop = True  
        self.screen_width = info.current_w  
        self.screen_height = info.current_h  
        self.DEFAULT_WIDTH = 1920  
        self.DEFAULT_HEIGHT = 1080  
        self.width_ratio = self.screen_width / self.DEFAULT_WIDTH  
        self.height_ratio = self.screen_height / self.DEFAULT_HEIGHT  
        self.screen = pygame.display.set_mode((self.screen_width, self.screen_height))  
  
        # Constants setup  
        self.BIG_TEXT_SIZE = 50  
        self.BASE_TEXT_SIZE = 15  
        self.TEXT_HEIGHT = 20  
        self.IMAGE_WIDTH = 64  
        self.IMAGE_HEIGHT = 64  
        self.IMAGE_PADDING = 20  
        self.BUTTON_PADDING = 85  
        self.CURVE = 10  
        self.THICKNESS = 2  
        self.BASE_BUTTON_WIDTH = 250  
        self.BASE_BUTTON_HEIGHT = 70  
  
        # Variable setup  
        self.image_width = int(self.IMAGE_WIDTH*self.width_ratio)  
        self.image_height = int(self.IMAGE_HEIGHT*self.height_ratio)  
        self.image_padding = int(self.IMAGE_PADDING*self.width_ratio)  
        self.button_padding = int(self.BUTTON_PADDING*self.height_ratio)  
        self.text_height = int(self.TEXT_HEIGHT*self.height_ratio)  
        self.base_text_size = int(self.BASE_TEXT_SIZE*self.height_ratio)  
        self.big_text_size = int(self.BIG_TEXT_SIZE*self.height_ratio)  
        self.curve = int(self.CURVE*self.height_ratio)  
        self.thickness = int(self.THICKNESS*self.height_ratio)  
        self.base_button_width = int(self.BASE_BUTTON_WIDTH*self.width_ratio)  
        self.base_button_height = int(self.BASE_BUTTON_HEIGHT*self.height_ratio)  
  
        # Images  
        self.icon = Images("Graphics/MainMenu/General/icon.png")  
        self.sunset_image = Images("Graphics/MainMenu/General/bg.png")  
        self.icon.display_icon()
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

# Setting up Objects For home
self.youtube_name = Text(self.text_height, "Fonts/Orbitron-Medium.ttf", (27,31,35), None, None, self.base_text_size)
self.github_name = Text(self.text_height, "Fonts/Orbitron-Medium.ttf", (27,31,35), None, None, self.base_text_size)
self.github_button = Button((27,31,35), (27,31,35), self.screen_width-(self.image_width/2)-self.image_padding, (self.image_width/2)+self.image_padding,
                           "Fonts/Orbitron-Regular.ttf", (27,31,35), self.image_width-self.image_height, "Image", None, self.big_text_size, self.curve, 'Graphics/MainMenu/Buttons/github.png')
self.youtube_button = Button((27,31,35), (27,31,35), self.screen_width-(self.image_width/2)-(self.image_padding/4), (self.image_width/2)+self.image_padding,
                           "Fonts/Orbitron-Regular.ttf", (27,31,35), self.image_width-self.image_height, "Image", None, self.big_text_size, self.curve, 'Graphics/MainMenu/Buttons/youtube.png')
self.custom_mouse = Mouse("Graphics/MainMenu/Mouse/mouse1.png", "Graphics/MainMenu/Mouse/mouse2.png", "Graphics/MainMenu/Mouse/mouse3.png")
self.start_button = Button((39, 174, 96), (27,31,35), self.screen_width/2, self.screen_height/2, self.base_button_width, self.base_button_height, "Start", Rectangle, None, int(self.big_text_size/1.3), self.curve)
self.options_button = Button((39, 96, 174), (240,240,240), self.screen_width/2, self.screen_height/2, "Fonts/Orbitron-Medium.ttf", (240,240,240), (39, 96, 174), self.base_button_width,
                             self.base_button_height, "Options", Rectangle, None, int(self.big_text_size/1.3), self.curve)
self.quit_button = Button((174, 39, 96), (240,240,240), self.screen_width/2, self.screen_height/2-self.button_padding, "Fonts/Orbitron-Medium.ttf", (240,240,240), (174, 39, 96), self.base_button_width,
                         self.base_button_height, "Quit", Rectangle, None, int(self.big_text_size/1.3), self.curve)
self.current_x = 0

# Options buttons
self.options_board = Button((27,31,35), (27,31,35), self.screen_width/2, self.screen_height/2, "Fonts/Orbitron-Regular.ttf", (27,31,35), (27,31,35), self.screen_width*0.7, self.screen_height*0.7,'',
                            'Rectangle', None, self.big_text_size, int(self.curve*1.5))
self.back = Button((174, 39, 96), (27,31,35), self.screen_width/2, self.screen_height*0.78, "Fonts/Orbitron-Medium.ttf", (27,31,35), (174, 39, 96), self.base_button_width, self.base_button_height,
                   'Back', Rectangle, None, int(self.big_text_size/1.3), self.curve)
self.control_settings = Text(self.text_height, "Fonts/Orbitron-Bold.ttf", (240,240,240), None, None, None, int(self.base_text_size*2))
self.audio_settings = Text(self.text_height, "Fonts/Orbitron-Bold.ttf", (240,240,240), None, None, None, int(self.base_text_size*2))
self.underline = Button((27,31,35), (240,240,240), (self.screen_width/2)*0.70, (self.screen_height/2)*0.46, "Fonts/Orbitron-Regular.ttf", (240,240,240), (240,240,240),
                       275*self.width_ratio, 3.5*self.height_ratio, 'Rectangle', None, self.big_text_size, self.curve)
self.underline2 = Button((27,31,35), (240,240,240), (self.screen_width/2)*1.3, (self.screen_height/2)*0.46, "Fonts/Orbitron-Regular.ttf", (240,240,240), (240,240,240),
                       245*self.width_ratio, 3.5*self.height_ratio, 'Rectangle', None, self.big_text_size, self.curve)

# Control Settings
self.box_control = Button((27,31,35), (27,31,35), (self.screen_width/2)*0.70, (self.screen_height/2)*0.70, "Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227),
                           self.base_button_width*2, self.base_button_height*7, 'Rectangle', None, self.big_text_size, self.curve)
self.key_binds = Button((27,31,35), (27,31,35), (self.screen_width/2)*0.685, (self.screen_height/2)*0.685, "Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227),
                        self.base_button_width, self.base_button_height, WASD, Rectangle, None, int(self.big_text_size/1.5), self.curve)
self.movement = Text(self.text_height, "Fonts/Orbitron-Medium.ttf", (240,240,240), None, None, None, int(self.base_text_size*2))
self.attack_text = Text(self.text_height, "Fonts/Orbitron-Medium.ttf", (240,240,240), None, None, None, int(self.base_text_size*2))
self.attack_btn = Button((27,31,35), (27,31,35), (self.screen_width/2)*0.70, (self.screen_height/2)*0.495, "Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227),
                        self.base_button_width, self.base_button_height, Space, 'Rectangle', None, int(self.big_text_size/1.5), self.curve)

# Audio Settings
self.box_audio = Button((27,31,35), (27,31,35), (self.screen_width/2)*1.3, (self.screen_height/2)*0.96, "Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), self.base_button_width*2,
                        self.base_button_height*7, 'Rectangle', None, self.big_text_size, self.curve)
self.volume_text = Text(self.text_height, "Fonts/Orbitron-Medium.ttf", (240,240,240), None, None, None, int(self.base_text_size*2))
self.volume_bar = Button((27,31,35), (27,31,35), (self.screen_width/2)*1.3, (self.screen_height/2)*0.685, "Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), self.base_button_width*1.5,
                        self.base_button_height*1, 'Rectangle', None, self.big_text_size, self.curve)
self.volume_line = Button((27,31,35), (27,31,35), (self.screen_width/2)*1.25, (self.screen_height/2)*0.685, "Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), self.base_button_width,
                        self.base_button_height*0.36, 'Rectangle', None, self.big_text_size, self.curve)
self.volume_button = Button((27,31,35), (51,96,158), (self.screen_width/2)*0.685, (self.screen_height/2)*0.685, "Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), self.base_button_height*0.3,
                           self.base_button_height*0.3, 'Rectangle', None, self.big_text_size, int(self.curve*2))
self.volume_indicator = Text(self.text_height, "Fonts/Orbitron-Regular.ttf", (240,240,240), None, None, None, int(self.base_text_size*2))
self.audio_text = Text(self.text_height, "Fonts/Orbitron-Medium.ttf", (240,240,240), None, None, None, int(self.base_text_size*2))
self.sound_text = Text(self.text_height, "Fonts/Orbitron-Medium.ttf", (240,240,240), None, None, None, int(self.base_text_size*2))
self.music_text = Text(self.text_height, "Fonts/Orbitron-Medium.ttf", (240,240,240), None, None, None, int(self.base_text_size*2))
self.sound_box = Button((27,31,35), (39,174,96), (self.screen_width/2)*1.45, (self.screen_height/2)*0.97, "Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), self.base_button_height*0.4,
                        self.base_button_height*0.4, 'Rectangle', None, self.big_text_size, self.curve)
self.music_box = Button((27,31,35), (39,174,96), (self.screen_width/2)*1.45, (self.screen_height/2)*1.08, "Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), self.base_button_height*0.4,
                        self.base_button_height*0.4, 'Rectangle', None, self.big_text_size, self.curve)
self.audio_box = Button((27,31,35), (27,31,35), (self.screen_width/2)*1.3, (self.screen_height/2)*1.03, "Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), self.base_button_width*1.5,
                        self.base_button_height*2.5, 'Rectangle', None, self.big_text_size, self.curve)
self.music_box = Button((27,31,35), (27,31,35), (self.screen_width/2)*1.45, (self.screen_height/2)*1.08, "Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), self.base_button_height*0.4,
                        self.base_button_height*0.4, 'Rectangle', None, self.big_text_size, self.curve)
self.audio_box = Button((27,31,35), (27,31,35), (self.screen_width/2)*1.3, (self.screen_height/2)*1.03, "Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), self.base_button_width*1.5,
                        self.base_button_height*2.5, 'Rectangle', None, self.big_text_size, self.curve)

# Start
self.start_board = Button((27,31,35), (27,31,35), self.screen_width/2, self.screen_height/2, "Fonts/Orbitron-Regular.ttf", (27,31,35), (27,31,35), self.screen_width*0.7, self.screen_height*0.7,'',
                           'Rectangle', None, self.big_text_size, int(self.curve*1.5))
self.start_back = Button((174, 39, 96), (27,31,35), self.screen_width/2, self.screen_height*0.78, "Fonts/Orbitron-Medium.ttf", (27,31,35), (174, 39, 96), self.base_button_width, self.base_button_height, 'Back',
                        'Rectangle', None, int(self.big_text_size/1.3), self.curve)
self.bullet_assault = Text(self.text_height, "Fonts/Orbitron-Bold.ttf", (240,240,240), None, None, None, int(self.base_text_size*3.5))
self.join_message = Text(self.text_height, "Fonts/Orbitron-Regular.ttf", (240,240,240), None, None, None, int(self.base_text_size*1.7))
self.ip_text_box = TextBox(self.base_button_width*2.5, self.base_button_height*1.5, (self.screen_width/2), (self.screen_height/2)*0.99, (240,240,240), (280,200,200), "Fonts/Orbitron-Regular.ttf", self.curve,
                        self.thickness, self.base_button_width*2.5)
self.join_boarder = Button((27,31,35), (27,31,35), (self.screen_width/2), (self.screen_height/2), "Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), self.base_button_width*3, self.base_button_height*6,
                           "", 'Rectangle', None, self.big_text_size, self.curve)
self.join_btn = Button((39, 174, 96), (27,31,35), (self.screen_width/2)*(1.445-self.button_padding), "Fonts/Orbitron-Medium.ttf", (27,31,35), (39, 174, 96), self.base_button_width,
                      self.base_button_height, Connect, Rectangle, None, int(self.big_text_size/1.3), self.curve)
self.server_ip_text = Text(self.text_height, "Fonts/Orbitron-Regular.ttf", (240,240,240), None, None, None, int(self.base_text_size*1.7))
self.name_text_box = TextBox(self.base_button_width*2.5, self.base_button_height*1.5, (self.screen_width/2), (self.screen_height/2)*0.70, (240,240,240), (280,200,200), "Fonts/Orbitron-Regular.ttf", self.curve,
                           self.thickness, self.base_button_width*2.5)
self.name_box_text = Text(self.text_height, "Fonts/Orbitron-Regular.ttf", (240,240,240), None, None, None, int(self.base_text_size*1.7))

# Settings
self.main_menu_pages = "home"
self.settings_screen = "control"
self.attack_keys = "Space"
self.keys = "NAD"
# Volume
self.min_vol_x = self.volume_line.x + self.volume_button.width/3 + 1
self.max_vol_x = self.volume_line.x + self.volume_line.width - (self.volume_button.width*2)/3 - 1
self.volume_ratio = 100 / (self.max_vol_x - self.min_vol_x)
self.volume = int(self.volume_ratio * ((self.volume_button.x + self.volume_button.width/4) - self.min_vol_x))
self.sound_change = 0
self.music_change = 0

def close(self):
    self.loop = False
    pgome.quit()
    sys.exit()

def options(self):
    self.main_menu_pages = "settings"

def home(self):
    self.main_menu_pages = "home"

def start_option(self):
    self.main_menu_pages = "start"

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
def draw_moving_background(self):
    # Draw the moving image on the screen
    self.sunset_image.draw(self.current_x, 0)
    self.sunset_image.draw((self.current_x - self.sunset_image.rect.width), 0)
    self.current_x += 1

    # If the image has moved beyond its width, reset it to 0
    if self.current_x >= self.sunset_image.rect.width:
        self.current_x = 0

def change_keys(self):
    if self.keys == 'WASD':
        self.keys = 'Arrow Keys'
    else:
        self.keys = 'WASD'

def control_settings_change(self):
    self.settings_screen = "control"

def audio_settings_change(self):
    self.settings_screen = "audio"

def change_attack(self):
    # Changes the key binds.
    if self.attack_keys == "Space":
        self.attack_keys = "L-Ctrl"
    elif self.attack_keys == "L-Ctrl":
        self.attack_keys = "R-Ctrl"
    else:
        self.attack_keys = "Space"

def sound_box_color(self):
    # Changes the sound box color
    self.sound_change += 1
    if self.sound_change == 1:
        self.sound_box.color = (39,174,96)
    else:
        self.sound_change = 0
        self.sound_box.color = (27,31,35)

def music_box_color(self):
    # Changes the music box check color.
    self.music_change += 1
    if self.music_change == 1:
        self.music_box.color = (39,174,96)
    else:
        self.music_change = 0
        self.music_box.color = (27,31,35)
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
def volume_settings(self):
    # Get the mouse and sets the volume to where the mouse is.
    mouse_pos = pygame.mouse.get_pos()
    if mouse_pos[0] - self.volume_button.width/3 > self.volume_line.x:
        if mouse_pos[0] + (self.volume_button.width/3)*2 < self.volume_line.x + self.volume_line.width:
            self.volume_button.x = mouse_pos[0] - self.volume_button.width/3
            self.volume = int(self.volume_ratio * (mouse_pos[0] - self.min_vol_x))

def start_game(self):
    self.starting_dict = {
        "settings": {
            "control": {
                "movement": self.keys,
                "offense": self.attack_keys
            },
            "audio": {
                "volume": self.volume,
                "sound": True if self.sound_change == 1 else False,
                "music": True if self.music_change == 1 else False
            }
        },
        "start": {
            "username": self.name_text_box.return_text() or "Player",
            "server_ip": self.ip_text_box.return_text() or "Server IP"
        }
    }
    self.loop = False

def draw_objects(self, events):
    # Draw the buttons and check for hover
    if self.main_menu_pages == "home":
        self.github_button.draw((27,31,35),None,"https://github.com/TheRealDL1/Simple-Client-Server")
        self.youtube_button.draw((27,31,35),None,"https://www.youtube.com/watch?v=dQw4w9WgXcQ")
        self.start_button.draw((27,31,35),self.start_option)
        self.options_button.draw((27,31,35),self.options)
        self.quit_button.draw((27,31,35),self.close,None,True)

        start_button_hover = self.start_button.is_hovered()
        options_button_hover = self.options_button.is_hovered()
        quit_button_hover = self.quit_button.is_hovered()
        github_button_hover = self.github_button.is_hovered()
        youtube_button_hover = self.youtube_button.is_hovered()

        start_button_click = self.start_button.is_clicking()
        options_button_click = self.options_button.is_clicking()
        quit_button_click = self.quit_button.is_clicking()
        github_button_click = self.github_button.is_clicking()
        youtube_button_click = self.youtube_button.is_clicking()
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

# Check if mouse is hovering over buttons
if start_button_hover or options_button_hover or quit_button_hover or github_button_hover or youtube_button_hover:
    # Draw hover text.
    if github_button_hover: self.github_name.draw("draw", "Github", self.screen_width-(self.image_width/2)-self.image_padding, (self.image_width/2)+self.image_padding*3.1)
    else: self.github_name.draw("undraw", "Github", self.screen_width-(self.image_width/2)-self.image_padding, (self.image_width/2)+self.image_padding*3.1)

    if youtube_button_hover: self.youtube_name.draw("draw", "Youtube", self.screen_width-(self.image_width/2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)
    else: self.youtube_name.draw("undraw", "Youtube", self.screen_width-(self.image_width/2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)

    if not start_button_click and not options_button_click and not quit_button_click and not github_button_click and not youtube_button_click:
        self.custom_mouse.mode = 1
    else:
        self.custom_mouse.mode = 2
else:
    self.custom_mouse.mode = 0
self.github_name.draw("undraw", "Github", self.screen_width-(self.image_width/2)-self.image_padding, (self.image_width/2)+self.image_padding*3.1)
self.youtube_name.draw("undraw", "Youtube", self.screen_width-(self.image_width/2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)

if self.main_menu_pages == "settings":
    self.options_board.draw((27, 31, 35), None, None, False)
    self.back.draw((240, 240, 240), self.home)
    self.control_settings.draw("draw", "Control Settings", (self.screen_width/2)*0.70, (self.screen_height/2)*0.42, self.control_settings_change)
    self.audio_settings.draw("draw", "Audio Settings", (self.screen_width/2)*1.3, (self.screen_height/2)*0.42, self.audio_settings_change)
    self.github_button.draw((27, 31, 35), None, "https://github.com/TheRealD1/Simple-Client-Server")
    self.youtube_button.draw((27, 31, 35), None, "https://www.youtube.com/@dominicpike")

    back_hover = self.back.is_hovered()
    github_button_hover = self.github_button.is_hovered()
    youtube_button_hover = self.youtube_button.is_hovered()
    key_binds_hover = self.control_settings.is_hovered()
    audio_hover = self.audio_settings.is_hovered()
    key_hover = self.key_binds.is_hovered()
    attack_hover = self.attack_btn.is_hovered()
    volume_btn_hover = self.volume_button.is_hovered()
    sound_box_hover = self.sound_box.is_hovered()
    music_box_hover = self.music_box.is_hovered()

    back_click = self.back.is_clicking()
    github_button_click = self.github_button.is_clicking()
    youtube_button_click = self.youtube_button.is_clicking()
    key_binds_click = self.control_settings.is_clicking()
    audio_click = self.audio_settings.is_clicking()
    key_click = self.key_binds.is_clicking()
    attack_click = self.attack_btn.is_clicking()
    volume_btn_click = self.volume_button.is_clicking()
    sound_box_click = self.sound_box.is_clicking()
    music_box_click = self.music_box.is_clicking()

if back_hover or github_button_hover or youtube_button_hover or key_binds_hover or audio_hover or key_hover or attack_hover or volume_btn_hover or sound_box_hover or music_box_hover:
    if github_button_hover: self.github_name.draw("draw", "Github", self.screen_width-(self.image_width/2)-self.image_padding, (self.image_width/2)+self.image_padding*3.1)
    else: self.github_name.draw("undraw", "Github", self.screen_width-(self.image_width/2)-self.image_padding, (self.image_width/2)+self.image_padding*3.1)

    if youtube_button_hover: self.youtube_name.draw("draw", "Youtube", self.screen_width-(self.image_width/2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)
    else: self.youtube_name.draw("undraw", "Youtube", self.screen_width-(self.image_width/2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)

    if not back_click and not github_button_click and not youtube_button_click and not key_binds_click and not audio_click and not key_click and not attack_click and not volume_btn_click and not sound_box_click and not music_box_click:
        self.custom_mouse.mode = 1
    else:
        if volume_btn_click:
            self.volume_settings()
            self.custom_mouse.mode = 2
else:
    self.custom_mouse.mode = 0
self.github_name.draw("undraw", "Github", self.screen_width-(self.image_width/2)-self.image_padding, (self.image_width/2)+self.image_padding*3.1)
self.youtube_name.draw("undraw", "Youtube", self.screen_width-(self.image_width/2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)

if self.settings_screen == "control":
    if audio_hover: self.underline2.draw(None, None, None, True, None, None, "draw")
    else: self.underline2.draw(None, None, None, True, None, None, "undraw")
    self.underline.draw(None, None, None, True, None, None, "draw")
    self.box_control.draw((240, 240, 240))
    self.movement.draw("draw", "Movement", (self.screen_width/2)*0.70, (self.screen_height/2)*0.57)
    self.key_binds.draw((240, 240, 240), self.change_keys, None, True, self.keys)
    self.attack_text.draw("draw", "Offense", (self.screen_width/2)*0.70, (self.screen_height/2)*0.82)
    self.attack_btn.draw((240, 240, 240), self.change_attack, None, True, self.attack_keys)

elif self.settings_screen == "audio":
    if key_binds_hover: self.underline.draw(None, None, None, True, None, None, "draw")
    else: self.underline.draw(None, None, None, True, None, None, "undraw")
    self.underline2.draw(None, None, None, True, None, None, "draw")
    self.box_audio.draw((240, 240, 240))
    self.volume_text.draw("draw", "Volume", (self.screen_width/2)*1.3, (self.screen_height/2)*0.57)
    self.volume_bar.draw((240, 240, 240))
    self.volume_line.draw((240, 240, 240))
    self.volume_button.draw((240, 240, 240))
    self.volume_indicator.draw("draw", str(self.volume), (self.screen_width/2)*1.435, (self.screen_height/2)*0.685)
    self.audio_text.draw("draw", "Audio", (self.screen_width/2)*1.3, (self.screen_height/2)*0.82)
    self.audio_box.draw((240, 240, 240))
    self.sound_text.draw("draw", "Sound", (self.screen_width/2)*1.185, (self.screen_height/2)*0.97)
    self.music_text.draw("draw", "Music", (self.screen_width/2)*1.18, (self.screen_height/2)*1.08)
    self.sound_box.draw((240, 240, 240), self.sound_box_color)
    self.music_box.draw((240, 240, 240), self.music_box_color)
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

if self.main_menu_pages == "start":
    self.options_board.draw((27,31,35),None, None, False)
    self.start_back.draw((240,240,240),self.home)
    self.github_button.draw((27,31,35),None,"https://github.com/TheRealDL/Simple-Client-Server")
    self.youtube_button.draw((27,31,35),None,"https://www.youtube.com/@dominicpike")
    self.join_boarder.draw((240,240,240))
    self.bullet_assault.draw("draw","Bullet Assault", (self.screen_width/2), (self.screen_height/2)*0.43)
    self.server_ip_text.draw("draw","Server IP Address", (self.screen_width/2), (self.screen_height/2)*0.955)
    message = "Enter the Server's IP Address that you want to join!"
    self.join_message.draw("draw",message, (self.screen_width/2), (self.screen_height/2)*0.52)
    self.ip_text_box.draw()
    self.ip_text_box.updateText(events)
    self.ip_text_box.update()
    self.join_btn.draw((240,240,240),self.start_game)

# Name Text Box
self.name_box_text.draw("draw","Username", (self.screen_width/2), (self.screen_height/2)*0.665)
self.name_text_box.draw()
self.name_text_box.updateText(events)
self.name_text_box.update()

github_button_hover = self.github_button.is_hovered()
youtube_button_hover = self.youtube_button.is_hovered()
start_back_hover = self.start_back.is_hovered()
join_hover = self.join_btn.is_hovered()
text_box_hover = self.ip_text_box.is_hovered()
name_hover = self.name_text_box.is_hovered()

github_button_click = self.github_button.is_clicking()
youtube_button_click = self.youtube_button.is_clicking()
start_back_click = self.start_back.is_clicking()
join_click = self.join_btn.is_clicking()
text_box_click = self.ip_text_box.is_clicking()
name_click = self.name_text_box.is_clicking()

if start_back_hover or github_button_hover or youtube_button_hover or join_hover or text_box_hover or name_hover:
    # Draw hover text.
    if github_button_hover: self.github_name.draw("draw","Github", self.screen_width-(self.image_width/2)-self.image_padding, (self.image_width/2)+self.image_padding*3.1)
    else: self.github_name.draw("undraw","Github", self.screen_width-(self.image_width/2)-self.image_padding, (self.image_width/2)+self.image_padding*3.1)

    if youtube_button_hover: self.youtube_name.draw("draw","Youtube", self.screen_width-(self.image_width*2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)
    else: self.youtube_name.draw("undraw","Youtube", self.screen_width-(self.image_width*2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)

    if not start_back_click and not github_button_click and not youtube_button_click and not join_click and not text_box_click and not name_click:
        self.custom_mouse.mode = 1
    else:
        self.custom_mouse.mode = 2
else:
    self.custom_mouse.mode = 0
self.github_name.draw("undraw","Github", self.screen_width-(self.image_width/2)-self.image_padding, (self.image_width/2)+self.image_padding*3.1)
self.youtube_name.draw("undraw","Youtube", self.screen_width-(self.image_width*2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)

def redraw_window(self, events):
    self.draw_moving_background()
    self.draw_objects(events)

def run(self):
    while self.loop:
        events = pygame.event.get()
        for event in events:
            if event.type == pygame.QUIT:
                self.close()
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_ESCAPE:
                    self.close()

            # Update the display and frame rate
            self.redraw_window(events)
            pygame.display.update()
            self.clock.tick(self.FPS)
        if not self.loop:
            return self.starting_dict

if __name__ == "__main__":
    while True:
        game = MainMenu()
        user_dict = game.run()
        client = Client(user_dict)
        client.main()
  
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Functions:

```
import pygame
import webbrowser
from Scripts.logger import *
logger.info("Pygame Functions 0.0.1 (Python 3.11.0)\nI am not affiliated
with pygame. https://github.com/TheRealDL1/pygame_functions")

class Images(object):
    def __init__(self, image):
        self.image = image
        self.screen = pygame.display.get_surface()
        self.load_image = self.load_image_from_file()
        self.rect = self.load_image.get_rect()

    def load_image_from_file(self):
        try:
            return pygame.image.load(self.image).convert_alpha()
        except:
            return pygame.image.load(f'../{self.image}').convert_alpha()

    def display_icon(self):
        pygame.display.set_icon(self.load_image)

    def draw(self, x=0, y=0):
        self.screen.blit(self.load_image, (x,y))

    def resize(self, width, height):
        self.load_image = pygame.transform.scale(self.load_image, (width,
height))

class Mouse(object):
    def __init__(self,mouse_image1, mouse_image2, mouse_image3):
        pygame.mouse.set_visible(False)
        self.screen = pygame.display.get_surface()
        self.mode = 0
        self.mouse_mode1 = Images(mouse_image1)
        self.mouse_mode2 = Images(mouse_image2)
        self.mouse_mode3 = Images(mouse_image3)

    def draw(self):
        # Get the current mouse position
        mouse_x, mouse_y = pygame.mouse.get_pos()

        # Draw a green square at the mouse cursor position
        if self.mode == 0:
            self.mouse_mode1.draw(mouse_x , mouse_y)
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
elif self.mode == 1:  
    self.mouse_mode2.draw(mouse_x , mouse_y)  
else:  
    self.mouse_mode3.draw(mouse_x , mouse_y)  
  
class WordButton(object):  
    def __init__(self, x, y, text, color1, color2, basefont, largefont,  
textSize=30,):  
        """Sets the values for button"""  
        self.color = color1  
        self.color2 = color2  
        self.x = x  
        self.y = y  
        self.text = text  
        self.textSize = textSize  
        self.largeSize = round(self.textSize * 1.25)  
        self.basefont = basefont  
        self.largefont = largefont  
        try:  
            self.base_font = pygame.font.Font(self.basefont,  
self.textSize)  
            self.large_font = pygame.font.Font(self.largefont,  
self.largeSize)  
        except:  
            self.base_font = pygame.font.Font(f"../{self.basefont}",  
self.textSize)  
            self.large_font = pygame.font.Font(f"../{self.largefont}",  
self.largeSize)  
        self.hover = False  
        self.click = False  
        self.release = False  
  
    def displayText(self, win, newText=None, action=None, link=None):  
        if newText != None:  
            self.text = newText  
        mouse = pygame.mouse.get_pos()  
        click = pygame.mouse.get_pressed()  
  
        # Render text with both fonts  
        text = self.large_font.render(self.text, 1, self.color2)  
        text2 = self.base_font.render(self.text, 1, self.color)  
  
        # Check if mouse is over the text  
        if self.x + text.get_width()/2 > mouse[0] > self.x -  
text.get_width()/2 and self.y + text.get_height()/2 > mouse[1] > self.y -  
text.get_height()/2:
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
self.hover = True
# Render text with larger font and lighter color
text = self.large_font.render(self.text, 1, self.color2)

# Center text vertically as well as horizontally
win.blit(text, (self.x - text.get_width() / 2, self.y -
text.get_height() / 2))

# Check for mouse click and run action if specified
if click[0] == 1:
    self.click = True

if self.click == True and not click[0]:
    self.release = True

if self.release:
    if action != None:
        action()
    if link != None:
        webbrowser.open(link)
    self.release = False
    self.click = False
else:
    self.hover = False
    self.click = False
    self.release = False
# Render text with base font and color
text2 = self.base_font.render(self.text, 1, self.color)

# Center text vertically as well as horizontally
win.blit(text2, (self.x - text2.get_width() / 2, self.y -
text2.get_height() / 2))

class Animation(object):
    def __init__(self, image, scale, scale_direction, scale_min,
scale_max, scale_speed, screen_y, screen_x):
        self.scale = scale #1.0
        self.image = image
        self.scale_direction = scale_direction #-1.35 # Start by zooming
out
        self.scale_min = scale_min #0.9
        self.scale_max = scale_max #1.2
        self.scale_speed = scale_speed #0.01 # The rate at which the
scale changes
        self.screen_center_x = screen_x
        self.screen_center_y = screen_y #324#win.get_height() // 2
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
self.monopoly_rect =  
self.image.get_rect(center=(self.screen_center_x, self.screen_center_y))  
  
def animation(self, win):  
    self.scale += self.scale_direction * self.scale_speed  
    if self.scale <= self.scale_min:  
        if self.scale_direction < 0:  
            self.scale_direction = self.scale_direction * -1  
        else:  
            self.scale_direction = self.scale_direction * 1# Start  
zooming in  
    elif self.scale >= self.scale_max:  
        if self.scale_direction > 0:  
            self.scale_direction = self.scale_direction * -1  
        else:  
            self.scale_direction = self.scale_direction * 1 # Start  
zooming out  
  
    # Scale the image and get its rect  
    scaled_monopoly = pygame.transform.rotozoom(self.image, 0,  
self.scale)  
    scaled_monopoly_rect =  
scaled_monopoly.get_rect(center=(self.screen_center_x,  
self.screen_center_y))  
  
    # Blit the scaled image to the screen and update the display  
win.blit(scaled_monopoly, scaled_monopoly_rect)  
  
class TextBox(object):  
    def __init__(self, width, height, x, y, color1, color2, textfont,  
curve, thickness, maxTextWidth, text_size=50):  
        self.text = ''  
        self.textSize = text_size  
        self.textfont = textfont  
        try:  
            self.base_font = pygame.font.Font(self.textfont,  
self.textSize)  
        except:  
            self.base_font = pygame.font.Font(f"../{self.textfont}",  
self.textSize)  
        self.active = False  
        self.width = width  
        self.height = height  
        self.x = x - self.width / 2  
        self.ogWidth = width  
        self.y = y
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
self.color_active = pygame.Color(color1)
self.color_passive = pygame.Color(color2)
self.color = self.color_passive
self.last_update = 0 # time of last update
self.show_cursor = False # whether to show cursor or not
self.cursor = 0
self.maxTextWidth = maxTextWidth
self.curve = curve
self.thickness = thickness
self.screen = pygame.display.get_surface()
self.hover = False
self.click = False

def draw(self):
    area = [self.x, self.y, self.width, self.height]
    pygame.draw.rect(self.screen, self.color, area, self.thickness,
self.curve)

def is_hovered(self):
    mouse = pygame.mouse.get_pos()
    return self.x + self.width > mouse[0] > self.x and self.y +
self.height > mouse[1] > self.y

def is_clicking(self):
    mouse = pygame.mouse.get_pos()
    click = pygame.mouse.get_pressed()
    return self.x + self.width > mouse[0] > self.x and self.y +
self.height > mouse[1] > self.y and click[0]

def checkTextBox(self):
    mouse = pygame.mouse.get_pos()
    click = pygame.mouse.get_pressed()

    if self.x + self.width > mouse[0] > self.x and self.y +
self.height > mouse[1] > self.y:
        self.hover = True
        if click[0]: self.click = True
        if self.click and not click[0]: self.click = False
        if self.active: self.active = False
        else: self.active = True

    else:
        self.hover = False
        self.active = False
        self.click = False
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
if self.active:  
    self.color = self.color_active  
else:  
    self.color = self.color_passive  
  
def update(self, color=(240,240,240),x_pos=None, y_pos=None):  
    middleOfX = self.screen.get_width() // 2  
    if x_pos: self.x = x_pos  
    if y_pos: self.y = y_pos  
  
    if self.text:  
        surface_area = self.base_font.render(self.text, True, color)  
        text_width = surface_area.get_width() + 20  
        if self.ogWidth > self.width:  
            self.width = self.ogWidth  
            self.x = middleOfX - self.width // 2  
        if text_width > self.width:  
            self.width = text_width  
            self.x = middleOfX - self.width // 2  
        elif self.width > text_width:  
            if self.ogWidth < self.width:  
                self.width = text_width  
                self.x = middleOfX - self.width // 2  
            self.screen.blit(surface_area, (self.x + 5, self.y +  
(self.height // 2 - surface_area.get_height() // 2)))  
        else:  
            self.width = self.ogWidth  
            self.x = middleOfX - self.width // 2  
            surface_area = self.base_font.render(self.text, True, color)  
            self.screen.blit(surface_area, (self.x + 5, self.y +  
(self.height // 2 - surface_area.get_height() // 2)))  
  
        time_since_last_update = pygame.time.get_ticks() -  
self.last_update  
        if self.active and time_since_last_update > 500:  
            self.show_cursor = not self.show_cursor  
            self.last_update = pygame.time.get_ticks()  
  
        if self.show_cursor and self.active == True:  
            cursor_width = 2  
            cursor_pos = self.base_font.size(self.text[:self.cursor])[0] -  
5  
            text_to_show = self.text[:self.cursor] + '|' +  
self.text[self.cursor:]  
            cursor_pos += self.base_font.size(' ')[0] * 0.8
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
surface_area = self.base_font.render(text_to_show, True,
color)
        cursor_area = pygame.Surface((cursor_width,
surface_area.get_height()))
        cursor_area.fill(color)
        self.screen.blit(cursor_area, (self.x + cursor_pos, self.y +
(self.height // 2 - cursor_area.get_height() // 2)), (0, 0, cursor_width,
cursor_area.get_height()))
    #else:
        #surface_area = self.base_font.render(self.text, True, color)
        #self.screen.blit(surface_area, (self.x + 5, self.y +
(self.height // 2 - surface_area.get_height() // 2)))

def updateText(self, events, color=(0,0,0), function=None):
    surface_area = self.base_font.render(self.text, True, color)
    text_width = surface_area.get_width() + 20
    for event in events:
        if event.type == pygame.MOUSEBUTTONDOWN:
            self.checkTextBox()
        if event.type == pygame.KEYDOWN:
            if self.active == True:
                if event.key == pygame.K_BACKSPACE:
                    if self.cursor > 0:
                        self.text = self.text[:self.cursor-1] +
self.text[self.cursor:]
                        self.cursor -= 1
                elif event.key == pygame.K_DELETE:
                    self.text = self.text[:self.cursor] +
self.text[self.cursor+1:]
                elif event.key == pygame.K_RETURN:
                    if function != None:
                        function()
                elif event.key == pygame.K_LEFT:
                    if self.cursor > 0:
                        self.cursor -= 1
                elif event.key == pygame.K_RIGHT:
                    if self.cursor < len(self.text):
                        self.cursor += 1
            else:
                if text_width < self.maxTextWidth and event.key is
not pygame.K_TAB:
                    new_text = self.text[:self.cursor] +
event.unicode + self.text[self.cursor:]
                    if len(new_text) > len(self.text):
                        self.text = new_text
                        self.cursor += 1
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
def return_text(self):
    return self.text

class Text(object):
    def __init__(self, height, baseFont, color=(0,0,0), x=None, y=None,
text=None, textSize=32):
        self.text = text
        self.x = x
        self.y = y
        self.baseFont = baseFont
        self.height = height
        self.textSize = textSize
        self.color = color
        self.screen = pygame.display.get_surface()
        try:
            self.base_font = pygame.font.Font(self.baseFont,
self.textSize)
        except:
            self.base_font = pygame.font.Font(f"../{self.baseFont}",

self.textSize)
        self.hover = False
        self.click = False
        self.release = False

        # Animation
        self.stop_start_animation = True
        self.animation_time = 150
        self.animation_speed = 13
        self.num_frames = 60
        self.time = 0
        self.time2 = 0

        # Opacity
        self.opacitya = 0
        self.opacityb = 255
        self.opacity_color1 = 0
        self.opacity_color2 = 0
        self.opacity_color = 0

    def draw(self, mode="draw", new_text=None, new_x=None, new_y=None,
function= None, link=None):
        if new_text: self.text = new_text
        if new_x: self.x = new_x
        if new_y: self.y = new_y
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
if mode == "normal":  
    surface_area = self.base_font.render(self.text, 1, self.color)  
    self.screen.blit(surface_area, (self.x -  
surface_area.get_width() / 2, self.y - surface_area.get_height() / 2))  
  
if mode == "draw":  
    self.stop_start_animation = False  
    self.time = pygame.time.get_ticks()  
    step = (self.time - self.time2) / (self.animation_time /  
self.animation_speed)  
    self.opacity_color1 = self.change_opacity(self.opacity_color2,  
self.opacityb, step)  
    self.opacity_color = self.opacity_color1  
  
if mode == "undraw":  
    self.time2 = pygame.time.get_ticks()  
    step = (self.time2 - self.time) / (self.animation_time /  
self.animation_speed)  
    self.opacity_color2 = self.change_opacity(self.opacity_color1,  
self.opacitya, step)  
    self.opacity_color = self.opacity_color2  
  
if not self.stop_start_animation:  
    surface_area = self.base_font.render(self.text, 1, self.color)  
    surface_area.set_alpha(self.opacity_color)  
    self.screen.blit(surface_area, (self.x -  
surface_area.get_width() / 2, self.y - surface_area.get_height() / 2))  
  
if self.is_hovered(): self.hover = True  
if self.is_clicking(): self.click = True  
if self.click and not self.is_clicking(): self.release = True  
  
if self.release:  
    if function: function()  
    if link: webbrowser.open(link)  
  
    self.release = False  
    self.click = False  
  
if not self.is_hovered:  
    self.hover = False  
    self.click = False  
    self.release = False  
  
def change_opacity(self, original_opacity, new_opacity,  
step_multiplier=0):
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
new_opacity_return = None
difference = new_opacity - original_opacity
opacity_step = difference / self.num_frames
if original_opacity > new_opacity:
    if int(original_opacity + opacity_step*step_multiplier) <
new_opacity:
        new_opacity_return = new_opacity
    else:
        new_opacity_return = int(original_opacity +
opacity_step*step_multiplier)
else:
    if int(original_opacity + opacity_step*step_multiplier) >
new_opacity:
        new_opacity_return = new_opacity
    else:
        new_opacity_return = int(original_opacity +
opacity_step*step_multiplier)

return new_opacity_return

def is_hovered(self):
    mouse_pos = pygame.mouse.get_pos()
    text_rect = self.base_font.render(self.text, 1,
self.color).get_rect()
    text_rect.center = (self.x, self.y)
    return text_rect.collidepoint(mouse_pos)

def is_clicking(self):
    mouse_pos = pygame.mouse.get_pos()
    mouse_buttons = pygame.mouse.get_pressed()
    text_rect = self.base_font.render(self.text, 1,
self.color).get_rect()
    text_rect.center = (self.x, self.y)
    return text_rect.collidepoint(mouse_pos) and mouse_buttons[0] == 1

class Button(object):
    """A class for all buttons"""
    def __init__(self, color, color2, x, y, basefont, text_color=(0,0,0),
text_color2=(0,0,0), width=None, height=None,
text='', buttonType='Rectangle', radius=None, textSize=30, curve=10,
image=''):
        """Sets the values for button"""
        self.screen = pygame.display.get_surface()
        try:
            self.x = x - width/2
            self.y = y - height/2
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
    self.width = width
    self.height = height
except:
    self.x = x
    self.y = y

if image != "":
    self.image = Images(image)
    if height and width:
        self.image.resize(width,height)
self.text = text
self.buttonType = buttonType
self.radius = radius
self.textSize = textSize
self.curve = curve
#Mouse clicking
self.hover = False
self.click = False
self.release = False
# Animation
self.stop_start_animation = True
self.animation_time = 150
self.animation_speed = 13
self.num_frames = 60
self.time = 0
self.time2 = 0
# Colors
self.color = color
self.color2 = color2
self.text_color = text_color
self.text_color2 = text_color2
self.text_trans_color = self.text_color
self.button_trans_color = self.color
self.text_trans_color2 = self.text_color
self.button_trans_color2 = self.color
# Opacity
self.opacitya = 255
self.opacityb = 180
self.opacity1 = 255
self.opacity2 = 180
# Fonts
self.basefont = basefont
try:
    self.base_font = pygame.font.Font(self.basefont,
self.textSize)
except:
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
    self.base_font = pygame.font.Font(f"../{self.basefont}",  
self.textSize)  
  
    def draw(self,outline=None,action=None, link=None, colorChange=True,  
newText=None, newX=None, newY=None, mode="normal"):  
        """New X, Y and Text values"""  
        if newText: self.text = newText  
        if newX: self.x = newX  
        if newY: self.y = newY  
  
        """Draws the button. Variable for mouse detection"""  
        mouse = pygame.mouse.get_pos()  
        click = pygame.mouse.get_pressed()  
  
        if self.buttonType == "Rectangle":  
            """If it is a rectangle it will draw it here"""  
            if mode == "normal":  
                if outline:  
                    """draws an outline"""  
                    pygame.draw.rect(self.screen, outline, (self.x-  
2,self.y-2,self.width+4,self.height+4),0,self.curve)  
                    if self.x+self.width > mouse[0] > self.x and  
self.y+self.height > mouse[1] > self.y and colorChange:  
                        self.hover = True  
                        self.stop_start_animation = False  
                        self.time2 = pygame.time.get_ticks()  
                        step = (self.time2 - self.time) / (self.animation_time  
/ self.animation_speed)  
                        """Draws a lighter version of the image"""  
                        self.button_trans_color =  
self.change_color(self.button_trans_color2, self.color2, step)  
                        pygame.draw.rect(self.screen, self.button_trans_color,  
(self.x,self.y,self.width,self.height), 0, self.curve)  
                        if self.text != '':  
                            self.text_trans_color =  
self.change_color(self.text_trans_color2, self.text_color2, step)  
                            text = self.base_font.render(self.text, 1,  
self.text_trans_color)  
                            self.screen.blit(text, (self.x + (self.width/2 -  
text.get_width()/2), self.y + (self.height/2 - text.get_height()/2)))  
  
                            # Clicking the Button  
                            if click[0] == 1:  
                                self.click = True  
  
                            if self.click == True and not click[0]:
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
        self.release = True

    if self.release:
        """If there is an action it is run"""
        if action:
            action()
        if link:
            webbrowser.open(link)
    self.release = False
    self.click = False

else:
    self.time = pygame.time.get_ticks()
    self.hover = False
    self.click = False
    self.release = False
    step = (self.time - self.time2) / (self.animation_time
/ self.animation_speed)
    """A darker version of image when the player isn't
hovering over"""
    if not self.stop_start_animation:
        self.button_trans_color2 =
self.change_color(self.button_trans_color, self.color, step)
        pygame.draw.rect(self.screen,
self.button_trans_color2, (self.x,self.y,self.width,self.height),0,
self.curve)
    else:
        pygame.draw.rect(self.screen, self.color,
(self.x,self.y,self.width,self.height),0, self.curve)
    if self.text != '':
        """Text is blit here"""
        if not self.stop_start_animation:
            self.text_trans_color2 =
self.change_color(self.text_trans_color, self.text_color, step)
            text = self.base_font.render(self.text, 1,
self.text_trans_color2)
        else:
            text = self.base_font.render(self.text, 1,
self.text_color)
            self.screen.blit(text, (self.x + (self.width/2 -
text.get_width()/2), self.y + (self.height/2 -
text.get_height()/2)))

else:
    if mode == "draw":
        self.time2 = pygame.time.get_ticks()
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
        step = (self.time2 - self.time) / (self.animation_time
/ self.animation_speed)
        """Draws a lighter version of the image"""
        self.button_trans_color =
self.change_color(self.button_trans_color2, self.color2, step)
            pygame.draw.rect(self.screen, self.button_trans_color,
(self.x,self.y,self.width,self.height), 0, self.curve)

        if mode == "undraw":
            self.time = pygame.time.get_ticks()
            step = (self.time - self.time2) / (self.animation_time
/ self.animation_speed)
            """A darker version of image when the player isn't
hovering over"""
            self.button_trans_color2 =
self.change_color(self.button_trans_color, self.color, step)
            pygame.draw.rect(self.screen,
self.button_trans_color2, (self.x,self.y,self.width,self.height),0,
self.curve)

        if self.buttonType == "Circle":
            """If it is a circle it will draw it here"""
            if outline:
                """draws an outline"""
                pygame.draw.circle(self.screen, outline, (self.x,
self.y),self.radius+2.5,0)
                """Trig for area"""
                differenceInX = mouse[0] - self.x
                differenceInY = mouse[1] - self.y
                difference = ( differenceInX**2 + differenceInY**2 )**0.5
                """Info on the circle"""
                if difference <= self.radius:
                    self.hover = True
                    self.stop_start_animation = False
                    self.time2 = pygame.time.get_ticks()
                    step = (self.time2 - self.time) / (self.animation_time /
self.animation_speed)
                    self.button_trans_color =
self.change_color(self.button_trans_color2, self.color2, step)
                    """Draws a lighter version of the image"""
                    pygame.draw.circle(self.screen, self.button_trans_color,
(self.x,self.y),self.radius,0)

                    if self.text != '':
                        """Text is blit here"""


```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
        self.text_trans_color =
self.change_color(self.text_trans_color2, self.text_color2, step)
    text = self.base_font.render(self.text, 1,
self.text_trans_color)
        self.screen.blit(text, (self.x - (text.get_width()/2),
self.y - (text.get_height()/2)))

    # Clicking the Button
    if click[0] == 1:
        self.click = True

    if self.click == True and not click[0]:
        self.release = True

    if self.release:
        """If there is an action it is run"""
        if action != None:
            action()
        if link != None:
            webbrowser.open(link)
        self.release = False
        self.click = False
    else:
        self.time = pygame.time.get_ticks()
        step = (self.time - self.time2) / (self.animation_time /
self.animation_speed)
        self.button_trans_color2 =
self.change_color(self.button_trans_color, self.color, step)
        self.hover = False
        self.click = False
        self.release = False
        """A darker version of image when the player isn't
hovering over"""
        pygame.draw.circle(self.screen, self.button_trans_color2,
(self.x, self.y), self.radius, 0)

        if self.text != '':
            """Text is blit here"""
            self.text_trans_color2 =
self.change_color(self.text_trans_color, self.text_color, step)
            text = self.base_font.render(self.text, 1,
self.text_trans_color2)
            self.screen.blit(text, (self.x - (text.get_width()/2),
self.y - (text.get_height()/2)))

    if self.buttonType == "Image":
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
if outline:  
    """draws an outline"""  
    pygame.draw.rect(self.screen, outline, (self.x-3,self.y-  
3,self.width+6,self.height+6),0,self.curve)  
  
    self.image.draw(self.x,self.y)  
  
    if self.x+self.width > mouse[0] > self.x and  
    self.y+self.height > mouse[1] > self.y:  
        self.time2 = pygame.time.get_ticks()  
        step = (self.time2 - self.time) / (self.animation_time /  
self.animation_speed)  
        self.opacity2 = self.change_opacity(self.opacity1,  
self.opacityb, step)  
        self.image.load_image.set_alpha(self.opacity2)  
        self.hover = True  
        # Clicking the Button  
        if click[0] == 1:  
            self.click = True  
  
        if self.click == True and not click[0]:  
            self.release = True  
  
        if self.release:  
            """If there is an action it is run"""  
            if action != None:  
                action()  
            if link != None:  
                webbrowser.open(link)  
            self.release = False  
            self.click = False  
        else:  
            self.time = pygame.time.get_ticks()  
            self.hover = False  
            self.click = False  
            self.release = False  
            step = (self.time - self.time2) / (self.animation_time /  
self.animation_speed)  
            self.opacity1 = self.change_opacity(self.opacity2,  
self.opacitya, step)  
            self.image.load_image.set_alpha(self.opacity1)  
  
def is_hovered(self):  
    return self.hover  
  
def is_clicking(self):
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
    return self.click

def change_color(self, original_color, new_color, step_multiplier=0):
    colors = []
    new_color_list = []
    for rgb1, rgb2 in zip(original_color, new_color):
        difference = rgb2 - rgb1
        colors.append(difference)

    color_step = [difference / self.num_frames for difference in
colors]

    for og_color_rgb, new_color_rgb, step in zip(original_color,
new_color, color_step):
        if og_color_rgb > new_color_rgb:
            if int(og_color_rgb + step_multiplier*step) <
new_color_rgb:
                new_color_list.append(new_color_rgb)
            else:
                new_color_list.append(int(og_color_rgb +
step_multiplier*step))
        else:
            if int(og_color_rgb + step_multiplier*step) >
new_color_rgb:
                new_color_list.append(new_color_rgb)
            else:
                new_color_list.append(int(og_color_rgb +
step_multiplier*step))

    return tuple(new_color_list)

def change_opacity(self, original_opacity, new_opacity,
step_multiplier=0):
    new_opacity_return = None
    difference = new_opacity - original_opacity
    opacity_step = difference / self.num_frames
    if original_opacity > new_opacity:
        if int(original_opacity + opacity_step*step_multiplier) <
new_opacity:
            new_opacity_return = new_opacity
        else:
            new_opacity_return = int(original_opacity +
opacity_step*step_multiplier)
    else:
        if int(original_opacity + opacity_step*step_multiplier) >
new_opacity:
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
        new_opacity_return = new_opacity
    else:
        new_opacity_return = int(original_opacity +
opacity_step*step_multiplier)

    return new_opacity_return
```

Player:

```
import pygame
from Scripts.logger import *
from Scripts.support import import_folder

logger.debug("RealDL Player Code.")

class Player(pygame.sprite.Sprite):
    def __init__(self, pos, image, groups, obstacle_sprites, username, id,
movement="WASD", offense="Space"):
        # Setting up player
        try:
            super().__init__(groups)
            self.screen = pygame.display.get_surface()
            try: self.image = pygame.image.load(image).convert_alpha()
            except: self.image =
pygame.image.load(f"../{image}").convert_alpha()
            self.image_name = f"../{image}"
            self.rect = self.image.get_rect(topleft = pos)
            self.hitbox = self.rect.inflate(0,-26)
            self.movement = movement
            self.offense = offense
            self.key_binds = {
                'move_up': pygame.K_w if self.movement == "WASD" else
pygame.K_UP,
                'move_down': pygame.K_s if self.movement == "WASD" else
pygame.K_DOWN,
                'move_left': pygame.K_a if self.movement == "WASD" else
pygame.K_LEFT,
                'move_right': pygame.K_d if self.movement == "WASD" else
pygame.K_RIGHT,
                'attack': pygame.K_SPACE if self.offense == "Space" else
pygame.K_LCTRL if self.offense == "L-Ctrl" else pygame.K_RCTRL
            }

            # graphics setup
            self.import_player_assets()
            self.status = 'down'
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
    self.frame_index = 0
    self.animation_speed = 0.15

    # Movement
    self.direction = pygame.math.Vector2()
    self.speed = 5
    self.attacking = False
    self.attack_cooldown = 400
    self.attack_time = None
    self.paused = False

    #Other stats
    self.username = username
    self.id = id
    self.basefont = "Fonts/Orbitron-Medium.ttf"
    self.textSize = 20
    try:
        self.font = pygame.font.Font(self.basefont, self.textSize)
    except:
        self.font = pygame.font.Font(f"../{self.basefont}",
self.textSize)
    self.sprite_type = "player"

    self.obstacle_sprites = obstacle_sprites
except FileNotFoundError as FileNotFoundError:
    logger.error(f"Failed to create Player Class: {FileNotFoundError}")

def import_player_assets(self):
    character_path = 'Graphics/Game/player/'
    self.animations = {'up': [], 'down': [], 'left': [], 'right': [],
                      'right_idle':[], 'left_idle':[], 'up_idle':[], 'down_idle':[],
                      'right_attack':[], 'left_attack':[], 'up_attack':[], 'down_attack':[]}
}

    self.animation_images = {'up': [], 'down': [], 'left': [], 'right': [],
                            'right_idle':[], 'left_idle':[], 'up_idle':[], 'down_idle':[],
                            'right_attack':[], 'left_attack':[], 'up_attack':[], 'down_attack':[]}
}

    for animation in self.animations.keys():
        full_path = character_path + animation
        self.animations[animation], self.animation_images[animation] =
import_folder(full_path)

def input(self):
    keys = pygame.key.get_pressed()
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
if not self.paused:

    if keys[self.key_binds['attack']]:
        self.attacking = True
        self.attack_time = pygame.time.get_ticks()
        logger.info('attack')

def cooldowns(self):
    current_time = pygame.time.get_ticks()

    if self.attacking:
        if current_time - self.attack_time >= self.attack_cooldown:
            self.attacking = False

def get_image_name(self):
    image_to_return = self.image_name
    if "../" in image_to_return:
        image_to_return = self.image_name.replace("../", "")
    return image_to_return

def animate(self):
    if not self.paused:
        animation = self.animations[self.status]
        image = self.animation_images[self.status]

        # loop over the frame index
        self.frame_index += self.animation_speed
        if self.frame_index >= len(animation):
            self.frame_index = 0

        # set the image
        self.image = animation[int(self.frame_index)]
        self.image_name = image[int(self.frame_index)]
        self.rect = self.image.get_rect(center = self.hitbox.center)

def draw(self, offset_pos):
    # Draw the player.
    text_surface = self.font.render(self.username, True, (240,240,240))
    text_rect = text_surface.get_rect()
    username_pos = (offset_pos[0] + self.rect.width // 2 -
text_surface.get_width() // 2, offset_pos[1] - 33)
    # Define the dimensions and position of the black box
    box_width = text_rect.width + 6 # Adjust the width as needed
    box_height = text_rect.height + 6 # Adjust the height as needed
    box_pos = (username_pos[0]-3, username_pos[1]-3) # Adjust the
position as needed
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
# Draw the black box
pygame.draw.rect(self.screen, (200,200,200), ((box_pos[0]-
2,box_pos[1]-2), (box_width+4, box_height+4)),0,10)
    pygame.draw.rect(self.screen, (27,31,35), (box_pos, (box_width,
box_height)),0,10)
    self.screen.blit(self.image, offset_pos)
    self.screen.blit(text_surface, username_pos)

def update(self):
    self.input()
    self.cooldowns()
    self.get_status()
    self.animate()
    self.move(self.speed)
```

Support:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
from csv import reader
from os import walk, path
import pygame

def import_csv_layout(path):
    terrain_map = []
    with open(path) as level_map:
        layout = reader(level_map, delimiter = ',')
        for row in layout:
            terrain_map.append(list(row))
    return terrain_map

def import_folder(folder_path):
    surface_list = []
    image_list = []
    send_back = False

    while not send_back:
        for root, dirs, img_files in walk(folder_path):
            for image in img_files:
                full_path = path.join(root, image)
                try:
                    image_surf = pygame.image.load(full_path).convert_alpha()
                except pygame.error:
                    alt_path = path.join('..', folder_path, image)
                    image_surf = pygame.image.load(alt_path).convert_alpha()
                surface_list.append(image_surf)
                image_list.append(full_path)
        if surface_list and image_list:
            send_back = True
        else:
            if not folder_path.startswith('../'):
                folder_path = '../' + folder_path
            else:
                send_back = True

    return surface_list, image_list
```

UI:

```
from Scripts.functions import *
from Scripts.logger import *
import pygame

logger.debug("RealDL UI Code.")

class UI:
    def __init__(self, custom_mouse, quit_function):
        # Setup Pygame Variables
        self.screen = pygame.display.get_surface()
        info = pygame.display.Info()
        self.screen_width = info.current_w
        self.screen_height = info.current_h
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
self.DEFAULT_WIDTH = 1920
self.DEFAULT_HEIGHT = 1080
self.width_ratio = self.screen_width / self.DEFAULT_WIDTH
self.height_ratio = self.screen_height / self.DEFAULT_HEIGHT
self.custom_mouse = custom_mouse
self.draw_ui = False
self.quit_function = quit_function

# Constants setup
self.BIG_TEXT_SIZE = 50
self.BASE_TEXT_SIZE = 15
self.TEXT_HEIGHT = 20
self.IMAGE_WIDTH = 64
self.IMAGE_HEIGHT = 64
self.IMAGE_PADDING = 20
self.BUTTON_PADDING = 85
self.CURVE = 10
self.THICKNESS = 2
self.BASE_BUTTON_WIDTH = 250
self.BASE_BUTTON_HEIGHT = 70

# Variable setup
self.image_width = int(self.IMAGE_WIDTH*self.width_ratio)
self.image_height = int(self.IMAGE_HEIGHT*self.height_ratio)
self.image_padding = int(self.IMAGE_PADDING*self.width_ratio)
self.button_padding = int(self.BUTTON_PADDING*self.height_ratio)
self.text_height = int(self.TEXT_HEIGHT*self.height_ratio)
self.base_text_size = int(self.BASE_TEXT_SIZE*self.height_ratio)
self.big_text_size = int(self.BIG_TEXT_SIZE*self.height_ratio)
self.curve = int(self.CURVE*self.height_ratio)
self.thickness = int(self.THICKNESS*self.height_ratio)
self.base_button_width = int(self.BASE_BUTTON_WIDTH*self.width_ratio)
self.base_button_height =
int(self.BASE_BUTTON_HEIGHT*self.height_ratio)

# UI Board
self.border = Button((27,31,35), (27,31,35), self.screen_width/2,
self.screen_height/2, "Fonts/Orbitron-Regular.ttf", (27,31,35), (27,31,35),
self.screen_width*0.7, self.screen_height*0.7,'', 'Rectangle', None,
self.big_text_size, int(self.curve*1.5))
self.join_boarder = Button((27,31,35), (27,31,35),
(self.screen_width/2), (self.screen_height/2), "Fonts/Orbitron-Regular.ttf",
(240,240,240), (136,173,227), self.base_button_width*3,
self.base_button_height*6,'', 'Rectangle', None, self.big_text_size,
self.curve)
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
    self.info = Text(self.text_height, "Fonts/Orbitron-  
Regular.ttf", (240, 240, 240), None, None, None, int(self.base_text_size*1.7))  
    self.info2 = Text(self.text_height, "Fonts/Orbitron-  
Regular.ttf", (240, 240, 240), None, None, None, int(self.base_text_size*1.7))  
    self.quit = Button((174, 39, 96), (27, 31, 35), self.screen_width/2,  
self.screen_height*0.78, "Fonts/Orbitron-Medium.ttf", (27, 31, 35), (174, 39,  
96), self.base_button_width, self.base_button_height, 'Quit', 'Rectangle', None,  
int(self.big_text_size/1.3), self.curve)  
    self.bullet_assault = Text(self.text_height, "Fonts/Orbitron-  
Bold.ttf", (240, 240, 240), None, None, None, int(self.base_text_size*3.5))  
    self.continue_btn = Button((39, 174, 96), (27, 31, 35),  
(self.screen_width/2), (self.screen_height/2)*1.445-self.button_padding,  
"Fonts/Orbitron-Medium.ttf", (27, 31, 35), (39, 174, 96),  
self.base_button_width, self.base_button_height, 'Continue', 'Rectangle', None,  
int(self.big_text_size/1.3), self.curve)  
  
def stop_drawing(self):  
    self.draw_ui = False  
  
def draw_menu(self):  
    if self.draw_ui:  
        self.border.draw((27, 31, 35), None, None, False)  
        self.join_boarder.draw((240, 240, 240))  
        self.info.draw("draw", "Game Paused: You have entered the main  
menu.", (self.screen_width/2), (self.screen_height/2)*0.665)  
        self.info2.draw("draw", "Movement is currently  
disabled.", (self.screen_width/2), (self.screen_height/2)*0.72)  
        self.quit.draw((240, 240, 240), self.quit_function)  
        self.continue_btn.draw((240, 240, 240), self.stop_drawing)  
        self.bullet_assault.draw("draw", "Bullet Assault",  
(self.screen_width/2), (self.screen_height/2)*0.43)  
  
        start_back_hover = self.quit.is_hovered()  
        join_hover = self.continue_btn.is_hovered()  
  
        start_back_click = self.quit.is_clicking()  
        join_click = self.continue_btn.is_clicking()  
  
        if start_back_hover or join_hover:  
            if not start_back_click and not join_click:  
                self.custom_mouse.mode = 1  
            else:  
                self.custom_mouse.mode = 2  
        else:  
            self.custom_mouse.mode = 0
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
# Draw the mouse
self.custom_mouse.draw()
```

Test proof (Tests 2.1 - 2.9):

See MP4 videos

Sprint 3

Aims for this sprint.

In addition, from sprint 2, I will be incorporating a slightly updated UI when it comes to the main menu UI (leaving and joining). I will also include in-game UI.

Updated main menu UI.

The update main menu UI will be almost the same as sprint two visually, however the mechanism for joining and leaving the game are updated and more efficient and less error prone. The difference visually, will be in the settings menu where another key bind is added, this one will be for the magic keys. Another update that I will include in this sprint is a UI feature to leave the game. Users will be able to toggle the main menu screen in the game, however there will be no settings option, but an option to quit the game.

In-game UI

The in-game UI will have a multitude of different elements that visually informs the users of their game status. Firstly, there will be a red bar and a blue bar at the top left-hand corner of the screen. The red bar represents their health, the health is a percentage, meaning that full means they have 100% health. The blue bar will represent the players energy. Energy is again displaying a percentage of their total energy as to not have a long energy bar. In addition, the energy bar only decreases when the player decides to use magic. This magic could be the heal animation or the flame animation. The blue bar shows how much energy the player has and whether it is enough to cast a spell. This system is implemented to prevent the health bar from becoming extremely long. Another feature of the in-game UI will be a “kill count” and a “players alive” count at the top right-hand side of the screen. This will not be working in this sprint, however in a later sprint we will have it working. Again, in the bottom right-hand corner of the screen. Finally, at the bottom left-hand corner, there are two boxes. One box is for the player’s weapon; and the other one is for the player’s magic option. The boxes display which weapon is currently being held. The weapon box will be able to change the weapon by holding the button “q,” and the magic box will be able to change by holding the button “e.” There will be eight different weapons and two different magic abilities. There will be, for the weapons, a sword, a lance, an axe, a rapier, a sai, a revolver, an SMG and pistol (I accidentally spelt the SMG gun as “msg” in my code). Finally, in sprint two, the mouse was visible on the screen. I will still be using the custom mouse however when the user is playing the game, we will hide the mouse and when the user toggles the menu screen, we will then reveal the mouse again. An important thing to mention will be that the UI in the game will not be fully functional yet. The UI will be necessary for the later sprints however as stated before, it will not be fully functional until I start implementing additional features in the server and client to support attacking.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Success Criteria

1. **Update the Main Menu UI:** Allow for the magic keys to be changed by users just like the movement keys and the attack keys.
2. **Improve the functionality when joining and leaving the game:** Implement a better system that can successfully allow users to leave and join efficiently without using inefficient while loops that could cause errors.
3. **Allow the users to have a main menu screen in the game:** The players should be able to toggle between the main menu screen in game by holding escape or by clicking the continue button on screen.
4. **Hide the mouse in game:** The mouse should be hidden in game. The players should not be able to see the mouse in game, only when the main menu screen is toggled.
5. **Have two bars for health and energy:** There should be a red bar for health and a blue bar for energy in the top-left hand corner. The bars are a percentage of their total health and energy. This does not need to be functional yet for this sprint.
6. **There should be two information boxes at the top-right hand corner:** There should be two information boxes. The first information box should show the kill count and the second should show the number of players still alive in the game. This does not need to be functional yet for this sprint.
7. **There should be an xp bar at the bottom-right hand corner of the screen:** This should increase as a player gets a kill. It will be used for upgrading the players stats; however, this is not currently functional yet.
8. **There should be two boxes displaying current weapons and magic animations:** The first box should be for weapons. The boxes will both be able to be toggled, the weapons box should be able to be toggled with the button “q” and the magic box should be able to be toggled with the button “e.” The second box will be for the magic animations. There must be 8 toggle options for the weapons and 2 toggle options for the magic animations.

Justification of sprint 3

Main menu interface

The reason I am developing some small additions to my user interface is because sprint 2 only has options for player movement and player weapon attacking. We need another option for magic attacks because the animations are a key part of my game. Another addition that will be added was the ability to quit the game via a new menu. This is needed to create a visually pleasing, yet functional way to quit the game.

Server-Client connection

I am going to redesign the functionality and methodology that is used to connect a client from the main menu screen to the game screen. I will only be changing the client side and not the server side as the connection is the same. We are only changing how the player can connect and disconnect. The first thing that will be changed will be the while loop. The while loop that will be creating a new main menu each time will be removed. Secondly, the main menu class will have its own folder for organisation. Another thing that will be changed will be in the client side. Firstly, within the client class, there will be another function that runs the main menu side first, not the actual game. Then what will happen is when the client connects we will have another function to connect the user to the game, and when the client leaves there will be a further function to determine that the player wants to leave, then we will close the connection between the server, and then we will reset the important values in the client so that if the player joins again they will have the correct settings.

Game UI

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Finally, the game UI will be created using a separate class and in a new file. This is necessary for organisation of code, but further creating game UI will become vital in future sprints when users will need the ability to see how many kills, they, how many players are left in the game and what weapon or magic option they have chosen. Game UI will help the player understand the key information in the game. How I will create the game UI is by creating multiple different buttons that can all be used to display information for the player. This will then be run in the client by creating an instance of the UI class. The UI will always be displayed during the game.

Pseudo Code

The Pseudo Code is in the Appendix.

Python Development

Program Code

Code is in the appendix.

Areas of complexity

Client Connection

```
def __init__(self):
    try:
        Config.__init__(self)
        self.gameMenu = MainMenu()
    except:
        logger.error("Error, couldn't initialize the main menu.")
        self.close()
```

Now, here in sprint 3, I have decided to inside call the main menu UI instead. Here we call run, instead of main. This is because when we run the main menu UI, and the user then joins the game. We will be able to receive the user dictionary containing all the necessary information about the players custom settings options. Once we have that, we can then initialise the client. This then works as normal because instead of calling the UI outside of the client we call it inside of the client, thus the user options are effectively saved.

```
def run(self):
    self.gameMenu.run()
    user_dict = self.gameMenu.start_game()
    self.initialize_client(user_dict)
    self.main()

if __name__ == "__main__":
    try:
        client = Client()
        client.run()
    except:
        logger.error("Couldn't run main menu or client.")
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

When the user decides to leave the game, we disconnect them like usual however, we close the main loop that runs the game, we also reset the player information and the information about other players. As shown below, then we reset the information about the UI.

```
def restart_menu(self):
    self.loop = True
    self.home()

def close(self):
    try:
        self.network.close()
    except:
        logger.exception("No server-client connection.")
    self.running = False
    self.player = None
    self.players = None
    self.gameMenu.restart_menu()
    self.run()
```

The two images show how the main menu screen is ran again as the UI main menu class object is reset to the original position and as we run “self.run()” again. This is much more efficient because firstly, main menu UI is in a separate class, another reason is that the while loop has been removed. The while loop in sprint 2 was a liability because it required creating new class instances of both the main menu and the client which is bad for performance and reliability as if the while loop broke, the game could effectively break.

Exit menu system.

In the game, when the user presses the escape key, we will show a menu system where the user can quit the game. The code below is used to draw objects they were defined above in the UI class. If the menu has been toggled, we will then draw the menu on screen. While the menu is on screen, players will not be able to move. This will then allow users to either quit the game and go back to the main menu or go back to playing the game.

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

def draw_menu(self):
    if self.draw_ui:
        self.border.draw((27, 31, 35), None, None, False)
        self.join_boarder.draw((240, 240, 240))
        self.info.draw("draw", "Game Paused: You have entered the main menu.", (self.screen_width / 2), (self.screen_height / 2) * 0.665)
        self.info2.draw("draw", "Movement is currently disabled.", (self.screen_width / 2), (self.screen_height / 2) * 0.72)
        self.quit.draw((240, 240, 240), self.quit_function)
        self.continue_btn.draw((240, 240, 240), self.stop_drawing)
        self.bullet_assault.draw("draw", "Bullet Assault", (self.screen_width / 2), (self.screen_height / 2) * 0.43)

        start_back_hover = self.quit.is_hovered()
        join_hover = self.continue_btn.is_hovered()

        start_back_click = self.quit.is_clicking()
        join_click = self.continue_btn.is_clicking()

        if start_back_hover or join_hover:
            if not start_back_click and not join_click:
                self.custom_mouse.mode = 1
            else:
                self.custom_mouse.mode = 2
        else:
            self.custom_mouse.mode = 0

    # Draw the mouse
    self.custom_mouse.draw()
  
```

In Game UI

The last difficult part of Sprint 3 was the in-Game UI. This is because I needed to create unique and individual buttons for each element. This was simple as I ran all the function from one function here:

```

def display(self, player):
    self.show_bar(player.health, player.stats['health'], self.health_bar_rect, self.HEALTH_COLOR)
    self.show_bar(player.energy, player.stats['energy'], self.energy_bar_rect, self.ENERGY_COLOR)

    self.show_exp(player.exp)
    self.show_info()

    self.weapon_overlay(player.weapon_index, not player.can_switch_weapon)
    self.magic_overlay(player.magic_index, not player.can_switch_magic)

def redraw_window(self, all_players_dict):
    try:
        self.update_players(all_players_dict)
        self.level.run(self.player)
        self.ui.display(self.player)
        self.ui.draw_menu()
        debug(f"Position: ({self.player.rect.x}, {self.player.rect.y})", self.WIDTH/2, 20)
        if self.ui.draw_ui: self.player.paused = True
        else: self.player.paused = False
        pygame.display.update()
        self.clock.tick(self.FPS)
    except:
        logger.error("Player has left the game.")
        self.close()
  
```

The two highlighted lines run the UI for the Game.

During iterative development

No.	Test Question	Expected Result	Actual Result	Proof
3.1	Main Menu UI Magic Key Change Test	Magic keys can be changed successfully in the UI.	The magic was changed to the selected key.	Figure 3.1

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

3.2	Joining and Leaving Functionality Test	Efficient joining and leaving without errors.	Users were able to join and leave without any errors as expected.	Figure 3.2
3.3	Main Menu Screen Toggle Test	Smooth transition between main menu and game.	This worked as expected.	Figure 3.3
3.4	Mouse Visibility Test	Mouse is hidden during gameplay; becomes visible in menu.	The mouse was invisible playing the game.	Figure 3.4
3.5	Health and Energy Bar Display Test	Bars in top-left corner representing percentage values.	The Game UI successfully displayed health and energy bars.	Figure 3.5

Stakeholders

Question	Roland	James
What do you think of the new connection system?	It is good. Much better than the old system.	Yes, it is much better.
Do you like the disconnect menu?	Yes. I like having a way to disconnect or pause the game.	Yep. It looks good.
Do you like the new UI?	Yes. The new UI is helpful.	Yes.
Do you like the animations for the weapon/magic toggles?	They look all right. I am happy with the animations.	Yes, they are fine.
Do you like the ability to change your settings?	Yes, this is helpful. Changing your settings is needed.	Yes, it is especially useful.
Do you like that your settings are saved when you disconnect?	That is fine.	That is good with me.

Evaluation

No.		Achieved
1	Update the Main Menu UI: Allow for the magic keys to be changed by users just like the movement keys and the attack keys.	Yes
2	Improve the functionality when joining and leaving the game: Implement a better system that can successfully allow users to leave and join efficiently without using inefficient while loops that could cause errors.	Yes
3	Allow the users to have a main menu screen in the game: The players should be able to toggle between the main menu screen in game by holding escape or by clicking the continue button on screen.	Yes
4	Hide the mouse in game: The mouse should be hidden in game. The players should not be able to see the mouse in game, only when the main menu screen is toggled.	Yes

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

5	Have two bars for health and energy: There should be a red bar for health and a blue bar for energy in the top-left hand corner. The bars are a percentage of their total health and energy. This does not need to be functional yet for this sprint.	Yes
6	There should be two information boxes at the top-right hand corner: There should be two information boxes. The first information box should show the kill count and the second should show the number of players still alive in the game. This does not need to be functional yet for this sprint.	Yes
7	There should be an xp bar at the bottom-right hand corner of the screen: This should increase as a player gets a kill. It will be used for upgrading the players stats; however, this is not currently functional yet.	Yes
8	There should be two boxes displaying current weapons and magic animations: The first box should be for weapons. The boxes will both be able to be toggled, the weapons box should be able to be toggled with the button “q” and the magic box should be able to be toggled with the button “e.” The second box will be for the magic animations. There must be 8 toggle options for the weapons and 2 toggle options for the magic animations.	Yes

In Sprint 3, my focus was on enhancing the UI and the way clients connect to the server. This has been a positive success. As shown from my stakeholder questions, they appreciated the flexibility introduced in the Main Menu UI, allowing them to seamlessly customise magic keys and the other keys as well. The improvement in joining and leaving functionality, eliminating inefficiencies, was successful as it provided users with a smoother experience during transitions without encountering errors.

The addition of a toggleable main menu screen and the effective hiding of the mouse during gameplay were noted as valuable improvements, enhancing the overall gaming immersion. The implementation of health and energy bars, information boxes for kill count and players alive, and the introduction of an XP bar were well-received by my stakeholders which signals progress in establishing a comprehensive and aesthetic game UI.

Feedback from stakeholders affirmed satisfaction with the new connection system, disconnect menu, UI, and animations for weapon/magic toggles. The retention of settings after disconnecting was recognised as a user-friendly feature which helps improve quality of life of the game. Sprint 3 has effectively addressed planned objectives, with positive stakeholder responses indicating a positive implementation and setting a solid foundation for subsequent development phases.

Next Sprint

Next sprint, I hope to add the animations for weapons and magic animations. This is so that I can get ahead for the attacking mechanics already having the attacking elements working so that hopefully implementing in the methods to attack other players can be effective in future sprints. This will be quite a large sprint because I will need to find mechanics to send over all the meta data effectively and successfully from each weapon/ magic class, then be able to receive it from the server and

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

effectively draw each weapon. I will start out next sprint but making weapon and magic animations solely client based then work on transferring the necessary data needed, then I will lastly focus on rendering another weapon or magic class with that data provided by the server.

Copyright

In this sprint again I am copying some code from Clear Code to effectively create my game. I am using the same UI as Clear Code as the game he creates on his YouTube channel is visually very similar to my game. I have used some elements of his code, but I have also adapted elements of his code. For my collisions I have used elements of how he creates collisions. Lastly, I have used his style of programming when creating my project. For example, having a settings file, a file to debug and file for file manipulation. This has overall made my programming better.

Appendix

Pseudo Code:

Client:

```
class Client inherits Config
    ...
    public procedure new()
        try
            Config._init_()
            gameMenu = new MainMenu()
        except
            logger.error("Error, couldn't initialize the main menu.")
            close()
        endtry
    endprocedure

    public procedure initialize_client(user_dict)
        logger.info("Client Connecting to Server")
        try
            initialize_pygame(user_dict)
            initialize_network()
        except
            logger.error("Error, couldn't initialize the client.")
            close()
        endtry
    endprocedure

    public procedure close()
        try
            network.close()
        except
            logger.exception("No server-client connection.")
        endtry
    endtry
    running = False
    player = None
    players = None
    gameMenu.restart_menu()
    run()
endprocedure
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
public procedure quit()
    running = False
    pygame.quit()
    sys.exit()
endprocedure

public procedure run()
    gameMenu.run()
    user_dict = new gameMenu.start_game()
    initialize_client(user_dict)
    main()
endprocedure

if __name__ == "__main__":
    try:
        client = new Client()
        client.run()
    except:
        logger.error("Couldn't run main menu OR client.")
    endtry
endif

main menu:
from Scripts.logger import *
try:
    import pygame, sys
    from Scripts.settings import Config
    from Scripts.encryption import *
    from Scripts.debug import debug
    from Scripts.functions import *

except:
    logger.critical("You do NOT have all the modules installed. Please install Pygame.")
endtry
logger.info("RealDL - Main Menu Code")
class MainMenu:
    private custom_mouse.mode
    private youtube_button.draw((27,31,35),None,"https://www.youtube.com/watch?v= new dQw4w9WgXcQ")
    private starting_dict
    private volume_button.x
    private music_box.color
    private sound_box.color
    private music_change
    private sound_change
    private volume
    private volume_ratio
    private max_vol_x
    private min_vol_x
    private keys
    private magic_keys
    private attack_keys
    private settings_screen
    private main_menu_pages
    private name_box_text
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
private name_text_box
private server_ip_text
private join_btn
private join_boarder
private ip_text_box
private join_message
private bullet_assault
private start_back
private start_board
private audio_box
private music_box
private sound_box
private music_text
private sound_text
private audio_text
private volume_indicator
private volume_button
private volume_line
private volume_bar
private volume_text
private box_audio
private magic_btn
private magic_text
private attack_btn
private attack_text
private movement
private key_birds
private box_control
private underline2
private underline
private audio_settings
private control_settings
private back
private options_board
private current_x
private quit_button
private options_button
private start_button
private custom_mouse
private youtube_button
private github_button
private github_name
private youtube_name
private sunset_image
private icon
private base_button_height
private base_button_width
private thickness
private curve
private big_text_size
private base_text_size
private text_height
private button_padding
private image_padding
private image_height
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
private image_width
private BASE_BUTTON_HEIGHT
private BASE_BUTTON_WIDTH
private THICKNESS
private CURVE
private BUTTON_PADDING
private IMAGE_PADDING
private IMAGE_HEIGHT
private IMAGE_WIDTH
private TEXT_HEIGHT
private BASE_TEXT_SIZE
private BIG_TEXT_SIZE
private screen
private height_ratio
private width_ratio
private DEFAULT_HEIGHT
private DEFAULT_WIDTH
private settings
private loop
private FPS
private clock

public procedure new()
// Pygame/ Game setup
pygame.display.set_caption('Bullet Assault')
clock = new pygame.time.Clock()
FPS = 60
loop = True
settings = new Config()
DEFAULT_WIDTH = 1920
DEFAULT_HEIGHT = 1080
width_ratio = settings.WIDTH / DEFAULT_WIDTH
height_ratio = settings.HEIGHT / DEFAULT_HEIGHT
screen = new pygame.display.set_mode((settings.WIDTH, settings.HEIGHT))
// Constants setup
BIG_TEXT_SIZE = 50
BASE_TEXT_SIZE = 15
TEXT_HEIGHT = 20
IMAGE_WIDTH = 64
IMAGE_HEIGHT = 64
IMAGE_PADDING = 20
BUTTON_PADDING = 85
CURVE = 10
THICKNESS = 2
BASE_BUTTON_WIDTH = 250
BASE_BUTTON_HEIGHT = 70
// Variable setup
image_width = new int(IMAGE_WIDTH*width_ratio)
image_height = new int(IMAGE_HEIGHT*height_ratio)
image_padding = new int(IMAGE_PADDING*width_ratio)
button_padding = new int(BUTTON_PADDING*height_ratio)
text_height = new int(TEXT_HEIGHT*height_ratio)
base_text_size = new int(BASE_TEXT_SIZE*height_ratio)
big_text_size = new int(BIG_TEXT_SIZE*height_ratio)
curve = new int(CURVE*height_ratio)
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
thickness = new int(THICKNESS*height_ratio)
base_button_width = new int(BASE_BUTTON_WIDTH*width_ratio)
base_button_height = new int(BASE_BUTTON_HEIGHT*height_ratio)
// Images
icon = new Images("Graphics/MainMenu/General/icon.png")
sunset_image = new Images("Graphics/MainMenu/General/bg.png")
icon.display_icon()
// Setting up Objects for home
youtube_name = new Text(text_height, "Graphics/Fonts/Orbitron-Medium.ttf", (27,31,35), None, None, None, None,
base_text_size)
github_name = new Text(text_height, "Graphics/Fonts/Orbitron-Medium.ttf", (27,31,35), None, None, None, None,
base_text_size)
github_button = new Button((27,31,35), (27,31,35), settings.WIDTH-(image_width/2)-
image_padding,(image_width/2)+image_padding, "Graphics/Fonts/Orbitron-Regular.ttf", (27,31,35), (27,31,35),
image_width, image_height,'Image', None, big_text_size, curve, 'Graphics/MainMenu/Buttons/github.png')
youtube_button = new Button((27,31,35), (27,31,35), settings.WIDTH-(image_width*2)-
(image_padding/4),(image_width/2)+image_padding, "Graphics/Fonts/Orbitron-Regular.ttf", (27,31,35), (27,31,35),
image_width, image_height,'Image', None, big_text_size, curve, 'Graphics/MainMenu/Buttons/youtube.png')
custom_mouse = new Mouse("Graphics/MainMenu/Mouse/mouse1.png",
"Graphics/MainMenu/Mouse/mouse2.png", "Graphics/MainMenu/Mouse/mouse3.png")
start_button = new Button((39, 174, 96), (240,240,240), settings.WIDTH/2, (settings.HEIGHT/2)-
button_padding, "Graphics/Fonts/Orbitron-Medium.ttf", (240,240,240), (39, 174, 96), base_button_width,
base_button_height,'Start','Rectangle', None, int(big_text_size/1.3), curve)
options_button = new Button((39, 96, 174), (240,240,240), settings.WIDTH/2, settings.HEIGHT/2,
"Graphics/Fonts/Orbitron-Medium.ttf", (240,240,240), (39, 96, 174), base_button_width,
base_button_height,'Options','Rectangle', None, int(big_text_size/1.3), curve)
quit_button = new Button((174, 39, 96), (240,240,240), settings.WIDTH/2,
(settings.HEIGHT/2)+button_padding, "Graphics/Fonts/Orbitron-Medium.ttf", (240,240,240), (174, 39, 96),
base_button_width, base_button_height,'Quit','Rectangle', None, int(big_text_size/1.3), curve)
current_x = 0
// Options buttons
options_board = new Button((27,31,35), (27,31,35), settings.WIDTH/2, settings.HEIGHT/2,
"Graphics/Fonts/Orbitron-Regular.ttf", (27,31,35), (27,31,35), settings.WIDTH*0.7,
settings.HEIGHT*0.7,'Rectangle', None, big_text_size, int(curve*1.5))
back = new Button((174, 39, 96), (27,31,35), settings.WIDTH/2, settings.HEIGHT*0.78,
"Graphics/Fonts/Orbitron-Medium.ttf", (27,31,35), (174, 39, 96), base_button_width,
base_button_height,'Back','Rectangle', None, int(big_text_size/1.3), curve)
control_settings = new Text(text_height, "Graphics/Fonts/Orbitron-Bold.ttf", (240,240,240), None, None,
None, int(base_text_size*2))
audio_settings = new Text(text_height, "Graphics/Fonts/Orbitron-Bold.ttf", (240,240,240), None, None, None,
int(base_text_size*2))
underline = new Button((27,31,35), (240,240,240), (settings.WIDTH/2)*0.70,
(settings.HEIGHT/2)*0.46,"Graphics/Fonts/Orbitron-Regular.ttf", (240,240,240), (240,240,240), 275*width_ratio,
3.5*height_ratio,'Rectangle', None, big_text_size, curve)
underline2 = new Button((27,31,35), (240,240,240), (settings.WIDTH/2)*1.3,
(settings.HEIGHT/2)*0.46,"Graphics/Fonts/Orbitron-Regular.ttf", (240,240,240), (240,240,240), 245*width_ratio,
3.5*height_ratio,'Rectangle', None, big_text_size, curve)
// Control Settings
box_control = new Button((27,31,35), (27,31,35), (settings.WIDTH/2)*0.70, (settings.HEIGHT/2)*0.96,
"Graphics/Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), base_button_width*2,
base_button_height*7,'Rectangle', None, big_text_size, curve)
key_binds = new Button((27,31,35), (27,31,35), (settings.WIDTH/2)*0.70, (settings.HEIGHT/2)*0.685,
"Graphics/Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), base_button_width,
base_button_height,'WASD','Rectangle', None, int(big_text_size/1.5), curve)
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
movement = new Text(text_height, "Graphics/Fonts/Orbitron-Medium.ttf", (240,240,240), None, None, None, int(base_text_size*2))  
attack_text = new Text(text_height, "Graphics/Fonts/Orbitron-Medium.ttf", (240,240,240), None, None, None, int(base_text_size*2))  
attack_btn = new Button((27,31,35), (27,31,35), (settings.WIDTH/2)*0.70, (settings.HEIGHT/2)*0.935, "Graphics/Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), base_button_width, base_button_height,'Space','Rectangle', None, int(big_text_size/1.5), curve)  
magic_text = new Text(text_height, "Graphics/Fonts/Orbitron-Medium.ttf", (240,240,240), None, None, None, int(base_text_size*2))  
magic_btn = new Button((27,31,35), (27,31,35), (settings.WIDTH/2)*0.70, (settings.HEIGHT/2)*1.185, "Graphics/Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), base_button_width, base_button_height,'Space','Rectangle', None, int(big_text_size/1.5), curve)  
// Audio Settings  
box_audio = new Button((27,31,35), (27,31,35), (settings.WIDTH/2)*1.3, (settings.HEIGHT/2)*0.96, "Graphics/Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), base_button_width*2, base_button_height*7,'Rectangle', None, big_text_size, curve)  
volume_text = new Text(text_height, "Graphics/Fonts/Orbitron-Medium.ttf", (240,240,240), None, None, None, int(base_text_size*2))  
volume_bar = new Button((27,31,35), (27,31,35), (settings.WIDTH/2)*1.3, (settings.HEIGHT/2)*0.685, "Graphics/Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), base_button_width*1.5, base_button_height*1,'Rectangle', None, big_text_size, curve)  
volume_line = new Button((27,31,35), (27,31,35), (settings.WIDTH/2)*1.25, (settings.HEIGHT/2)*0.685, "Graphics/Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), base_button_width, base_button_height*0.36,'Rectangle', None, big_text_size, curve)  
volume_button = new Button((42, 109, 201), (60, 122, 207), (settings.WIDTH/2)*1.3, (settings.HEIGHT/2)*0.685, "Graphics/Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), base_button_height*0.3, base_button_height*0.3,'Rectangle', None, big_text_size, int(curve/2))  
volume_indicator = new Text(text_height, "Graphics/Fonts/Orbitron-Regular.ttf", (240,240,240), None, None, None, int(base_text_size*2))  
audio_text = new Text(text_height, "Graphics/Fonts/Orbitron-Medium.ttf", (240,240,240), None, None, None, int(base_text_size*2))  
sound_text = new Text(text_height, "Graphics/Fonts/Orbitron-Medium.ttf", (240,240,240), None, None, None, int(base_text_size*2))  
music_text = new Text(text_height, "Graphics/Fonts/Orbitron-Medium.ttf", (240,240,240), None, None, None, int(base_text_size*2))  
sound_box = new Button((27,31,35), (39,174,96), (settings.WIDTH/2)*1.45, (settings.HEIGHT/2)*0.97, "Graphics/Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), base_button_height*0.4, base_button_height*0.4,'Rectangle', None, big_text_size, curve)  
music_box = new Button((27,31,35), (39,174,96), (settings.WIDTH/2)*1.45, (settings.HEIGHT/2)*1.08, "Graphics/Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), base_button_height*0.4, base_button_height*0.4,'Rectangle', None, big_text_size, curve)  
audio_box = new Button((27,31,35), (27,31,35), (settings.WIDTH/2)*1.3, (settings.HEIGHT/2)*1.03, "Graphics/Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), base_button_width*1.5, base_button_height*2.5,'Rectangle', None, big_text_size, curve)  
// Start  
start_board = new Button((27,31,35), (27,31,35), settings.WIDTH/2, settings.HEIGHT/2, "Graphics/Fonts/Orbitron-Regular.ttf", (27,31,35), (27,31,35), settings.WIDTH*0.7, settings.HEIGHT*0.7,'Rectangle', None, big_text_size, int(curve*1.5))  
start_back = new Button((174, 39, 96), (27,31,35), settings.WIDTH/2, settings.HEIGHT*0.78, "Graphics/Fonts/Orbitron-Medium.ttf", (27,31,35), (174, 39, 96), base_button_width, base_button_height,'Back','Rectangle', None, int(big_text_size/1.3), curve)  
bullet_assault = new Text(text_height, "Graphics/Fonts/Orbitron-Bold.ttf", (240,240,240), None, None, None, int(base_text_size*3.5))  
join_message = new Text(text_height, "Graphics/Fonts/Orbitron-Regular.ttf", (240,240,240), None, None, None, int(base_text_size*1.7))
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
ip_text_box = new TextBox(base_button_width*2.5, base_button_height*1.5, (settings.WIDTH/2),  
(settings.HEIGHT/2)*0.99, (240,240,240), (200,200,200),"Graphics/Fonts/Orbitron-  
Regular.ttf",curve,thickness,base_button_width*2.5)  
join_boarder = new Button((27,31,35), (27,31,35), (settings.WIDTH/2), (settings.HEIGHT/2),  
"Graphics/Fonts/Orbitron-Regular.ttf", (240,240,240), (136,173,227), base_button_width*3,  
base_button_height*6,'Rectangle', None, big_text_size, curve)  
join_btn = new Button((39, 174, 96), (27,31,35), (settings.WIDTH/2), (settings.HEIGHT/2)*1.445-  
button_padding, "Graphics/Fonts/Orbitron-Medium.ttf", (27,31,35), (39, 174, 96), base_button_width,  
base_button_height,'Connect','Rectangle', None, int(big_text_size/1.3), curve)  
server_ip_text = new Text(text_height, "Graphics/Fonts/Orbitron-Regular.ttf", (240,240,240), None, None,  
None, int(base_text_size*1.7))  
name_text_box = new TextBox(base_button_width*2.5, base_button_height*1.5, (settings.WIDTH/2),  
(settings.HEIGHT/2)*0.70, (240,240,240), (200,200,200),"Graphics/Fonts/Orbitron-  
Regular.ttf",curve,thickness,base_button_width*2.5)  
name_box_text = new Text(text_height, "Graphics/Fonts/Orbitron-Regular.ttf", (240,240,240), None, None,  
None, int(base_text_size*1.7))  
// Settings  
main_menu_pages = "home"  
settings_screen = "control"  
attack_keys = "Space"  
magic_keys = "L-Shift"  
endif  
keys = "WASD"  
// Volume  
min_vol_x = volume_line.x + volume_button.width/3 + 1  
max_vol_x = new volume_line.x + volume_line.width - (volume_button.width*2)/3 - 1  
volume_ratio = new 100 / (max_vol_x - min_vol_x)  
volume = new int(volume_ratio * ((volume_button.x + volume_button.width/4) - min_vol_x))  
sound_change = 0  
music_change = 0  
endprocedure  
  
public procedure close()  
loop = False  
pygame.quit()  
sys.exit()  
endprocedure  
  
public procedure options()  
main_menu_pages = "settings"  
endprocedure  
  
public procedure home()  
main_menu_pages = "home"  
endprocedure  
  
public procedure start_option()  
main_menu_pages = "start"  
endprocedure  
  
public procedure draw_moving_background()  
// Draw the moving image on the screen  
sunset_image.draw(current_x, 0)  
sunset_image.draw((current_x - sunset_image.rect.width), 0)  
current_x += 1
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
// If the image has moved beyond its width, reset it to 0
if current_x >= sunset_image.rect.width then
    current_x = 0
endif
endprocedure

public procedure change_keys()
if keys == 'WASD' then
    keys = 'Arrow Keys'
else
    keys = 'WASD'
endif
endprocedure

public procedure control_settings_change()
    settings_screen = "control"
endprocedure

public procedure audio_settings_change()
    settings_screen = "audio"
endprocedure

public procedure change_attack()
// Changes the key binds.
if attack_keys == "Space" then
    attack_keys = "L-Ctrl"
endif
elseif attack_keys == "L-Ctrl" then
    attack_keys = "R-Ctrl"
else
    attack_keys = "Space"
endif
endprocedure

public procedure change_magic()
if magic_keys == "L-Shift" then
    magic_keys = "R-Shift"
endif
endif
elseif magic_keys == "R-Shift" then
    magic_keys = "Enter"
else
    magic_keys = "L-Shift"
endif
endif
endprocedure

public procedure sound_box_color()
// Changes the sound box color
sound_change += 1
if sound_change == 1 then
    sound_box.color = new (39,174,96)
else
    sound_change = 0
    sound_box.color = new (27,31,35)
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
        endif
endprocedure

public procedure music_box_color()
    // Changes the music box check color.
    music_change += 1
    if music_change == 1 then
        music_box.color = new (39,174,96)
    else
        music_change = 0
        music_box.color = new (27,31,35)
    endif
endprocedure

public procedure volume_settings()
    // Get the mouse and sets the volume to where the mouse is.
    mouse_pos = new pygame.mouse.get_pos()
    if mouse_pos[0] - volume_button.width/3 > volume_line.x then
        if mouse_pos[0] + (volume_button.width/3)*2 < volume_line.x + volume_line.width then
            volume_button.x = mouse_pos[0] - volume_button.width/3
            volume = new int(volume_ratio * (mouse_pos[0] - min_vol_x))
        endif
    endif
endprocedure

function create_dict()
    starting_dict = {
        "settings": {
            "control": {
                "movement": keys,
                "offense": attack_keys,
                "magic": magic_keys
            },
            "audio": {
                "volume": volume,
                "sound" then True if sound_change == 1 else False,
                "music" then True if music_change == 1 else False
            }
        },
        "start": {
            "username": name_text_box.return_text() OR "Player",
            "server_ip": ip_text_box.return_text() OR "192.168.0.223"
        }
    }
    loop = False
    // Check the values
    //
    for category, subcategories in starting_dict.items()
        logger.info(f"{category}:")
        for subcategory, values in subcategories.items()
            logger.info(f" {subcategory}:")
            if isinstance(values, dict) then
                for key, value in values.items()
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

    logger.info(f" {key}: {value}")
else
  next key, value
  logger.info(f" {values}")//
endif
next subcategory, values
next category, subcategories
endfunction

function start_game()
if NOT loop then
  return starting_dict
else
  return None
endif
endfunction

public procedure restart_menu()
loop = True
home()
endprocedure

public procedure draw_objects(events)
// Draw the buttons and check for hover
if main_menu_pages == "home" then
  github_button.draw((27,31,35),None,"https://github.com/TheRealDL1/Simple-Client-Server")
  youtube_button.draw((27,31,35),None,"https://www.youtube.com/watch?v= new dQw4w9WgXcQ")
  start_button.draw((27,31,35),start_option)
  options_button.draw((27,31,35),options)
  quit_button.draw((27,31,35),close,None,True)
  start_button_hover = new start_button.is_hovered()
  options_button_hover = new options_button.is_hovered()
  quit_button_hover = new quit_button.is_hovered()
  github_button_hover = new github_button.is_hovered()
  youtube_button_hover = new youtube_button.is_hovered()
  start_button_click = new start_button.is_clicking()
  options_button_click = new options_button.is_clicking()
  quit_button_click = new quit_button.is_clicking()
  github_button_click = new github_button.is_clicking()
  youtube_button_click = new youtube_button.is_clicking()
endif
endprocedure

// Check if mouse is hovering over buttons
if start_button_hover OR options_button_hover OR quit_button_hover OR github_button_hover OR
youtube_button_hover then
  // Draw hover text.
  if github_button_hover then github_name.draw("draw","Github", settings.WIDTH-(image_width/2)-
image_padding, (image_width/2)+image_padding*3.1)
    else github_name.draw("undraw","Github", settings.WIDTH-(image_width/2)-image_padding,
(image_width/2)+image_padding*3.1)
  endif
  if youtube_button_hover then youtube_name.draw("draw","Youtube", settings.WIDTH-(image_width*2)-
(image_padding/4), (image_width/2)+image_padding*3.1)

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
    else youtube_name.draw("undraw","Youtube", settings.WIDTH-(image_width*2)-(image_padding/4),
(image_width/2)+image_padding*3.1)
    endif
    if NOT start_button_click AND NOT options_button_click AND NOT quit_button_click AND NOT
github_button_click AND NOT youtube_button_click then
        custom_mouse.mode = 1
    else
        custom_mouse.mode = 2
    else
        endif
        custom_mouse.mode = 0
    github_name.draw("undraw","Github", settings.WIDTH-(image_width/2)-image_padding,
(image_width/2)+image_padding*3.1)
        youtube_name.draw("undraw","Youtube", settings.WIDTH-(image_width*2)-(image_padding/4),
(image_width/2)+image_padding*3.1)
    endif
if main_menu_pages == "settings" then
    options_board.draw((27,31,35),None, None, False)
    back.draw((240,240,240),home)
    control_settings.draw("draw","Control Settings", (settings.WIDTH/2)*0.70,
(settings.HEIGHT/2)*0.42,control_settings_change)
    audio_settings.draw("draw","Audio Settings", (settings.WIDTH/2)*1.3,
(settings.HEIGHT/2)*0.42,audio_settings_change)
    github_button.draw((27,31,35),None,"https://github.com/TheRealDL1/Simple-Client-Server")
    youtube_button.draw((27,31,35),None,"https://www.youtube.com/@dominicpike")
    back_hover = new back.is_hovered()
    github_button_hover = new github_button.is_hovered()
    youtube_button_hover = new youtube_button.is_hovered()
    key_binds_hover = new control_settings.is_hovered()
    audio_hover = new audio_settings.is_hovered()
    key_hover = new key_binds.is_hovered()
    attack_hover = new attack_btn.is_hovered()
    volume_btn_hover = new volume_button.is_hovered()
    sound_box_hover = new sound_box.is_hovered()
    music_box_hover = new music_box.is_hovered()
    magic_btn_hover = new magic_btn.is_hovered()
    back_click = new back.is_clicking()
    github_button_click = new github_button.is_clicking()
    youtube_button_click = new youtube_button.is_clicking()
    key_binds_click = new control_settings.is_clicking()
    audio_click = new audio_settings.is_clicking()
    key_click = new key_binds.is_clicking()
    attack_click = new attack_btn.is_clicking()
    volume_btn_click = new volume_button.is_clicking()
    sound_box_click = new sound_box.is_clicking()
    music_box_click = new music_box.is_clicking()
    magic_btn_click = new magic_btn.is_clicking()
    if back_hover OR github_button_hover OR youtube_button_hover OR key_binds_hover OR audio_hover OR
key_hover OR attack_hover OR volume_btn_hover OR sound_box_hover OR music_box_hover OR
magic_btn_hover then
        if github_button_hover then github_name.draw("draw","Github", settings.WIDTH-(image_width/2)-
image_padding, (image_width/2)+image_padding*3.1)
        else github_name.draw("undraw","Github", settings.WIDTH-(image_width/2)-image_padding,
(image_width/2)+image_padding*3.1)
    endif
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
if youtube_button_hover then youtube_name.draw("draw","Youtube", settings.WIDTH-(image_width*2)-(image_padding/4), (image_width/2)+image_padding*3.1)
else youtube_name.draw("undraw","Youtube", settings.WIDTH-(image_width*2)-(image_padding/4), (image_width/2)+image_padding*3.1)
endif
if NOT back_click AND NOT github_button_click AND NOT youtube_button_click AND NOT key_binds_click AND NOT audio_click AND NOT key_click AND NOT attack_click AND NOT volume_btn_click AND NOT sound_box_click AND NOT music_box_click AND NOT magic_btn_click then
    custom_mouse.mode = 1
else
    if volume_btn_click then
        volume_settings()
    endif
    custom_mouse.mode = 2
endif
custom_mouse.mode = 0
github_name.draw("undraw","Github", settings.WIDTH-(image_width/2)-image_padding, (image_width/2)+image_padding*3.1)
youtube_name.draw("undraw","Youtube", settings.WIDTH-(image_width*2)-(image_padding/4), (image_width/2)+image_padding*3.1)
endif
if settings_screen == "control" then
    if audio_hover then underline2.draw(None,None,None,True,None,None,None,"draw")
    else underline2.draw(None,None,None,True,None,None,None,"undraw")
    endif
    underline.draw(None,None,None,True,None,None,None,"draw")
    box_control.draw((240,240,240))
    movement.draw("draw","Movement", (settings.WIDTH/2)*0.70, (settings.HEIGHT/2)*0.57)
    key_binds.draw((240,240,240),change_keys, None, True, keys)
    attack_text.draw("draw","Offense", (settings.WIDTH/2)*0.70, (settings.HEIGHT/2)*0.82)
    attack_btn.draw((240,240,240),change_attack, None, True, attack_keys)
    magic_text.draw("draw","Magic", (settings.WIDTH/2)*0.70, (settings.HEIGHT/2)*1.07)
    magic_btn.draw((240,240,240),change_magic, None, True, magic_keys)
endif
elseif settings_screen == "audio" then
    if key_binds_hover then underline.draw(None,None,None,True,None,None,None,"draw")
    else underline.draw(None,None,None,True,None,None,None,"undraw")
    endif
    underline2.draw(None,None,None,True,None,None,None,"draw")
    box_audio.draw((240,240,240))
    volume_text.draw("draw","Volume", (settings.WIDTH/2)*1.3, (settings.HEIGHT/2)*0.57)
    volume_bar.draw((240,240,240))
    volume_line.draw((240,240,240))
    volume_button.draw((240,240,240))
    volume_indicator.draw("draw",str(volume), (settings.WIDTH/2)*1.435, (settings.HEIGHT/2)*0.685)
    audio_text.draw("draw","Audio", (settings.WIDTH/2)*1.3, (settings.HEIGHT/2)*0.82)
    audio_box.draw((240,240,240))
    sound_text.draw("draw","Sound", (settings.WIDTH/2)*1.185, (settings.HEIGHT/2)*0.97)
    music_text.draw("draw","Music", (settings.WIDTH/2)*1.18, (settings.HEIGHT/2)*1.08)
    sound_box.draw((240,240,240),sound_box_color)
    music_box.draw((240,240,240),music_box_color)
endif
endif
if main_menu_pages == "start" then
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

options_board.draw((27,31,35),None, None, False)
start_back.draw((240,240,240),home)
github_button.draw((27,31,35),None,"https://github.com/TheRealDL1/Simple-Client-Server")
youtube_button.draw((27,31,35),None,"https://www.youtube.com/@dominicpike")
join_boarder.draw((240,240,240))
bullet_assault.draw("draw","Bullet Assault", (settings.WIDTH/2), (settings.HEIGHT/2)*0.43)
server_ip_text.draw("draw","Server IP Address", (settings.WIDTH/2), (settings.HEIGHT/2)*0.955)
message = "Enter the Server's IP Address that you want to join!"
join_message.draw("draw",message, (settings.WIDTH/2), (settings.HEIGHT/2)*0.52)
ip_text_box.draw()
ip_text_box.updateText(events)
ip_text_box.update()
join_btn.draw((240,240,240),create_dict)
// Name Text Box
name_box_text.draw("draw","Username", (settings.WIDTH/2), (settings.HEIGHT/2)*0.665)
name_text_box.draw()
name_text_box.updateText(events)
name_text_box.update()
github_button_hover = new github_button.is_hovered()
youtube_button_hover = new youtube_button.is_hovered()
start_back_hover = new start_back.is_hovered()
join_hover = new join_btn.is_hovered()
text_box_hover = new ip_text_box.is_hovered()
name_hover = new name_text_box.is_hovered()
github_button_click = new github_button.is_clicking()
youtube_button_click = new youtube_button.is_clicking()
start_back_click = new start_back.is_clicking()
join_click = new join_btn.is_clicking()
text_box_click = new ip_text_box.is_clicking()
name_click = new name_text_box.is_clicking()
if start_back_hover OR github_button_hover OR youtube_button_hover OR join_hover OR text_box_hover
OR name_hover then
  // Draw hover text.
  if github_button_hover then github_name.draw("draw","Github", settings.WIDTH-(image_width/2)-
image_padding, (image_width/2)+image_padding*3.1)
    else github_name.draw("undraw","Github", settings.WIDTH-(image_width/2)-image_padding,
(image_width/2)+image_padding*3.1)
    endif
  if youtube_button_hover then youtube_name.draw("draw","Youtube", settings.WIDTH-(image_width*2)-
(image_padding/4), (image_width/2)+image_padding*3.1)
    else youtube_name.draw("undraw","Youtube", settings.WIDTH-(image_width*2)-(image_padding/4),
(image_width/2)+image_padding*3.1)
    endif
  if NOT start_back_click AND NOT github_button_click AND NOT youtube_button_click AND NOT
join_click AND NOT text_box_click AND NOT name_click then
    custom_mouse.mode = 1
  else
    custom_mouse.mode = 2
  else
    endif
  custom_mouse.mode = 0
  github_name.draw("undraw","Github", settings.WIDTH-(image_width/2)-image_padding,
(image_width/2)+image_padding*3.1)
  youtube_name.draw("undraw","Youtube", settings.WIDTH-(image_width*2)-(image_padding/4),
(image_width/2)+image_padding*3.1)

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
        endif
    endif
    // Draw the mouse
    custom_mouse.draw()
public procedure redraw_window(events)
    draw_moving_background()
    draw_objects(events)
endprocedure

public procedure run()
    while loop
        events = new pygame.event.get()
        for event in events
            if event.type == pygame.QUIT then
                close()
            endif
            if event.type == pygame.KEYDOWN then
                if event.key == pygame.K_ESCAPE then
                    close()
                endif
            endif
            next event
        // Update the display and frame rate
        redraw_window(events)
        pygame.display.update()
        clock.tick(FPS)
    endwhile
endprocedure
```

Player:

```
from Scripts.logger import *
try
    import pygame
    from Scripts.support import import_folder
    from Scripts.settings import Config
except:
    logger.critical("You do NOT have all the modules installed. Please install Pygame.")
endtry
logger.debug("RealDL Player Code.")

class Player inherits pygame.sprite.Sprite
    private hitbox.top
    private hitbox.bottom
    private hitbox.left
    private hitbox.right
    private rect.center
    private hitbox.y
    private hitbox.x
    private direction.x
    private direction.y
    private animation_images
    private animations
    private obstacle_sprites
    private sprite_type
    private font
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
private textSize
private basefont
private id
private username
private invulnerability_duration
private hurt_time
private vulnerable
private exp
private energy
private health
private upgrade_cost
private max_stats
private stats
private magic_switch_time
private can_switch_magic
private magic_index
private switch_duration_cooldown
private weapon_switch_time
private can_switch_weapon
private weapon
private weapon_index
private paused
private attack_time
private attack_cooldown
private attacking
private speed
private direction
private animation_speed
private frame_index
private status
private 'magic'
private 'attack'
private 'move_right'
private 'move_left'
private 'move_down'
private 'move_up'
private key_binds
private magic
private offense
private movement
private hitbox
private rect
private image_name
private image
private screen
private settings

public procedure new(pos, image, groups, obstacle_sprites, username, id, movement= new "WASD", offense=
new "Space", magic= new "L-Shift") then
    // Setting up player
    try
        super.new(groups)
        settings = new Config()
        screen = new pygame.display.get_surface()
        try
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
image = new pygame.image.load(image).convert_alpha()
image_name = image
except
    image = new pygame.image.load(f"../{image}").convert_alpha()
    image_name = f"../{image}"
endtry
rect = new image.get_rect(topleft = new pos)
hitbox = new rect.inflate(0,-26)
movement = movement
offense = offense
magic = magic
key_binds = {
    'move_up' then pygame.K_w if movement == "WASD" else pygame.K_UP,
    'move_down' then pygame.K_s if movement == "WASD" else pygame.K_DOWN,
    'move_left' then pygame.K_a if movement == "WASD" else pygame.K_LEFT,
    'move_right' then pygame.K_d if movement == "WASD" else pygame.K_RIGHT,
    'attack' then pygame.K_SPACE if offense == "Space" else pygame.K_LCTRL if offense == "L-Ctrl" else
pygame.K_RCTRL,
    'magic' then pygame.K_LSHIFT if magic == "L-Shift" else pygame.K_RSHIFT if magic == "R-Shift" else
pygame.K_RETURN
    endif
    endif
    endif
    endif
    endif
    endif
}
// graphics setup
import_player_assets()
status = 'down'
frame_index = 0
animation_speed = 0.15
// Movement
direction = new pygame.math.Vector2()
speed = 5
attacking = False
attack_cooldown = 400
attack_time = None
paused = False
// weapon
weapon_index = 0
weapon = new list(settings.weapon_data.keys())[weapon_index]
can_switch_weapon = True
weapon_switch_time = None
switch_duration_cooldown = 200
// magic
magic_index = 0
magic = new list(settings.magic_data.keys())[magic_index]
can_switch_magic = True
magic_switch_time = None
// stats
stats = {'health': 100,'energy':60,'attack': 10,'magic': 4,'speed': 5}
max_stats = {'health': 300, 'energy': 140, 'attack': 20, 'magic' : 10, 'speed': 10}
upgrade_cost = {'health': 100, 'energy': 100, 'attack': 100, 'magic' : 100, 'speed': 100}
health = stats['health'] * 0.5
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
energy = stats['energy'] * 0.8
exp = 5000
speed = stats['speed']
// damage timer
vulnerable = True
hurt_time = None
invulnerability_duration = 500
//Other stats
username = username
id = id
basefont = "Graphics/Fonts/Orbitron-Medium.ttf"
textSize = 20
try
    font = new pygame.font.Font(basefont, textSize)
except
    font = new pygame.font.Font(f"../{basefont}", textSize)
endtry
sprite_type = "player"
obstacle_sprites = obstacle_sprites
except
    logger.error("Failed to create Player Class")
endtry
endprocedure

public procedure input()
    keys = new pygame.key.get_pressed()
    if NOT paused then
        if NOT attacking then
            if keys[key_binds['move_up']] then
                direction.y = -1
                status = 'up'
            endif
            elseif keys[key_binds['move_down']] then
                direction.y = 1
                status = 'down'
            else
                direction.y = 0
            endif
            if keys[key_binds['move_right']] then
                direction.x = 1
                status = 'right'
            endif
            elseif keys[key_binds['move_left']] then
                direction.x = -1
                status = 'left'
            else
                direction.x = 0
            endif
            if keys[key_binds['attack']] then
                attacking = True
                attack_time = new pygame.time.get_ticks()
                logger.info('attack')
            endif
            // magic input
            if keys[key_binds['magic']] then
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

attacking = True
attack_time = new pygame.time.get_ticks()
logger.info("magic")
endif
if keys[pygame.K_q] AND can_switch_weapon then
  can_switch_weapon = False
  weapon_switch_time = new pygame.time.get_ticks()
  if weapon_index < list(settings.weapon_data.keys().length)) - 1
    weapon_index += 1
  else
    weapon_index = 0
  endif
  weapon = new list(settings.weapon_data.keys())[weapon_index]
endif
if keys[pygame.K_e] AND can_switch_magic then
  can_switch_magic = False
  magic_switch_time = new pygame.time.get_ticks()
  if magic_index < list(settings.magic_data.keys().length)) - 1
    magic_index += 1
  else
    magic_index = 0
  endif
  magic = new list(settings.magic_data.keys())[magic_index]
endif
endif
endif
endprocedure

```

Settings:

```

from Scripts.logger import *
try
  from socket import gethostname, gethostbyname
  import pickle, pygame

except
  logger.critical("You do NOT have all the modules installed. Please install Pygame.")
endtry
pygame.init()
logger.info("Github: https://github.com/TheRealDL1/Simple-Client-Server")
logger.debug("RealDL Settings Code.")

class Config
  private UPGRADE_BG_COLOR_SELECTED
  private BAR_COLOR_SELECTED
  private BAR_COLOR
  private TEXT_COLOR_SELECTED
  private UI_BORDER_COLOR_ACTIVE
  private ENERGY_COLOR
  private HEALTH_COLOR
  private TEXT_COLOR
  private UI_BORDER_COLOR
  private UI_BG_COLOR
  private WATER_COLOR
  private UI_FONT_SIZE
  private UI_FONT

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
private ITEM_BOX_SIZE
private ENERGY_BAR_WIDTH
private HEALTH_BAR_WIDTH
private BAR_HEIGHT
private magic_data
private weapon_data
private TILE_ID
private TILESIZEx
private ENCRYPTION_DATA_SIZE
private SMALL_DATA
private DATA_SIZE
private BG_COLOR
private ID_STRING_LENGTH
private PORT
private SERVER
private HOST_NAME
private FPS
private BITS
private SQUARE_SIZE
private WIDTH
private HEIGHT

public procedure new()
// Screen Width and Height
info = new pygame.display.Info()
max_width = info.current_w
max_height = info.current_h
HEIGHT = max_height // 720
WIDTH = max_width // 1280
// Other Server/ Client Settings
SQUARE_SIZE = 64
BITS = 256
FPS = 60
HOST_NAME = new gethostname()
SERVER = new gethostbyname(HOST_NAME)
PORT = 5555
ID_STRING_LENGTH = 20
BG_COLOR = new (113, 221, 238)
DATA_SIZE = 4096
SMALL_DATA = 32
ENCRYPTION_DATA_SIZE = 1024
TILESIZEx = 64
TILE_ID = 'Tile'
// weapons
weapon_data = {
    'sword': {'cooldown': 100, 'damage': 15,'graphic':'Graphics/Game/weapons/sword/full.png'},
    'lance': {'cooldown': 400, 'damage': 30,'graphic':'Graphics/Game/weapons/lance/full.png'},
    'axe': {'cooldown': 300, 'damage': 20, 'graphic':'Graphics/Game/weapons/axe/full.png'},
    'rapier': {'cooldown': 50, 'damage': 8, 'graphic':'Graphics/Game/weapons/rapier/full.png'},
    'sai': {'cooldown': 80, 'damage': 10, 'graphic':'Graphics/Game/weapons/sai/full.png'},
    'revolver': {'cooldown': 200, 'damage': 25, 'graphic':'Graphics/Game/weapons/revolver/full.png'},
    'msg': {'cooldown': 60, 'damage': 6, 'graphic':'Graphics/Game/weapons/msg/full.png'},
    'pistol': {'cooldown': 160, 'damage': 16, 'graphic':'Graphics/Game/weapons/pistol/full.png'}}
// magic
magic_data = {
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

'flame': {'strength': 5,'cost': 20,'graphic':'Graphics/Game/particles/flame/fire.png'},  

'heal' : {'strength': 20,'cost': 10,'graphic':'Graphics/Game/particles/heal/heal.png'}}  

// ui  

BAR_HEIGHT = 20  

HEALTH_BAR_WIDTH = 200  

ENERGY_BAR_WIDTH = 140  

ITEM_BOX_SIZE = 90  

UI_FONT = 'Graphics/Fonts/Orbitron-Medium.ttf'  

UI_FONT_SIZE = 18  

// general colors  

WATER_COLOR = new (113, 221, 238)  

UI_BG_COLOR = new (34, 34, 34)  

UI_BORDER_COLOR = new (17, 17, 17)  

TEXT_COLOR = new (238, 238, 238)  

// ui colors  

HEALTH_COLOR = new (255, 0, 0)  

ENERGY_COLOR = new (23, 108, 235)  

UI_BORDER_COLOR_ACTIVE = new (255, 215, 0)  

// upgrade menu  

TEXT_COLOR_SELECTED = new (17, 17, 17)  

BAR_COLOR = new (238, 238, 238)  

BAR_COLOR_SELECTED = new (17, 17, 17)  

UPGRADE_BG_COLOR_SELECTED = new (238, 238, 238)  

endprocedure
  
```

Server:

```

from Scripts.logger import *  

try  

  from _thread import *  

  import random  

  import string, math, socket  

  from Scripts.settings import Config  

  from Scripts.encryption import *  

  from Scripts.debug import debug

except
  logger.critical("You do NOT have all the modules installed. Please install Pygame, RSA AND Pycryptodome.")
endtry
logger.debug("RealDL Server Code.")

class Server inherits Config
  private coordinates
  private rsa_encrypt
  private public_key,
  private rsa_keys
  private s
  private running
  private connections
  private players
  private port
  private server

  public procedure new()
    try
      Config.__init__()
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

initialize_server()
except
  logger.error(f"Couldn't initialize server.")
endtry
endprocedure

public procedure initialize_server()
  server = SERVER
  port = PORT
  players = {}
  connections = 0
  running = True
  s = new socket.socket(socket.AF_INET, socket.SOCK_STREAM)
  rsa_keys = new RSA_Keys(ENCRYPTION_DATA_SIZE)
  public_key, private_key = new rsa_keys.export_keys()
  rsa_encrypt = new RSA_Encryption(public_key)
  coordinates = [
    [[1030, 1270],[1278, 1616]],
    [[1100, 1660],[1278, 2180]],
    [[1420, 580],[2494, 432]],
    [[1670, 245],[1865, 592]],
    [[2250, 2485],[2622, 2896]],
    [[3134, 2485],[2818, 2832]]
  ]
  try
    s.bind((server, port))
  except socket.error as e
    logger.error(f"Socket Error: {e}")
  endtry
  s.listen()
  logger.info("Waiting for connections, Server Started")
  logger.info(f"Server IP Address: {SERVER}")
endprocedure

public procedure get_player_position()
  function create_coords()
    random_area = new random.choice(coordinates)
    if random_area == coordinates[2] then
      x = new random.randint(random_area[0][0], random_area[1][0])
      y = new random.randint(random_area[1][1], random_area[0][1])
    endif
    elseif random_area == coordinates[5] then
      x = new random.randint(random_area[1][0], random_area[0][0])
      y = new random.randint(random_area[0][1], random_area[1][1])
    else
      x = new random.randint(random_area[0][0], random_area[1][0])
      y = new random.randint(random_area[0][1], random_area[1][1])
    endif
    return x,y
  endfunction

  try
    x, y = new create_coords()
    while any(is_touching(x, y, p['x'], p['y'], SQUARE_SIZE) for p in players.values())
      x, y = new create_coords()
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

  endwhile
except
  x, y = new create_coords()
endtry
return x, y
endprocedure

function validate_movement(player_pos, player_data)
  // Calculate the distance moved between updates
  player_speed = 10
  leeway = 5
  average_player_distance = new (player_speed*10) + leeway // Safety to ensure legit players have some
leeway.
  delta_x = player_data['x'] - player_pos[0][0]
  delta_y = player_data['y'] - player_pos[0][1]
  distance_moved = new math.sqrt(delta_x**2 + delta_y**2)
endfunction

if distance_moved > average_player_distance then
  return False
endif
return True
function validate_health(player)
  leeway = 5
  max_health = 300
  if player['health'] > max_health + leeway then
    return max_health
  else
    return player['health']
  endif
endfunction

function validate_player_status(player, data)
  if player['status'] == "dead" AND data['status'] == 'alive' then
    return player['status']
  else
    return player['status']
  endif
endfunction

public procedure handle_client_communication(conn, key_string, aes_encryption)
  running = True
  player_pos = []
  while running
    try
      // Receive data from player and check it is valid.
      data = new aes_encryption.decrypt(unserialize(conn.recv(DATA_SIZE)))
      data['status'] = new validate_player_status(players[key_string], data)
      players[key_string] = data
      // Validate the player data to ensure they are not hacking.
      player_pos.append([players[key_string]['x'], players[key_string]['y']])
      if player_pos.length > 10 player_pos.pop(0)
      endif
      if NOT validate_movement(player_pos, players[key_string]) then

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

running = False
logger.info(f"Player {key_string} disconnected because they were speed hacking.")
endif
players[key_string]['health'] = new validate_health(players[key_string])
logger.info(f"Received Player Dict: {data}.")
if NOT data then
  logger.info(f"Player {key_string} disconnected.")
  running = False
else
  reply = players
  encrypted_reply = new serialize(aes_encryption.encrypt(reply))
endif
conn.sendall(encrypted_reply)
logger.info(f"Sending All Player Dict: {reply}.")
except
  logger.info(f"Player {key_string} lost connection.")
  running = False
endtry
endwhile
logger.info(f"Connection Closed for Player {key_string}.")
try
  del players[key_string]
except
  logger.info(f"No player with the ID: {key_string} exists.")
endtry
connections -= 1
conn.close()
endprocedure

public procedure run()
  while running
    conn, addr = new s.accept()
    connections += 1
    logger.info(f"Connected to: {addr}")
    logger.info(f"There {'is' if connections= new = new 1 else 'are'} {connections} {'client' if connections= new = new 1 else 'clients'} connected to the server!")
    endif
    start_new_thread(threaded_client, (conn,))
  endwhile
endprocedure

if __name__ == "__main__":
  try
    server = new Server()
    server.run()
  except
    logger.critical("Server has failed to launch.")
  endtry
endif

support:
from Scripts.logger import *
try
  from csv import reader
  from os import walk, path

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
import pygame
except
    logger.critical("You do NOT have all the modules installed. Please install Pygame.")
endtry
function import_csv_layout(path)
    terrain_map = []
    try
        with open(path) as level_map:
            layout = new reader(level_map,delimiter = new ',')
            for row in layout
                terrain_map.append(list(row))
            next row
    except
        with open(f"../{path}") as level_map:
            layout = new reader(level_map,delimiter = new ',')
            for row in layout
                terrain_map.append(list(row))
            next row
    endtry
    return terrain_map
endfunction

function import_folder(folder_path,return_image_list= new False)
    surface_list = []
    image_list = []
    values = False
    while NOT values
        for root, dirs, img_files in walk(folder_path)
            for image in img_files
                full_path = new path.join(root, image)
                try
                    image_surf = new pygame.image.load(full_path).convert_alpha()
                except
                    image_surf = new pygame.image.load(f"../{full_path}").convert_alpha()
                endtry
                surface_list.append(image_surf)
                image_list.append(full_path)
            next image
        next root, dirs, img_files
    if surface_list AND image_list then
        values = True
    else
        if "../" in folder_path then
            values = True
        else
            folder_path = "../" + folder_path
        endif
    endif
endwhile
if return_image_list then
    return surface_list, image_list
else
    return surface_list
endif
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

endfunction

UI:

```
from Scripts.logger import *
try
    from Scripts.functions import *
    from Scripts.settings import Config
    import pygame

except
    logger.critical("You do NOT have all the modules installed. Please install Pygame.")
endtry
logger.debug("RealDL UI Code.")

class UI inherits Config
    private custom_mouse.mode
    private magic_graphics
    private weapon_graphics
    private energy_bar_rect
    private health_bar_rect
    private players_alive
    private boarder_stats
    private kill_count
    private player_count
    private font
    private continue_btn
    private bullet_assault
    private quit
    private info2
    private info
    private join_boarder
    private border
    private base_button_height
    private base_button_width
    private thickness
    private curve
    private big_text_size
    private base_text_size
    private text_height
    private button_padding
    private image_padding
    private image_height
    private image_width
    private BASE_BUTTON_HEIGHT
    private BASE_BUTTON_WIDTH
    private THICKNESS
    private CURVE
    private BUTTON_PADDING
    private IMAGE_PADDING
    private IMAGE_HEIGHT
    private IMAGE_WIDTH
    private TEXT_HEIGHT
    private BASE_TEXT_SIZE
    private BIG_TEXT_SIZE
    private quit_function
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
private draw_ui
private custom_mouse
private height_ratio
private width_ratio
private DEFAULT_HEIGHT
private DEFAULT_WIDTH
private screen_height
private screen_width
private screen

public procedure new(custom_mouse, quit_function, player_count, kill_count)
    // Setup Pygame Variables
    Config._init_()
    screen = new pygame.display.get_surface()
    info = new pygame.display.Info()
    screen_width = info.current_w
    screen_height = info.current_h
    DEFAULT_WIDTH = 1920
    DEFAULT_HEIGHT = 1080
    width_ratio = screen_width / DEFAULT_WIDTH
    height_ratio = screen_height / DEFAULT_HEIGHT
    custom_mouse = custom_mouse
    draw_ui = False
    quit_function = quit_function
    // Constants setup
    BIG_TEXT_SIZE = 50
    BASE_TEXT_SIZE = 15
    TEXT_HEIGHT = 20
    IMAGE_WIDTH = 64
    IMAGE_HEIGHT = 64
    IMAGE_PADDING = 20
    BUTTON_PADDING = 85
    CURVE = 10
    THICKNESS = 2
    BASE_BUTTON_WIDTH = 250
    BASE_BUTTON_HEIGHT = 70
    // Variable setup
    image_width = new int(IMAGE_WIDTH * width_ratio)
    image_height = new int(IMAGE_HEIGHT * height_ratio)
    image_padding = new int(IMAGE_PADDING * width_ratio)
    button_padding = new int(BUTTON_PADDING * height_ratio)
    text_height = new int(TEXT_HEIGHT * height_ratio)
    base_text_size = new int(BASE_TEXT_SIZE * height_ratio)
    big_text_size = new int(BIG_TEXT_SIZE * height_ratio)
    curve = new int(CURVE * height_ratio)
    thickness = new int(THICKNESS * height_ratio)
    base_button_width = new int(BASE_BUTTON_WIDTH * width_ratio)
    base_button_height = new int(BASE_BUTTON_HEIGHT * height_ratio)
    // UI Board
    border = new Button((27, 31, 35), (27, 31, 35), screen_width / 2, screen_height / 2, "Graphics/Fonts/Orbitron-Regular.ttf", (27, 31, 35), (27, 31, 35), screen_width * 0.7, screen_height * 0.7, 'Rectangle', None, big_text_size, int(curve * 1.5))
    join_boarder = new Button((27, 31, 35), (27, 31, 35), (screen_width / 2), (screen_height / 2), "Graphics/Fonts/Orbitron-Regular.ttf", (240, 240, 240), (136, 173, 227), base_button_width * 3, base_button_height * 6, 'Rectangle', None, big_text_size, curve)
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

info = new Text(text_height, "Graphics/Fonts/Orbitron-Regular.ttf", (240, 240, 240), None, None, None,
int(base_text_size * 1.7))
info2 = new Text(text_height, "Graphics/Fonts/Orbitron-Regular.ttf", (240, 240, 240), None, None, None,
int(base_text_size * 1.7))
quit = new Button((174, 39, 96), (27, 31, 35), screen_width / 2, screen_height * 0.78,
"Graphics/Fonts/Orbitron-Medium.ttf", (27, 31, 35), (174, 39, 96), base_button_width, base_button_height, 'Quit',
'Rectangle', None, int(big_text_size / 1.3), curve)
bullet_assault = new Text(text_height, "Graphics/Fonts/Orbitron-Bold.ttf", (240, 240, 240), None, None, None,
int(base_text_size * 3.5))
continue_btn = new Button((39, 174, 96), (27, 31, 35), (screen_width / 2), (screen_height / 2) * 1.445 -
button_padding, "Graphics/Fonts/Orbitron-Medium.ttf", (27, 31, 35), (39, 174, 96), base_button_width,
base_button_height, 'Continue', 'Rectangle', None, int(big_text_size / 1.3), curve)
// In-game UI. kill count,
// general
try
  font = new pygame.font.Font(UI_FONT, UI_FONT_SIZE)
except
  font = new pygame.font.Font(f"../{UI_FONT}", UI_FONT_SIZE)
endtry
player_count = player_count
kill_count = kill_count
// Kill count and player count
boarder_stats = new Button(UI_BG_COLOR, UI_BG_COLOR, WIDTH-(HEALTH_BAR_WIDTH)/2-10-2,
10+(BAR_HEIGHT*1.25)+2, "Graphics/Fonts/Orbitron-Medium.ttf", TEXT_COLOR, TEXT_COLOR,
(HEALTH_BAR_WIDTH), BAR_HEIGHT*2.5, "", 'Rectangle', None, UI_FONT_SIZE, 0)
kill_count = new Button(UI_BG_COLOR, UI_BG_COLOR, WIDTH-(HEALTH_BAR_WIDTH)/2-10-2,
13.5+(BAR_HEIGHT/2)+2, "Graphics/Fonts/Orbitron-Medium.ttf", TEXT_COLOR, TEXT_COLOR,
(HEALTH_BAR_WIDTH), BAR_HEIGHT, f'Kill count: {kill_count}', 'Rectangle', None, UI_FONT_SIZE, 0)
players_alive = new Button(UI_BG_COLOR, UI_BG_COLOR, WIDTH-(HEALTH_BAR_WIDTH)/2-10-2,
BAR_HEIGHT+16.5+(BAR_HEIGHT/2)+2, "Graphics/Fonts/Orbitron-Medium.ttf", TEXT_COLOR, TEXT_COLOR,
(HEALTH_BAR_WIDTH), BAR_HEIGHT, f'Players alive: {player_count}', 'Rectangle', None, UI_FONT_SIZE, 0)
// bar setup
health_bar_rect = new pygame.Rect(10, 10, HEALTH_BAR_WIDTH, BAR_HEIGHT)
energy_bar_rect = new pygame.Rect(10, 34, ENERGY_BAR_WIDTH, BAR_HEIGHT)
// convert weapon dictionary
weapon_graphics = []
for weapon in weapon_data.values()
  weapon_path = weapon['graphic']
  try
    weapon = new pygame.image.load(weapon_path).convert_alpha()
  except
    weapon = new pygame.image.load(f"../{weapon_path}").convert_alpha()
  endtry
  weapon_graphics.append(weapon)
next weapon
// convert magic dictionary
magic_graphics = []
for magic in magic_data.values()
  magic_path = magic['graphic']
  try
    magic = new pygame.image.load(magic_path).convert_alpha()
  except
    magic = new pygame.image.load(f"../{magic_path}").convert_alpha()
  endtry
  magic_graphics.append(magic)

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
next magic
endprocedure

public procedure stop_drawing()
    draw_ui = False
endprocedure

public procedure show_info()
    boarder_stats.draw(UI_BORDER_COLOR,None,None,False)
    kill_count.draw(None,None,None,False)
    players_alive.draw(None,None,None,False)
endprocedure

public procedure show_bar(current, max_amount, bg_rect, color)
    // draw bg
    pygame.draw.rect(screen, UI_BG_COLOR, bg_rect)
    // converting stat to pixel
    ratio = current / max_amount
    current_width = bg_rect.width * ratio
    current_rect = new bg_rect.copy()
    current_rect.width = current_width
    // drawing the bar
    pygame.draw.rect(screen, color, current_rect)
    pygame.draw.rect(screen, UI_BORDER_COLOR, bg_rect, 3)
endprocedure

public procedure show_exp(exp)
    text_surf = new font.render(str(int(exp)), False, TEXT_COLOR)
    x = new screen.get_size()[0] - 20
    y = new screen.get_size()[1] - 20
    text_rect = new text_surf.get_rect(bottomright= new(x, y))
    pygame.draw.rect(screen, UI_BG_COLOR, text_rect.inflate(20, 20))
    screen.blit(text_surf, text_rect)
    pygame.draw.rect(screen, UI_BORDER_COLOR, text_rect.inflate(20, 20), 3)
endprocedure

function selection_box(left, top, has_swapped)
    bg_rect = new pygame.Rect(left, top, ITEM_BOX_SIZE, ITEM_BOX_SIZE)
    pygame.draw.rect(screen, UI_BG_COLOR, bg_rect)
    if has_swapped then
        pygame.draw.rect(screen, UI_BORDER_COLOR_ACTIVE, bg_rect, 3)
    else
        pygame.draw.rect(screen, UI_BORDER_COLOR, bg_rect, 3)
    endif
    return bg_rect
endfunction

public procedure weapon_overlay(weapon_index, has_swapped)
    bg_rect = new selection_box(10, HEIGHT-ITEM_BOX_SIZE-10, has_swapped)
    weapon_surf = weapon_graphics[weapon_index]
    weapon_rect = new weapon_surf.get_rect(center= newbg_rect.center)
    screen.blit(weapon_surf, weapon_rect)
endprocedure

public procedure magic_overlay(magic_index, has_swapped)
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
bg_rect = new selection_box(10+ITEM_BOX_SIZE+10, HEIGHT-ITEM_BOX_SIZE-10, has_switched)
magic_surf = magic_graphics[magic_index]
magic_rect = new magic_surf.get_rect(center= newbg_rect.center)
screen.blit(magic_surf, magic_rect)
endprocedure

public procedure display(player)
    show_bar(player.health, player.stats['health'], health_bar_rect, HEALTH_COLOR)
    show_bar(player.energy, player.stats['energy'], energy_bar_rect, ENERGY_COLOR)
    show_exp(player.exp)
    show_info()
    weapon_overlay(player.weapon_index, NOT player.can_switch_weapon)
    magic_overlay(player.magic_index, NOT player.can_switch_magic)
endprocedure
```

Python Code:

Client:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
class Client(Config):
    def __init__(self):
        try:
            Config.__init__(self)
            self.gameMenu = MainMenu()
        except:
            logger.error("Error, couldn't initialize the main menu.")
            self.close()

    def initialize_client(self, user_dict):
        logger.info("Client Connecting to Server")
        try:
            self.initialize_pygame(user_dict)
            self.initialize_network()
        except:
            logger.error("Error, couldn't initialize the client.")
            self.close()

    def close(self):
        try:
            self.network.close()
        except:
            logger.exception("No server-client connection.")
        self.running = False
        self.player = None
        self.players = None
        self.gameMenu.restart_menu()
        self.run()

    def quit(self):
        self.running = False
        pygame.quit()
        sys.exit()

    def run(self):
        self.gameMenu.run()
        user_dict = self.gameMenu.start_game()
        self.initialize_client(user_dict)
        self.main()

if __name__ == "__main__":
    try:
        client = Client()
        client.run()
    except:
        logger.error("Couldn't run main menu or client.")
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Main menu:

```
from Scripts.logger import *
try:
    import pygame, sys
    from Scripts.settings import Config
    from Scripts.encryption import *
    from Scripts.debug import debug
    from Scripts.functions import *
except:
    logger.critical("You do not have all the modules installed. Please install Pygame.")

logger.info("RealDL - Main Menu Code")

class MainMenu:
    def __init__(self):
        # Pygame/ Game setup
        pygame.display.set_caption('Bullet Assault')
        self.clock = pygame.time.Clock()
        self.FPS = 60
        self.loop = True
        self.settings = Config()
        self.DEFAULT_WIDTH = 1920
        self.DEFAULT_HEIGHT = 1080
        self.width_ratio = self.settings.WIDTH / self.DEFAULT_WIDTH
        self.height_ratio = self.settings.HEIGHT / self.DEFAULT_HEIGHT
        self.screen = pygame.display.set_mode((self.settings.WIDTH, self.settings.HEIGHT))

        # Constants setup
        self.BIG_TEXT_SIZE = 50
        self.BASE_TEXT_SIZE = 15
        self.TEXT_HEIGHT = 20
        self.IMAGE_WIDTH = 64
        self.IMAGE_HEIGHT = 64
        self.IMAGE_PADDING = 20
        self.BUTTON_PADDING = 85
        self.CURVE = 10
        self.THICKNESS = 2
        self.BASE_BUTTON_WIDTH = 250
        self.BASE_BUTTON_HEIGHT = 70
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
def close(self):
    self.loop = False
    pygame.quit()
    sys.exit()

def options(self):
    self.main_menu_pages = "settings"

def home(self):
    self.main_menu_pages = "home"

def start_option(self):
    self.main_menu_pages = "start"

def draw_moving_background(self):
    # Draw the moving image on the screen
    self.sunset_image.draw(self.current_x, 0)
    self.sunset_image.draw((self.current_x - self.sunset_image.rect.width), 0)
    self.current_x += 1

    # If the image has moved beyond its width, reset it to 0
    if self.current_x >= self.sunset_image.rect.width:
        self.current_x = 0

def change_keys(self):
    if self.keys == 'WASD':
        self.keys = 'Arrow Keys'
    else:
        self.keys = 'WASD'

def control_settings_change(self):
    self.settings_screen = "control"

def audio_settings_change(self):
    self.settings_screen = "audio"

def change_attack(self):
    # Changes the key binds.
    if self.attack_keys == "Space":
        self.attack_keys = "L-Ctrl"
    elif self.attack_keys == "L-Ctrl":
        self.attack_keys = "R-Ctrl"
    else:
        self.attack_keys = "Space"

def change_magic(self):
    if self.magic_keys == "L-Shift":
        self.magic_keys = "R-Shift"
    elif self.magic_keys == "R-Shift":
        self.magic_keys = "Enter"
    else:
        self.magic_keys = "L-Shift"
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
def sound_box_color(self):
    # Changes the sound box color
    self.sound_change += 1
    if self.sound_change == 1:
        self.sound_box.color = (39,174,96)
    else:
        self.sound_change = 0
        self.sound_box.color = (27,31,35)

def music_box_color(self):
    # Changes the music box check color.
    self.music_change += 1
    if self.music_change == 1:
        self.music_box.color = (39,174,96)
    else:
        self.music_change = 0
        self.music_box.color = (27,31,35)

def volume_settings(self):
    # Get the mouse and sets the volume to where the mouse is.
    mouse_pos = pygame.mouse.get_pos()
    if mouse_pos[0] - self.volume_button.width/3 > self.volume_line.x:
        if mouse_pos[0] + (self.volume_button.width/3)*2 < self.volume_line.x + self.volume_line.width:
            self.volume_button.x = mouse_pos[0] - self.volume_button.width/3
            self.volume = int(self.volume_ratio * (mouse_pos[0] - self.min_vol_x))

def create_dict(self):
    self.starting_dict = {
        "settings": {
            "control": {
                "movement": self.keys,
                "offense": self.attack_keys,
                "magic": self.magic_keys
            },
            "audio": {
                "volume": self.volume,
                "sound": True if self.sound_change == 1 else False,
                "music": True if self.music_change == 1 else False
            }
        },
        "start": {
            "username": self.name_text_box.return_text() or "Player",
            "server_ip": self.ip_text_box.return_text() or "192.168.0.223"
        }
    }
    self.loop = False
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

def start_game(self):
  if not self.loop:
    | return self.starting_dict
  else:
    | return None

def restart_menu(self):
  self.loop = True
  self.home()

def draw_objects(self, events):
  # Draw the buttons and check for hover
  if self.main_menu_pages == "home":
    self.github_button.draw((27,31,35),None,"https://github.com/TheRealDL1/Simple-Client-Server")
    self.youtube_button.draw((27,31,35),None,"https://www.youtube.com/watch?v=dQw4w9wgXcQ")
    self.start_button.draw((27,31,35),self.start_option)
    self.options_button.draw((27,31,35),self.options)
    self.quit_button.draw((27,31,35),self.close,None,True)

    start_button_hover = self.start_button.is_hovered()
    options_button_hover = self.options_button.is_hovered()
    quit_button_hover = self.quit_button.is_hovered()
    github_button_hover = self.github_button.is_hovered()
    youtube_button_hover = self.youtube_button.is_hovered()

    start_button_click = self.start_button.is_clicking()
    options_button_click = self.options_button.is_clicking()
    quit_button_click = self.quit_button.is_clicking()
    github_button_click = self.github_button.is_clicking()
    youtube_button_click = self.youtube_button.is_clicking()

  # Check if mouse is hovering over buttons
  if start_button_hover or options_button_hover or quit_button_hover or github_button_hover or youtube_button_hover:
    # Draw hover text.
    if github_button_hover: self.github_name.draw("draw","Github", self.settings.WIDTH-(self.image_width/2)-self.image_padding, (self.image_width/2)+self.image_padding*3.1)
    else: self.github_name.draw("undraw","Github", self.settings.WIDTH-(self.image_width/2)-self.image_padding, (self.image_width/2)+self.image_padding*3.1)

    if youtube_button_hover: self.youtube_name.draw("draw","Youtube", self.settings.WIDTH-(self.image_width/2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)
    else: self.youtube_name.draw("undraw","Youtube", self.settings.WIDTH-(self.image_width/2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)

    if not start_button_click and not options_button_click and not quit_button_click and not github_button_click and not youtube_button_click:
      self.custom_mouse.mode = 1
    else:
      self.custom_mouse.mode = 2
  else:
    self.custom_mouse.mode = 0
    self.github_name.draw("undraw","Github", self.settings.WIDTH-(self.image_width/2)-self.image_padding, (self.image_width/2)+self.image_padding*3.1)
    self.youtube_name.draw("undraw","Youtube", self.settings.WIDTH-(self.image_width/2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)

  if self.main_menu_pages == "settings":
    self.options_board.draw((27,31,35),None, None, False)
    self.back.draw((240,240,240),self.home)
    self.control_settings.draw("draw","Control Settings", (self.settings.WIDTH/2)*0.70, (self.settings.HEIGHT/2)*0.42,self.control_settings_change)
    self.audio_settings.draw("draw","Audio Settings", (self.settings.WIDTH/2)*1.3, (self.settings.HEIGHT/2)*0.42,self.audio_settings_change)
    self.github_button.draw((27,31,35),None,"https://github.com/TheRealDL1/Simple-Client-Server")
    self.youtube_button.draw((27,31,35),None,"https://www.youtube.com/@dominickpike")

    back_hover = self.back.is_hovered()
    github_button_hover = self.github_button.is_hovered()
    youtube_button_hover = self.youtube_button.is_hovered()
    key_binds_hover = self.control_settings.is_hovered()
    audio_hover = self.audio_settings.is_hovered()
    key_hover = self.key_binds.is_hovered()
    attack_hover = self.attack_btn.is_hovered()
    volume_btn_hover = self.volume_button.is_hovered()
    sound_box_hover = self.sound_box.is_hovered()
    music_box_hover = self.music_box.is_hovered()
    magic_btn_hover = self.magic_btn.is_hovered()

    back_click = self.back.is_clicking()
    github_button_click = self.github_button.is_clicking()
    youtube_button_click = self.youtube_button.is_clicking()
    key_binds_click = self.control_settings.is_clicking()
    audio_click = self.audio_settings.is_clicking()
    key_click = self.key_binds.is_clicking()
    attack_click = self.attack_btn.is_clicking()
    volume_btn_click = self.volume_button.is_clicking()
    sound_box_click = self.sound_box.is_clicking()
    music_box_click = self.music_box.is_clicking()
    magic_btn_click = self.magic_btn.is_clicking()

  if back_hover or github_button_hover or youtube_button_hover or key_binds_hover or audio_hover or key_hover or attack_hover or volume_btn_hover or sound_box_hover or music_box_hover or magic_btn_hover:
    if github_button_hover: self.github_name.draw("draw","Github", self.settings.WIDTH-(self.image_width/2)-self.image_padding, (self.image_width/2)+self.image_padding*3.1)
    else: self.github_name.draw("undraw","Github", self.settings.WIDTH-(self.image_width/2)-self.image_padding, (self.image_width/2)+self.image_padding*3.1)

    if youtube_button_hover: self.youtube_name.draw("draw","Youtube", self.settings.WIDTH-(self.image_width/2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)
    else: self.youtube_name.draw("undraw","Youtube", self.settings.WIDTH-(self.image_width/2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)

    if not back_click and not github_button_click and not youtube_button_click and not key_binds_click and not audio_click and not key_click and not attack_click and not volume_btn_click and not sound_box_click and not music_box_click and not magic_btn_click:
      self.custom_mouse.mode = 1
    else:
      if volume_btn_click:
        self.volume_settings()
      self.custom_mouse.mode = 2
  else:
    self.custom_mouse.mode = 0
    self.github_name.draw("undraw","Github", self.settings.WIDTH-(self.image_width/2)-self.image_padding, (self.image_width/2)+self.image_padding*3.1)
    self.youtube_name.draw("undraw","Youtube", self.settings.WIDTH-(self.image_width/2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

if self.settings_screen == "control":
  if audio_hover: self.underline2.draw(None,None,None,True,None,None,None,"draw")
  else: self.underline2.draw(None,None,None,True,None,None,None,"undraw")
  self.underline.draw(None,None,None,True,None,None,None,"draw")
  self.box_control.draw((240,240,240))
  self.movement.draw("draw","Movement", (self.settings.WIDTH/2)*0.70, (self.settings.HEIGHT/2)*0.57)
  self.key_binds.draw((240,240,240),self.change_keys, None, True, self.keys)
  self.attack_text.draw("draw","Offense", (self.settings.WIDTH/2)*0.70, (self.settings.HEIGHT/2)*0.82)
  self.attack_btn.draw((240,240,240),self.change_attack, None, True, self.attack_keys)
  self.magic_text.draw("draw","Magic", (self.settings.WIDTH/2)*0.70, (self.settings.HEIGHT/2)*1.07)
  self.magic_btn.draw((240,240,240),self.change_magic, None, True, self.magic_keys)

elif self.settings_screen == "audio":
  if key_binds_hover: self.underline.draw(None,None,None,True,None,None,None,"draw")
  else: self.underline.draw(None,None,None,True,None,None,None,"undraw")
  self.underline2.draw(None,None,None,True,None,None,None,"draw")
  self.box_audio.draw((240,240,240))
  self.volume_text.draw("draw","Volume", (self.settings.WIDTH/2)*1.3, (self.settings.HEIGHT/2)*0.57)
  self.volume_bar.draw((240,240,240))
  self.volume_line.draw((240,240,240))
  self.volume_button.draw((240,240,240))
  self.volume_indicator.draw("draw",str(self.volume), (self.settings.WIDTH/2)*1.435, (self.settings.HEIGHT/2)*0.685)
  self.audio_text.draw("draw","Audio", (self.settings.WIDTH/2)*1.3, (self.settings.HEIGHT/2)*0.82)
  self.audio_box.draw((240,240,240))
  self.sound_text.draw("draw","Sound", (self.settings.WIDTH/2)*1.185, (self.settings.HEIGHT/2)*0.97)
  self.music_text.draw("draw","Music", (self.settings.WIDTH/2)*1.18, (self.settings.HEIGHT/2)*1.08)
  self.sound_box.draw((240,240,240),self.sound_box_color)
  self.music_box.draw((240,240,240),self.music_box_color)

if self.main_menu_pages == "start":
  self.options_board.draw((27,31,35),None, None, False)
  self.start_back.draw((240,240,240),self.home)
  self.github_button.draw((27,31,35),None,"https://github.com/TheRealDL1/Simple-Client-Server")
  self.youtube_button.draw((27,31,35),None,"https://www.youtube.com/@dominicpike")
  self.join_boarder.draw((240,240,240))
  self.bullet_assault.draw("draw","Bullet Assault", (self.settings.WIDTH/2), (self.settings.HEIGHT/2)*0.43)
  self.server_ip_text.draw("draw","Server IP Address", (self.settings.WIDTH/2), (self.settings.HEIGHT/2)*0.955)
  message = "Enter the Server's IP Address that you want to join!"
  self.join_message.draw("draw",message, (self.settings.WIDTH/2), (self.settings.HEIGHT/2)*0.52)
  self.ip_text_box.draw()
  self.ip_text_box.updateText(events)
  self.ip_text_box.update()
  self.join_btn.draw((240,240,240),self.create_dict)

# Name Text Box
self.name_box_text.draw("draw","Username", (self.settings.WIDTH/2), (self.settings.HEIGHT/2)*0.665)
self.name_text_box.draw()
self.name_text_box.updateText(events)
self.name_text_box.update()

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

github_button_hover = self.github_button.is_hovered()
youtube_button_hover = self.youtube_button.is_hovered()
start_back_hover = self.start_back.is_hovered()
join_hover = self.join_btn.is_hovered()
text_box_hover = self.ip_text_box.is_hovered()
name_hover = self.name_text_box.is_hovered()

github_button_click = self.github_button.is_clicking()
youtube_button_click = self.youtube_button.is_clicking()
start_back_click = self.start_back.is_clicking()
join_click = self.join_btn.is_clicking()
text_box_click = self.ip_text_box.is_clicking()
name_click = self.name_text_box.is_clicking()

if start_back_hover or github_button_hover or youtube_button_hover or join_hover or text_box_hover or name_hover:
    # Draw hover text.
    if github_button_hover: self.github_name.draw("draw","Github", self.settings.WIDTH-(self.image_width/2)-self.image_padding, (self.image_width/2)+self.image_padding*3.1)
    else: self.github_name.draw("undraw","Github", self.settings.WIDTH-(self.image_width/2)-self.image_padding, (self.image_width/2)+self.image_padding*3.1)

    if youtube_button_hover: self.youtube_name.draw("draw","Youtube", self.settings.WIDTH-(self.image_width/2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)
    else: self.youtube_name.draw("undraw","Youtube", self.settings.WIDTH-(self.image_width/2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)

    if not start_back_click and not github_button_click and not youtube_button_click and not join_click and not text_box_click and not name_click:
        | self.custom_mouse.mode = 1
    else:
        | self.custom_mouse.mode = 2
else:
    self.custom_mouse.mode = 0
    self.github_name.draw("undraw","Github", self.settings.WIDTH-(self.image_width/2)-self.image_padding, (self.image_width/2)+self.image_padding*3.1)
    self.youtube_name.draw("undraw","Youtube", self.settings.WIDTH-(self.image_width/2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)

# Draw the mouse
self.custom_mouse.draw()

def redraw_window(self, events):
    self.draw_moving_background()
    self.draw_objects(events)

def run(self):
    while self.loop:
        events = pygame.event.get()
        for event in events:
            if event.type == pygame.QUIT:
                | self.close()
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_ESCAPE:
                    | self.close()

        # Update the display and frame rate
        self.redraw_window(events)
        pygame.display.update()
        self.clock.tick(self.FPS)

```

Player:

```

from Scripts.logger import *
try:
    import pygame
    from Scripts.support import import_folder
    from Scripts.settings import Config
except:
    logger.critical("You do not have all the modules installed. Please install Pygame.")

logger.debug("RealDL Player Code.")

class Player(pygame.sprite.Sprite):
    def __init__(self, pos, image, groups, obstacle_sprites, username, id, movement="WASD", offense="Space", magic="L-Shift"):
        # Setting up player
        try:
            super().__init__(groups)
            self.settings = Config()
            self.screen = pygame.display.get_surface()
            try:
                self.image = pygame.image.load(image).convert_alpha()
                self.image_name = image
            except:

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
        self.image = pygame.image.load(f"../{image}").convert_alpha()
        self.image_name = f"../{image}"
    self.rect = self.image.get_rect(topleft = pos)
    self.hitbox = self.rect.inflate(0,-26)
    self.movement = movement
    self.offense = offense
    self.magic = magic
    self.key_binds = {
        'move_up': pygame.K_w if self.movement == "WASD" else
pygame.K_UP,
        'move_down': pygame.K_s if self.movement == "WASD" else
pygame.K_DOWN,
        'move_left': pygame.K_a if self.movement == "WASD" else
pygame.K_LEFT,
        'move_right': pygame.K_d if self.movement == "WASD" else
pygame.K_RIGHT,
        'attack': pygame.K_SPACE if self.offense == "Space" else
pygame.K_LCTRL if self.offense == "L-Ctrl" else pygame.K_RCTRL,
        'magic': pygame.K_LSHIFT if self.magic == "L-Shift" else
pygame.K_RSHIFT if self.magic == "R-Shift" else pygame.K_RETURN
    }

    # graphics setup
    self.import_player_assets()
    self.status = 'down'
    self.frame_index = 0
    self.animation_speed = 0.15

    # Movement
    self.direction = pygame.math.Vector2()
    self.speed = 5
    self.attacking = False
    self.attack_cooldown = 400
    self.attack_time = None
    self.paused = False

    # weapon
    self.weapon_index = 0
    self.weapon =
list(self.settings.weapon_data.keys())[self.weapon_index]
    self.can_switch_weapon = True
    self.weapon_switch_time = None
    self.switch_duration_cooldown = 200

    # magic
    self.magic_index = 0
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
        self.magic =
list(self.settings.magic_data.keys())[self.magic_index]
        self.can_switch_magic = True
        self.magic_switch_time = None

        # stats
        self.stats = {'health': 100, 'energy': 60, 'attack': 10, 'magic':
4, 'speed': 5}
        self.max_stats = {'health': 300, 'energy': 140, 'attack': 20,
'magic' : 10, 'speed': 10}
        self.upgrade_cost = {'health': 100, 'energy': 100, 'attack': 100,
'magic' : 100, 'speed': 100}
        self.health = self.stats['health'] * 0.5
        self.energy = self.stats['energy'] * 0.8
        self.exp = 5000
        self.speed = self.stats['speed']

        # damage timer
        self.vulnerable = True
        self.hurt_time = None
        self.invulnerability_duration = 500

#Other stats
        self.username = username
        self.id = id
        self.basefont = "Graphics/Fonts/Orbitron-Medium.ttf"
        self.textSize = 20
        try:
            self.font = pygame.font.Font(self.basefont, self.textSize)
        except:
            self.font = pygame.font.Font(f"../{self.basefont}",
self.textSize)
        self.sprite_type = "player"

        self.obstacle_sprites = obstacle_sprites
    except:
        logger.error("Failed to create Player Class")

def input(self):
    keys = pygame.key.get_pressed()
    if not self.paused:
        if not self.attacking:
            if keys[self.key_binds['move_up']]:
                self.direction.y = -1
                self.status = 'up'
            elif keys[self.key_binds['move_down']]:
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
        self.direction.y = 1
        self.status = 'down'
    else:
        self.direction.y = 0

    if keys[self.key_binds['move_right']]:
        self.direction.x = 1
        self.status = 'right'
    elif keys[self.key_binds['move_left']]:
        self.direction.x = -1
        self.status = 'left'
    else:
        self.direction.x = 0

    if keys[self.key_binds['attack']]:
        self.attacking = True
        self.attack_time = pygame.time.get_ticks()
        logger.info('attack')

    # magic input
    if keys[self.key_binds['magic']]:
        self.attacking = True
        self.attack_time = pygame.time.get_ticks()
        logger.info("magic")

    if keys[pygame.K_q] and self.can_switch_weapon:
        self.can_switch_weapon = False
        self.weapon_switch_time = pygame.time.get_ticks()

        if self.weapon_index <
len(list(self.settings.weapon_data.keys())) - 1:
            self.weapon_index += 1
        else:
            self.weapon_index = 0

        self.weapon =
list(self.settings.weapon_data.keys())[self.weapon_index]

    if keys[pygame.K_e] and self.can_switch_magic:
        self.can_switch_magic = False
        self.magic_switch_time = pygame.time.get_ticks()

        if self.magic_index <
len(list(self.settings.magic_data.keys())) - 1:
            self.magic_index += 1
        else:
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
        self.magic_index = 0

        self.magic =
list(self.settings.magic_data.keys())[self.magic_index]

def update(self):
    self.input()
    self.cooldowns()
    self.get_status()
    self.animate()
    self.move(self.speed)
    self.energy_recovery()
```

```
server:
from Scripts.logger import *
try:
    from _thread import *
    import random
    import string, math, socket
    from Scripts.settings import Config
    from Scripts.encryption import *
    from Scripts.debug import debug
except:
    logger.critical("You do not have all the modules installed. Please install Pygame, RSA and Pycryptodome.")

logger.debug("RealDL Server Code.")

class Server(Config):
    def __init__(self):
        try:
            Config.__init__(self)
            self.initialize_server()
        except:
            logger.error(f"Couldn't initialize server.")

    def initialize_server(self):
        self.server = self.SERVER
        self.port = self.PORT
        self.players = {}
        self.connections = 0
        self.running = True
        self.s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.rsa_keys = RSA_Keys(self.ENCRYPTION_DATA_SIZE)
        self.public_key, self.private_key = self.rsa_keys.export_keys()
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
self.rsa_encrypt = RSA_Encryption(self.public_key)
self.coordinates = [
    [[1030, 1270],[1278, 1616]],
    [[1100, 1660],[1278, 2180]],
    [[1420, 580],[2494, 432]],
    [[1670, 245],[1865, 592]],
    [[2250, 2485],[2622, 2896]],
    [[3134, 2485],[2818, 2832]]
]

try:
    self.s.bind((self.server, self.port))
except socket.error as e:
    logger.error(f"Socket Error: {e}")
self.s.listen()
logger.info("Waiting for connections, Server Started")
logger.info(f"Server IP Address: {self.SERVER}")

def get_player_position(self):
    def create_coords():
        random_area = random.choice(self.coordinates)
        if random_area == self.coordinates[2]:
            x = random.randint(random_area[0][0], random_area[1][0])
            y = random.randint(random_area[1][1], random_area[0][1])
        elif random_area == self.coordinates[5]:
            x = random.randint(random_area[1][0], random_area[0][0])
            y = random.randint(random_area[0][1], random_area[1][1])
        else:
            x = random.randint(random_area[0][0], random_area[1][0])
            y = random.randint(random_area[0][1], random_area[1][1])
        return x,y

    try:
        x, y = create_coords()
        while any(self.is_touching(x, y, p['x'], p['y'], self.SQUARE_SIZE)
for p in self.players.values()):
            x, y = create_coords()
    except:
        x, y = create_coords()
    return x, y

def validate_movement(self, player_pos, player_data):
    # Calculate the distance moved between updates
    player_speed = 10
    leeway = 5
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
average_player_distance = (player_speed*10) + leeway # Safety to
ensure legit players have some leeway.
delta_x = player_data[ 'x' ] - player_pos[0][0]
delta_y = player_data[ 'y' ] - player_pos[0][1]
distance_moved = math.sqrt(delta_x**2 + delta_y**2)

if distance_moved > average_player_distance:
    return False
return True

def validate_health(self, player):
    leeway = 5
    max_health = 300
    if player['health'] > max_health + leeway:
        return max_health
    else:
        return player['health']

def validate_player_status(self, player, data):
    if player['status'] == "dead" and data['status'] == 'alive':
        return player['status']
    else:
        return player['status']

def handle_client_communication(self, conn, key_string, aes_encryption):
    running = True
    player_pos = []
    while running:
        try:
            # Receive data from player and check it is valid.
            data =
aes_encryption.decrypt(self.unserialize(conn.recv(self.DATA_SIZE)))
            data[ 'status' ] =
self.validate_player_status(self.players[key_string], data)
            self.players[key_string] = data

            # Validate the player data to ensure they are not hacking.
            player_pos.append([self.players[key_string]['x'],
self.players[key_string]['y']])
            if len(player_pos) > 10: player_pos.pop(0)
            if not self.validate_movement(player_pos,
self.players[key_string]):
                running = False
                logger.info(f"Player {key_string} disconnected because
they were speed hacking.")
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
        self.players[key_string]['health'] =  
self.validate_health(self.players[key_string])  
        logger.info(f"Received Player Dict: {data}.")  
  
        if not data:  
            logger.info(f"Player {key_string} disconnected.")  
            running = False  
        else:  
            reply = self.players  
            encrypted_reply =  
self.serialize(aes_encryption.encrypt(reply))  
  
            conn.sendall(encrypted_reply)  
            logger.info(f"Sending All Player Dict: {reply}.")  
    except:  
        logger.info(f"Player {key_string} lost connection.")  
        running = False  
  
logger.info(f"Connection Closed for Player {key_string}.")  
try:  
    del self.players[key_string]  
except:  
    logger.info(f"No player with the ID: {key_string} exists.")  
self.connections -= 1  
conn.close()  
  
def run(self):  
    while self.running:  
        conn, addr = self.s.accept()  
        self.connections += 1  
        logger.info(f"Connected to: {addr}")  
        logger.info(f"There {'is' if self.connections==1 else 'are'}  
{self.connections} {'client' if self.connections==1 else 'clients'} connected  
to the server!")  
  
        start_new_thread(self.threaded_client, (conn,))  
  
if __name__ == "__main__":  
    try:  
        server = Server()  
        server.run()  
    except:  
        logger.critical("Server has failed to launch.")
```

settings:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
from Scripts.logger import *
try:
    from socket import gethostname, gethostbyname
    import pickle, pygame
except:
    logger.critical("You do not have all the modules installed. Please install Pygame.")
pygame.init()

logger.info("Github: https://github.com/TheRealDL1/Simple-Client-Server")
logger.debug("RealDL Settings Code.")

class Config():
    def __init__(self):
        # Screen Width and Height
        info = pygame.display.Info()
        max_width = info.current_w
        max_height = info.current_h
        self.HEIGHT = max_height # 720
        self.WIDTH = max_width # 1280

        # Other Server/ Client Settings
        self.SQUARE_SIZE = 64
        self.BITS = 256
        self.FPS = 60
        self.HOST_NAME = gethostname()
        self.SERVER = gethostbyname(self.HOST_NAME)
        self.PORT = 5555
        self.ID_STRING_LENGTH = 20
        self.BG_COLOR = (113, 221, 238)
        self.DATA_SIZE = 4096
        self.SMALL_DATA = 32
        self.ENCRYPTION_DATA_SIZE = 1024
        self.TILESIZE = 64
        self.TILE_ID = 'Tile'

        # weapons
        self.weapon_data = {
            'sword': {'cooldown': 100, 'damage': 15, 'graphic': 'Graphics/Game/weapons/sword/full.png'},
            'lance': {'cooldown': 400, 'damage': 30, 'graphic': 'Graphics/Game/weapons/lance/full.png'},
            'axe': {'cooldown': 300, 'damage': 20, 'graphic': 'Graphics/Game/weapons/axe/full.png'},
            'rapier': {'cooldown': 50, 'damage': 8, 'graphic': 'Graphics/Game/weapons/rapier/full.png'},
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
'sai':{'cooldown': 80, 'damage': 10,
'graphic':'Graphics/Game/weapons/sai/full.png'},
      'revolver':{'cooldown': 200, 'damage': 25,
'graphic':'Graphics/Game/weapons/revolver/full.png'},
      'msg':{'cooldown': 60, 'damage': 6,
'graphic':'Graphics/Game/weapons/msg/full.png'},
      'pistol':{'cooldown': 160, 'damage': 16,
'graphic':'Graphics/Game/weapons/pistol/full.png'}}
```

```
# magic
self.magic_data = {
    'flame': {'strength': 5,'cost':
20,'graphic':'Graphics/Game/particles/flame/fire.png'},
    'heal' : {'strength': 20,'cost':
10,'graphic':'Graphics/Game/particles/heal/heal.png'}}
```

```
# ui
self.BAR_HEIGHT = 20
self.HEALTH_BAR_WIDTH = 200
self.ENERGY_BAR_WIDTH = 140
self.ITEM_BOX_SIZE = 90
self.UI_FONT = 'Graphics/Fonts/Orbitron-Medium.ttf'
self.UI_FONT_SIZE = 18
```

```
# general colors
self.WATER_COLOR = (113, 221, 238)
self.UI_BG_COLOR = (34, 34, 34)
self.UI_BORDER_COLOR = (17, 17, 17)
self.TEXT_COLOR = (238, 238, 238)
```

```
# ui colors
self.HEALTH_COLOR = (255, 0, 0)
self.ENERGY_COLOR = (23, 108, 235)
self.UI_BORDER_COLOR_ACTIVE = (255, 215, 0)
```

```
# upgrade menu
self.TEXT_COLOR_SELECTED = (17, 17, 17)
self.BAR_COLOR = (238, 238, 238)
self.BAR_COLOR_SELECTED = (17, 17, 17)
self.UPGRADE_BG_COLOR_SELECTED = (238, 238, 238)
```

support:

```
from Scripts.logger import *
try:
    from csv import reader
    from os import walk, path
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
import pygame
except:
    logger.critical("You do not have all the modules installed. Please install Pygame.")

def import_csv_layout(path):
    terrain_map = []

    try:
        with open(path) as level_map:
            layout = reader(level_map, delimiter = ',')
            for row in layout:
                terrain_map.append(list(row))
    except:
        with open(f"../{path}") as level_map:
            layout = reader(level_map, delimiter = ',')
            for row in layout:
                terrain_map.append(list(row))
    return terrain_map

def import_folder(folder_path, return_image_list=False):
    surface_list = []
    image_list = []
    values = False

    while not values:
        for root, dirs, img_files in walk(folder_path):
            for image in img_files:
                full_path = path.join(root, image)
                try:
                    image_surf = pygame.image.load(full_path).convert_alpha()
                except:
                    image_surf =
                pygame.image.load(f"../{full_path}").convert_alpha()
                surface_list.append(image_surf)
                image_list.append(full_path)
        if surface_list and image_list:
            values = True
        else:
            if "../" in folder_path:
                values = True
            else:
                folder_path = "../" + folder_path

    if return_image_list:
        return surface_list, image_list
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
else:  
    return surface_list
```

UI:

```
from Scripts.logger import *  
try:  
    from Scripts.functions import *  
    from Scripts.settings import Config  
    import pygame  
except:  
    logger.critical("You do not have all the modules installed. Please install Pygame.")  
  
logger.debug("RealDL UI Code.")  
  
class UI(Config):  
    def __init__(self, custom_mouse, quit_function, player_count, kill_count):  
        # Setup Pygame Variables  
        Config.__init__(self)  
        self.screen = pygame.display.get_surface()  
        info = pygame.display.Info()  
        self.screen_width = info.current_w  
        self.screen_height = info.current_h  
        self.DEFAULT_WIDTH = 1920  
        self.DEFAULT_HEIGHT = 1080  
        self.width_ratio = self.screen_width / self.DEFAULT_WIDTH  
        self.height_ratio = self.screen_height / self.DEFAULT_HEIGHT  
        self.custom_mouse = custom_mouse  
        self.draw_ui = False  
        self.quit_function = quit_function  
  
        # Constants setup  
        self.BIG_TEXT_SIZE = 50  
        self.BASE_TEXT_SIZE = 15  
        self.TEXT_HEIGHT = 20  
        self.IMAGE_WIDTH = 64  
        self.IMAGE_HEIGHT = 64  
        self.IMAGE_PADDING = 20  
        self.BUTTON_PADDING = 85  
        self.CURVE = 10  
        self.THICKNESS = 2  
        self.BASE_BUTTON_WIDTH = 250  
        self.BASE_BUTTON_HEIGHT = 70  
  
        # Variable setup  
        self.image_width = int(self.IMAGE_WIDTH * self.width_ratio)
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
self.image_height = int(self.IMAGE_HEIGHT * self.height_ratio)
self.image_padding = int(self.IMAGE_PADDING * self.width_ratio)
self.button_padding = int(self.BUTTON_PADDING * self.height_ratio)
self.text_height = int(self.TEXT_HEIGHT * self.height_ratio)
self.base_text_size = int(self.BASE_TEXT_SIZE * self.height_ratio)
self.big_text_size = int(self.BIG_TEXT_SIZE * self.height_ratio)
self.curve = int(self.CURVE * self.height_ratio)
self.thickness = int(self.THICKNESS * self.height_ratio)
self.base_button_width = int(self.BASE_BUTTON_WIDTH *
self.width_ratio)
self.base_button_height = int(self.BASE_BUTTON_HEIGHT *
self.height_ratio)

# UI Board
self.border = Button((27, 31, 35), (27, 31, 35), self.screen_width /
2, self.screen_height / 2, "Graphics/Fonts/Orbitron-Regular.ttf", (27, 31,
35), (27, 31, 35), self.screen_width * 0.7, self.screen_height * 0.7, '',
'Rectangle', None, self.big_text_size, int(self.curve * 1.5))
self.join_boarder = Button((27, 31, 35), (27, 31, 35),
(self.screen_width / 2), (self.screen_height / 2), "Graphics/Fonts/Orbitron-
Regular.ttf", (240, 240, 240), (136, 173, 227), self.base_button_width * 3,
self.base_button_height * 6, '', 'Rectangle', None, self.big_text_size,
self.curve)
self.info = Text(self.text_height, "Graphics/Fonts/Orbitron-
Regular.ttf", (240, 240, 240), None, None, None, int(self.base_text_size *
1.7))
self.info2 = Text(self.text_height, "Graphics/Fonts/Orbitron-
Regular.ttf", (240, 240, 240), None, None, None, int(self.base_text_size *
1.7))
self.quit = Button((174, 39, 96), (27, 31, 35), self.screen_width / 2,
self.screen_height * 0.78, "Graphics/Fonts/Orbitron-Medium.ttf", (27, 31, 35),
(174, 39, 96), self.base_button_width, self.base_button_height, 'Quit',
'Rectangle', None, int(self.big_text_size / 1.3), self.curve)
self.bullet_assault = Text(self.text_height, "Graphics/Fonts/Orbitron-
Bold.ttf", (240, 240, 240), None, None, None, int(self.base_text_size * 3.5))
self.continue_btn = Button((39, 174, 96), (27, 31, 35),
(self.screen_width / 2), (self.screen_height / 2) * 1.445 -
self.button_padding, "Graphics/Fonts/Orbitron-Medium.ttf", (27, 31, 35), (39,
174, 96), self.base_button_width, self.base_button_height, 'Continue',
'Rectangle', None, int(self.big_text_size / 1.3), self.curve)

# In-game UI. kill count,
# general
try:
    self.font = pygame.font.Font(self.UI_FONT, self.UI_FONT_SIZE)
except:
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
        self.font = pygame.font.Font(f"../{self.UI_FONT}",  
self.UI_FONT_SIZE)  
        self.player_count = player_count  
        self.kill_count = kill_count  
  
        # Kill count and player count  
        self.boarder_stats = Button(self.UI_BG_COLOR, self.UI_BG_COLOR,  
self.WIDTH-(self.HEALTH_BAR_WIDTH)/2-10-2, 10+(self.BAR_HEIGHT*1.25)+2,  
"Graphics/Fonts/Orbitron-Medium.ttf", self.TEXT_COLOR, self.TEXT_COLOR,  
(self.HEALTH_BAR_WIDTH), self.BAR_HEIGHT*2.5, "", 'Rectangle', None,  
self.UI_FONT_SIZE, 0)  
        self.kill_count = Button(self.UI_BG_COLOR, self.UI_BG_COLOR,  
self.WIDTH-(self.HEALTH_BAR_WIDTH)/2-10-2, 13.5+(self.BAR_HEIGHT/2)+2,  
"Graphics/Fonts/Orbitron-Medium.ttf", self.TEXT_COLOR, self.TEXT_COLOR,  
(self.HEALTH_BAR_WIDTH), self.BAR_HEIGHT, f'Kill count: {self.kill_count}',  
'Rectangle', None, self.UI_FONT_SIZE, 0)  
        self.players_alive = Button(self.UI_BG_COLOR, self.UI_BG_COLOR,  
self.WIDTH-(self.HEALTH_BAR_WIDTH)/2-10-2,  
self.BAR_HEIGHT+16.5+(self.BAR_HEIGHT/2)+2, "Graphics/Fonts/Orbitron-  
Medium.ttf", self.TEXT_COLOR, self.TEXT_COLOR, (self.HEALTH_BAR_WIDTH),  
self.BAR_HEIGHT, f"Players alive: {self.player_count}", 'Rectangle', None,  
self.UI_FONT_SIZE, 0)  
  
        # bar setup  
        self.health_bar_rect = pygame.Rect(10, 10, self.HEALTH_BAR_WIDTH,  
self.BAR_HEIGHT)  
        self.energy_bar_rect = pygame.Rect(10, 34, self.ENERGY_BAR_WIDTH,  
self.BAR_HEIGHT)  
  
        # convert weapon dictionary  
        self.weapon_graphics = []  
        for weapon in self.weapon_data.values():  
            weapon_path = weapon['graphic']  
            try:  
                weapon = pygame.image.load(weapon_path).convert_alpha()  
            except:  
                weapon =  
pygame.image.load(f"../{weapon_path}").convert_alpha()  
            self.weapon_graphics.append(weapon)  
  
        # convert magic dictionary  
        self.magic_graphics = []  
        for magic in self.magic_data.values():  
            magic_path = magic['graphic']  
            try:  
                magic = pygame.image.load(magic_path).convert_alpha()
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
except:  
    magic = pygame.image.load(f"../{magic_path}").convert_alpha()  
    self.magic_graphics.append(magic)  
  
def show_info(self):  
    self.boarder_stats.draw(self.UI_BORDER_COLOR, None, None, False)  
    self.kill_count.draw(None, None, None, False)  
    self.players_alive.draw(None, None, None, False)  
  
def show_bar(self, current, max_amount, bg_rect, color):  
    # draw bg  
    pygame.draw.rect(self.screen, self.UI_BG_COLOR, bg_rect)  
  
    # converting stat to pixel  
    ratio = current / max_amount  
    current_width = bg_rect.width * ratio  
    current_rect = bg_rect.copy()  
    current_rect.width = current_width  
  
    # drawing the bar  
    pygame.draw.rect(self.screen, color, current_rect)  
    pygame.draw.rect(self.screen, self.UI_BORDER_COLOR, bg_rect, 3)  
  
def show_exp(self, exp):  
    text_surf = self.font.render(str(int(exp)), False, self.TEXT_COLOR)  
    x = self.screen.get_size()[0] - 20  
    y = self.screen.get_size()[1] - 20  
    text_rect = text_surf.get_rect(bottomright=(x, y))  
  
    pygame.draw.rect(self.screen, self.UI_BG_COLOR, text_rect.inflate(20,  
20))  
    self.screen.blit(text_surf, text_rect)  
    pygame.draw.rect(self.screen, self.UI_BORDER_COLOR,  
text_rect.inflate(20, 20), 3)  
  
def selection_box(self, left, top, has_switted):  
    bg_rect = pygame.Rect(left, top, self.ITEM_BOX_SIZE,  
self.ITEM_BOX_SIZE)  
    pygame.draw.rect(self.screen, self.UI_BG_COLOR, bg_rect)  
    if has_switted:  
        pygame.draw.rect(self.screen, self.UI_BORDER_COLOR_ACTIVE,  
bg_rect, 3)  
    else:  
        pygame.draw.rect(self.screen, self.UI_BORDER_COLOR, bg_rect, 3)  
    return bg_rect
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
def weapon_overlay(self, weapon_index, has_swapped):
    bg_rect = self.selection_box(10, self.HEIGHT-self.ITEM_BOX_SIZE-10,
has_swapped)
    weapon_surf = self.weapon_graphics[weapon_index]
    weapon_rect = weapon_surf.get_rect(center=bg_rect.center)

    self.screen.blit(weapon_surf, weapon_rect)

def magic_overlay(self, magic_index, has_swapped):
    bg_rect = self.selection_box(10+self.ITEM_BOX_SIZE+10, self.HEIGHT-
self.ITEM_BOX_SIZE-10, has_swapped)
    magic_surf = self.magic_graphics[magic_index]
    magic_rect = magic_surf.get_rect(center=bg_rect.center)

    self.screen.blit(magic_surf, magic_rect)

def display(self, player):
    self.show_bar(player.health, player.stats['health'],
self.health_bar_rect, self.HEALTH_COLOR)
    self.show_bar(player.energy, player.stats['energy'],
self.energy_bar_rect, self.ENERGY_COLOR)

    self.show_exp(player.exp)
    self.show_info()

    self.weapon_overlay(player.weapon_index, not player.can_switch_weapon)
    self.magic_overlay(player.magic_index, not player.can_switch_magic)
```

Test Proof (Fig 3.1 - 3.5):

See MP4 videos.

Sprint 4

Aims for this sprint.

Adding on from sprint 3 with UI changes, I want to start developing the mechanics of attacking with weapons and magic, as well as the functional differences between them. I do not currently intend to implement collision mechanics between multiple players, as I will do that at a later sprint (particularly after having implemented attacks). However, I will start on creating the animations and other visuals to have a better framework with which to implement collisions.

Weapon and Animation Mechanics

Firstly, I will split this up into two phases. The first phase will be developing the animations on each client individually. The second phases will be using the server to then send and receive each players

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

animation. In the first phase I will purely design and implement the animation and visuals of the animations into the game. I will create the code that creates and destroys the weapons. I will implement the code to create bullets and finally the code for the two animations: fire and heal.

Updating the server

After the visuals for each weapon works for each client individually, the server will need to be updated. What will happen is we will now include meta data for the weapons, bullets, and animations. The server will then be able to process this data, effectively passing data from one client to the other clients. When we create the player we will include the data for the weapons, bullets, and animations. This is to ensure that each player has their own weapons, bullets, and animations; meaning that we know which attack belongs to each player so that when we come to killing players, we will be able to know who gets the kill and who has been attacked.

Updating the client

After the server is updated, the client must be updated to match the server. This means that the server and client will be able to send the meta data for each attack back and forth. How this will be done is the dictionary that contains all the meta data for the player will include meta data for the new attack data and that will be sent over to the client. Adding this to the client is necessary because it will ensure that the data is effectively transferred between the server and clients, it will also ensure that there are no errors and that we know each player has their own weapons, bullets, and animations. This is not necessary for this sprint but is vital in the later sprint because we will need a mechanism to determine whether a player has been hit which should result in a reduction of health, furthermore when a player dies, we will ensure that the server knows which player has killed that player.

Weapons

Weapons will include a sword, a lance, an axe, a rapier, a sai, a revolver an smg, and a pistol. Something important to note is that my code refers to the smg as “msg” because I spelt it wrong accidentally. Each weapon, including the guns, will be able to appear for a split second then disappear. However, if you hold the weapon, the weapon will stay until the user releases the weapon. Players will not be able to move while they are attacking. This is because the weapon must stay in place and moving the character will mess up the weapon. In addition, a moving weapon would be too powerful as the player could just run into another player, which is why players will remain still when attacking. After the player has attacked there will be a brief cooldown time to ensure that if the player just holds the attack button it will not look visually bad and to allow some time for the player to be vulnerable. In addition, weapons have different attack damages and cooldown speeds. This is done to create a more immersive and interesting game.

Bullets

Bullets are different to the weapon. They are technically a weapon and still would do damage like another weapon however they shoot bullets. Players are not able to move while shooting and there is another cooldown between players attacking when they stop attacking. Something important about the bullet is, bullets must consistently shoot. There will be a function that creates a small cooldown between each bullet. This is different for each weapon as they are meant to have slightly different damages and cooldowns.

Animations

There will be two animations in my game. My stakeholders wanted an ability to use a power up. These animations are effectively power ups because they enhance the player. The first animation is a heal animation. This animation heals the player. It will increase the players health by a select

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

amount. Each animation will have its own duration, meaning how long the animation lasts as well as a cooldown after the animation. The heal animation will be a predesigned animation that last less than a second. The other animation is a fire animation. This animation will be multiple little fires that come out from the player's hand. There will be about twelve animations for the fire animation. After the player attacks with the fire animation, there will be a cooldown. The cooldown will be the same for each attack, however the duration of the fire attack is greater. This is only for the fire animation, but I will randomise the number of total fires. There could be 4 to 6 fire animations when the player uses the fire animation.

Attacking mechanics

The player attacking mechanics will be different for the weapons and magic animations. From last sprint the player was able to choose the keys that they used for the attacking and magic keys, however, in this sprint they will effectively work. The player will be able to choose between three different keys for each attack. Finally, the keys will be different to ensure that the player does not have the same key for an attack, this will ensure that each attack is separate.

Success Criteria

1. **Create weapons:** Create weapons that can be displayed on screen for a period and then disappear after the set duration or after the player lets go of the attack key. Include damage and cooldown stats for each weapon with different animations.
2. **Shooting bullets:** Create a method, like the weapon, that allows the player to shoot a bullet each time after a set cooldown by holding down the attack button. A cooldown should be applied after each bullet and when the player stops shooting there should be a cooldown to stop the player from shooting straight again.
3. **Fire and heal animations:** Have fire and heal animations that display on screen. Have the fire animation have individual flames (between 4 and 6). The flame animation should also have a duration and a cooldown period after being used. The heal animations should be one animation, however it should contain data about how much to increase the health by.
4. **Attacking mechanics:** Players should not be able to move while attacking regardless of if it is a melee attack or an attack with a flame animation (players should still not move when healing too).
5. **Update the server:** Update the server by changing the function where the player is created to include lists for the weapons, bullets, and animations.
6. **Update the client:** Update the client to be able to send and receive the updated player dictionary.
7. **Create attacking visuals privately:** Create the attacking visuals privately first. No other clients should be able to see other clients attacking visuals.
8. **Incorporate all client attacks to be seen by all:** All attacks by any player should be drawn onto the screen by all clients. The new connection should allow the meta data to be shared and drawn onto the screen.

Justification of Sprint 4

Attacking mechanics

Players should not move when they attack via melee, guns or by using magic. This is to ensure that firstly the animations look smooth when running. The melee weapons do not move, so the player moving would look out of place. Furthermore, moving while shooting is complicated to implement

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

and makes the game unfair for the other players because it is easy to run into other players. This would cause players to take lots of damage, therefore, I am implementing a method to stop players from moving when they attack to make the game smoother and fairer.

Attacking cooldowns and durations

When a player attacks, that attack will last for a duration of time. This effect is created to allow the other players to move out of the way when the player attacks. It is also implemented to slow down the game so that players do not die instantly. The game should be fun but also be a decent range. This is because my stakeholders did not really want to play a game that was long. Cooldowns after attacks are implemented in the game to allow other players to have the opportunity to fight back. It also makes the game more balanced as they cannot attack all the time. In addition, I added the cooldowns to make the game look slightly more visually pleasing. Lastly, cooldowns keep the game in check by ensuring that players cannot attack all the time and that they remain vulnerable to attacks for a brief period.

Updating the client and server

Updating the client and server is an imperative for the new attacking visuals to even be added to the game. Since my stakeholders wanted animations and attacking graphics the client code must be updated to send over all the meta data related to each attacking sprite class. This should be compiled into one large data set that should be sent over to the server. The change in the server is minimal but it is necessary it is changed because to facilitate the animations to work for all clients. If the server is not changed, firstly the code will break, and secondly the client will never receive any data from about the other clients' attacking visuals. Therefore, updating the server and client is an imperative to receiving data and then updating the screen to show the animations of one client's animation to all clients.

Creating the weapon

Creating the weapon is important for my future sprints. This is because weapons will be used to attack and kill other players. This is an imperative in the game because there needs to be visuals and effective mechanics that allow clients to attack each other, take damage and eventually die and turn into ghosts. My stakeholders wanted a game that is a last man standing game and to effectively make that game I must include a way to attack.

Creating bullets

In my analysis I said I would add guns to my game. I asked my stakeholders about the guns, and they were happy to have guns in the game. Guns are a good addition to the game because they improve the overall game quality and enjoyment of my stakeholders because it adds a ranged weapon that each have different stats. This is done purely to make the game more engaging and entertaining for my stakeholders. Therefore, guns are a necessity to my game to make it fun, engaging, and unique for my stakeholders to play.

Adding animations

Adding on to the last point, I will be adding animations into the game because my stakeholders wanted to include powerups into the game. The best way of doing this was by adding two different animations. Firstly, the heal animation will increase the players health when used but not more than full health. The heal animation is a powerup in the game as it is a special ability in the game. It allows players to increase their health by a set amount each time. I added the heal animation to meet my stakeholders needs of a powerup but also, I incorporated it because it helps diversify my game and make my game aesthetically pleasing and enjoyable playing. The fire animation is also a powerup because it shoots fire from the player. Each powerup/animation uses up energy which slowly

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

increases over time, however this power up is unique because it shoots multiple small fires. My stakeholders wanted something fun and unique when I designed my game, and this is what I have decided to add. The fire animation is a good addition to the game because it splits the attacking objects into three groups: melee weapons, guns, and fire (which is magic). Therefore, adding a third type of attacking object is great at meeting my stakeholders needs of a unique and fun game, but also helps at diversifying the game with multiple ways to attack.

Pseudo Code

The Pseudo Code is in the Appendix.

Python Development

Program Code

Code is in the appendix.

Areas of complexity

Updated Server

The server code previously is like the current server code. As shown below we still use the same method to send and receive data. However, what has changed is how we create the player.

```
def create_new_player(self, username):
    key_string = self.create_random_string()
    player_x, player_y = self.get_player_position()
    return {
        "player": {
            "x": player_x,
            "y": player_y,
            "image": "Graphics/Game/player/down_idle/idle_down.png",
            "username": username,
            "status": "alive",
            "health": 100,
            "id": key_string
        },
        "weapon": [],
        "bullets": [],
        "magic": []
    }, key_string
```

Here we create the player, but the key difference is that we have added a weapon, bullets, and magic list to the player data.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
def create_new_player(self, username):
    key_string = self.create_random_string()
    player_x, player_y = self.get_player_position()
    return {
        "player": {
            "x": player_x,
            "y": player_y,
            "image": "Graphics/Game/player/down_idle/idle_down.png",
            "username": username,
            "status": "alive",
            "health": 100,
            "id": key_string
        },
        "weapon": [],
        "bullets": [],
        "magic": []
    }, key_string
```

This is vital for the server to be able to receive the client data and update it. The server code here adds three lists that now enables the client and server to communicate together.

Updated Client Dictionary

As the server has been updated, we have updated the client to then send the data to the client. We must get the relevant information from the player like their x-value, y-value and more to send over all the necessary information to the server as well as sending over the three new lists (weapon, bullets, magic).

```
# Get player data
player_dict = {'x': self.player.rect.x,
               'y': self.player.rect.y,
               'image': self.player.get_image_name(),
               'username': self.username,
               'status': self.status,
               'health': self.player.health,
               'id': self.id}
player_total_dict = {'player': player_dict,
                     'weapon': self.return_weapon(),
                     'bullets': self.return_bullets(),
                     'magic': self.return_magic_data()
}
```

The three functions above are used to return the data for each list in the dictionary. How it works is every sprite is given a “sprite_type” and they are added the visible sprites group in level.py. For

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

example, in the Melee class where we create the weapons, we initialise the groups here. Here above is where we create the attack. We use “self.level.visible_sprites” to act as a sprite group that contains all the sprites that are to be drawn on screen. This is done for the bullets and magic as well shown below.

All the images above show that we use the sprite groups to draw the animations on screen. They are drawn below here in level.py. All the sprite from the “self.level.visible_sprites” group are drawn.

As all the sprites are stored in “self.level.visible_sprites” we can access all the relevant sprites by using those three methods to return the necessary meta data for each sprite in a dictionary all compiled into a list that gets sent over to the server. The client is shown to be able to send the updated data back and forth to the server, however, the client cannot do anything with that data unless we turn that data into class objects.

Creating weapons

In the client we have two functions that create weapons. These two functions effectively create and destroy weapons. However, since the player is in another class it cannot access these methods. This is when I decided to pass through those functions as attributes of my player.

As shown previously, the four methods (in blue) are used to create weapons, destroy weapons, create bullets, and create magic. These are all passed through to the player in client.py.

In the player, we set those methods to different variables to be used later.

In the input method in my player, we check if the player has attacked or used a magic ability. If they have, we simply run that command that was given to the player in the beginning. This is good to pass through functions because it does not affect the detection of collisions in future and it can help saves lines of code that are not needed, and we are also able to access the current weapon being used in the client too. As for the weapon when we finish attacking there is a cooldown.

This cooldown effectively destroys the weapon and stops the player from attacking. After attacking we are then allowed to move again because in the input method, we stop players moving if they are attacking or magic attacking.

In weapon.py here is where we create the actual weapon.

As shown, this function is called in the client, however the goal of this code is to create the weapon by sourcing it from the player class and getting the attribute. Another thing is we get the correct orientation for the weapon. This allows the correct weapon to be drawn as well as creating a unique ID for the weapon.

Creating bullets

In the client, we create a method called “create_bullet” that is passed through to the player.py file which uses that method in the player class to create multiple instances of the bullet.

Below is the create_bullet initialised as a variable.

When the game is not paused and the player is not currently attacking via melee attacks, guns or magic attacks, the player can then shoot. Here is the code that allows that to happen.

Something important to note is that this is the same function used in melee attacks as they require the same functionality except with the guns, we create a bullet only if the player is holding a gun.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

The final thing about the bullets is that we need to constantly fire them after a certain period. We do this in the cooldowns function. If the player is still holding the attack button, they are still attacking, this allows the player to shoot after every interval. A new bullet clone is created until the player stops holding the space button. After that, the if statement above the bottom one creates a cooldown after using the gun which leaves the player vulnerable to give a small advantage to the other players, which helps keep the game balanced. Furthermore, as shown previously where we created the weapon in weapon.py, the bullet is also created here.

Although the bullet and weapon are similar when it comes to creating them, the bullet has two extra functions. Update and Collision are both used to detect if a bullet has hit an obstacle. If so, it deletes that bullet.

Creating magic

Firstly, magic is more complicated as it requires extra classes and files to do the job.

This function is passed through to the player to be used when magic is run. Below is the function in the player where we check if the player has pressed the magic key and run the create_magic function.

Something else to mention is that in the client we initialise animation_player and magic_player that are both used to help create the magic for the flame and heal animations.

The magic animations require two new files. Magic.py is used to run the animations using particles.py.

This class has a heal and a flame function. The heal function ensures that the energy is greater than the cost and increases the players health and then runs the animations. The flame animation determines the direction of the player, then creates 5 flame animations that all slightly vary. Below is particles.py with the classes that are used.

AnimationPlayer holds the information of the data of image surfaces for each type of animation in a dictionary. The frames dictionary holds the surfaces of each image, whereas images hold the location of each image file. Create_particles function is used to get the frame and image needed for the ParticleEffect class to run through the image. The frame is the image that was converted into a surface by Pygame. ParticleEffect is used to return the image to the client when requested. This is to ensure that when the image is requested in the client, we can return it. Another thing to mention is that this animation class requires animate and update. This is an imperative so that we can loop through each frame and display the frame for each animation.

Drawing other players attack visuals on screen.

First, there must be a system to draw the weapons on the screen for the other players. The code below is all the code that is required for other players to be able to see the weapons. This is done through the function “update players” and requires the dictionary that holds information about all players, but more specifically, all the weapon data for each player.

The order for the code above goes update weapons, delete weapons and finally, create weapons. This order is necessary because otherwise visually, weapons are not drawn onto the screen

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

correctly. The update weapons function is quite simple. It gets all the weapon data into several new variables, then ensures that each weapon that is currently being drawn on the screen is in the dictionary. If it is, then it is updated, if it is not, then it is to be deleted for that client. This happens simultaneously for all clients. If a weapon is not on screen, however it is in the dictionary, then it should be created since it exists and is valid in the current game. As you can see, there is a class called “Weapon” that is creating another weapon. This is a ghost class as it creates a ghost image on other clients’ screens. They can see the image; however, it has a different sprite type, and it uses a different class. The “Weapons” class creates an x-value, a y-value and an image using the data from the dictionary that aims to copy a real weapon from another client so that other clients can visually see different clients’ weapons.

Now for the bullets. Updating the bullets for all clients are like updating the weapon since it requires the same mechanics. We update the bullets, then delete them then we create new bullets. The only difference is that there are new variable names. Apart from the new variable names and getting the data for the bullet and changing the sprite type name, there is almost no difference. The mechanisms are the same. Something to also mention that is like the weapon is that we are continuing to use the “Weapon” class to create a ghost bullet. This class is crucial when it comes to creating all the temporary weapons, bullets, and animations. You might think that using that class for collisions might be difficult, however, collisions can be done using sprite groups. The class takes a parameter called “groups” and this will allow me to create a collisions function in the client using a sprite group that contains all the attacking sprites like weapons, bullets, and the flame animation. Lastly, there are the animations. This section of code is like the sections above. All that is different is variable names. We get the information about all the animations from each player’s magic list. As seen before, we have variables that hold all the ids, images, x-values, and y-values for the animations. We then find the index of the magic object and create a temporary animation using the Weapons class. If any animation ID is not found in the dictionary but is drawn on screen, that animation will be deleted. Finally, if there is an animation that is not being drawn on screen, however it is valid within the dictionary that contains data about all players, it will then be created using the Weapons class again. Any animation ID that is still in the dictionary and an object on screen, it will be updated. All animations will be able to be seen by all players via this code as the server can connect all clients and draw all the images on the screen.

During iterative development

The testing below will test my game during the iterative development and after the iterative development. I will include several tests throughout development that will seek to determine whether I am successfully implementing all my success criteria into the game and if my game meets all the requirements in the tests at the end of development.

Test No.	Test Data	Expected Result	Actual Result
4.1	Player presses the attack button once.	The weapon should appear for a brief period then disappear.	Figure 4.1
4.2	Player switches weapons, attacks with different weapons.	The player can change weapon, then attack with the weapon shown in the menu GUI.	Figure 4.2
4.3	The player switches to a gun, then holds the attack button.	A gun should be drawn on screen with the correct orientation. Bullets should	Figure 4.3

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

		then be shooting out from the gun at regular intervals.	
4.4	The player presses the magic attack button with the flame option selected.	The client should be able to see a flame animation show that last for about a second.	Figure 4.4
4.5	The player presses the magic attack button with the heal option selected.	The client should increase their health if damaged and see the heal animation.	Figure 4.5
4.6	A client should press the attack button with the sword selected.	All clients will see that the sword is being displayed by that client.	Figure 4.6
4.7	A client should press the attack button with the revolver selected and holding space for a few seconds.	All the clients should see the revolver and shooting bullets at a consistent rate.	Figure 4.7
4.8	The client should choose one of the animations and press the magic attack button.	All clients should see the animations working visually on screen.	Figure 4.8
4.9	Server and Client should be communicating player data about the weapons, bullets, and animations.	The server sends all the data to all the clients, the clients receive all the data.	Figure 4.9

Stakeholders

Question	Roland	James
Do you like the weapon animation?	I like the simplicity of the weapons. I think that they are a good addition.	Yes. The weapons look good, and I like that they disappear after a certain time.
Do you like the bullets and the gun mechanism.	Yes. They look good. I like the duration between each bullet.	Yes. Shooting in the game can be useful and good.
Do you like the magic animations?	Yes. They look unique and aesthetic which is great for the game.	Yes, I like them. They are both unique and add some variety to the game.
Do you like that the heal animation increases the players health?	Being able to heal in the game is useful. It keeps the game going for longer.	I like this feature. It increases the game length.
What do you think of the methods used to send and receive different animations?	I think they are good. Sending and receiving important data is an imperative.	I think the methods are good. It is needed in this game and needs to be efficient.
What do you think of the attacking mechanics when it comes to cooldowns between weapons?	This is a good feature. It helps add suspense and diversity into the game.	I think they are good.

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

Evaluation

No.		Achieved
1	Create weapons: Create weapons that can be displayed on screen for a period and then disappear after the set duration or after the player lets go of the attack key. Include damage and cooldown stats for each weapon with different animations.	Yes
2	Shooting bullets: Create a method, like the weapon, that allows the player to shoot a bullet each time after a set cooldown by holding down the attack button. A cooldown should be applied after each bullet and when the player stops shooting there should be a cooldown to stop the player from shooting straight again.	Yes
3	Fire and heal animations: Have fire and heal animations that display on screen. Have the fire animation have individual flames (between 4 and 6). The flame animation should also have a duration and a cooldown period after being used. The heal animations should be one animation, however it should contain data about how much to increase the health by.	Yes
4	Attacking mechanics: Players should not be able to move while attacking regardless of if it is a melee attack or an attack with a flame animation (players should still not move when healing too).	Yes
5	Update the server: Update the server by changing the function where the player is created to include lists for the weapons, bullets, and animations.	Yes
6	Update the client: Update the client to be able to send and receive the updated player dictionary.	Yes
7	Create attacking visuals privately: Create the attacking visuals privately first. No other clients should be able to see other clients attacking visuals.	Yes
8	Incorporate all client attacks to be seen by all: All attacks by any player should be drawn onto the screen by all clients. The new connection should allow the meta data to be shared and drawn onto the screen.	Yes

In Sprint 4, my focus was on establishing new connection between the client and server in which the players meta data containing data about their weapons, bullets and magic animations can be shared between all clients and the server. This was the first stage of my sprint where I aimed to be able to send and receive the data that I needed. I successfully created all the elements needs to send and receive that data; however, I need to be able to create the weapons before sending that data to the server.

As seen in the success criteria I created the weapons, bullets, and magic animations privately first, meaning that only the client that did that attack could see the animations. This is imperative because I needed to first have the data about the animations, so I created the animations privately. Then once I had the animations I could effectively send and receive all the data which I successfully did. I believe that I was able to create and send data about the weapons, bullets shot or animations about each player.

Finally, the last step was to effectively turn the information from each player into graphical information that can be displayed on screen. I did this by reading through the list about a weapon, then making each weapon, bullet, or magic animation an object that can be displayed on screen in level.py. I believe I have effectively developed a system that can display the weapons, bullet, and magic animations on the screen.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

My stakeholders also like this sprint as they have given me positive feedback about each of my points about the game. In conclusion I believe that this sprint is successful as I have developed what I said I would develop, and I did it effectively to the delight of my stakeholders.

Next sprint

Next sprint, I want to add collisions between the players. I also want to add mechanisms to damage and kill other players with the weapons created in this sprint. I also want to update the kill count and player count that was created in sprint 3. Furthermore, I want to update the server to detect whether a player has been killed or not. I also want to ensure that players cannot harm themselves. In addition, I will add mechanisms to make sure that when players die, they cannot attack with a weapon or an animation and cannot heal themselves. In the next sprint I will also ensure that players will visually turn into ghosts that can be seen only by other ghosts. This is to ensure that the user requirements in my analysis is met and that I meet my stakeholders needs for this game. Lastly, I will try to add some optimisation into the game to ensure that the game runs smoothly with multiple player connections.

Copyright

For the copyright this sprint, I have used again elements from Clear Code. I have used class code from Clear Codes weapon and magic files. These files were necessary to create the animations. Although admittedly they are sourced from Clear Code. However, what I have added is the ability for magic and weapons to be seen by other players. This is what I have done this sprint. I am using and adapting portions of his code to make my game. Overall, for this sprint I have required many elements from his code although I have used those elements so that I could create the weapons visually be sent over the network.

Appendix

Pseudo code:

Client:

```
from Scripts.logger import *
try:
    import pygame, sys
    from Scripts.network import Network
    from Scripts.settings import Config
    from Scripts.level import Level
    from Scripts.player import Player
    from Scripts.weapon import *
    from Scripts.encryption import *
    from Scripts.debug import debug
    from Scripts.functions import *
    from Scripts.ui import UI
    from Scripts.main_menu import MainMenu
    from Scripts.magic import MagicPlayer
    from Scripts.particles import AnimationPlayer
except:
    logger.critical("You do not have all the modules installed. Please install Pygame, RSA and Pycryptodome.")

logger.info("RealDL - Client Code")
pygame.init()
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

class Client(Config):

  def update_players(self, player_dict):
    # Update existing player instances and remove players that are not in player_dict
    try:
      ### Players ###

      # Update players
      players_to_remove = []
      for player in self.players:
        if player.id in player_dict:
          player_data = player_dict[player.id]['player']
          player.rect.x = player_data['x']
          player.rect.y = player_data['y']
          player.health = player_data['health']
          try: player.image = pygame.image.load(player_data['image']).convert_alpha()
          except: player.image = pygame.image.load(f"../{player_data['image']}").convert_alpha()
        else:
          logger.debug(f"Removing player instance with id: {player.id}")
          players_to_remove.append(player)

      # Delete players.
      for player in players_to_remove:
        self.players.remove(player)
        self.level.visible_sprites.remove(player)

    # Create new player instances for players not already in self.players
    for player_data in player_dict.values():
      player_info = player_data['player']
      player_ids = [player.id for player in self.players]
      if player_info['id'] not in player_ids:
        logger.debug(f"Creating new player instance with ID: {player_info['id']}")
        new_player = Player(
          [player_info['x'], player_info['y']],
          player_info['image'],
          [self.level.visible_sprites],
          self.level.obstacle_sprites,
          player_info['username'],
          player_info['id'],
          player_info['health']
        )
        self.players.append(new_player)

    ### Weapons ###

    # Update Weapons
    weapons_to_remove = []
    for weapon in self.weapons:
      weapon_info = player_data['weapon']
      weapon_dict_ids = [weapon_dict['id'] for weapon_dict in weapon_info]
      weapon_dict_x = [weapon_dict['x'] for weapon_dict in weapon_info]
      weapon_dict_y = [weapon_dict['y'] for weapon_dict in weapon_info]
      weapon_dict_image = [weapon_dict['image'] for weapon_dict in weapon_info]
      if weapon.id in weapon_dict_ids:

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

weapon_index = weapon_dict_ids.index(weapon.id)
weapon.rect.x = weapon_dict_x[weapon_index]
weapon.rect.y = weapon_dict_y[weapon_index]
try: weapon.image = pygame.image.load(weapon_dict_image[weapon_index]).convert_alpha()
except: weapon.image =
pygame.image.load(f"../{weapon_dict_image[weapon_index]}").convert_alpha()
else:
  logger.debug(f"Removing weapon instance with id: {weapon.id}")
  weapons_to_remove.append(weapon)

# Delete weapons.
for weapon in weapons_to_remove:
  self.weapons.remove(weapon)
  self.level.visible_sprites.remove(weapon)

# Create weapons
for player_data in player_dict.values():
  weapon_info = player_data['weapon']
  weapon_ids = [weapon.id for weapon in self.weapons]
  weapon_dict_ids = [weapon_dict['id'] for weapon_dict in weapon_info]
  for weapon_id, weapon_data in zip(weapon_dict_ids, weapon_info):
    if weapon_id not in weapon_ids:
      logger.debug(f"Creating a new weapon with the ID: {weapon_data['id']}")
      new_weapon = Weapon([self.level.visible_sprites],
                          weapon_data['x'],
                          weapon_data['y'],
                          weapon_data['id'],
                          weapon_data['image'])
      self.weapons.append(new_weapon)

#### Bullets ####

# Update Bullets
bullets_to_remove = []
for bullet in self.bullets:
  bullet_info = player_data['bullets']
  bullet_dict_ids = [bullet_dict['id'] for bullet_dict in bullet_info]
  bullet_dict_x = [bullet_dict['x'] for bullet_dict in bullet_info]
  bullet_dict_y = [bullet_dict['y'] for bullet_dict in bullet_info]
  bullet_dict_image = [bullet_dict['image'] for bullet_dict in bullet_info]
  if bullet.id in bullet_dict_ids:
    bullet_index = bullet_dict_ids.index(bullet.id)
    bullet.rect.x = bullet_dict_x[bullet_index]
    bullet.rect.y = bullet_dict_y[bullet_index]
    try: bullet.image = pygame.image.load(bullet_dict_image[bullet_index]).convert_alpha()
    except: bullet.image = pygame.image.load(f"../{bullet_dict_image[bullet_index]}").convert_alpha()
  else:
    logger.debug(f"Removing bullet instance with id: {bullet.id}")
    bullets_to_remove.append(bullet)

# Delete Bullets
for bullet in bullets_to_remove:
  self.bullets.remove(bullet)
  self.level.visible_sprites.remove(bullet)
  
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
# Create Bullets
for player_data in player_dict.values():
    bullet_info = player_data['bullets']
    bullet_ids = [bullet.id for bullet in self.bullets]
    bullet_dict_ids = [bullet_dict['id'] for bullet_dict in bullet_info]
    for bullet_id, bullet_data in zip(bullet_dict_ids, bullet_info):
        if bullet_id not in bullet_ids:
            logger.debug(f"Creating a new weapon with the ID: {bullet_data['id']}")"
            new_bullet = Weapon([self.level.visible_sprites],
                                bullet_data['x'],
                                bullet_data['y'],
                                bullet_data['id'],
                                bullet_data['image'],
                                "bullet_copy")
            self.bullets.append(new_bullet)

### Magic ###

# Update Magic Animation

magic_to_remove = []
for magic in self.magic_animations:
    magic_info = player_data['magic']
    magic_dict_ids = [magic_dict['id'] for magic_dict in magic_info]
    magic_dict_x = [magic_dict['x'] for magic_dict in magic_info]
    magic_dict_y = [magic_dict['y'] for magic_dict in magic_info]
    magic_dict_image = [magic_dict['image'] for magic_dict in magic_info]
    if magic.id in magic_dict_ids:
        magic_index = magic_dict_ids.index(magic.id)
        magic.rect.x = magic_dict_x[magic_index]
        magic.rect.y = magic_dict_y[magic_index]
        try: magic.image = pygame.image.load(magic_dict_image[magic_index]).convert_alpha()
        except: magic.image = pygame.image.load(f"../{magic_dict_image[magic_index]}").convert_alpha()
    else:
        logger.debug(f"Removing bullet instance with id: {magic.id}")
        magic_to_remove.append(magic)

# Delete Bullets
for magic in magic_to_remove:
    self.magic_animations.remove(magic)
    self.level.visible_sprites.remove(magic)

# Create Magic Animation
for player_data in player_dict.values():
    magic_info = player_data['magic']
    magic_ids = [magic.id for magic in self.magic_animations]
    magic_dict_ids = [magic_dict['id'] for magic_dict in magic_info]
    for magic_id, magic_data in zip(magic_dict_ids, magic_info):
        if magic_id not in magic_ids:
            logger.debug(f"Creating a new Magic Animation with the ID: {magic_data['id']}")"
            new_magic_animation = Weapon([self.level.visible_sprites],
                                         magic_data['x'],
                                         magic_data['y'],
                                         magic_data['id'],
                                         magic_data['image'],
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

        "magic_copy")
self.magic_animations.append(new_magic_animation)

except:
    logger.error("Player disconnected.")
    self.close()

def redraw_window(self, all_players_dict):
    try:
        self.update_players(all_players_dict)
        self.level.run(self.player)
        self.ui.display(self.player)
        debug(f"Position: ({self.player.rect.x}, {self.player.rect.y})",self.WIDTH/2, 20)
        self.ui.draw_menu()
        if self.ui.draw_ui: self.player.paused = True
        else: self.player.paused = False
        pygame.display.update()
        self.clock.tick(self.FPS)
    except:
        logger.error("Player has left the game.")
        self.close()

def create_attack(self):
    logger.info("Create attack")
    try:
        self.current_attack = Melee(self.player, [self.level.visible_sprites, self.level.attack_sprites])
    except:
        self.current_attack = None
        logger.error("Failed to render attack image.")

def create_bullet(self):
    logger.info("Bullet!")
    self.bullet = Bullets(self.player, [self.level.visible_sprites, self.level.attack_sprites], self.level.obstacle_sprites)

def destroy_attack(self):
    logger.info("destroy attack.")
    try:
        if self.current_attack:
            self.current_attack.kill()
        self.current_attack = None
    except:
        self.current_attack = None
        logger.error("Failed to destroy attack.")

def create_magic(self, style, strength, cost):
    if style == 'heal':
        self.magic_player.heal(self.player, strength, cost, [self.level.visible_sprites])

    if style == 'flame':
        self.magic_player.flame(self.player, cost, [self.level.visible_sprites, self.level.attack_sprites])

def return_weapon(self):
    weapon = []
    for sprite in self.level.visible_sprites:

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

if sprite.sprite_type == "weapon":
    weapon_dict = {"x":sprite.rect.x,"y":sprite.rect.y,"image":sprite.full_path,"id":sprite.id}
    weapon.append(weapon_dict)

return weapon

def return_bullets(self):
    bullets = []

    for sprite in self.level.visible_sprites:
        if sprite.sprite_type == "bullet":
            bullet_dict = {"x":sprite.rect.x,
                           "y":sprite.rect.y,
                           "image":sprite.full_path,
                           "id":sprite.id}
            bullets.append(bullet_dict)
    return bullets

def return_magic_data(self):
    magic = []
    for sprite in self.level.visible_sprites:
        if sprite.sprite_type == "magic":
            magic_dict = {"x":sprite.rect.x,
                           "y":sprite.rect.y,
                           "image":sprite.return_image(),
                           "id":sprite.id}
            magic.append(magic_dict)
    return magic

def main(self):
    try:
        while self.running:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    self.quit()
                if event.type == pygame.KEYDOWN:
                    if event.key == pygame.K_ESCAPE:
                        self.ui.draw_ui = not self.ui.draw_ui

        try:
            # Get player data
            player_dict = {'x': self.player.rect.x,
                           'y': self.player.rect.y,
                           'image': self.player.get_image_name(),
                           'username':self.username,
                           'status':self.status,
                           'health':self.player.health,
                           'id': self.id}
            player_total_dict =
{'player':player_dict,'weapon':self.return_weapon(),'bullets':self.return_bullets(),'magic':self.return_magic_data()}
            player_encrypted_dict = self.serialize(self.encryption.encrypt(player_total_dict))
            self.network.send(player_encrypted_dict)
            all_players_dict = self.encryption.decrypt(self.deserialize(self.network.receive(self.DATA_SIZE)))

            logger.debug(f"Sending player data: {player_total_dict}")

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

        logger.debug(f"Received players dictionary: {all_players_dict}")

        self.redraw_window(all_players_dict)
    except:
        logger.error("Failed to send over player to server.")
        self.close()
    except:
        logger.error("Failed to run main loop.")
        self.close()

def run(self):
    self.gameMenu.run()
    user_dict = self.gameMenu.start_game()
    self.initialize_client(user_dict)
    self.main()

if __name__ == "__main__":
    try:
        client = Client()
        client.run()
    except:
        logger.error("Couldn't run main menu or client.")
    
```

Level:

```

from Scripts.logger import *
try
    import pygame
    from Scripts.settings import *
    from Scripts.tile import Tile
    from Scripts.support import *
    from random import choice
except:
    logger.critical("You do NOT have all the modules installed. Please install Pygame.")
endtry
logger.info("RealDL Level Code.")

class Level inherits Config
    private attackable_sprites
    private attack_sprites
    private current_attack
    private obstacle_sprites
    private visible_sprites
    private display_surface

    public procedure new()
        Config._init_()
        // get the display surface
        display_surface = new pygame.display.get_surface()
        // sprite group setup
        visible_sprites = new YSortCameraGroup()
        obstacle_sprites = new pygame.sprite.Group()
        // attack sprites
        current_attack = None
        attack_sprites = new pygame.sprite.Group()
        attackable_sprites = new pygame.sprite.Group()
    
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

// sprite setup
create_map()
endprocedure

class YSortCameraGroup inherits pygame.sprite.Group

public procedure custom_draw(player)
  // getting the offset
  offset.x = player.rect.centerx - half_width
  offset.y = player.rect.centery - half_height
  // drawing the floor
  display_surface.fill(settings.BG_COLOR)
  floor_offset_pos = floor_rect.topleft - offset
  display_surface.blit(floor_surf, floor_offset_pos)
  // for sprite in sprites():
  for sprite in sorted(sprites(), key= new lambda sprite sprite.rect.centery)
    offset_pos = sprite.rect.topleft - offset
    if sprite.sprite_type == "player" then
      if sprite.id == player.id then
        sprite.draw(offset_pos, (50, 190, 40))
      else
        sprite.draw(offset_pos)
    else
      endif
      display_surface.blit(sprite.image, offset_pos)
    endif
  next sprite
endprocedure
  
```

Magic:

```

from Scripts.logger import *
try
  import pygame, random
  from Scripts.settings import *

except
  logger.critical("You do NOT have all the modules installed. Please install Pygame.")
endtry
logger.debug("RealDL Magic Code.")

class MagicPlayer inherits Config
  private animation_player

  public procedure new(animation_player)
    Config.__init__()
    animation_player = animation_player
  endprocedure

  public procedure heal(player, strength, cost, groups)

    if player.energy >= cost then
      player.health += strength
      //player.energy -= cost
      if player.health >= player.stats['health'] then
  
```

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

        player.health = player.stats['health']
    endif
    animation_player.create_particles('aura',player.rect.center,groups)
    animation_player.create_particles('heal',player.rect.center,groups)
endif
endprocedure

public procedure flame(player,cost,groups)
    if player.energy >= cost then
        //player.energy -= cost
        if player.status.split('_')[0] == new == new 'right' then direction = new
pygame.math.Vector2(1,0)
        elseif player.status.split('_')[0] == new == new 'left' then direction = new
pygame.math.Vector2(-1,0)
        elseif player.status.split('_')[0] == new == new 'up' then direction = new
pygame.math.Vector2(0,-1)
        else direction = new pygame.math.Vector2(0,1)
    endif
    for i in range(1,6)
        if direction.x then //horizontal
            offset_x = new (direction.x * i) * TILESIZE
            x = new player.rect.centerx + offset_x + random.randint(-TILESIZE
DIV 3, TILESIZE DIV 3)
            y = new player.rect.centery + random.randint(-TILESIZE DIV 3,
TILESIZE DIV 3)
            animation_player.create_particles('flame',(x,y),groups)
        else // vertical
            offset_y = new (direction.y * i) * TILESIZE
            x = new player.rect.centerx + random.randint(-TILESIZE DIV 3,
TILESIZE DIV 3)
            y = new player.rect.centery + offset_y + random.randint(-TILESIZE
DIV 3, TILESIZE DIV 3)
            animation_player.create_particles('flame',(x,y),groups)
        endif
    next i
endif
endprocedure

```

Particles:

```

from Scripts.logger import *
try

```

```

    import pygame, string, random
    from Scripts.settings import Config
    from Scripts.support import import_folder

```

```

except

```

```

    logger.critical("You do NOT have all the modules installed. Please install Pygame.")
endtry

```

```

logger.debug("RealDL Magic Code.")


```

```

class AnimationPlayer
    private images
    private frames
    private particles

```

```

public procedure new()

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
particles = None
frames = {
    // magic
    'flame': import_folder('Graphics/Game/particles/flame/frames'),
    'aura': import_folder('Graphics/Game/particles/aura'),
    'heal': import_folder('Graphics/Game/particles/heal/frames'),
}
images = {
    // magic
    'flame': import_folder('Graphics/Game/particles/flame/frames',None),
    'aura': import_folder('Graphics/Game/particles/aura',None),
    'heal': import_folder('Graphics/Game/particles/heal/frames',None),
}

endprocedure

public procedure create_particles(animation_type,pos,groups)
    animation_frames = frames[animation_type]
    animation_images = images[animation_type]
    particles = new ParticleEffect(pos,animation_frames,animation_images,groups)
endprocedure

class ParticleEffect inherits pygame.sprite.Sprite
    private full_path
    private rect
    private image
    private images
    private frames
    private animation_speed
    private frame_index
    private id
    private sprite_type
    private settings

    public procedure new(pos,animation_frames,animation_images,groups)
        super.new(groups)
        settings = new Config()
        sprite_type = 'magic'
        id = new create_id()
        frame_index = 0
        animation_speed = 0.15
        frames = animation_frames
        images = animation_images
        image = frames[frame_index]
        rect = new image.get_rect(center = new pos)
    endprocedure

    function animate()
        frame_index += animation_speed
        if frame_index >= frames.length
            kill()
        else
            return_image()
            image = new frames[int(frame_index)]
    endfunction
```

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

endfunction

function return_image()
    full_path = new images[int(frame_index)]
    if "../" in full_path then
        full_path = new full_path.replace("../", "")
    endif
    return full_path
endfunction

function create_id()
    try
        characters = string.ascii_letters + string.digits
        return "join(random.choice(characters) for _ in range(settings.ID_STRING_LENGTH))"
    except
        logger.error("Something went wrong with creating a unique ID.")
    endtry
endfunction

public procedure update()
    animate()
endprocedure

```

Player:

```

from Scripts.logger import *
try
    import pygame
    from Scripts.support import import_folder
    from Scripts.settings import Config
except
    logger.critical("You do NOT have all the modules installed. Please install Pygame.")
endtry
logger.debug("RealDL Player Code.")

class Player inherits pygame.sprite.Sprite
    private animation_images
    private animations
    private obstacle_sprites
    private sprite_type
    private large_font
    private font
    private textSize
    private largefont
    private basefont
    private id
    private username
    private invulnerability_duration
    private hurt_time
    private vulnerable
    private exp
    private attack_strength
    private energy
    private health
    private upgrade_cost
    private max_stats

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
private stats
private magic_switch_time
private can_switch_magic
private magic
private create_magic
private magic_index
private switch_duration_cooldown
private weapon_switch_time
private can_switch_weapon
private weapon_type
private weapon
private weapon_index
private create_bullet
private destroy_attack
private create_attack
private paused
private attack_time
private attack_cooldown
private holding_magic_btn
private holding_attack_btn
private magic_attacking
private attacking
private speed
private direction
private animation_speed
private frame_index
private status
private key_binds
private magic_key
private offense
private movement
private hitbox
private rect
private image_name
private image
private screen
private settings

public procedure new(pos, image, groups, obstacle_sprites, username, id, health, create_attack= new None,
destroy_attack= new None, create_bullet= new None, create_magic= new None, movement= new "WASD",
offense= new "Space", magic= new "L-Shift") then
    // Setting up player
    try
        super.new(groups)
        settings = new Config()
        screen = new pygame.display.get_surface()
        try
            image = new pygame.image.load(image).convert_alpha()
            image_name = image
        except
            image = new pygame.image.load(f"../{image}").convert_alpha()
            image_name = f"../{image}"
        endtry
        rect = new image.get_rect(topleft = new pos)
        hitbox = new rect.inflate(0,-26)
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
movement = movement
offense = offense
magic_key = magic
key_binds = {
    'move_up' then pygame.K_w if movement == "WASD" else pygame.K_UP,
    'move_down' then pygame.K_s if movement == "WASD" else pygame.K_DOWN,
    'move_left' then pygame.K_a if movement == "WASD" else pygame.K_LEFT,
    'move_right' then pygame.K_d if movement == "WASD" else pygame.K_RIGHT,
    'attack' then pygame.K_SPACE if offense == "Space" else pygame.K_LCTRL if offense == "L-Ctrl" else
pygame.K_RCTRL,
    'magic' then pygame.K_LSHIFT if magic_key == "L-Shift" else pygame.K_RSHIFT if magic_key == "R-
Shift" else pygame.K_RETURN
}
// graphics setup
import_player_assets()
status = 'down'
frame_index = 0
animation_speed = 0.15
// Movement
direction = new pygame.math.Vector2()
speed = 5
attacking = False
magic_attacking = False
holding_attack_btn = False
holding_magic_btn = False
attack_cooldown = 400
attack_time = None
paused = False
// weapon
create_attack = create_attack
destroy_attack = destroy_attack
create_bullet = create_bullet
weapon_index = 0
weapon = new list(settings.weapon_data.keys())[weapon_index]
weapon_type = settings.weapon_data[weapon]['type']
can_switch_weapon = True
weapon_switch_time = None
switch_duration_cooldown = 200
// magic
magic_index = 0
create_magic = create_magic
magic = new list(settings.magic_data.keys())[magic_index]
can_switch_magic = True
magic_switch_time = None
// stats
stats = {'health': 100,'energy':60,'attack': 10,'magic': 4,'speed': 5}
max_stats = {'health': 300, 'energy': 140, 'attack': 20, 'magic' : 10, 'speed': 10}
upgrade_cost = {'health': 100, 'energy': 100, 'attack': 100, 'magic' : 100, 'speed': 100}
health = health
energy = stats['energy'] * 0.8
attack_strength = stats['attack']
exp = 500
speed = stats['speed']
// damage timer
vulnerable = True
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
hurt_time = None
invulnerability_duration = 500
//Other stats
username = username
id = id
basefont = "Graphics/Fonts/Orbitron-Medium.ttf"
largefont = "Graphics/Fonts/Orbitron-Bold.ttf"
textSize = 20
try
    font = new pygame.font.Font(basefont, textSize)
    large_font = new pygame.font.Font(largefont, textSize)
except
    font = new pygame.font.Font(f"../{basefont}", textSize)
    large_font = new pygame.font.Font(f"../{largefont}", textSize)
endtry
sprite_type = "player"
obstacle_sprites = obstacle_sprites
except
    logger.error("Failed to create Player Class")
endtry
endprocedure

public procedure input()
keys = new pygame.key.get_pressed()
if NOT paused then
    if NOT attacking AND NOT magic_attacking then
        if keys[key_binds['move_up']] then
            direction.y = -1
            status = 'up'
        endif
        elseif keys[key_binds['move_down']] then
            direction.y = 1
            status = 'down'
        else
            direction.y = 0
        endif
        if keys[key_binds['move_right']] then
            direction.x = 1
            status = 'right'
        endif
        elseif keys[key_binds['move_left']] then
            direction.x = -1
            status = 'left'
        else
            direction.x = 0
        endif
        if keys[key_binds['attack']] then
            attacking = True
            holding_attack_btn = True
            attack_time = new pygame.time.get_ticks()
            if create_attack != None then
                create_attack()
            if weapon_type == "gun" then
                if create_bullet then
                    create_bullet()
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

      endif
      endif
      endif
      logger.info('attack')
    endif
    // magic input
    if keys[key_binds['magic']] then
      magic_attacking = True
      holding_magic_btn = True
      attack_time = new pygame.time.get_ticks()
      style = new list(settings.magic_data.keys())[magic_index]
      strength = new list(settings.magic_data.values())[magic_index]['strength'] + stats['magic']
      cost = new list(settings.magic_data.values())[magic_index]['cost']
      if create_magic then
        create_magic(style,strength,cost)
      endif
      logger.info("magic")
    endif
    if keys[pygame.K_q] AND can_switch_weapon then
      can_switch_weapon = False
      weapon_switch_time = new pygame.time.get_ticks()
      if weapon_index < list(settings.weapon_data.keys().length)) - 1
        weapon_index += 1
      else
        weapon_index = 0
      endif
      weapon = new list(settings.weapon_data.keys())[weapon_index]
      weapon_type = settings.weapon_data[weapon]['type']
    endif
    if keys[pygame.K_e] AND can_switch_magic then
      can_switch_magic = False
      magic_switch_time = new pygame.time.get_ticks()
      if magic_index < list(settings.magic_data.keys().length)) - 1
        magic_index += 1
      else
        magic_index = 0
      endif
      magic = new list(settings.magic_data.keys())[magic_index]
    else
      endif
      if NOT keys[key_binds['attack']] then
        holding_attack_btn = False
      endif
      if NOT keys[key_binds['magic']] then
        holding_magic_btn = False
      endif
    endif
  endif
endprocedure

```

```

public procedure cooldowns()
  current_time = new pygame.time.get_ticks()
  if attacking then
    if current_time - attack_time >= attack_cooldown + settings.weapon_data[weapon]['cooldown'] then

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

if NOT holding_attack_btn then
  attacking = False
  if destroy_attack != None then
    destroy_attack()
  endif
endif
if current_time - attack_time >= settings.weapon_data[weapon]['cooldown'] then
  if holding_attack_btn AND weapon_type == "gun" then
    if create_bullet then
      create_bullet()
      attack_time = current_time
    endif
  endif
endif
if magic_attacking then
  if current_time - attack_time >= settings.magic_data[magic]['cooldown'] then
    if NOT holding_magic_btn then
      magic_attacking = False
    endif
  endif
endif
if NOT can_switch_weapon then
  if current_time - weapon_switch_time >= switch_duration_cooldown then
    can_switch_weapon = True
  endif
endif
if NOT can_switch_magic then
  if current_time - magic_switch_time >= switch_duration_cooldown then
    can_switch_magic = True
  endif
endif
if NOT vulnerable then
  if current_time - hurt_time >= invulnerability_duration then
    vulnerable = True
  endif
endif
endprocedure
  
```

Server:

```

from Scripts.logger import *
try
  from _thread import *
  import random
  import string, math, socket
  from Scripts.settings import Config
  from Scripts.encryption import *
  from Scripts.debug import debug

except
  logger.critical("You do NOT have all the modules installed. Please install Pygame, RSA AND Pycryptodome.")
endtry
logger.debug("RealDL Server Code.")
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

class Server inherits Config

function create_new_player(username)
  key_string = new create_random_string()
  player_x, player_y = new get_player_position()
  username = new username_validity(username)
  return {
    "player":{
      "x": player_x,
      "y": player_y,
      "image": "Graphics/Game/player/down_idle/idle_down.png",
      "username": username,
      "status": "alive",
      "health": 100,
      "id": key_string
    },
    "weapon":[],
    "bullets":[],
    "magic":[]}, key_string
endfunction

public procedure handle_client_communication(conn, key_string, aes_encryption)
  running = True
  player_pos = []
  while running
    try
      // Receive data from player and check it is valid.
      player_data = new aes_encryption.decrypt(unserialize(conn.recv(DATA_SIZE)))
      data = player_data['player']
      data['status'] = new validate_player_status(players[key_string]['player'], data)
      players[key_string] = player_data
      // Validate the player data to ensure they are not hacking.
      player_pos.append([players[key_string]['player']['x'], players[key_string]['player']['y']])
      if player_pos.length > 10 player_pos.pop(0)
      endif
      if NOT validate_movement(player_pos, players[key_string]['player']) then
        running = False
        logger.info(f"Player {key_string} disconnected because they were speed hacking.")
      endif
      players[key_string]['player']['health'] = new validate_health(players[key_string]['player'])
      logger.info(f"Received Player Dict: {player_data}.")
      if NOT data then
        logger.info(f"Player {key_string} disconnected.")
        running = False
      else
        reply = players
        encrypted_reply = new serialize(aes_encryption.encrypt(reply))
      endif
      conn.sendall(encrypted_reply)
      logger.info(f"Sending All Player Dict: {reply}.")
    except
      logger.info(f"Player {key_string} lost connection.")
      running = False
    endtry
  endwhile

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

logger.info(f"Connection Closed for Player {key_string}.")
try
  del players[key_string]
except
  logger.info(f"No player with the ID: {key_string} exists.")
endtry
connections -= 1
conn.close()
endprocedure

public procedure run()
  while running
    conn, addr = new s.accept()
    connections += 1
    logger.info(f"Connected to: {addr}")
    logger.info(f"There {'is' if connections= new = new 1 else 'are'} {connections} {'client' if connections= new = new 1 else 'clients'} connected to the server!")
  endif
  start_new_thread(threaded_client, (conn,))
  endwhile
endprocedure

if __name__ == "__main__":
  try
    server = new Server()
    server.run()
  except
    logger.critical("Server has failed to launch.")
  endtry
endif
  
```

Settings:

```

from Scripts.logger import *
try
  from socket import gethostname, gethostbyname
  import pickle, pygame

except
  logger.critical("You do NOT have all the modules installed. Please install Pygame.")
endtry
pygame.init()
logger.info("Github: https://github.com/TheRealDL1/Simple-Client-Server")
logger.debug("RealDL Settings Code.")

class Config
  private magic_data
  private bullet_type
  private weapon_data

  public procedure new()
    // weapons
    weapon_data = {
      'sword': {'cooldown': 100, 'speed': 0, 'damage':
15,'graphic':'Graphics/Game/weapons/sword/full.png','type':'melee'},
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

'lance': {'cooldown': 400, 'speed': 0, 'damage':
30,'graphic':'Graphics/Game/weapons/lance/full.png','type':'melee'},
  'axe': {'cooldown': 300, 'speed': 0, 'damage': 20,
'graphic':'Graphics/Game/weapons/axe/full.png','type':'melee'},
  'rapier': {'cooldown': 50, 'speed': 0, 'damage': 8,
'graphic':'Graphics/Game/weapons/rapier/full.png','type':'melee'},
  'sai': {'cooldown': 80, 'speed': 0, 'damage': 10,
'graphic':'Graphics/Game/weapons/sai/full.png','type':'melee'},
  'revolver': {'cooldown': 1000, 'speed': 20, 'damage': 25,
'graphic':'Graphics/Game/weapons/revolver/full.png','type':'gun'},
  'msg': {'cooldown': 175, 'speed': 10, 'damage': 5,
'graphic':'Graphics/Game/weapons/msg/full.png','type':'gun'},
  'pistol': {'cooldown': 600, 'speed': 15, 'damage': 18,
'graphic':'Graphics/Game/weapons/pistol/full.png','type':'gun'}}}

// Bullets
bullet_type = {
  'msg': {
    'down': 'Graphics/Game/bullets/msg/down.png',
    'left': 'Graphics/Game/bullets/msg/left.png',
    'right': 'Graphics/Game/bullets/msg/right.png',
    'up': 'Graphics/Game/bullets/msg/up.png',
  },
  'pistol': {
    'down': 'Graphics/Game/bullets/pistol/down.png',
    'left': 'Graphics/Game/bullets/pistol/left.png',
    'right': 'Graphics/Game/bullets/pistol/right.png',
    'up': 'Graphics/Game/bullets/pistol/up.png',
  },
  'revolver': {
    'down': 'Graphics/Game/bullets/revolver/down.png',
    'left': 'Graphics/Game/bullets/revolver/left.png',
    'right': 'Graphics/Game/bullets/revolver/right.png',
    'up': 'Graphics/Game/bullets/revolver/up.png',
  }
}
// magic
magic_data = {
  'flame': {'strength': 5, 'cost':
20,'graphic':'Graphics/Game/particles/flame/fire.png','type':'magic','cooldown':1000},
  'heal': {'strength': 20, 'cost':
10,'graphic':'Graphics/Game/particles/heal/heal.png','type':'magic','cooldown':800}}
endprocedure
  
```

Weapon:

```

from Scripts.logger import *
try
  import pygame, string, random
  from Scripts.settings import *

except
  logger.critical("You do NOT have all the modules installed. Please install Pygame.")
endtry
logger.debug("RealDL Weapon Code.")

class Melee inherits pygame.sprite.Sprite
  
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
private rect
private image
private full_path
private last_shot_time
private time
private id
private settings
private sprite_type

public procedure new(player, groups)
    try
        super.new(groups)
        sprite_type = "weapon"
        settings = new Config()
        id = new create_id()
        time = new pygame.time.get_ticks()
        last_shot_time = 0 // Initialize the last shot time to 0
        direction = new player.status.split('_')[0]
        // Load the image
        full_path = f'Graphics/Game/weapons/{player.weapon}/{direction}.png'
        try
            image = new pygame.image.load(full_path).convert_alpha()
        except
            image = new pygame.image.load(f"../{full_path}").convert_alpha()
        endtry
        // Set the placement of the sprite
        if direction == 'right' then
            rect = new image.get_rect(midleft= newplayer.rect.midright + pygame.math.Vector2(0, 16))
        endif
        elseif direction == 'left' then
            rect = new image.get_rect(midright= newplayer.rect.midleft + pygame.math.Vector2(0, 16))
        endif
        elseif direction == 'down' then
            rect = new image.get_rect(midtop= newplayer.rect.midbottom + pygame.math.Vector2(-10, 0))
        else
            rect = new image.get_rect(midbottom= newplayer.rect.midtop + pygame.math.Vector2(-10, 0))
        endif
    except
        logger.error("Failed to create Weapon")
    endtry
endprocedure

function create_id()
    try
        characters = string.ascii_letters + string.digits
        return ".join(random.choice(characters) for _ in range(settings.ID_STRING_LENGTH))"
    except
        logger.error("Something went wrong with creating a unique ID.")
    endtry
endfunction

class Weapon inherits pygame.sprite.Sprite
    private rect.y
    private rect.x
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
private rect
private image
private full_path
private y
private x
private id
private sprite_type

public procedure new(groups, x, y, id, image, sprite_type= new "weapon_copy")
    super.new(groups)
    sprite_type = sprite_type
    id = id
    x = x
    y = y
    full_path = image
    try
        image = new pygame.image.load(full_path).convert_alpha()
    except
        image = new pygame.image.load(f"../{full_path}").convert_alpha()
    endtry
    rect = new image.get_rect()
    rect.x = x
    rect.y = y
endprocedure

class Bullets inherits pygame.sprite.Sprite
    private rect.y
    private rect.x
    private rect
    private image
    private full_path
    private collided
    private cooldown
    private damage
    private speed
    private direction
    private groups
    private player
    private id
    private sprite_type
    private obstacle_sprites
    private settings

    public procedure new(player, groups, obstacle_sprites)
        super.new(groups)
        settings = new Config()
        obstacle_sprites = obstacle_sprites
        sprite_type = "bullet"
        id = new create_id()
        player = player
        groups = groups
        direction = new player.status.split('_')[0]
        gun = player.weapon
        speed = settings.weapon_data[gun]['speed']
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

damage = player.attack_strength + settings.weapon_data[gun]['damage']
cooldown = settings.weapon_data[gun]['cooldown']
collided = False
full_path = settings.bullet_type[gun][direction]
try
  image = new pygame.image.load(full_path).convert_alpha()
except
  image = new pygame.image.load(f"../{full_path}").convert_alpha()
endtry
// Set the placement of the sprite
if direction == 'right' then
  rect = new image.get_rect(midleft= newplayer.rect.midright + pygame.math.Vector2(20, 8))
endif
elseif direction == 'left' then
  rect = new image.get_rect(midright= newplayer.rect.midleft + pygame.math.Vector2(-20, 8))
endif
elseif direction == 'down' then
  rect = new image.get_rect(midtop= newplayer.rect.midbottom + pygame.math.Vector2(-18, 21))
else
  rect = new image.get_rect(midbottom= newplayer.rect.midtop + pygame.math.Vector2(-18, -21))
endif
endprocedure

public procedure update()
collision('horizontal')
collision('vertical')
if NOT collided then
  if direction == 'right' then
    rect.x += speed
  endif
  elseif direction == 'left' then
    rect.x -= speed
  endif
  elseif direction == 'down' then
    rect.y += speed
  else
    rect.y -= speed
  endif
endif
if collided then
  kill()
endif
endprocedure

function create_id()
try
  characters = string.ascii_letters + string.digits
  return "join(random.choice(characters) for _ in range(settings.ID_STRING_LENGTH))"
except
  logger.error("Something went wrong with creating a unique ID.")
endtry
endfunction

public procedure collision(direction)
  if direction == 'horizontal' then

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
for sprite in obstacle_sprites
    if sprite.hitbox.colliderect(rect) then
        collided = True
    endif
next sprite
endif
if direction == 'vertical' then
    for sprite in obstacle_sprites
        if sprite.hitbox.colliderect(rect) then
            collided = True
        endif
    next sprite
endif
// You may want to set its initial position and direction here
endprocedure
```

Python code:

Client:

```
from Scripts.logger import *
try: ...
except:
    logger.critical("You do not have all the modules installed. Please install Pygame, RSA and Pycryptodome.")

logger.info("RealDL - Client Code")
pygame.init()

class Client(Config):
    def __init__(self): ...

    def initialize_client(self, user_dict): ...

    def initialize_pygame(self, user_dict):
        try:
            self.players = []
            self.bullets = []
            self.weapons = []
            self.magic_animations = []
            self.player_count = len(self.players)
            self.kill_count = 0
            self.level = Level()
            # particles
            self.animation_player = AnimationPlayer()
            self.magic_player = MagicPlayer(self.animation_player)
            self.ui = UI(self.custom_mouse, self.close, self.player_count, self.kill_count)
            self.running = True

            # attack sprites
            self.current_attack = None
            self.bullet = None

        except:
            logger.error("Couldn't correctly initialize pygame.")
            self.close()

    def initialize_network(self): ...

    def initialize_player(self, player_info): ...
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

def update_players(self, player_dict):
    # Update existing player instances and remove players that are not in player_dict
    try:
        ### Players ####

        # Update players
        players_to_remove = []
        for player in self.players:
            if player.id in player_dict:
                player_data = player_dict[player.id]['player']
                player.rect.x = player_data['x']
                player.rect.y = player_data['y']
                player.health = player_data['health']
                try: player.image = pygame.image.load(player_data['image']).convert_alpha()
                except: player.image = pygame.image.load(f"../{player_data['image']}").convert_alpha()
            else:
                logger.debug(f"Removing player instance with id: {player.id}")
                players_to_remove.append(player)

        # Delete players.
        for player in players_to_remove:
            self.players.remove(player)
            self.level.visible_sprites.remove(player)

        # Create new player instances for players not already in self.players
        for player_data in player_dict.values():
            player_info = player_data['player']
            player_ids = [player.id for player in self.players]
            if player_info['id'] not in player_ids:
                logger.debug(f"Creating new player instance with ID: {player_info['id']}")
                new_player = Player(
                    [player_info['x'], player_info['y']],
                    player_info['image'],
                    [self.level.visible_sprites],
                    self.level.obstacle_sprites,
                    player_info['username'],
                    player_info['id'],
                    player_info['health']
                )
                self.players.append(new_player)

    # Update Weapons
    weapons_to_remove = []
    for weapon in self.weapons:
        weapon_info = player_data['weapon']
        weapon_dict_ids = [weapon_dict['id'] for weapon_dict in weapon_info]
        weapon_dict_x = [weapon_dict['x'] for weapon_dict in weapon_info]
        weapon_dict_y = [weapon_dict['y'] for weapon_dict in weapon_info]
        weapon_dict_image = [weapon_dict['image'] for weapon_dict in weapon_info]
        if weapon.id in weapon_dict_ids:
            weapon_index = weapon_dict_ids.index(weapon.id)
            weapon.rect.x = weapon_dict_x[weapon_index]
            weapon.rect.y = weapon_dict_y[weapon_index]
            try: weapon.image = pygame.image.load(weapon_dict_image[weapon_index]).convert_alpha()
            except: weapon.image = pygame.image.load(f"../{weapon_dict_image[weapon_index]}").convert_alpha()
        else:
            logger.debug(f"Removing weapon instance with id: {weapon.id}")
            weapons_to_remove.append(weapon)

    # Delete weapons.
    for weapon in weapons_to_remove:
        self.weapons.remove(weapon)
        self.level.visible_sprites.remove(weapon)

    # Create weapons
    for player_data in player_dict.values():
        weapon_info = player_data['weapon']
        weapon_ids = [weapon.id for weapon in self.weapons]
        weapon_dict_ids = [weapon_dict['id'] for weapon_dict in weapon_info]
        for weapon_id, weapon_data in zip(weapon_dict_ids, weapon_info):
            if weapon_id not in weapon_ids:
                logger.debug(f"Creating a new weapon with the ID: {weapon_data['id']}")
                new_weapon = Weapon([self.level.visible_sprites],
                                    weapon_data['x'],
                                    weapon_data['y'],
                                    weapon_data['id'],
                                    weapon_data['image'])
                self.weapons.append(new_weapon)
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

### Bullets ###

# Update Bullets
bullets_to_remove = []
for bullet in self.bullets:
  bullet_info = player_data['bullets']
  bullet_dict_ids = [bullet_dict['id'] for bullet_dict in bullet_info]
  bullet_dict_x = [bullet_dict['x'] for bullet_dict in bullet_info]
  bullet_dict_y = [bullet_dict['y'] for bullet_dict in bullet_info]
  bullet_dict_image = [bullet_dict['image'] for bullet_dict in bullet_info]
  if bullet.id in bullet_dict_ids:
    bullet_index = bullet_dict_ids.index(bullet.id)
    bullet.rect.x = bullet_dict_x[bullet_index]
    bullet.rect.y = bullet_dict_y[bullet_index]
    try: bullet.image = pygame.image.load(bullet_dict_image[bullet_index]).convert_alpha()
    except: bullet.image = pygame.image.load(f"../{bullet_dict_image[bullet_index]}").convert_alpha()
  else:
    logger.debug(f"Removing bullet instance with id: {bullet.id}")
    bullets_to_remove.append(bullet)

# Delete Bullets
for bullet in bullets_to_remove:
  self.bullets.remove(bullet)
  self.level.visible_sprites.remove(bullet)

# Create Bullets
for player_data in player_dict.values():
  bullet_info = player_data['bullets']
  bullet_ids = [bullet.id for bullet in self.bullets]
  bullet_dict_ids = [bullet_dict['id'] for bullet_dict in bullet_info]
  for bullet_id, bullet_data in zip(bullet_dict_ids, bullet_info):
    if bullet_id not in bullet_ids:
      logger.debug(f"Creating a new weapon with the ID: {bullet_data['id']}")
      new_bullet = Weapon([self.level.visible_sprites],
                          bullet_data['x'],
                          bullet_data['y'],
                          bullet_data['id'],
                          bullet_data['image'],
                          "bullet_copy")
      self.bullets.append(new_bullet)

### Magic ###

# Update Magic Animation

magic_to_remove = []
for magic in self.magic_animations:
  magic_info = player_data['magic']
  magic_dict_ids = [magic_dict['id'] for magic_dict in magic_info]
  magic_dict_x = [magic_dict['x'] for magic_dict in magic_info]
  magic_dict_y = [magic_dict['y'] for magic_dict in magic_info]
  magic_dict_image = [magic_dict['image'] for magic_dict in magic_info]
  if magic.id in magic_dict_ids:
    magic_index = magic_dict_ids.index(magic.id)
    magic.rect.x = magic_dict_x[magic_index]
    magic.rect.y = magic_dict_y[magic_index]
    try: magic.image = pygame.image.load(magic_dict_image[magic_index]).convert_alpha()
    except: magic.image = pygame.image.load(f"../{magic_dict_image[magic_index]}").convert_alpha()
  else:
    logger.debug(f"Removing bullet instance with id: {magic.id}")
    magic_to_remove.append(magic)

# Delete Bullets
for magic in magic_to_remove:
  self.magic_animations.remove(magic)
  self.level.visible_sprites.remove(magic)

# Create Magic Animation
for player_data in player_dict.values():
  magic_info = player_data['magic']
  magic_ids = [magic.id for magic in self.magic_animations]
  magic_dict_ids = [magic_dict['id'] for magic_dict in magic_info]
  for magic_id, magic_data in zip(magic_dict_ids, magic_info):
    if magic_id not in magic_ids:
      logger.debug(f"Creating a new Magic Animation with the ID: {magic_data['id']}")
      new_magic_animation = Weapon([self.level.visible_sprites],
                                  magic_data['x'],
                                  magic_data['y'],
                                  magic_data['id'],
                                  magic_data['image'],
                                  "magic_copy")
      self.magic_animations.append(new_magic_animation)

except:
  logger.error("Player disconnected.")
  self.close()

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

def redraw_window(self, all_players_dict):
    try:
        self.update_players(all_players_dict)
        self.level.run(self.player)
        self.ui.display(self.player)
        debug(f"Position: ({self.player.rect.x}, {self.player.rect.y})", self.WIDTH/2, 20)
        self.ui.draw_menu()
        if self.ui.draw_ui: self.player.paused = True
        else: self.player.paused = False
        pygame.display.update()
        self.clock.tick(self.FPS)
    except:
        logger.error("Player has left the game.")
        self.close()

def create_attack(self):
    logger.info("Create attack")
    try:
        self.current_attack = Melee(self.player, [self.level.visible_sprites, self.level.attack_sprites])
    except:
        self.current_attack = None
        logger.error("Failed to render attack image.")

def create_bullet(self):
    logger.info("Bullet!")
    self.bullet = Bullets(self.player, [self.level.visible_sprites, self.level.attack_sprites], self.level.obstacle_sprites)

def destroy_attack(self):
    logger.info("destroy attack")
    try:
        if self.current_attack:
            self.current_attack.kill()
        self.current_attack = None
    except:
        self.current_attack = None
        logger.error("Failed to destroy attack.")

def create_magic(self, style, strength, cost):
    if style == 'heal':
        self.magic_player.heal(self.player, strength, cost, [self.level.visible_sprites])
    if style == 'flame':
        self.magic_player.flame(self.player, cost, [self.level.visible_sprites, self.level.attack_sprites])

def return_weapon(self):
    weapon = []
    for sprite in self.level.visible_sprites:
        if sprite.sprite_type == "weapon":
            weapon_dict = {"x":sprite.rect.x, "y":sprite.rect.y, "image":sprite.full_path, "id":sprite.id}
            weapon.append(weapon_dict)

    return weapon

def return_bullets(self):
    bullets = []

    for sprite in self.level.visible_sprites:
        if sprite.sprite_type == "bullet":
            bullet_dict = {"x":sprite.rect.x,
                          "y":sprite.rect.y,
                          "image":sprite.full_path,
                          "id":sprite.id}
            bullets.append(bullet_dict)

    return bullets

def return_magic_data(self):
    magic = []
    for sprite in self.level.visible_sprites:
        if sprite.sprite_type == "magic":
            magic_dict = {"x":sprite.rect.x,
                          "y":sprite.rect.y,
                          "image":sprite.return_image(),
                          "id":sprite.id}
            magic.append(magic_dict)

    return magic
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

def main(self):
    try:
        while self.running:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    self.quit()
                if event.type == pygame.KEYDOWN:
                    if event.key == pygame.K_ESCAPE:
                        self.ui.draw_ui = not self.ui.draw_ui

        try:
            # Get player data
            player_dict = {'x': self.player.rect.x,
                           'y': self.player.rect.y,
                           'image': self.player.get_image_name(),
                           'username': self.username,
                           'status': self.status,
                           'health': self.player.health,
                           'id': self.id}
            player_total_dict = {'player': player_dict, 'weapon': self.return_weapon(), 'bullets': self.return_bullets(), 'magic': self.return_magic_data()}
            player_encrypted_dict = self.serialize(self.encryption.encrypt(player_total_dict))
            self.network.send(player_encrypted_dict)
            all_players_dict = self.encryption.decrypt(self.deserialize(self.network.receive(self.DATA_SIZE)))

            logger.debug(f"Sending player data: {player_total_dict}")
            logger.debug(f"Received players dictionary: {all_players_dict}")

            self.redraw_window(all_players_dict)
        except:
            logger.error("Failed to send over player to server.")
            self.close()
    except:
        logger.error("Failed to run main loop.")
        self.close()

    def run(self):
        self.gameMenu.run()
        user_dict = self.gameMenu.start_game()
        self.initialize_client(user_dict)
        self.main()

if __name__ == "__main__":
    try:
        client = Client()
        client.run()
    except:
        logger.error("Couldn't run main menu or client.")
  
```

Level:

```

from Scripts.logger import *
try: ...
except:
    logger.critical("You do not have all the modules installed. Please install Pygame.")

logger.info("RealDL Level Code.")

class Level(Config):
    def __init__(self):
        Config.__init__(self)

        # get the display surface
        self.display_surface = pygame.display.get_surface()

        # sprite group setup
        self.visible_sprites = YSortCameraGroup()
        self.obstacle_sprites = pygame.sprite.Group()

        # attack sprites
        self.current_attack = None
        self.attack_sprites = pygame.sprite.Group()
        self.attackable_sprites = pygame.sprite.Group()
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

class YSortCameraGroup(pygame.sprite.Group):
    def __init__(self): ...

    def custom_draw(self, player):
        # getting the offset
        self.offset.x = player.rect.centerx - self.half_width
        self.offset.y = player.rect.centery - self.half_height

        # drawing the floor
        self.display_surface.fill(self.settings.BG_COLOR)
        floor_offset_pos = self.floor_rect.topleft - self.offset
        self.display_surface.blit(self.floor_surf, floor_offset_pos)

        # for sprite in self.sprites():
        for sprite in sorted(self.sprites(), key=lambda sprite: sprite.rect.centery):
            offset_pos = sprite.rect.topleft - self.offset
            if sprite.sprite_type == "player":
                if sprite.id == player.id:
                    sprite.draw(offset_pos, (50, 190, 40))
                else:
                    sprite.draw(offset_pos)
            else:
                self.display_surface.blit(sprite.image, offset_pos)
  
```

Magic:

```

from Scripts.logger import *
try:
    import pygame, random
    from Scripts.settings import *
except:
    logger.critical("You do not have all the modules installed. Please install Pygame.")

logger.debug("RealDL Magic Code.")

class MagicPlayer(Config):
    def __init__(self, animation_player):
        Config.__init__(self)
        self.animation_player = animation_player

    def heal(self, player, strength, cost, groups):
        if player.energy >= cost:
            player.health += strength
            #player.energy -= cost
            if player.health >= player.stats['health']:
                player.health = player.stats['health']
            self.animation_player.create_particles('aura', player.rect.center, groups)
            self.animation_player.create_particles('heal', player.rect.center, groups)

    def flame(self, player, cost, groups):
        if player.energy >= cost:
            #player.energy -= cost

            if player.status.split('_')[0] == 'right': direction = pygame.math.Vector2(1,0)
            elif player.status.split('_')[0] == 'left': direction = pygame.math.Vector2(-1,0)
            elif player.status.split('_')[0] == 'up': direction = pygame.math.Vector2(0,-1)
            else: direction = pygame.math.Vector2(0,1)

            for i in range(1,6):
                if direction.x: #horizontal
                    offset_x = (direction.x * i) * self.TILESIZE
                    x = player.rect.centerx + offset_x + random.randint(-self.TILESIZE // 3, self.TILESIZE // 3)
                    y = player.rect.centery + random.randint(-self.TILESIZE // 3, self.TILESIZE // 3)
                    self.animation_player.create_particles('flame',(x,y),groups)
                else: # vertical
                    offset_y = (direction.y * i) * self.TILESIZE
                    x = player.rect.centerx + random.randint(-self.TILESIZE // 3, self.TILESIZE // 3)
                    y = player.rect.centery + offset_y + random.randint(-self.TILESIZE // 3, self.TILESIZE // 3)
                    self.animation_player.create_particles('flame',(x,y),groups)
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

Particles:

```

from Scripts.logger import *
try:
    import pygame, string, random
    from Scripts.settings import Config
    from Scripts.support import import_folder
except:
    logger.critical("You do not have all the modules installed. Please install Pygame.")

logger.debug("RealDL Magic Code.")

class AnimationPlayer:
    def __init__(self):
        self.particles = None
        self.frames = {
            # magic
            'flame': import_folder('Graphics/Game/particles/flame/frames'),
            'aura': import_folder('Graphics/Game/particles/aura'),
            'heal': import_folder('Graphics/Game/particles/heal/frames'),
        }

        self.images = {
            # magic
            'flame': import_folder('Graphics/Game/particles/flame/frames',None),
            'aura': import_folder('Graphics/Game/particles/aura',None),
            'heal': import_folder('Graphics/Game/particles/heal/frames',None),
        }

    def create_particles(self,animation_type,pos,groups):
        animation_frames = self.frames[animation_type]
        animation_images = self.images[animation_type]
        self.particles = ParticleEffect(pos,animation_frames,animation_images,groups)

class ParticleEffect(pygame.sprite.Sprite):
    def __init__(self,pos,animation_frames,animation_images,groups):
        super().__init__(groups)
        self.settings = Config()
        self.sprite_type = 'magic'
        self.id = self.create_id()
        self.frame_index = 0
        self.animation_speed = 0.15
        self.frames = animation_frames
        self.images = animation_images
        self.image = self.frames[self.frame_index]
        self.rect = self.image.get_rect(center = pos)

    def animate(self):
        self.frame_index += self.animation_speed
        if self.frame_index >= len(self.frames):
            self.kill()
        else:
            self.return_image()
            self.image = self.frames[int(self.frame_index)]

    def return_image(self):
        self.full_path = self.images[int(self.frame_index)]
        if "../" in self.full_path:
            self.full_path = self.full_path.replace("../","")
        return self.full_path

    def create_id(self):
        try:
            characters = string.ascii_letters + string.digits
            return ''.join(random.choice(characters) for _ in range(self.settings.ID_STRING_LENGTH))
        except:
            logger.error("Something went wrong with creating a unique ID.")

    def update(self):
        self.animate()
    |
  
```

Player:

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

from Scripts.logger import *
try:
except:
  logger.critical("You do not have all the modules installed. Please install Pygame.")

logger.debug("RealDL Player Code.")

class Player(pygame.sprite.Sprite):
  def __init__(self, pos, image, groups, obstacle_sprites, username, id, health, create_attack=None, destroy_attack=None, create_bullet=None,
               create_magic=None, movement="WASD", offense="Space", magic="L-Shift"):
    # Setting up player
    try:
      super().__init__(groups)
      self.settings = Config()
      self.screen = pygame.display.get_surface()
      try:
        self.image = pygame.image.load(image).convert_alpha()
        self.image_name = image
      except:
        self.image = pygame.image.load(f"../{image}").convert_alpha()
        self.image_name = f"../{image}"
      self.rect = self.image.get_rect(topleft=pos)
      self.hitbox = self.rect.inflate(0,-26)
      self.movement = movement
      self.offense = offense
      self.magic_key = magic
      self.key_binds = {
        'move_up': pygame.K_w if self.movement == "WASD" else pygame.K_UP,
        'move_down': pygame.K_s if self.movement == "WASD" else pygame.K_DOWN,
        'move_left': pygame.K_a if self.movement == "WASD" else pygame.K_LEFT,
        'move_right': pygame.K_d if self.movement == "WASD" else pygame.K_RIGHT,
        'attack': pygame.K_SPACE if self.offense == "Space" else pygame.K_LCTRL if self.offense == "L-Ctrl" else pygame.K_RCTRL,
        'magic': pygame.K_LSHIFT if self.magic_key == "L-Shift" else pygame.K_RSHIFT if self.magic_key == "R-Shift" else pygame.K_RETURN
      }

      # graphics setup
      self.import_player_assets()
      self.status = 'down'
      self.frame_index = 0
      self.animation_speed = 0.15

    # Movement
    self.direction = pygame.math.Vector2()
    self.speed = 5
    self.attacking = False
    self.magic_attacking = False
    self.holding_attack_btn = False
    self.holding_magic_btn = False
    self.attack_cooldown = 400
    self.attack_time = None
    self.paused = False

    # weapon
    self.create_attack = create_attack
    self.destroy_attack = destroy_attack
    self.create_bullet = create_bullet
    self.weapon_index = 0
    self.weapon = list(self.settings.weapon_data.keys())[self.weapon_index]
    self.weapon_type = self.settings.weapon_data[self.weapon]['type']
    self.can_switch_weapon = True
    self.weapon_switch_time = None
    self.switch_duration_cooldown = 200

    # magic
    self.magic_index = 0
    self.create_magic = create_magic
    self.magic = list(self.settings.magic_data.keys())[self.magic_index]
    self.can_switch_magic = True
    self.magic_switch_time = None

    # stats
    self.stats = {'health': 100, 'energy': 60, 'attack': 10, 'magic': 4, 'speed': 5}
    self.max_stats = {'health': 300, 'energy': 140, 'attack': 20, 'magic': 10, 'speed': 10}
    self.upgrade_cost = {'health': 100, 'energy': 100, 'attack': 100, 'magic': 100, 'speed': 100}
    self.health = health
    self.energy = self.stats['energy'] * 0.8
    self.attack_strength = self.stats['attack']
    self.exp = 500
    self.speed = self.stats['speed']

    # damage timer
    self.vulnerable = True
    self.hurt_time = None
    self.invulnerability_duration = 500
  
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
#Other stats
self.username = username
self.id = id
self.basefont = "Graphics/Fonts/Orbitron-Medium.ttf"
self.largefont = "Graphics/Fonts/Orbitron-Bold.ttf"
self.textSize = 20
try:
    self.font = pygame.font.Font(self.basefont, self.textSize)
    self.large_font = pygame.font.Font(self.largefont, self.textSize)
except:
    self.font = pygame.font.Font(f"../{self.basefont}", self.textSize)
    self.large_font = pygame.font.Font(f"../{self.largefont}", self.textSize)

self.sprite_type = "player"

self.obstacle_sprites = obstacle_sprites
except:
    logger.error("Failed to create Player Class")

def import_player_assets(self): ...

def input(self):
    keys = pygame.key.get_pressed()
    if not self.paused:
        if not self.attacking and not self.magic_attacking: ...

        else:
            if not keys[self.key_binds['attack']]:
                self.holding_attack_btn = False
            if not keys[self.key_binds['magic']]:
                self.holding_magic_btn = False

def cooldowns(self):
    current_time = pygame.time.get_ticks()

    if self.attacking:
        if current_time - self.attack_time >= self.attack_cooldown + self.settings.weapon_data[self.weapon]['cooldown']:
            if not self.holding_attack_btn:
                self.attacking = False
                if self.destroy_attack != None:
                    self.destroy_attack()
        if current_time - self.attack_time >= self.settings.weapon_data[self.weapon]['cooldown']:
            if self.holding_attack_btn and self.weapon_type == "gun":
                if self.create_bullet:
                    self.create_bullet()
                    self.attack_time = current_time

    if self.magic_attacking:
        if current_time - self.attack_time >= self.settings.magic_data[self.magic]['cooldown']:
            if not self.holding_magic_btn:
                self.magic_attacking = False

    if not self.can_switch_weapon:
        if current_time - self.weapon_switch_time >= self.switch_duration_cooldown:
            self.can_switch_weapon = True

    if not self.can_switch_magic:
        if current_time - self.magic_switch_time >= self.switch_duration_cooldown:
            self.can_switch_magic = True

    if not self.vulnerable:
        if current_time - self.hurt_time >= self.invulnerability_duration:
            self.vulnerable = True
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

Server:

```

from Scripts.logger import *
try: ...
except:
    logger.critical("You do not have all the modules installed. Please install Pygame, RSA and Pycryptodome.")

logger.debug("RealDL Server Code.")

class Server(Config):
    def __init__(self):
        try:
            Config.__init__(self)
            self.initialize_server()
        except:
            logger.error(f"Couldn't initialize server.")

    def initialize_server(self): ...

    def create_random_string(self): ...

    def is_touching(self, x, y, other_x, other_y, threshold): ...

    def get_player_position(self): ...

    def create_new_player(self, username):
        key_string = self.create_random_string()
        player_x, player_y = self.get_player_position()
        username = self.username_validity(username)
        return {
            "player": {
                "x": player_x,
                "y": player_y,
                "image": "Graphics/Game/player/down_idle/idle_down.png",
                "username": username,
                "status": "alive",
                "health": 100,
                "id": key_string
            },
            "weapon": [],
            "bullets": [],
            "magic": []
        }, key_string

    def handle_client_communication(self, conn, key_string, aes_encryption):
        running = True
        player_pos = []
        while running:
            try:
                # Receive data from player and check it is valid.
                player_data = aes_encryption.decrypt(self.deserialize(conn.recv(self.DATA_SIZE)))
                data = player_data['player']
                data['status'] = self.validate_player_status(self.players[key_string]['player'], data)
                self.players[key_string] = player_data

                # Validate the player data to ensure they are not hacking.
                player_pos.append([self.players[key_string]['player']['x'], self.players[key_string]['player']['y']])
                if len(player_pos) > 10: player_pos.pop(0)
                if not self.validate_movement(player_pos, self.players[key_string]['player']):
                    running = False
                    logger.info(f"Player {key_string} disconnected because they were speed hacking.")
                self.players[key_string]['player']['health'] = self.validate_health(self.players[key_string]['player'])
                logger.info(f"Received Player Dict: {player_data}.")

                if not data:
                    logger.info(f"Player {key_string} disconnected.")
                    running = False
                else:
                    reply = self.players[key_string]
                    encrypted_reply = self.serialize(aes_encryption.encrypt(reply))

                    conn.sendall(encrypted_reply)
                    logger.info(f"Sending All Player Dict: {reply}.")
            except:
                logger.info(f"Player {key_string} lost connection.")
                running = False

        logger.info(f"Connection Closed for Player {key_string}.")
        try:
            del self.players[key_string]
        except:
            logger.info(f"No player with the ID: {key_string} exists.")
        self.connections -= 1
        conn.close()
    
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

Settings:

```

from Scripts.logger import *
try:
except:
pygame.init()

logger.info("Github: https://github.com/TheRealDL1/Simple-Client-Server")
logger.debug("RealDL Settings Code.")

class Config():
  def __init__(self):
    # Screen Width and Height
    info = pygame.display.Info()
    max_width = info.current_w
    max_height = info.current_h
    self.HEIGHT = 720 #max_height # 720
    self.WIDTH = 1280 #max_width # 1280

    # weapons
    self.weapon_data = {
      'sword': {'cooldown': 100, 'speed': 0, 'damage': 15, 'graphic':'Graphics/Game/weapons/sword/full.png','type':'melee'},
      'lance': {'cooldown': 400, 'speed': 0, 'damage': 30,'graphic':'Graphics/Game/weapons/lance/full.png','type':'melee'},
      'axe': {'cooldown': 300, 'speed': 0, 'damage': 20, 'graphic':'Graphics/Game/weapons/axe/full.png','type':'melee'},
      'rapier': {'cooldown': 50, 'speed': 0, 'damage': 8, 'graphic':'Graphics/Game/weapons/rapier/full.png','type':'melee'},
      'sai': {'cooldown': 80, 'speed': 0, 'damage': 10, 'graphic':'Graphics/Game/weapons/sai/full.png','type':'melee'},
      'revolver': {'cooldown': 1000, 'speed': 20, 'damage': 25, 'graphic':'Graphics/Game/weapons/revolver/full.png','type':'gun'},
      'msg': {'cooldown': 175, 'speed': 10, 'damage': 5, 'graphic':'Graphics/Game/weapons/msg/full.png','type':'gun'},
      'pistol': {'cooldown': 600, 'speed': 15, 'damage': 18, 'graphic':'Graphics/Game/weapons/pistol/full.png','type':'gun'}}
    }

    # Bullets

    self.bullet_type = {
      'msg': {
        'down': 'Graphics/Game/bullets/msg/down.png',
        'left': 'Graphics/Game/bullets/msg/left.png',
        'right': 'Graphics/Game/bullets/msg/right.png',
        'up': 'Graphics/Game/bullets/msg/up.png',
      },
      'pistol': {
        'down': 'Graphics/Game/bullets/pistol/down.png',
        'left': 'Graphics/Game/bullets/pistol/left.png',
        'right': 'Graphics/Game/bullets/pistol/right.png',
        'up': 'Graphics/Game/bullets/pistol/up.png',
      },
      'revolver': [
        'down': 'Graphics/Game/bullets/revolver/down.png',
        'left': 'Graphics/Game/bullets/revolver/left.png',
        'right': 'Graphics/Game/bullets/revolver/right.png',
        'up': 'Graphics/Game/bullets/revolver/up.png',
      ],
    }

    # magic
    self.magic_data = {
      'flame': {'strength': 5,'cost': 20,'graphic':'Graphics/Game/particles/flame/fire.png','type':'magic','cooldown':1000},
      'heal' : {'strength': 20,'cost': 10,'graphic':'Graphics/Game/particles/heal/heal.png','type':'magic','cooldown':800}}
    }

    # ui
  
```

Weapon:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
Computer Science Project - S21 Main > Sprint 4 > Python Code > Melee > create_id
from Scripts.logger import *
try:
    import pygame, string, random
    from Scripts.settings import *
except:
    logger.critical("You do not have all the modules installed. Please install Pygame.")

logger.debug("RealDL Weapon Code.")

class Melee(pygame.sprite.Sprite):
    def __init__(self, player, groups):
        try:
            super().__init__(groups)
            self.sprite_type = "weapon"
            self.settings = Config()
            self.id = self.create_id()
            self.time = pygame.time.get_ticks()
            self.last_shot_time = 0 # Initialize the last shot time to 0
            direction = player.status.split('_')[0]

            # Load the image
            self.full_path = f'Graphics/Game/weapons/{player.weapon}/{direction}.png'
            try:
                self.image = pygame.image.load(self.full_path).convert_alpha()
            except:
                self.image = pygame.image.load(f"../{self.full_path}").convert_alpha()

            # Set the placement of the sprite
            if direction == 'right':
                self.rect = self.image.get_rect(midleft=player.rect.midright + pygame.math.Vector2(0, 16))
            elif direction == 'left':
                self.rect = self.image.get_rect(midright=player.rect.midleft + pygame.math.Vector2(0, 16))
            elif direction == 'down':
                self.rect = self.image.get_rect(midtop=player.rect.midbottom + pygame.math.Vector2(-10, 0))
            else:
                self.rect = self.image.get_rect(midbottom=player.rect.midtop + pygame.math.Vector2(-10, 0))
        except:
            logger.error("Failed to create Weapon")

    def create_id(self):
        try:
            characters = string.ascii_letters
            (variable) characters: LiteralString
            return ''.join(random.choice(characters) for _ in range(self.settings.ID_STRING_LENGTH))
        except:
            logger.error("Something went wrong with creating a unique ID.")
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
class Weapon(pygame.sprite.Sprite):
    def __init__(self, groups, x, y, id, image, sprite_type="weapon_copy"):
        super().__init__(groups)
        self.sprite_type = sprite_type
        self.id = id
        self.x = x
        self.y = y
        self.full_path = image
        try:
            self.image = pygame.image.load(self.full_path).convert_alpha()
        except:
            self.image = pygame.image.load(f"../{self.full_path}").convert_alpha()

        self.rect = self.image.get_rect()
        self.rect.x = self.x
        self.rect.y = self.y

class Bullets(pygame.sprite.Sprite):
    def __init__(self, player, groups, obstacle_sprites):
        super().__init__(groups)
        self.settings = Config()
        self.obstacle_sprites = obstacle_sprites
        self.sprite_type = "bullet"
        self.id = self.create_id()
        self.player = player
        self.groups = groups
        self.direction = player.status.split('_')[0]
        gun = player.weapon
        self.speed = self.settings.weapon_data[gun]['speed']
        self.damage = player.attack_strength + self.settings.weapon_data[gun]['damage']
        self.cooldown = self.settings.weapon_data[gun]['cooldown']
        self.collided = False

        self.full_path = self.settings.bullet_type[gun][self.direction]
        try:
            self.image = pygame.image.load(self.full_path).convert_alpha()
        except:
            self.image = pygame.image.load(f"../{self.full_path}").convert_alpha()

        # Set the placement of the sprite
        if self.direction == 'right':
            self.rect = self.image.get_rect(midleft=player.rect.midright + pygame.math.Vector2(20, 8))
        elif self.direction == 'left':
            self.rect = self.image.get_rect(midright=player.rect.midleft + pygame.math.Vector2(-20, 8))
        elif self.direction == 'down':
            self.rect = self.image.get_rect(midtop=player.rect.midbottom + pygame.math.Vector2(-18, 21))
        else:
            self.rect = self.image.get_rect(midbottom=player.rect.midtop + pygame.math.Vector2(-18, -21))
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
def update(self):
    self.collision('horizontal')
    self.collision('vertical')
    if not self.collided:
        if self.direction == 'right':
            self.rect.x += self.speed
        elif self.direction == 'left':
            self.rect.x -= self.speed
        elif self.direction == 'down':
            self.rect.y += self.speed
        else:
            self.rect.y -= self.speed

    if self.collided:
        self.kill()

def create_id(self):
    try:
        characters = string.ascii_letters + string.digits
        return ''.join(random.choice(characters) for _ in range(self.settings.ID_STRING_LENGTH))
    except:
        logger.error("Something went wrong with creating a unique ID.")

def collision(self, direction):
    if direction == 'horizontal':
        for sprite in self.obstacle_sprites:
            if sprite.hitbox.colliderect(self.rect):
                self.collided = True

    if direction == 'vertical':
        for sprite in self.obstacle_sprites:
            if sprite.hitbox.colliderect(self.rect):
                self.collided = True

# You may want to set its initial position and direction here
```

Proof tests (Fig 4.1-4.9):

See MP4 videos.

Sprint 5

Aims for this sprint.

After completing sprint 4, my game is coming together increasingly. However, there are still key features from my analysis that need to implemented still. I still need to implement collisions between players as well as updating the UI to update the kill count and players alive. In addition, I still need to add mechanism to kill other players that turn them into ghosts. I will also need to optimise my code to support a diverse game with multiple connected players. Finally, I am going to implement something called time delta into my game which will help reduce lag and increase frames per second.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Detecting collisions

Detecting collisions will be done on the server side. I will detect the weapons of other players against the individual client's player. As shown in sprint 4, weapons that are not from a client's player have the word "copy" in their "sprite_type." This is an imperative for collisions as I will have to detect whether an object from the sprite group has touched the player. Something that is important to note is that I will need to detect if players have killed by a weapon, bullet or magic ID that is does not share the same player ID in the objects attributes otherwise this could cause players to effectively damage themselves. A final thing to mention is that if a player has been hit, their health will be reduced until it reaches 0 and they die.

Updating the UI

I am going to update the kill count and the players alive UI boxes on the screen. Updating the number of players will be easy as I only need to sort through all the alive players in the players dictionary then display the total number of players that are alive. This will be done client side. However, updating the kill count will be harder as that requires updating the server. I will need to send over the player ID from the weapon or bullet or flame animation that killed the player. I will also need to send over the ID of the player that was killed. The server will then need to recognise that a player has died, and it should increase the kill count for that player with the player data. This can then be sent back to each client and the client that killed will be able to increment the kill count by the number of kills they got.

Ghosts

When a player loses all their health, their status as a player will change from "alive" to "dead." Their appearance on screen will change as their name badge will be grey and their character will have a lower opacity. An alive player will not be able to see a dead player, however dead players will be able to see other players. Once a player dies, the player that killed the player will have their kill count increase by one and the players alive count will decrease by one. Ghosts will have a grey username and their character will have their opacity reduced to create a ghost effect which comes under my user requirements in my analysis. Dead players called ghosts will not be able to interact with any player. They cannot attack players with weapons or guns or animations. They cannot take damage too and can only observe what happens in the game.

Time delta and optimisation

Time delta needed in my game because it measures how long the program takes to execute in real time. In this small time slot, the inputs for the player and outputs happen as they are drawn on screen many times a second. This feature will be used in the game to ensure that all players, regardless of their FPS will move at the same speed. Before, a player's speed could be affected by a higher or lower frame rate, which is not good as it would cause imbalanced in the game. It would mean other players cannot even damage a player as they are moving too fast. All players are supposed to move the same speed in sprint 5. Time delta will mean that I will have to update the players speeds and find new speeds that are not too slow and that are not too fast. Overall, time delta just allows for a smoother game as having higher frames will mean that each player has the game as optimised as possible. Finally, I will use this optimisation to run a faster game. This in turn will create a better gaming experience for my stakeholders.

Server anti-cheat

The last thing that will be implemented in this sprint is server anti-cheat. This is to stop health hacking, speed hacking and to stop the player from becoming alive again after being killed. Health hacking is increasing a player's health artificially over the max limit of 100, this will result in a kick.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Speed hacking will compare the position of the player at a time and the position of the player at another time. If their speed is greater than 10, then they will get kicked. Finally, if a player is switching their status from alive to dead, then they will get kicked as clients shouldn't be able to interfere with that. The reason why their measures are necessary and important is because it helps keep the game balanced and fair for all players and it also is good for server maintainability since a good server will stop clients from cheating.

Success Criteria

1. **Detect collisions:** Detect multiple collisions between weapons and players client-side for each player. When a player has been hit remove their health by the damage of the weapon that hit them.
2. **Update UI:** Update the number of players in the game for each player by sorting through the players dictionary to get the total number of players alive. Update the kill count for each player server side when a player on the client side dies and gives the server the ID of the player that killed them and their own ID to give the player the kill count.
3. **Dead players or Ghosts:** Players should visually become ghosts with a grey username and their character should have a reduced opacity. Ghosts should not be able to interact with the game or other players at all and they are only allowed to observe the game. They can only be seen by other dead players and their alive status will be updated to be "dead."
4. **Time delta and optimization:** Integrate delta time into the game to ensure that player movement is consistent whilst also increasing the overall FPS for all players. Gameplay should be smooth for multiple players connected to the server.
5. **Server anti-cheat:** The server should be able to identify players that speed hack, health hack or try to change their alive status. The server should then take the appropriate measures and remove those players from the game smoothly without interrupting players.

Justification for Sprint 5

Detecting collisions

Detecting collisions is necessary and important in my game. This is because in my analysis my game is a last man standing game which requires players to be able to kill each other. Furthermore, my stakeholders wanted a game which had variety and was interesting and implementing a feature like this is imperative. In addition, my user requirements state that this is necessary because players need a way for the weapons to hit other players and to be detected. Collisions in the game will allow for players to get kills and inevitably have one winner in the game. Finally, collisions in the game will allow me to complete my game and will also allow me to create a game with variety and diverse.

Updating UI

Updating the UI is an imperative for my game because in sprint 3 when I created the UI, I stated that I would come back and ensure that the UI would be functional in the future sprints. In sprint 5 I am going to update the player count and the kill count. The kill count is individual for each player. This is good to let them know how many kills they are and is required by my stakeholders and a quality-of-life feature stated in my user requirements. The player count is also a quality-of-life feature; however, this is a useful feature for both alive and dead players as they know how many players are still in the game which can be useful if a player is off screen. Overall, these are both useful features to have as they are both required in my analysis and by my stakeholders but leaves the players informed with some information of what is currently happening in the game.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Ghosts

Having dead players as ghosts is required in my analysis and in my user requirements. Ghosts are essential for my game as since players can die while the game is still running, there must be a procedure to deal with these dead players without them affecting the game for the other players. Effectively having ghosts allows them to still be in the game without needing to kick that player and allows them to still view the game however, they have no effect on the outcome as ghosts are invisible to alive players and only visible to other dead players, furthermore, they cannot interact with the game at all as they can only observe.

Time delta and optimisation

Time delta is necessary for game because without it, movement is directly correlated to FPS. This game breaking because FPS is set to 60, however if a sudden wave of lag happens, it could cause a sudden decrease in speed, or vice versa, a sudden increase in speed. This cannot happen in the game as it would make it unfair to players with high FPS and unreliable to players in general. This is the reason why time delta, which mean the average time for the program to do one loop, will be implemented into the game. Time delta will then allow for game optimisation as it will allow players to have higher FPS which is a great quality-of-life feature and great for optimisation. Overall, time delta will help optimise the game for the server and players to allow for an enjoyable experience without too much lag or interference.

Server anti-cheat

Some sort of server anti cheat is an imperative to have a functioning server that can handle different scenarios when it comes to clients. Although you could say my project doesn't need anti cheat as it is only a small game, it is in fact good to have, even for a small multiplayer game. It makes the game look more legitimate and seem more professional as the server has effectively been designed to handle hacking players. Furthermore, handling hacking players is good for the non-cheating players as they will have a chance to win if they are kicked since health hacking or speed hacking would make it extremely hard to win.

Pseudo Code

The Pseudo Code is in the Appendix (I will email the Appendix to you separately).

Python Development

Program Code

Code is in the appendix. (I will email the Appendix to you separately).

Areas of complexity

Detecting collisions

The following code above is what I use to detect collisions. Firstly, I check whether there are damage sprites that are currently active. Once we do that the code removes all any weapon, bullet or animations that has the same player id as the client running this script. This is an imperative as it stops the player from being damaged by their own weapons. Once a collision is detected, we determine if the player has been damaged by the damage sprite and we ensure that this sprite has

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

not already been used to attack a player. After that we damage the player, kill the bullet if we can and we set the player id who killed the player if their health reaches 0. This type of collision detection only detects whether their own client has been damaged by another player. This could be seen as a disadvantage as the collision detection is client side which does rely on each client having a solid connection to the server. This also means that the client code could be edited to be invulnerable if this section of code was simply removed.

Updating the UI

This is the kill count function in the server. This function gets a list of all players that have killed someone. When we have that list, we check whether each id in the list is equal to the ID of the current client. We then return the updated kill count of the current client.

```
# Validate if the player has been killed.  
player_data['player']['kill_count'], killed_players = self.kill_count(key_string, player_data['player']['kill_count'], killed_players)
```

The UI for the game is the same except now all we are doing is updating the kill count and the number of players currently alive. Here in the client, we get the number of players alive by looping through the players dictionary and then we increment the player count by 1 if a player is found alive.

```
def players_number(self):  
    player_count = 0  
    for player in self.players:  
        if player.status_alive == "alive":  
            player_count += 1  
    return player_count
```

As shown above, we then simply update the UI values in the redraw window procedure.

Ghosts

Firstly, in the input procedure in the player class we use this line of code to ensure that dead players cannot damage or kill other players.

In the player class, in the draw function, like in the previous sprints where we draw the player with a username, we are now drawing players with grey usernames and their character is slightly transparent as the opacity is decreased.

Here if a player is dead, we change the alpha and colour values so that drawing the player is the same. We draw the image and username as same but instead with the changed values. In level.py where we draw all the sprites, this section of code is used to ensure that only ghosts can see each other and other players and that players cannot see ghosts.

Firstly, the players are sorted into y-axis values to create the top down 2D effect. If the sprite is not a player, we draw the sprite as normal, however, if the sprite is a player, we then check if the player is our own sprite, if it is then we draw the username border green. If the player is alive, we filter out all the ghosts and don't draw them, however if the player is dead, we draw all the players.

Time Delta Optimisation

The first part of time delta optimisation is having the previous time and time difference set to 0. As shown in the second image, previous time is set to the current time in seconds. Self.dt is the fraction of time between the first time and the new time. If you divide 1 by self.dt you will get the frames per second or the number of times that the loop is ran.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

As seen self.dt is used to work out the FPS, however this small decimal is used to ensure that speed is equal between all players. Self.dt is passed through to level.py where the player is drawn.

As seen here self.dt is passed through to level.py. Once we have the time different, we update all the different sprites so that they all run at the same speed regardless of the frame rate. In the player we use dt to update the animation speed and their movement speed.

In the animation we use dt here. This regulates the frame index so that a higher dt will have fewer frames but a lower dt will have more frames.

Finally, in move we use dt here that regulates the distance that the player moves every time the loop is ran.

The weapon doesn't have a dt time because it is a stationary weapon and doesn't move but the bullets and animations do. As seen below the speeds of the bullet are updated.

Finally, the animations require dt.

As used in the player, the frame index is updated so that it runs according to the frame rate. The bullet movement and player movement are similar, and the magic animations are like the player animations when it comes to how dt is used.

Server anti-cheat

These two images (above and below) are the functions for the anti-cheat mechanisms and where they are run. The first function validate_movement is used to find a speed from the two given points and the given time. We find the total time from the sum of all the dt values in player_pos. Moreover, when we have the speed, we return false if the speed is greater than allowed speed and true if it is within the limit. Validate health just checks if their health is not greater than 100 and validate_player_status checks whether their status has not changed from dead to alive. As health validation and status validation are quite simple and only require a few lines they don't need any more explanation however the speed checking does as that is the hard one. Firstly, we collect data about the players position which is their x and y, and we use dt to find the different between each time. After we have the dt, we append that x and y as a list and the time into a 2D list. After this list reaches 10 values, we remove the first item in the list to allow another item in the list so that there are always 10 items in the list. This is then passed through to the function where it uses the data to find their speed.

During iterative development

Now for the testing of this sprint. I will have testing during the development and testing after the development has been completed.

No.	Test Data	Expected Result	Actual Result
5.1	Collision Detection Testing – Ensure player collisions are detected and their health is decreased when hit.	Player and damage sprite collisions will leave the player with a decreased health.	Figure 5.1

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

5.2	UI Update Testing – make sure that when a player gets a kill their kill count increases and the number of players alive decreases. E.g.: have 4 players and kill 2 players with one player. There should be a player with 2 kills and 2 players left.	Whenever a player gets a kill, their own kill count should increase by one and the overall players count should decrease by one.	Figure 5.2
5.3	Ghosts Functionality Testing – Ensure players turn into ghosts when they have no more health. Make sure ghosts cannot interact with any player and that they can still see players and ghosts. Ensure that ghosts are invisible to players.	Ghosts should not be seen by other players. They should not be able to collide or interact with any players. They can be seen only by other ghosts.	Figure 5.3
5.4	Time Delta Integration Testing – Ensure that animation speed and the players speed is linked to the delta time. Vary frame rates to check if speeds are the same.	Speeds should be the same regardless of FPS. Animations should be correlated to the FPS as well.	Figure 5.4
5.5	Optimisation testing – Ensure that the game is optimised for a high frame rate that does not affect the game.	The game should run smooth with multiple players all with a high FPS.	Figure 5.5
5.6	Server anti-cheat testing – ensure that no player can go above the speed of 10 and no player can hack at all. Increase the speed to 20 and their health to 300 and they should get kicked.	Any player speed hacking, health hacking or changing their status should be kicked.	Figure 5.6
5.7	Final integration – all the components of this sprint should work smoothly with each like the UI and optimisation.	All the modules of this sprint should effectively and smoothly work.	Figure 5.7

Stakeholder Questions

Question	Roland	James
What are your thoughts collision detection for player interactions?	Collision detection is a big part of the game. It brings the game alive.	I think the collisions are good. They are effective and are good at engaging players.
How do you feel about the UI updating and displaying player stats and kill counts?	This is nice quality of life feature. Not necessary but nice to know.	This is good for more competitive players but overall nice to know.
What do you think about the concept of ghost players into the game upon player death?	This is a unique and cool feature. It makes the game more engaging for dead players.	Yes, I like this. It is good for players to still feel engaged even though they have no outcome in the game.
What are your thoughts on delta time and optimisation measures will impact gameplay experience?	I think this is good as it allows player to have a higher FPS and have a better game experience.	This is good for efficiency and good for the game experience.
Do the implemented server anti-cheat measures maintain balance and fairness?	This good for all players I believe. Since cheating makes the game	Anti-cheat is good for all players. It keeps the game

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

	unenjoyable it stops one player from dominating unfairly.	fair and keeps the flow of the game.
What are your thoughts on updating the kill count on the server?	I think this is a good feature.	Yes, that seems good to do.
What specific metrics would you propose to measure the success of optimisation efforts in terms of FPS and overall game performance?	I would say that players have an FPS of something above 60 and that the game is generally smooth.	Generally, that FPS doesn't drop or if it does it doesn't drop too low, and that FPS is high, and the game is smooth.

Evaluation

No.	Achieved
1	Detect collisions: Detect multiple collisions between weapons and players client-side for each player. When a player has been hit remove their health by the damage of the weapon that hit them.
2	Update UI: Update the number of players in the game for each player by sorting through the players dictionary to get the total number of players alive. Update the kill count for each player server side when a player on the client side dies and gives the server the ID of the player that killed them and their own ID to give the player the kill count.
3	Dead players or Ghosts: Players should visually become ghosts with a grey username and their character should have a reduced opacity. Ghosts should not be able to interact with the game or other players at all and they are only allowed to observe the game. They can only be seen by other dead players and their alive status will be updated to be "dead."
4	Time delta and optimization: Integrate delta time into the game to ensure that player movement is consistent whilst also increasing the overall FPS for all players. Gameplay should be smooth for multiple players connected to the server.
5	Server anti-cheat: The server should be able to identify players that speed hack, health hack or try to change their alive status. The server should then take the appropriate measures and remove those players from the game smoothly without interrupting players.

From the completion of my success criteria this sprint to the feedback received from my stakeholders as well as the proof of testing, I believe that I have been able to complete this sprint successfully and effectively as intended. Firstly, my collisions detections work. When creating sprint 5, I had problems with collisions as sometimes they would not work and that the player could damage themselves however, after this sprint and work I was able to find the problem. This was because I did not filter out all the sprites that were copies so I needed to give each damage sprite the ID of the player who created it.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Updating the UI was simple. Implementation of the player count was easy however the kill count was harder as in the client we only detect if other players damage us. Overall, this worked and was fine except I need to tell the server which player kill that specific client and update the kill count for the player who got that kill on the server. I would have rather done with on the client side for each player detecting if they had killed another player, but it was better do on the server side so that no confusion was made. This would avoid a player getting a kill even though the player they killed did not die.

I believe turning players into ghosts was easy and successful. This was one of the easier elements to complete in this sprint as all I needed was to determine whether a player dies which is easy as we set the players health to zero. It only required a line of code in the player that stops ghosts from attacking players and stopping ghosts interfering with the collisions and the game play. Finally, changing the colour of the dead players in ghosts was not too hard, overall, this was an easy objective to complete in this sprint as it did not require too much programming to implement. The delta time optimisation I believe was a success. Even though it did take time to set it up, I was eventually able to update the player movement to match the same speed as another player, whilst also having animations be related the time delta difference. I had to play around with the animation speeds and the speeds of the player because multiplying the speed by delta time makes the player slow, so I produced a method to increase the speed. This means that the speed is slightly faster, but it means that players move at the same speed regardless of FPS.

Finally, the server anti was an easier objective to implement. Especially for the health and status checks as they required a few lines of code. I originally wanted to send over the speed of a player to the server but was not good as players could manipulate the speed they are going. What I instead did was store the latest 10 position locations of the player as well as the delta time difference between each position. This was good as what I did was compare the last location from the first location to get the total distance travelled by using Pythagorean theorem. Once I had the total distance, I then summed all the delta time difference to get one time. After that I had the distance and time with, I could then find the speed by dividing the distance by the time. This is simple after breaking it down, however I still needed to pop the first item from list each time to constantly be updating the items in the list. Overall, this was tricky as I needed to have multiple different steps to find the speed, but this was the best way to do it and the fairest way to do it since it means that no players could lie about their speed.

To sum up the last 5 points, I believe from the feedback from my stakeholders and the completion of all my success criteria for this sprint, that this sprint was successful, and I was able to implement everything I intended to, and I improved my game so that it could become playable for players now.

Next sprint

As this sprint was a success, next sprint I intend to add structure to the game, music, and sound effects. Firstly, I will break down what structure means in my game. Currently, there is no way to win the game. The first thing that I will do is when players join is put them in a lobby. This lobby will be a dark screen that indicates to players how many players are in the lobby and a start button. Players will only be able to start the game if there are at least two players. Once the game starts, no more players can join the game. Now for the second phase, when there is only one player left alive a leaderboard or players kill count and a small game over animation will appear. After that players will disconnect automatically from the game after 15 seconds then they will join will be able to join the game again after the server resets its main variables. Lastly, I will implement music for all players in

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

the main menu and game but not in the waiting zone. Sound effects and music will be optional for the player, but they will be able to hear the sound effects of other players if they enable sound.

Copyright

Admittedly this sprint I did not use as my code from Clear Code since this sprint I had to create the collision detection myself as well as making ghosts. The first few sprints I had to use lots of code from clear code since I was establishing my game, but now that the game has collisions and a way to be damaged and die. It is really brought everything together. Overall, I am still using the same concepts and the code from Clear Code, this sprint I have not needed to use code from Clear Code since this part of my project is where I create the collisions, deaths, and other game mechanics.

Appendix

Pseudo code:

Client:

```
from Scripts.logger import *
try
    import pygame, sys, time
    from Scripts.network import Network
    from Scripts.settings import Config
    from Scripts.level import Level
    from Scripts.player import Player
    from Scripts.weapon import *
    from Scripts.encryption import *
    from Scripts.debug import debug
    from Scripts.functions import *
    from Scripts.ui import UI
    from Scripts.main_menu import MainMenu
    from Scripts.magic import MagicPlayer
    from Scripts.particles import AnimationPlayer

except
    logger.critical("You do NOT have all the modules installed. Please install Pygame, RSA AND Pycryptodome.")
endtry
logger.info("RealDL - Client Code")
pygame.init()

class Client inherits Config

public procedure initialize_pygame(user_dict)
    try
        player_count = players.length
        kill_count = 0
        level = new Level()

        previous_time = 0
        dt = 0
    except
        logger.error("Couldn't correctly initialize pygame.")
        close()
    endtry
endprocedure

function players_number()
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

player_count = 0
for player in players
  if player.status_alive == "alive" then
    player_count += 1
  endif
next player
return player_count
endfunction

public procedure redraw_window(all_players_dict)
try
  update_players(all_players_dict) // could be this
  level.run(player, dt)
  //player_attack_logic() # could be this//
  player_attack_collisions()
  ui.player_count = new_players_number()
  ui.player_kill_count = kill_count
  ui.display(player)
  debug(f"Position: {{player.rect.x}, {player.rect.y}}", WIDTH/2, 20)
  if dt != new 0 then debug(f"FPS then {int(1/dt)}", WIDTH/2, 50)
  endif
  debug(f"{player.direction}", WIDTH/2, 80)
  ui.draw_menu()
  if ui.draw_ui then player.paused = True
  else player.paused = False
  endif
  pygame.display.update()
  // clock.tick(FPS)
except
  logger.error("Player has left the game.")
  close()
endtry
endprocedure

public procedure player_attack_collisions()
  // Removes attack sprite id that are no longer in the game
  sprite_ids = [sprite.id for sprite in level.visible_sprites]
  for sprite_id in attacked_sprites_ids
    if sprite_id NOT in sprite_ids then
      attacked_sprites_ids.remove(sprite_id)
    endif
  next sprite_id
  if level.attack_sprites then
    for attack_sprite in level.attack_sprites
      if "copy" in attack_sprite.sprite_type then
        // Checks if the attack is not from our player
        if attack_sprite.player_id != player.id then
          player_collisions = new pygame.sprite.spritecollide(attack_sprite, level.player_sprites, False)
          if player_collisions then
            for player in player_collisions
              if attack_sprite.damaged_player != True AND attack_sprite.id NOT in attacked_sprites_ids then
                if player.id == attack_sprite.id AND player.health > 0 then
                  attacked_sprites_ids.append(attack_sprite.id)
                  attack_sprite.damaged_player = True
                  old_player_health = player.health
                  player.damage_player(attack_sprite.damage)
                  if player.health == 0 AND old_player_health > 0 AND killed_by == "" then
                    killed_by = attack_sprite.player_id
                  endif
                if "bullet" in attack_sprite.sprite_type then

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

      attack_sprite.kill()
    endif
  endif
  next player
endif
next attack_sprite
endif
endprocedure

public procedure main()
try
  previous_time = new time.time()
while running
  current_time = new time.time()
  dt = current_time - previous_time
  previous_time = current_time
  handle_events()
try
  all_players_dict = new send_player_data()
  logger.debug(f"Sending player data: {all_players_dict}")
  logger.debug(f"Received players dictionary: {all_players_dict}")
  redraw_window(all_players_dict)
except Exception as e
  logger.error(f"Failed to send/receive player data: {e}")
  close()
endtry
endwhile
except Exception as e
  logger.error(f"Failed to run main loop: {e}")
  close()
endtry
endprocedure

public procedure handle_events()
for event in pygame.event.get()
  if event.type == pygame.QUIT then
    quit()
  endif
  elseif event.type == pygame.KEYDOWN AND event.key == pygame.K_ESCAPE then
    ui.draw_ui = NOT ui.draw_ui
  endif
  next event
endprocedure

function send_player_data()
player_dict = {
  'x': player.rect.x,
  'y': player.rect.y,
  'image': player.get_image_name(),
  'username': username,
  'status': player.return_alive_status(),
  'health': player.health,
  'id': id,
  'kill_count': kill_count,
  'killed_by': killed_by
}
player_total_dict =

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

'player': player_dict,
'weapon': return_weapon(),
'bullets': return_bullets(),
'magic': return_magic_data()
}
player_encrypted_dict = new serialize(encryption.encrypt(player_total_dict))
network.send(player_encrypted_dict)
all_players_dict = new encryption.decrypt(unserialize(network.receive(DATA_SIZE)))
return all_players_dict
endfunction

public procedure run()
  gameMenu.run()
  user_dict = new gameMenu.start_game()
  initialize_client(user_dict)
  main()
endprocedure

if __name__ == "__main__":
  try:
    client = new Client()
    client.run()
  except:
    logger.error("Couldn't run main menu OR client.")
  endtry
endif
  
```

Level:

```

from Scripts.logger import *
try:
  import pygame
  from Scripts.settings import Config
  from Scripts.tile import Tile
  from Scripts.support import *
  from random import choice
except:
  logger.critical("You do NOT have all the modules installed. Please install Pygame.")
endtry
logger.info("RealDL Level Code.")

class Level inherits Config
  private tile_id
  private player_sprites
  private attackable_sprites
  private attack_sprites
  private current_attack
  private obstacle_sprites
  private visible_sprites
  private display_surface

  public procedure new()
    Config.__init__()
    // get the display surface
    display_surface = new pygame.display.get_surface()
    // sprite group setup
    visible_sprites = new YSortCameraGroup()
    obstacle_sprites = new pygame.sprite.Group()
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

// attack sprites
current_attack = None
attack_sprites = new pygame.sprite.Group()
attackable_sprites = new pygame.sprite.Group()
player_sprites = new pygame.sprite.Group()
// sprite setup
tile_id = 0
create_map()
endprocedure

public procedure run(player, dt)
  // update and draw the game
  visible_sprites.custom_draw(player)
  visible_sprites.update(dt)
endprocedure

class YSortCameraGroup inherits pygame.sprite.Group
  public procedure custom_draw(player)
    // getting the offset
    offset.x = player.rect.centerx - half_width
    offset.y = player.rect.centery - half_height
    // drawing the floor
    display_surface.fill(settings.BG_COLOR)
    floor_offset_pos = floor_rect.topleft - offset
    display_surface.blit(floor_surf, floor_offset_pos)
    // for sprite in sprites():
    for sprite in sorted(sprites(), key= new lambda sprite sprite.rect.centery)
      offset_pos = sprite.rect.topleft - offset
      if sprite.sprite_type == "player" then
        if sprite.id == player.id then
          sprite.draw(offset_pos, (50, 190, 40))
        else
          if player.status_alive == "alive" then
            if NOT sprite.status_alive == "dead" then
              sprite.draw(offset_pos)
            else
              endif
              sprite.draw(offset_pos)
            else
              endif
              endif
              display_surface.blit(sprite.image, offset_pos)
            endif
          next sprite
        endprocedure
      endprocedure
    endfor
  endprocedure
endclass
  
```

Particle:

```

from Scripts.logger import *
try
  import pygame, string, random
  from Scripts.settings import Config
  from Scripts.support import import_folder
except
  logger.critical("You do NOT have all the modules installed. Please install Pygame.")
  
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
endtry
logger.debug("RealDL Magic Code.")

class AnimationPlayer
    private images
    private frames
    private particles

    ...

class ParticleEffect inherits pygame.sprite.Sprite

    ...

public procedure animate(dt)
    frame_index += animation_speed * dt
    if frame_index >= frames.length
        kill()
    else
        image = new frames[int(frame_index)]
    endif
endprocedure

function return_type()
    return animation_type
endfunction

function return_strength()
    return strength
endfunction

function return_image()
    full_path = new images[int(frame_index)]
    if "./" in full_path then
        full_path = new full_path.replace("./", "")
    endif
    return full_path
endfunction

function create_id()
    try
        characters = string.ascii_letters + string.digits
        return "join(random.choice(characters) for _ in range(settings.ID_STRING_LENGTH))"
    except
        logger.error("Something went wrong with creating a unique ID.")
    endtry
endfunction

public procedure update(dt)
    animate(dt)
endprocedure
```

Player:

```
from Scripts.logger import *
try
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
import pygame, math
from Scripts.support import import_folder
from Scripts.settings import Config
except:
    logger.critical("You do NOT have all the modules installed. Please install Pygame.")
endtry
logger.debug("RealDL Player Code.")

class Player inherits pygame.sprite.Sprite

    public procedure new(pos, image, groups, obstacle_sprites, username, id, health, create_attack= new None,
destroy_attack= new None, create_bullet= new None, create_magic= new None, movement= new "WASD",
offense= new "Space", magic= new "L-Shift",status_alive= new "alive") then
        // Setting up player
        try
            super.new(groups)
            settings = new Config()
            screen = new pygame.display.get_surface()
            try
                image = new pygame.image.load(image).convert_alpha()
                image_name = image
            except
                image = new pygame.image.load(f"../{image}").convert_alpha()
                image_name = f"../{image}"
            endtry
            rect = new image.get_rect(topleft = new pos)
            hitbox = new rect.inflate(0,-26)
            old_hitbox = new hitbox.copy()
            pos = new pygame.math.Vector2(hitbox.topleft)
            movement = movement
            offense = offense
            magic_key = magic
            key_binds = {
                'move_up' then pygame.K_w if movement == "WASD" else pygame.K_UP,
                'move_down' then pygame.K_s if movement == "WASD" else pygame.K_DOWN,
                'move_left' then pygame.K_a if movement == "WASD" else pygame.K_LEFT,
                'move_right' then pygame.K_d if movement == "WASD" else pygame.K_RIGHT,
                'attack' then pygame.K_SPACE if offense == "Space" else pygame.K_LCTRL if offense == "L-Ctrl" else
pygame.K_RCTRL,
                'magic' then pygame.K_LSHIFT if magic_key == "L-Shift" else pygame.K_RSHIFT if magic_key == "R-
Shift" else pygame.K_RETURN
            }
            // graphics setup
            import_player_assets()
            status = 'down'
            frame_index = 0
            animation_speed = new 0.15*(settings.frame_increase_rate)*0.6
            // Movement
            direction = new pygame.math.Vector2()
            attacking = False
            magic_attacking = False
            holding_attack_btn = False
            holding_magic_btn = False
            attack_cooldown = 400
            attack_time = None
        
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
paused = False
// weapon
create_attack = create_attack
destroy_attack = destroy_attack
create_bullet = create_bullet
weapon_index = 0
weapon = new list(settings.weapon_data.keys())[weapon_index]
weapon_type = settings.weapon_data[weapon]['type']
can_switch_weapon = True
weapon_switch_time = None
switch_duration_cooldown = 200
// magic
magic_index = 0
create_magic = create_magic
magic = new list(settings.magic_data.keys())[magic_index]
can_switch_magic = True
magic_switch_time = None
// magic and weapons
attacking_reset = False
finish_attacking = 0
// stats
stats = {'health': 100,'energy':60,'attack': 10,'magic': 4,'speed': 5}
max_stats = {'health': 300, 'energy': 140, 'attack': 20, 'magic' : 10, 'speed': 10}
upgrade_cost = {'health': 100, 'energy': 100, 'attack': 100, 'magic' : 100, 'speed': 100}
health = health
energy = stats['energy'] * 0.8
attack_strength = stats['attack']
exp = 500
speed = new stats['speed']*(settings.frame_increase_rate/1.55)
status_alive = status_alive
// damage timer
vulnerable = True
hurt_time = None
invulnerability_duration = 500
alpha = 255
image_alpha = 255
//Other stats
username = username
id = id
basefont = "Graphics/Fonts/Orbitron-Medium.ttf"
largefont = "Graphics/Fonts/Orbitron-Bold.ttf"
textSize = 20
try
    font = new pygame.font.Font(basefont, textSize)
    large_font = new pygame.font.Font(largefont, textSize)
except
    font = new pygame.font.Font(f"../{basefont}", textSize)
    large_font = new pygame.font.Font(f"../{largefont}", textSize)
endtry
sprite_type = "player"
obstacle_sprites = obstacle_sprites
except
    logger.error("Failed to create Player Class")
endtry
endprocedure
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

public procedure move(speed, dt)
if NOT paused then
  old_hitbox = new hitbox.copy()
  if direction.magnitude() != new 0 then
    direction = new direction.normalize()
  endif
  pos.x += new round(direction.x * speed * dt)
  hitbox.x = pos.x
  collision('horizontal')
  pos.y += new round(direction.y * speed * dt)
  hitbox.y = pos.y
  collision('vertical')
  rect.center = hitbox.center
endif
endprocedure

public procedure collision(direction)
  collision_sprites = new pygame.sprite.spritecollide(obstacle_sprites,False)
  if collision_sprites then
    if direction == 'horizontal' then
      for sprite in collision_sprites
        // Collision on the right
        if hitbox.right >= sprite.hitbox.left AND old_hitbox.right <= sprite.old_hitbox.left then
          hitbox.right = sprite.hitbox.left
          pos.x = hitbox.x
        endif
        // Collision on the left
        if hitbox.left <= sprite.hitbox.right AND old_hitbox.left >= sprite.old_hitbox.right then
          hitbox.left = sprite.hitbox.right
          pos.x = hitbox.x
        endif
      next sprite
    endif
    if direction == 'vertical' then
      for sprite in collision_sprites
        // Collision on the bottom
        if hitbox.bottom >= sprite.hitbox.top AND old_hitbox.bottom <= sprite.old_hitbox.top then
          hitbox.bottom = sprite.hitbox.top
          pos.y = hitbox.y
        endif
        // Collision on the top
        if hitbox.top <= sprite.hitbox.bottom AND old_hitbox.top >= sprite.old_hitbox.bottom then
          hitbox.top = sprite.hitbox.bottom
          pos.y = hitbox.y
        endif
      next sprite
    endif
  endif
endprocedure

public procedure cooldowns()
  current_time = new pygame.time.get_ticks()
  if attacking then
    if current_time - attack_time >= attack_cooldown + settings.weapon_data[weapon]['cooldown'] then
  
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
if NOT holding_attack_btn then
    attacking = False
    attacking_reset = True
    finish_attacking = new pygame.time.get_ticks()
    if destroy_attack != None then
        destroy_attack()
    endif
endif
endif
if current_time - attack_time >= settings.weapon_data[weapon]['cooldown'] then
    if holding_attack_btn AND weapon_type == "gun" then
        if create_bullet then
            create_bullet()
            attack_time = current_time
        endif
    endif
endif
endif
if magic_attacking then
    if current_time - attack_time >= settings.magic_data[magic]['cooldown'] then
        if NOT holding_magic_btn then
            magic_attacking = False
            attacking_reset = True
            finish_attacking = new pygame.time.get_ticks()
        endif
    endif
endif
if attacking_reset then
    if current_time - finish_attacking >= settings.attack_or_magic_cooldown then
        attacking_reset = False
    endif
endif
if NOT can_switch_weapon then
    if current_time - weapon_switch_time >= switch_duration_cooldown then
        can_switch_weapon = True
    endif
endif
if NOT can_switch_magic then
    if current_time - magic_switch_time >= switch_duration_cooldown then
        can_switch_magic = True
    endif
endif
if NOT vulnerable then
    if current_time - hurt_time >= invulnerability_duration then
        vulnerable = True
    endif
endif
endprocedure

public procedure animate(dt)
    if NOT paused then
        animation = animations[status]
        image = animation_images[status]
        // loop over the frame index
        frame_index += animation_speed * dt
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
if frame_index >= animation.length
    frame_index = 0
endif
// set the image
image = new animation[int(frame_index)]
image_name = new image[int(frame_index)]
rect = new image.get_rect(center = new hitbox.center)
//flicker
flicker()
endif
endprocedure

public procedure set_opacity()
    image.set_alpha(alpha)
endprocedure

function return_alpha()
    return image_alpha
endfunction

public procedure flicker()
    // flicker
    if NOT vulnerable then
        image_alpha = new wave_value()
        image.set_alpha(image_alpha)
    else
        image.set_alpha(255)
    endif
endprocedure

public procedure damage_player(damage)
    if vulnerable then
        vulnerable = False
        hurt_time = new pygame.time.get_ticks()
        if health - damage <= 0 then
            health = 0
            status_alive = "dead"
        else
            health -= damage
        endif
    endif
endprocedure

function return_alive_status()
    return status_alive
endfunction

function wave_value()
    value = new math.sin(pygame.time.get_ticks())
    if value >= 0 then
        return 255
    else
        return 0
    endif
endfunction
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

public procedure draw(offset_pos, color= new (190, 40, 50))
  // Draw the player.
  alpha = 255
  if status_alive == "dead" then
    color = new (128, 128, 128)
    alpha = 135
  endif
  draw_image = new image.copy()
  draw_image.set_alpha(alpha)
  text_surface = new large_font.render(username, True, color)
  text_rect = new text_surface.get_rect()
  username_pos = new (offset_pos[0] + rect.width DIV 2 - text_surface.get_width() DIV 2, offset_pos[1] - 35)
  // Define the dimensions and position of the black box
  box_width = text_rect.width + 6 // Adjust the width as needed
  box_height = text_rect.height + 6 // Adjust the height as needed
  box_pos = new (username_pos[0]-3, username_pos[1]-3) // Adjust the position as needed
  // Draw the black box
  pygame.draw.rect(screen, (27,31,35), ((box_pos[0]-2,box_pos[1]-2), (box_width+4, box_height+4)),3,10)
  //pygame.draw.rect(screen, (27,31,35), (box_pos, (box_width, box_height)),0,10)
  screen.blit(draw_image, offset_pos)
  screen.blit(text_surface, username_pos)
endprocedure
  
```

server:

```

from Scripts.logger import *
try
  from _thread import *
  from random import randint, choice
  import string, math, socket, time
  from Scripts.settings import Config
  from Scripts.encryption import *
  from Scripts.debug import debug

except
  logger.critical("You do NOT have all the modules installed. Please install Pygame, RSA AND Pycryptodome.")
endtry
logger.debug("RealDL Server Code.")
  
```

class Server inherits Config

```

function kill_count(key_string, kill_count, killed_players)
  kills = []
  // gets a list of all ids of players that have killed someone
  for player in players.values()
    if player['player']['killed_by'] != "" then
      if player['player']['id'] NOT in killed_players then
        kills.append(player['player']['killed_by'])
        killed_players.append(player['player']['id'])
      endif
    endif
  next player
  // checks if this player has killed anyone
  for kill_id in kills
    // logger.debug(kill_count, kill_id, key_string)
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

if kill_id == key_string then
  kill_count += 1
endif
next kill_id
return kill_count, killed_players
endfunction

public procedure handle_client_communication(conn, key_string, aes_encryption)
  running = True
  player_pos = []
  killed_players = []
  last_time = new time.time()
  while running
    try
      // Recieve player data
      player_data = new aes_encryption.decrypt(unserialize(conn.recv(DATA_SIZE)))
      // check the player's status is correct (this doesn't result in a kick)
      data = player_data['player']
      player_data['player']['status'] = new validate_player_status(players[key_string]['player'], data)
      // Validate if the player has been killed.
      player_data['player']['kill_count'], killed_players = new kill_count(key_string,
player_data['player']['kill_count'], killed_players)
      players[key_string] = player_data
      // Calculate the speed of play (FPS).
      delta_time = new time.time() - last_time
      last_time = new time.time()
      // Validate the player data to ensure they are not speed hacking. (this does result in a kick)
      player_pos.append([players[key_string]['player']['x'], players[key_string]['player']['y'], delta_time])
      if player_pos.length > 20 player_pos.pop(0)
    endif
    if NOT validate_movement(player_pos) then
      running = False
      logger.info(f"Player {key_string} disconnected because they were speed hacking.")
    endif
    // Validate player health to ensure they are not health hacking. (this does result in a kick)
    if NOT validate_health(players[key_string]['player']) then
      running = False
      logger.info(f"Player {key_string} disconnected because they were health hacking.")
    endif
    logger.info(f"Received Player Dict: {player_data}.")
    if NOT data then
      logger.info(f"Player {key_string} disconnected.")
      running = False
    else
      reply = players
      encrypted_reply = new serialize(aes_encryption.encrypt(reply))
    endif
    conn.sendall(encrypted_reply)
    logger.info(f"Sending All Player Dict: {reply}.")
  except
    logger.info(f"Player {key_string} lost connection.")
    running = False
  endtry
endwhile
logger.info(f"Connection Closed for Player {key_string}.")
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

try
  del players[key_string]
except
  logger.info(f"No player with the ID: {key_string} exists.")
endtry
connections -= 1
conn.close()
endprocedure
  
```

Weapon:

```

from Scripts.logger import *
try
  import pygame, string, random
  from Scripts.settings import *

except
  logger.critical("You do NOT have all the modules installed. Please install Pygame.")
endtry
logger.debug("RealDL Weapon Code.")

class Melee inherits pygame.sprite.Sprite

  public procedure new(player, groups, player_id)
    try
      super.new(groups)
      sprite_type = "weapon"
      settings = new Config()
      id = new create_id()
      player_id = player_id
      time = new pygame.time.get_ticks()
      last_shot_time = 0 // Initialize the last shot time to 0
      damage = player.attack_strength + settings.weapon_data[player.weapon]['damage']
      damaged_player = False
      direction = new player.status.split('_')[0]
      // Load the image
      full_path = f'Graphics/Game/weapons/{player.weapon}/{direction}.png'
      try
        image = new pygame.image.load(full_path).convert_alpha()
      except
        image = new pygame.image.load(f"../{full_path}").convert_alpha()
      endtry
      // Set the placement of the sprite
      if direction == 'right' then
        rect = new image.get_rect(midleft= newplayer.rect.midright + pygame.math.Vector2(0, 16))
      endif
      elseif direction == 'left' then
        rect = new image.get_rect(midright= newplayer.rect.midleft + pygame.math.Vector2(0, 16))
      endif
      elseif direction == 'down' then
        rect = new image.get_rect(midtop= newplayer.rect.midbottom + pygame.math.Vector2(-10, 0))
      else
        rect = new image.get_rect(midbottom= newplayer.rect.midtop + pygame.math.Vector2(-10, 0))
      endif
    except
      logger.error("Failed to create Weapon")
    
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
endtry
endprocedure

function create_id()
    try
        characters = string.ascii_letters + string.digits
        return "join(random.choice(characters) for _ in range(settings.ID_STRING_LENGTH))"
    except
        logger.error("Something went wrong with creating a unique ID.")
    endtry
endfunction

class Bullets inherits pygame.sprite.Sprite

    public procedure new(player, groups, obstacle_sprites, player_id, player_group)
        super.new(groups)
        settings = new Config()
        obstacle_sprites = obstacle_sprites
        sprite_type = "bullet"
        id = new create_id()
        player_id = player_id
        player_group = player_group
        player = player
        direction = new player.status.split('_')[0]
        gun = player.weapon
        speed = settings.weapon_data[gun]['speed']*settings.frame_increase_rate/1.5
        damage = player.attack_strength + settings.weapon_data[gun]['damage']
        damaged_player = False
        cooldown = settings.weapon_data[gun]['cooldown']

    endprocedure

    public procedure update(dt)
        collision('obstacle')
        collision('player')
        if NOT collided then
            if direction == 'right' then
                rect.x += speed * dt
            endif
            elseif direction == 'left' then
                rect.x -= speed * dt
            endif
            elseif direction == 'down' then
                rect.y += speed * dt
            else
                rect.y -= speed * dt
            endif
        endif
        if collided then
            kill()
        endif
    endprocedure

    function create_id()
        try
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

characters = string.ascii_letters + string.digits
return ''.join(random.choice(characters) for _ in range(settings.ID_STRING_LENGTH))
except:
  logger.error("Something went wrong with creating a unique ID.")
endtry
endfunction

public procedure collision(collision_type)
  if collision_type == "obstacle" then
    collision_sprites = new pygame.sprite.spritecollide(obstacle_sprites,False)
    if collision_sprites then
      collided = True
    endif
  endif
  if collision_type == "player" then
    collision_sprites = new pygame.sprite.spritecollide(player_group,False)
    if collision_sprites then
      collided = True
    endif
  endif

  // You may want to set its initial position and direction here
endprocedure

```

Python code:

Client:

```

from Scripts.logger import *
> try:
> except:
  logger.critical("You do not have all the modules installed. Please install Pygame, RSA and Pycryptodome.")

logger.info("RealDL - Client Code")
pygame.init()

class Client(Config):
  def __init__(self): ...

  def initialize_client(self, user_dict): ...

  def initialize_pygame(self, user_dict): ...

  def initialize_network(self): ...

  def initialize_player(self, player_info): ...

  def update_players(self, player_dict): ...

  def players_number(self):
    player_count = 0
    for player in self.players:
      if player.status_alive == "alive":
        player_count += 1
    return player_count

  def redraw_window(self, all_players_dict):
    try:
      self.update_players(all_players_dict) # could be this
      self.level.run(self.player, self.dt)
      """self.player_attack_logic() # could be this"""
      self.player_attack_collisions()
      self.ui.player_count = self.players_number()
      self.ui.player_kill_count = self.kill_count
      self.ui.display(self.player)
      debug(f"Position: ({self.player.rect.x}, {self.player.rect.y})", self.WIDTH/2, 20)
      if self.dt != 0: debug(f"FPS: {int(1/self.dt)}", self.WIDTH/2, 50)
      debug(f"{self.player.direction}", self.WIDTH/2, 80)
      self.ui.draw_menu()
      if self.ui.draw_ui: self.player.paused = True
      else: self.player.paused = False
      pygame.display.update()
      # self.clock.tick(self.FPS)
    except:
      logger.error("Player has left the game.")
      self.close()

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

def player_attack_collisions(self):
    # Removes attack sprite id that are no longer in the game
    sprite_ids = [sprite.id for sprite in self.level.visible_sprites]
    for sprite_id in self.attacked_sprites_ids:
        if sprite_id not in sprite_ids:
            self.attacked_sprites_ids.remove(sprite_id)

    if self.level.attack_sprites:
        for attack_sprite in self.level.attack_sprites:
            if "copy" in attack_sprite.sprite_type:
                # Checks if the attack is not from our player
                if attack_sprite.player_id != self.player.id:
                    player_collisions = pygame.sprite.spritecollide(attack_sprite, self.level.player_sprites, False)
                    if player_collisions:
                        for player in player_collisions:
                            if attack_sprite.damaged_player != True and attack_sprite.id not in self.attacked_sprites_ids:
                                if player.id == self.player.id and self.player.health > 0:
                                    self.attacked_sprites_ids.append(attack_sprite.id)
                                    attack_sprite.damaged_player = True
                                    old_player_health = self.player.health
                                    self.player.damage_player(attack_sprite.damage)
                                    if self.player.health == 0 and old_player_health > 0 and self.killed_by == '':
                                        self.killed_by = attack_sprite.player_id
                                if "bullet" in attack_sprite.sprite_type:
                                    attack_sprite.kill()

def main(self):
    try:
        self.previous_time = time.time()
        while self.running:
            current_time = time.time()
            self.dt = current_time - self.previous_time
            self.previous_time = current_time

            self.handle_events()

            try:
                all_players_dict = self.send_player_data()
                logger.debug(f"Sending player data: {all_players_dict}")
                logger.debug(f"Received players dictionary: {all_players_dict}")
                self.redraw_window(all_players_dict)
            except Exception as e:
                logger.error(f"Failed to send/receive player data: {e}")
                self.close()

        except Exception as e:
            logger.error(f"Failed to run main loop: {e}")
            self.close()

    def handle_events(self):
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                self.quit()
            elif event.type == pygame.KEYDOWN and event.key == pygame.K_ESCAPE:
                self.ui.draw_ui = not self.ui.draw_ui

    def send_player_data(self):
        player_dict = {
            'x': self.player.rect.x,
            'y': self.player.rect.y,
            'image': self.player.get_image_name(),
            'username': self.username,
            'status': self.player.return_alive_status(),
            'health': self.player.health,
            'id': self.id,
            'kill_count': self.kill_count,
            'killed_by': self.killed_by
        }
        player_total_dict = {
            'player': player_dict,
            'weapon': self.return_weapon(),
            'bullets': self.return_bullets(),
            'magic': self.return_magic_data()
        }
        player_encrypted_dict = self.serialize(self.encryption.encrypt(player_total_dict))
        self.network.send(player_encrypted_dict)
        all_players_dict = self.encryption.decrypt(self.unserialize(self.network.receive(self.DATA_SIZE)))
        return all_players_dict

    def run(self):
        self.gameMenu.run()
        user_dict = self.gameMenu.start_game()
        self.initialize_client(user_dict)
        self.main()

```

Level:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
from Scripts.logger import *
> try: ...
except:
    logger.critical("You do not have all the modules installed. Please install Pygame.")

logger.info("RealDL Level Code.")

class Level(Config):
>     def __init__(self): ...

>     def create_map(self): ...

def run(self, player, dt):
    # update and draw the game
    self.visible_sprites.custom_draw(player)
    self.visible_sprites.update(dt)

class YSortCameraGroup(pygame.sprite.Group):
>     def __init__(self): ...

def custom_draw(self, player):
    # getting the offset
    self.offset.x = player.rect.centerx - self.half_width
    self.offset.y = player.rect.centery - self.half_height

    # drawing the floor
    self.display_surface.fill(self.settings.BG_COLOR)
    floor_offset_pos = self.floor_rect.topleft - self.offset
    self.display_surface.blit(self.floor_surf, floor_offset_pos)

    # for sprite in self.sprites():
    for sprite in sorted(self.sprites(), key=lambda sprite: sprite.rect.centery):
        offset_pos = sprite.rect.topleft - self.offset
        if sprite.sprite_type == "player":
            if sprite.id == player.id:
                sprite.draw(offset_pos, (50, 190, 40))
            else:
                if player.status_alive == "alive":
                    if not sprite.status_alive == "dead":
                        sprite.draw(offset_pos)
                else:
                    sprite.draw(offset_pos)
        else:
            self.display_surface.blit(sprite.image, offset_pos)
```

Particles:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
from Scripts.logger import *
try:
    import pygame, string, random
    from Scripts.settings import Config
    from Scripts.support import import_folder
except:
    logger.critical("You do not have all the modules installed. Please install Pygame.")

logger.debug("RealDL Magic Code.")

class AnimationPlayer:
    def __init__(self): ...
    def create_particles(self, animation_type, pos, strength, groups, player_id): ...

    class ParticleEffect(pygame.sprite.Sprite):
        def __init__(self, pos, animation_frames, animation_images, animation_type, groups, strength=None, player_id=None): ...
        def animate(self, dt): ...
        def return_type(self):
            return self.animation_type
        def return_strength(self):
            return self.strength
        def return_image(self): ...
        def create_id(self): ...
        def update(self, dt):
            self.animate(dt)
```

Player:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
from Scripts.logger import *
try:
    import pygame, math
    from Scripts.support import import_folder
    from Scripts.settings import Config
except:
    logger.critical("You do not have all the modules installed. Please install Pygame.")

logger.debug("RealDL Player Code.")

class Player(pygame.sprite.Sprite):
    def __init__(self, pos, image, groups, obstacle_sprites, username, id, health, create_...
        def import_player_assets(self):...
        def input(self):...
        def get_status(self):...
        def move(self, speed, dt):
            if not self.paused:
                self.old_hitbox = self.hitbox.copy()
                if self.direction.magnitude() != 0:
                    self.direction = self.direction.normalize()

                self.pos.x += round(self.direction.x * speed * dt)
                self.hitbox.x = self.pos.x
                self.collision('horizontal')

                self.pos.y += round(self.direction.y * speed * dt)
                self.hitbox.y = self.pos.y
                self.collision('vertical')
                self.rect.center = self.hitbox.center

        def collision(self, direction):
            collision_sprites = pygame.sprite.spritecollide(self, self.obstacle_sprites, False)
            if collision_sprites:
                if direction == 'horizontal':
                    for sprite in collision_sprites:
                        # Collision on the right
                        if self.hitbox.right >= sprite.hitbox.left and self.old_hitbox.right <= sprite.old_hitbox.left:
                            self.hitbox.right = sprite.hitbox.left
                            self.pos.x = self.hitbox.x

                        # Collision on the left
                        if self.hitbox.left <= sprite.hitbox.right and self.old_hitbox.left >= sprite.old_hitbox.right:
                            self.hitbox.left = sprite.hitbox.right
                            self.pos.x = self.hitbox.x

                if direction == 'vertical':
                    for sprite in collision_sprites:
                        # Collision on the bottom
                        if self.hitbox.bottom >= sprite.hitbox.top and self.old_hitbox.bottom <= sprite.old_hitbox.top:
                            self.hitbox.bottom = sprite.hitbox.top
                            self.pos.y = self.hitbox.y

                        # Collision on the top
                        if self.hitbox.top <= sprite.hitbox.bottom and self.old_hitbox.top >= sprite.old_hitbox.bottom:
                            self.hitbox.top = sprite.hitbox.bottom
                            self.pos.y = self.hitbox.y
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

def cooldowns(self):
    current_time = pygame.time.get_ticks()

    if self.attacking:
        if current_time - self.attack_time >= self.attack_cooldown + self.settings.weapon_data[self.weapon]['cooldown']:
            if not self.holding_attack_btn:
                self.attacking = False
                self.attacking_reset = True
                self.finish_attacking = pygame.time.get_ticks()
                if self.destroy_attack != None:
                    self.destroy_attack()
            self.attack_time = current_time

    if current_time - self.attack_time >= self.settings.weapon_data[self.weapon]['cooldown']:
        if self.holding_attack_btn and self.weapon_type == "gun":
            if self.create_bullet:
                self.create_bullet()
            self.attack_time = current_time

    if self.magic_attacking:
        if current_time - self.attack_time >= self.settings.magic_data[self.magic]['cooldown']:
            if not self.holding_magic_btn:
                self.magic_attacking = False
                self.attacking_reset = True
                self.finish_attacking = pygame.time.get_ticks()

    if self.attacking_reset:
        if current_time - self.finish_attacking >= self.settings.attack_or_magic_cooldown:
            self.attacking_reset = False

    if not self.can_switch_weapon:
        if current_time - self.weapon_switch_time >= self.switch_duration_cooldown:
            self.can_switch_weapon = True

    if not self.can_switch_magic:
        if current_time - self.magic_switch_time >= self.switch_duration_cooldown:
            self.can_switch_magic = True

    if not self.vulnerable:
        if current_time - self.hurt_time >= self.invulnerability_duration:
            self.vulnerable = True

def animate(self, dt):
    if not self.paused:
        animation = self.animations[self.status]
        image = self.animation_images[self.status]

        # loop over the frame index
        self.frame_index += self.animation_speed * dt
        if self.frame_index >= len(animation):
            self.frame_index = 0

        # set the image
        self.image = animation[int(self.frame_index)]
        self.image_name = image[int(self.frame_index)]
        self.rect = self.image.get_rect(center = self.hitbox.center)

        #flicker
        self.flicker()

    def set_opacity(self):
        self.image.set_alpha(self.alpha)

    def return_alpha(self):
        return self.image_alpha

    def flicker(self):
        # flicker
        if not self.vulnerable:
            self.image_alpha = self.wave_value()
            self.image.set_alpha(self.image_alpha)
        else:
            self.image.set_alpha(255)
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

def damage_player(self,damage):
    if self.vulnerable:
        self.vulnerable = False
        self.hurt_time = pygame.time.get_ticks()
    if self.health - damage <= 0:
        self.health = 0
        self.status_alive = "dead"
    else:
        self.health -= damage

def return_alive_status(self):
    return self.status_alive

def wave_value(self):
    value = math.sin(pygame.time.get_ticks())
    if value >= 0:
        return 255
    else:
        return 0

def draw(self, offset_pos, color=(190, 40, 50)):
    # Draw the player.
    alpha = 255
    if self.status_alive == "dead":
        color = (128, 128, 128)
        alpha = 135
    draw_image = self.image.copy()
    draw_image.set_alpha(alpha)
    text_surface = self.large_font.render(self.username, True, color)
    text_rect = text_surface.get_rect()
    username_pos = (offset_pos[0] + self.rect.width // 2 - text_surface.get_width() // 2, offset_pos[1] - 35)

    # Define the dimensions and position of the black box
    box_width = text_rect.width + 6 # Adjust the width as needed
    box_height = text_rect.height + 6 # Adjust the height as needed
    box_pos = (username_pos[0]-3, username_pos[1]-3) # Adjust the position as needed

    # Draw the black box
    pygame.draw.rect(self.screen, (27,31,35), ((box_pos[0]-2,box_pos[1]-2), (box_width+4, box_height+4)),3,10)
    #pygame.draw.rect(self.screen, (27,31,35), (box_pos, (box_width, box_height)),0,10)
    self.screen.blit(draw_image, offset_pos)
    self.screen.blit(text_surface, username_pos)

def update(self, dt):
    self.input()
    self.cooldowns()
    self.get_status()
    self.animate(dt)
    self.move(self.speed, dt)
    self.energy_recovery()
  
```

Server:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
from Scripts.logger import *
try: ...
except:
    logger.critical("You do not have all the modules installed. Please install Pygame, RSA and Pycryptodome.")

logger.debug("RealDL Server Code.")

class Server(Config):
    def __init__(self): ...

    def initialize_server(self): ...

    def create_random_string(self): ...

    def is_touching(self, player_x, player_y, threshold=100): ...

    def get_player_position(self): ...

    def create_new_player(self, username): ...

    def validate_movement(self, player_pos): ...

    def validate_health(self, player): ...

    def validate_player_status(self, player, data): ...

    def kill_count(self, key_string, kill_count, killed_players):
        kills = []

        # gets a list of all ids of players that have killed someone
        for player in self.players.values():
            if player['player']['killed_by'] != "":
                if player['player']['id'] not in killed_players:
                    kills.append(player['player']['id'])
                    killed_players.append(player['player']['id'])

        # checks if this player has killed anyone
        for kill_id in kills:
            # logger.debug(kill_count, kill_id, key_string)
            if kill_id == key_string:
                kill_count += 1
        return kill_count, killed_players

    def handle_client_communication(self, conn, key_string, aes_encryption):
        running = True
        player_pos = []
        killed_players = []
        last_time = time.time()
        while running:
            try:
                # Recieve player data
                player_data = aes_encryption.decrypt(self.deserialize(conn.recv(self.DATA_SIZE)))

                # check the player's status is correct (this doesn't result in a kick)
                data = player_data['player']
                player_data['player']['status'] = self.validate_player_status(self.players[key_string]['player'], data)

                # Validate if the player has been killed.
                player_data['player']['kill_count'], killed_players = self.kill_count(key_string, player_data['player']['kill_count'], killed_players)

                self.players[key_string] = player_data

                # Calculate the speed of play (FPS).
                delta_time = time.time() - last_time
                last_time = time.time()

                # Validate the player data to ensure they are not speed hacking. (this does result in a kick)
                player_pos.append([self.players[key_string]['player']['x'], self.players[key_string]['player']['y'], delta_time])
                if len(player_pos) > 20: player_pos.pop(0)
                if not self.validate_movement(player_pos):
                    running = False
                    logger.info(f"Player {key_string} disconnected because they were speed hacking.")

                # Validate player health to ensure they are not health hacking. (this does result in a kick)
                if not self.validate_health(self.players[key_string]['player']):
                    running = False
                    logger.info(f"Player {key_string} disconnected because they were health hacking.")

                logger.info(f"Received Player Dict: {player_data}.")

                if not data:
                    logger.info(f"Player {key_string} disconnected.")
                    running = False
                else:
                    reply = self.players[key_string]
                    encrypted_reply = self.serialize(aes_encryption.encrypt(reply))

            except:
                logger.error(f"An error occurred while handling player {key_string}.")
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
        conn.sendall(encrypted_reply)
        logger.info(f"Sending All Player Dict: {reply}.")
    except:
        logger.info(f"Player {key_string} lost connection.")
        running = False

    logger.info(f"Connection Closed for Player {key_string}.")
    try:
        del self.players[key_string]
    except:
        logger.info(f"No player with the ID: {key_string} exists.")
    self.connections -= 1
    conn.close()

def threaded_client(self, conn): ...

def run(self): ...

if __name__ == "__main__":
    try:
        server = Server()
        server.run()
    except:
        logger.critical("Server has failed to launch.")
```

Weapon:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
from Scripts.logger import *
> try: ...
except:
    logger.critical("You do not have all the modules installed. Please install Pygame.")

logger.debug("RealDL Weapon Code.")

class Melee(pygame.sprite.Sprite):
    def __init__(self, player, groups, player_id):
        try:
            super().__init__(groups)
            self.sprite_type = "weapon"
            self.settings = Config()
            self.id = self.create_id()
            self.player_id = player_id
            self.time = pygame.time.get_ticks()
            self.last_shot_time = 0 # Initialize the last shot time to 0
            self.damage = player.attack_strength + self.settings.weapon_data[player.weapon]['damage']
            self.damaged_player = False
            direction = player.status.split('_')[0]

            # Load the image
            self.full_path = f'Graphics/Game/weapons/{player.weapon}/{direction}.png'
        >     try: ...
        > except: ...

            # Set the placement of the sprite
        >     if direction == 'right':...
        >     elif direction == 'left':...
        >     elif direction == 'down':...
        >     else:...
        >         except:
                logger.error("Failed to create Weapon")
        >     def create_id(self):...
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
class Bullets(pygame.sprite.Sprite):
    def __init__(self, player, groups, obstacle_sprites, player_id, player_group):
        super().__init__(groups)
        self.settings = Config()
        self.obstacle_sprites = obstacle_sprites
        self.sprite_type = "bullet"
        self.id = self.create_id()
        self.player_id = player_id
        self.player_group = player_group
        self.player = player
        self.direction = player.status.split('_')[0]
        gun = player.weapon
        self.speed = self.settings.weapon_data[gun]['speed']*self.settings.frame_increase_rate/1.5
        self.damage = player.attack_strength + self.settings.weapon_data[gun]['damage']
        self.damaged_player = False
        self.cooldown = self.settings.weapon_data[gun]['cooldown']
        self.collided = False

        self.full_path = self.settings.bullet_type[gun][self.direction]
        try: ...
        except: ...

        # Set the placement of the sprite
        if self.direction == 'right':...
        elif self.direction == 'left':...
        elif self.direction == 'down':...
        else:...

    def update(self, dt):
        self.collision('obstacle')
        self.collision('player')
        if not self.collided:
            if self.direction == 'right':
                self.rect.x += self.speed * dt
            elif self.direction == 'left':
                self.rect.x -= self.speed * dt
            elif self.direction == 'down':
                self.rect.y += self.speed * dt
            else:
                self.rect.y -= self.speed * dt

        if self.collided:
            self.kill()

    def collision(self, collision_type):
        if collision_type == "obstacle":
            collision_sprites = pygame.sprite.spritecollide(self, self.obstacle_sprites, False)
            if collision_sprites:
                self.collided = True

        if collision_type == "player":
            collision_sprites = pygame.sprite.spritecollide(self, self.player_group, False)
            if collision_sprites:
                self.collided = True
```

Tests proof (Fig 5.1 - 5.7):

See MP4 videos.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Sprint 6

Aims for this sprint.

Waiting Zone

As described at the end of my last sprint, I will be adding in a queue area for players to wait in before the game starts. In my user requirements in the analysis, it states in point 4 that players should be able to join within a certain period. In the analysis I asked my stakeholders about this, and they were not too keen on this. Therefore, instead automatically making all players join the game after a minute, I have opted to give users more control and allowing one of them to automatically start the game for all players whenever they want if there are two players. Another thing that has changed about the game is that the player count will be limited at 5. There is an important reason for this since allowing an unlimited number of players to join is terribly slow and can cause collision errors. In addition, an unlimited number of players will lead to lower player satisfaction due to the game breaking collision errors and due to the low FPS. Therefore, I have determined that it is necessary to limit the player count to 5. To conclude, the waiting zone should consist of a button to start the game, with a dark background displaying the number of players in the game.

Music and Sound

Music and sound are a good addition to my game. Firstly, because it makes the game more immersive since the animation with the sound creates a more captivating effect as opposed to no sound effects. In sprint 2 I created the main menu UI and as stated in sprint 2, I needed to implement music and sound into the game as there are settings options for audio. There will be background music playing in the main menu and in the game, however there will be no music playing in the waiting zone. There will be sound effects for the flame and heal animations as well as the attacking motions (weapons and guns). Furthermore, there will be a sound effect for the death of a player and when a player gets damaged. Lastly, sound effects from one player will be able to be heard by all players. This means that one player that does a flame attack, it will be heard by all other players.

Game over and Leaderboard.

As mentioned above about the waiting zone, there are some slight changes that differ from my user requirements. In point 13 of my user requirements, rather than showing a player's statistics, instead I am just going to display a leaderboard with each player kill count in descending order. In user requirements 5 I said that ghosts will turn back to players when the game is over. What will happen is that after 15 seconds when a player has won the game, all clients will be kicked. This will delete the player data on the client side and server side if the necessary variables are reset in the server. After that players will be able to join the game again as normal and the server does not need to be reran again.

Success Criteria

1. **Waiting zone:** There should be a waiting zone where players can join and wait in lobby which has a start button, with a dark background that states the number of players that are in the game. Players should be able to connect until there are 5 players. Once the game starts no more players can join until the game resets.
2. **Music and Sound:** Player's should have the options to toggle music and sound effects. The music or main theme song should play in the main menu screen and the game but not in the

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

waiting zone. Sounds effects should be heard by all players and there should be sound effects for the flame and heal animations, as well as a death and damage sound effect and finally a sound effect for the weapons/ guns and bullets.

3. **Game over and Leaderboard:** The game over animation should appear when there is only one player left alive. The leaderboard should automatically display as well that displays each player and their kill count in descending order. After 15 seconds all users will be kicked from the game and server will reset.

Justification of Sprint 6

Waiting Zone

The waiting zone is particularly important for my game. The reason is because the waiting zone is highly effective at creating game structure when it comes to starting the game. It stops players from joining mid game as well as it now requires a minimum of 2 players to start the game. Without the waiting zone, there would be no structure to the game. If 4 players joined, then another player joined it would be unfair for the dead players as they played fair and now another player has the chance to win. In addition, it would be unfair for the last player alive to die to the new player that joined. Lastly, as this is a last man standing style of game, it would not make sense to allow players to join all the time like an FFA (free for all) game. Therefore, it is justifiable to restrict players to join the game once it has already started.

Music and Sound

To have music and sound in a game is a nice feature. Music and sound help to increase immersion into the game and is good for overall player satisfaction. As seen in movies, music and good sound effects can help create a joyful atmosphere as well as a spooky or scary atmosphere depending on the music or sound effects. Therefore, music and sound effects in my game will be implemented to try and increase the overall engagement of players. The sound effects are good because they can help enhance the already good animations. For example, the heal sound with the heal animation helps improve the sound effect. Overall, music and sound effects are a good feature in game, and they help create a game that is more diverse and interesting to the player.

Game over and Leaderboard

As stated in my user requirements, I said that I would implement a game over animation and a leaderboard for users. A game over animation is good to indicate to players that the game is over, and a player has won. This is a good feature to have, especially for a last man standing style of game as there is a winner compared to a regular FFA game. The leaderboard is a good feature to have, although it is not a necessary feature it is a good element to have in the game especially for those players that are interested in comparing their stats with other players. Overall, a game over animation is good as well as a leaderboard to display to players who has the most kills. Lastly, a 15 second timer before kicking all players is a good feature. It effectively resets the server as there will be no players in the game and it means all players need to do is rejoin the game. There could have been a different solution where players rejoin the waiting zone, however, that would not work as players must have their health, kill count and position reset as well as other variables to be reset otherwise it would be unfair. Therefore, I deemed it as necessary to kick all players as that is the most efficient thing to do.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Pseudo Code

The Pseudo Code is in the Appendix (I will email the Appendix to you separately).

Python Development

Program Code

Code is in the appendix. (I will email the Appendix to you separately).

Areas of complexity

Waiting zone

As seen in previous sprints, we create an instance of the server main loop using threading. Instead of that I changed that line to read this “self.waiting_zone(conn, key_string, aes_encryption).”

Therefore, we ran waiting zone instead of the main loop. As seen in self.run, we now only allow a player to connect if the game has not yet started and that the connections are less than 5.

```
def run(self):
    while self.running:
        ## Limit on 5 players
        conn, addr = self.s.accept()
        logger.debug(f"Connections: {str(self.connections)}")

        if self.ready_to_play and self.connections == 0:
            self.ready_to_play = False

        if self.ready_to_play or self.connections >= 5:
            conn.close()
        else:
            self.connections += 1
            #logger.info(f"Connected to: {addr}")
            #logger.info(f"There {'is' if self.connections==1 else 'are' } {self.connections} players connected")
            start_new_thread(self.threaded_client, (conn,))
```

The waiting zone is not big but it is important to notice that we send over the number of players connected, as well as whether the game has started. We receive if the game should start assuming one of the players wants the game to start. That is why we set self.ready_to_play to true as that is a variable all waiting_zone instances have access to.

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

def waiting_zone(self, conn, key_string, aes_encryption):
    running = True
    while running:
        try:
            # send over number of players connected
            reply = {"connections":self.connections, "started":self.ready_to_play}
            encrypted_reply = self.serialize(aes_encryption.encrypt(reply))

            conn.sendall(encrypted_reply)
            # Receive start or not.
            start_dict = aes_encryption.decrypt(self.unserialize(conn.recv(self.DATA_SIZE)))
            start = start_dict["ready"]
            if start and self.connections >= 2:
                self.ready_to_play = True
        except:
            running = False
            logger.error("Something went wrong.")

        if start:
            self.handle_client_communication(conn, key_string, aes_encryption)

    try:
        del self.players[key_string]
        self.connections -= 1
    except:
        logger.info(f"No player with the ID: {key_string} exists.")
    conn.close()
  
```

In the client, we receive the player dictionary, and the game map is already draw out. This means that when we start the game it is instant for all players as everything has already been loaded.

```

while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            self.quit()
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
                self.quit()

    try:
        connected_players = self.encryption.decrypt(self.unserialize(self.network.receive(self.DATA_SIZE)))
        connections = connected_players['connections']
        started = connected_players['started']
        if self.join_game and connections < 2:
            self.join_game = False
        if started:
            self.join_game = started
        game_dict = {"ready":self.join_game}
        start_the_game = self.serialize(self.encryption.encrypt(game_dict))
        self.network.send(start_the_game)
        if self.join_game or started:
            self.join_game = True
            running = False
    except:
        running = False
        logger.error("Something failed.")
        self.close()

    redraw_window()
  
```

As seen in the loading zone, we have a redraw window function. This effectively draws everything that is needed for the waiting zone. The dark background, some text stating the number of players and a start button. It is simple but it is effective as that is all the information needed. The important thing about the waiting zone on the client side is receiving the number of players and whether the

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

game has started. If the game has already started, then all we do is send it back to the server meaning that one player starts the game for all players.

Sound and Music

Firstly, when the player is being setup. We have these three variables.

```
self.volume = settings['audio']['volume']
self.sound = settings['audio']['sound']
self.music = settings['audio']['music']
```

Volume is between 0 and 100 as an integer. Sound and music are both integers either 0 or 1. 1 meaning true and 0 meaning false. In the main menu, we play the music.

```
def run(self):
    self.background_music.play(-1)
    while self.loop:
        if self.music_change == 0:
            self.background_music.set_volume(0)
        else:
            self.background_music.set_volume(self.volume/100)
```

The first thing to notice is that instead of stopping the music, we reduce the volume to 0. This way if the player decides to turn on the music they can, and it will start not start from the beginning again. Above the while loop, the play (-1) means that the music will play on repeat. Something important to know is that we create the sound effects here in the client instead of in settings.py. The sole reason for this is that the game load very slowly as it was loading all the sound effects and the music which caused the game generation to be much slower than usual therefore, I defined it separately in the client. Previously as seen in the player_attack_collisions procedure, we detected when the player got hit. Here we check whether sound is enabled then we play it.

```
if self.sound == 1:
    self.damage_sound.play()
self.player.damage_player(attack_sprite.damage)
if self.player.health == 0 and old_player_health > 0 and self.killed_by == '':
    if self.sound == 1:
        self.death_sound.play()
    self.killed_by = attack_sprite.player_id
```

When we play the sounds for the player, all I did was add in a couple lines to check if sound has been enabled then we play the sounds. Since the bullet and weapons/ guns use the same sound, If the weapon is a gun, we do not play the sound effect for the melee weapon as it plays the sound effect twice otherwise.

The only difficult part about the sounds was playing the sound effect of other players for all players. This is tricky because we need a copy of the dictionary containing all the information about all players. Here is how we created a copy of the dictionary.

This is vital since to determine if the player was hit, we need to know their previous health. If it was higher than their current health, then they were hit. Similarly, if the player status is now dead but it was alive, then they just died. This is the reason as why we need a copy of the old and new dictionary as we cannot detect the collisions of another client on a client script. As we do not need to detect collisions for other clients this is a better method of detecting when a sound should be played.

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

Playing the weapon, gun and animation sound effects were harder, but still required the old and new dictionary. As we need to know when that weapon, gun or animation was created. If we detected that a new damage sprite was created, we could then play the sound for that weapon if its player ID is different from the clients ID.

The bullet sound was easier since if there was a sound, we would just play the sound, however with the weapon, we needed to get the index of the weapon so that we could determine the location of that weapon. This meant we could find out the animation type which was essential to not play the gun sound effect multiple times. The animations are like the weapon as we also need to determine the animation type so that we can play the according sound.

Game over and Leaderboard

For the game over screen, I created a procedure that would run everything on the client side for me all in one location. In redraw window, we would run the function and check whether there is one player over or if the game is already over. This way if a player leaves the game, then the game over and leaderboard just does not disappear. The variables for the leaderboard and game over animation are created when the client is initialised. When there is one player left in the game, The Game over sign is drawn as well as the leaderboard is drawn. Using my own personally pygame functions to make the buttons and text was easy, however, the difficult part was the leaderboard as pygame only allows you to draw one text at a time on the same line, I needed multiple lines. Therefore, I split up the leaderboard text into separate lists which contains the player and their kill count. Then using the theory of concurrent processing, I looped round that list and then drew each line separately. In reality, only one text line is drawn at a time, however when the game is running at high FPS it looks like there are multiple text lines as intended. In the server we check if there is less than one player left in the game. When there is we stop updating the end_time variable which means that after 15 seconds in real time, the server kicks all client from the game as required.

During iterative development

No.	Test Data	Expected Result	Actual Result
6.1	Connect one player to the game, try to start the game.	The player should not be able to start the game.	Figure 6.1
6.2	Try to connect 6 players to the game.	The 6 th player should not be able to join.	Figure 6.2
6.3	Try to join the game while it is running.	That player should be kicked.	Figure 6.3
6.4	Toggle music and sound off when the game runs.	There should be no sound.	Figure 6.4
6.5	Do the flame animation and heal animation.	All players should be able to hear the animations.	Figure 6.5
6.6	Ensure one player wins the game. Assuming the game over and leaderboard sign shows up, leave the game.	The game over and leaderboard sign should still be displayed.	Figure 6.6
6.7	Player can hear the sound effects of other players.	All players hear the sound effects of other players.	Figure 6.7

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

Stakeholder Questions

Question	Roland	James
Do you like the new waiting zone feature?	Yes, I think this is a good feature. It creates game structure.	Yep, I think it is good.
Do you think that limiting the player count to 5 is a good idea considering overall FPS for players?	Considering game playability, I think this is a good move.	Yep. It will stop the frames from being very low, so it is a good change.
Do you like the idea of limiting players from joining the game before there are at least two players as well as restricting players access to join the game when it is running?	Yep. This is a good idea. I think it is good to require at least two players to join. I also think that restricting players from join when the game is running is a good idea.	Yep, I think that they are both good. It creates more game structure.
Do you like the ability to hear sound effects of other players?	Sound effects generally help boost a game. I think this is a good idea.	Yes, I think sound effects are a good addition.
What are your opinions on the music and sound effects? Do they add to the game?	Yes, they do add to the game. It makes the game more immersive.	I like sound effects in a game. It will make the game better.
Do you like that the game now has a winner with a game over animation with a leaderboard.	This is a very good feature. Displaying to all users that the game is over with a leaderboard is good.	Yes. I think that having a winner is a good idea.
Do you like the fact that the game kicks all players when the game is over.	Yes, I like that feature.	That seems fine to me.
What are your opinions on the game overall as a whole now it is completed?	Overall, I think the game is good. When it comes to the UI, game layout and the overall objective of the game I think it is good.	Generally, it is a good game. It has many different features all required in a game.

Evaluation

No.		Achieved
1	Waiting zone: There should be a waiting zone where players can join and wait in lobby which has a start button, with a dark background that states the number of players that are in the game. Players should be able to connect until there are 5 players. Once the game starts no more players can join until the game resets.	Yes
2	Music and Sound: Player's should have the options to toggle music and sound effects. The music or main theme song should play in the main menu screen and the game but not in the waiting zone. Sounds effects should be heard by all players and there should be sound effects for the flame and heal animations, as well as a death and damage sound effect and finally a sound effect for the weapons/ guns and bullets.	Yes

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

3	Game over and Leaderboard: The game over animation should appear when there is only one player left alive. The leaderboard should automatically display as well that displays each player and their kill count in descending order. After 15 seconds all users will be kicked from the game and the server will reset.	Yes

Finally, as this is my final sprint, I believe that comparing what I have completed in this sprint compared to my original 15 user requirements in the analysis I have completed most of them apart from a few additional changes that were added due to time and finding out a better and more effective way to implement them as well as a better solution to the problem.

Original Goals

My original goals for this sprint were to create structure in the game and to implement audio. I believe from the responses of my stakeholders and the fact that I have completed this sprint as I intended means I have added structure to my game. The first element of structure was the waiting zone. This was used to limit the number of players that could join and create a minimum of two players required to start the game. This feature was successful and stopped more than 5 players from being able to connect to the game. The overall premise of the waiting zone was to allow players to join if the game has not started then when the game is running, it would not allow players to join, which is what I intended to do and is what worked.

Game Continuation

The other element to game structure is having a winner as well to keep the game flow continuous. I first wanted to add some graphical output to display to all players that indicates whether a player has won as well as a leaderboard for players to compare their stats to each other. This feature was successful as the animation worked as intended after a player had won the game as well as the leaderboard being displayed. Even if player count decreased to 0, it wouldn't mean that the leaderboard and game over animation would stop being displayed.

Music and Sound effects

My other intention was to implement music and sound effects. This is to improve the overall player satisfaction. As seen in sprint 2 I created the UI with the audio settings, however this sprint I decided to implement music and sound effects. In conclusion, I believe I did achieve what I set out to do. The sound effects work on all clients as expected as well as the music creating a good game atmosphere. To sum up the above, I believe in this sprint I have accomplished what I intended to do which was a turn my game from a loosely FFA style of game into the intended last man standing game that I originally intended to develop in my analysis.

Copyright

As this is my last sprint, I haven't used any code from Clear Code when it comes to the waiting zone or the game over and leaderboard. However, I have used the music and sound effects from Clear Code's game. As I stated previously, I am still using the same architecture that was developed in the previous sprints, however I did use the sound effects and music for my game as they do in fact make my game better.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Appendix

Pseudo code:

Client:

```
from Scripts.logger import *
try
    import pygame, sys, time
    from Scripts.network import Network
    from Scripts.settings import Config
    from Scripts.level import Level
    from Scripts.player import Player
    from Scripts.weapon import *
    from Scripts.encryption import *
    from Scripts.debug import debug
    from Scripts.functions import *
    from Scripts.ui import UI
    from Scripts.main_menu import MainMenu
    from Scripts.magic import MagicPlayer
    from Scripts.particles import AnimationPlayer

except
    logger.critical("You do NOT have all the modules installed. Please install Pygame, RSA AND Pycryptodome.")
endtry
logger.info("RealDL - Client Code")
pygame.init()

class Client inherits Config

public procedure initialize_pygame(user_dict)
    try
        // Sound and Music
        try
            sword_sound = new pygame.mixer.Sound("Audio/sword.wav")
            damage_sound = new pygame.mixer.Sound("Audio/hit.wav")
            heal_sound = new pygame.mixer.Sound("Audio/heal.wav")
            fire_sound = new pygame.mixer.Sound("Audio/Fire.wav")
            death_sound = new pygame.mixer.Sound("Audio/death.wav")
            background_music = new pygame.mixer.Sound("Audio/main.ogg")
        except
            sword_sound = new pygame.mixer.Sound("../Audio/sword.wav")
            damage_sound = new pygame.mixer.Sound("../Audio/hit.wav")
            heal_sound = new pygame.mixer.Sound("../Audio/heal.wav")
            fire_sound = new pygame.mixer.Sound("../Audio/Fire.wav")
            death_sound = new pygame.mixer.Sound("../Audio/death.wav")
            background_music = new pygame.mixer.Sound("../Audio/main.ogg")
        endtry

        // Game Over
        game_over = new Text(gameMenu.text_height, "Graphics/Fonts/Orbitron-ExtraBold.ttf", (247,155,16),
None, None, None, gameMenu.base_text_size*10)
        game_over_board = new Button((27,31,35), (27,31,35), gameMenu.settings.WIDTH/2,
gameMenu.settings.HEIGHT/2*0.4, "Graphics/Fonts/Orbitron-Regular.ttf", (27,31,35), (27,31,35),
gameMenu.settings.WIDTH*0.55, gameMenu.settings.HEIGHT*0.2,'Rectangle', None, gameMenu.big_text_size,
int(gameMenu.curve*1.5))
        // Leaderboard
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

leaderboard = new Button((27,31,35), (27,31,35), gameMenu.settings.WIDTH/2*1.78,
gameMenu.settings.HEIGHT/2, "Graphics/Fonts/Orbitron-Regular.ttf", (27,31,35), (27,31,35),
gameMenu.settings.WIDTH*0.2, gameMenu.settings.HEIGHT*0.7,'Rectangle', None, gameMenu.big_text_size,
int(gameMenu.curve*1.5))
leaderboard_text = new Text(gameMenu.text_height, "Graphics/Fonts/Orbitron-Bold.ttf", (240,240,240),
None, None, None, gameMenu.base_text_size*3)
leaderboard2 = new Button((27,31,35), (27,31,35), gameMenu.settings.WIDTH/2*1.78,
gameMenu.settings.HEIGHT/2*1.06, "Graphics/Fonts/Orbitron-Regular.ttf", (27,31,35), (27,31,35),
gameMenu.settings.WIDTH*0.18, gameMenu.settings.HEIGHT*0.6,'Rectangle', None, gameMenu.big_text_size,
int(gameMenu.curve*1.5))
value_board = new Text(gameMenu.text_height, "Graphics/Fonts/Orbitron-Medium.ttf", (240,240,240),
None, None, None, gameMenu.base_text_size*3)
game_finished = False
except
  logger.error("Couldn't correctly initialize pygame.")
  close()
endtry
endprocedure

public procedure update_players(player_dict, dictionary_copy)
  // Update existing player instances and remove players that are not in player_dict
  try
    ///// Player Sounds /////
    for player_data, old_player_data in zip(player_dict.values(), dictionary_copy.values())
      player_info = player_data['player']
      old_player_info = old_player_data['player']
      if player_info['id'] != player.id then
        ///// Player Got Hit /////
        if player_info['health'] < old_player_info['health'] then
          if sound == 1 then
            damage_sound.play()
          endif
        endif
        ///// Player dies /////
        if player_info['status'] == "dead" AND old_player_info['status'] == "alive" then
          if sound == 1 then
            death_sound.play()
          endif
        endif
        ///// Weapons /////
        weapon_ids = [weapon['player_id'] for weapon in player_data['weapon']]
        weapon_type = [weapon['type'] for weapon in player_data['weapon']]
        old_weapon_ids = [old_weapon['player_id'] for old_weapon in old_player_data['weapon']]
        new_weapon_ids = new set(weapon_ids) - set(old_weapon_ids)
        if new_weapon_ids then
          for new_weapon_id in new_weapon_ids
            weapon_index = new_weapon_ids.index(new_weapon_id)
            animation_type = weapon_type[weapon_index]
            if sound == 1 then
              if animation_type == "melee" then
                sword_sound.play()
              endif
            endif
          next new_weapon_id
        endif
      endif
    endfor
  endtry
endprocedure
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

///// Bullets /////
bullet_ids = [bullet['id'] for bullet in player_data['bullets']]
old_bullet_ids = [old_bullet['id'] for old_bullet in old_player_data['bullets']]
new_bullet_ids = new set(bullet_ids) - set(old_bullet_ids)
if new_bullet_ids then
  for new_bullet_id in new_bullet_ids
    if sound == 1 then
      sword_sound.play()
    endif
  next new_bullet_id
endif
///// Fire Animation /////
magic_ids = [magic['player_id'] for magic in player_data['magic']]
magic_type = [magic['type'] for magic in player_data['magic']]
old_magic_ids = [old_magic['player_id'] for old_magic in old_player_data['magic']]
new_magic_ids = new set(magic_ids) - set(old_magic_ids)
if new_magic_ids then
  for new_magic_id in new_magic_ids
    magic_index = new magic_ids.index(new_magic_id)
    animation_type = magic_type[magic_index]
    if sound == 1 then
      if animation_type == "flame" then
        fire_sound.play()
      endif
      if animation_type == "aura" then
        heal_sound.play()
      endif
    endif
  next new_magic_id
endif
next player_data, old_player_data
except
  logger.error("Player has left the game.")
  close()
endtry
endprocedure

public procedure create_attack(weapon_type= new "melee")
  //logger.info("Create attack")
  try
    if sound == 1 AND weapon_type == "melee" then
      sword_sound.play()
    endif
    current_attack = new Melee(player, [level.visible_sprites], player.id)
  except
    current_attack = None
    //logger.error("Failed to render attack image.")
  endtry
endprocedure

public procedure create_bullet()
  //logger.info("Bullet!")
  if sound == 1 then
    sword_sound.play()

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
endif
bullet = new Bullets(player, [level.visible_sprites], level.obstacle_sprites, player.id, level.player_sprites)
endprocedure

public procedure create_magic(style, strength, cost)
if style == 'heal' then
    if sound == 1 AND player.energy > cost then
        heal_sound.play()
    endif
    magic_player.heal(player, strength, cost, [level.visible_sprites], player.id)
endif
if style == 'flame' then
    if sound == 1 AND player.energy > cost then
        fire_sound.play()
    endif
    magic_player.flame(player, strength, cost, [level.visible_sprites], player.id)
endif
endprocedure

public procedure player_attack_collisions()
// Removes attack sprite id that are no longer in the game
sprite_ids = [sprite.id for sprite in level.visible_sprites]
for sprite_id in attacked_sprites_ids
    if sprite_id NOT in sprite_ids then
        attacked_sprites_ids.remove(sprite_id)
    endif
next sprite_id
if level.attack_sprites then
    for attack_sprite in level.attack_sprites
        if "copy" in attack_sprite.sprite_type then
            // Checks if the attack is not from our player
            if attack_sprite.player_id != player.id then
                player_collisions = new pygame.sprite.spritecollide(attack_sprite, level.player_sprites, False)
                if player_collisions then
                    for player in player_collisions
                        if attack_sprite.damaged_player != True AND attack_sprite.id NOT in attacked_sprites_ids then
                            if player.id == player.id AND player.health > 0 then
                                attacked_sprites_ids.append(attack_sprite.id)
                                attack_sprite.damaged_player = True
                                old_player_health = player.health
                                if sound == 1 then
                                    damage_sound.play()
                                endif
                                player.damage_player(attack_sprite.damage)
                            if player.health == 0 AND old_player_health > 0 AND killed_by == "" then
                                if sound == 1 then
                                    death_sound.play()
                                endif
                                killed_by = attack_sprite.player_id
                            endif
                            if "bullet" in attack_sprite.sprite_type then
                                attack_sprite.kill()
                            endif
                        endif
                    endif
                endif
            endif
        endif
    endif
endif
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

      next player
    endif
    endif
    endif
  next attack_sprite
endif
endprocedure

public procedure main()
dictionary_copy = {}
background_music.play(-1)
if music == 1 then
  background_music.set_volume(volume/100)
else
  background_music.set_volume(0)
endif
try
  previous_time = new time.time()
  while running
    current_time = new time.time()
    dt = current_time - previous_time
    previous_time = current_time
    handle_events()
    try
      all_players_dict = new send_player_data()
      //logger.debug(f"Sending player data: {all_players_dict}")
      //logger.debug(f"Received players dictionary: {all_players_dict}")
      redraw_window(all_players_dict, dictionary_copy)
      dictionary_copy = all_players_dict
    except Exception as e
      logger.error(f"Failed to send/receive player data: {e}")
      close()
    endtry
  endwhile
except Exception as e
  logger.error(f"Failed to run main loop: {e}")
  close()
endtry
endprocedure

public procedure one_player_remaining()
if players_number() = new= new 1 OR game_finished then
  //// Player has won.
  game_finished = True
  //// Animation
  game_over_board.draw((27,31,35), None, None, True, None, None, None, "draw")
  game_over.draw("draw", "Game Over!",(gameMenu.settings.WIDTH/2),
(gameMenu.settings.HEIGHT/2*0.4))
  //// Display kill counts for all players
  leaderboard.draw((27,31,35))
  leaderboard2.draw((240,240,240))
  leaderboard_text.draw("draw", "Leaderboard",(gameMenu.settings.WIDTH/2*1.78),
(gameMenu.settings.HEIGHT/2*0.375))
  leaderboard_text = new "\n".join([f"{player.username} {player.kill_count}" for player in sorted(players, key=
newlambda x x.kill_count, reverse= newTrue)])

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

next player
leaderboard_list = new leaderboard_text.split('\n')
for index, leaderboard_index in enumerate(leaderboard_list)
  spacing_index = 0.52 + 0.1*index
  value_board.draw("draw",leaderboard_index,(gameMenu.settings.WIDTH/2*1.78),
(gameMenu.settings.HEIGHT/2*spacing_index))
  next index, leaderboard_index
  //// Reset key player data
  //// Log out after a 30 second.
else
  game_over.draw("undraw", "Game Over!",(gameMenu.settings.WIDTH/2),
(gameMenu.settings.HEIGHT/2*0.4))
endif
endprocedure

public procedure handle_events()
for event in pygame.event.get()
  if event.type == pygame.QUIT then
    quit()
  endif
  elseif event.type == pygame.KEYDOWN AND event.key == pygame.K_ESCAPE then
    ui.draw_ui = NOT ui.draw_ui
  endif
  next event
endprocedure

function send_player_data()
player_dict = {
  'x': player.rect.x,
  'y': player.rect.y,
  'image': player.get_image_name(),
  'username': username,
  'status': player.return_alive_status(),
  'health': player.health,
  'id': id,
  'kill_count': kill_count,
  'killed_by': killed_by
}
player_total_dict = {
  'player': player_dict,
  'weapon': return_weapon(),
  'bullets': return_bullets(),
  'magic': return_magic_data()
}
player_encrypted_dict = new serialize(encryption.encrypt(player_total_dict))
network.send(player_encrypted_dict)
all_players_dict = new encryption.decrypt(unserialize(network.receive(DATA_SIZE)))
return all_players_dict
endfunction

public procedure loading_zone()
procedure redraw_window()
  screen.fill((27,31,35))
  join_btn.draw((240,240,240),join_the_game)

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

players.draw("draw", f"{connections} {'player' if connections= new = new 1 else 'players'} {'is' if
connections= new = new 1 else 'are'} connected.", (WIDTH / 2), (HEIGHT / 2) * 0.4)
endif
join_hover = new join_btn.is_hovered()
join_click = new join_btn.is_clicking()
if join_hover then
  if NOT join_click then
    mouse.mode = 1
  else
    mouse.mode = 2
else
  endif
  mouse.mode = 0
endif
mouse.draw()
pygame.display.update()
endprocedure

procedure join_the_game()
  join_game = True
endprocedure

// Receive number of players
running = True
mouse = new Mouse("Graphics/MainMenu/Mouse/mouse1.png",
"Graphics/MainMenu/Mouse/mouse2.png", "Graphics/MainMenu/Mouse/mouse3.png")
width_ratio = WIDTH / 1920
height_ratio = HEIGHT / 1080
// Variables
players = new Text(gameMenu.text_height, "Graphics/Fonts/Orbitron-Regular.ttf", (240, 240, 240), None,
None, None, int(gameMenu.base_text_size * 3))
join_btn = new Button((39, 174, 96), (27,31,35), (gameMenu.settings.WIDTH/2),
(gameMenu.settings.HEIGHT/2)*1.4-gameMenu.button_padding, "Graphics/Fonts/Orbitron-Medium.ttf",
(27,31,35), (39, 174, 96), gameMenu.base_button_width, gameMenu.base_button_height,'Start','Rectangle', None,
int(gameMenu.big_text_size/1.3), gameMenu.curve)
connections = 0
while running
  for event in pygame.event.get()
    if event.type == pygame.QUIT then
      quit()
    endif
    if event.type == pygame.KEYDOWN then
      if event.key == pygame.K_ESCAPE then
        quit()
      endif
    endif
  next event
  try
    connected_players = new encryption.decrypt(unserialize(network.receive(DATA_SIZE)))
    connections = connected_players['connections']
    started = connected_players['started']
    if join_game AND connections < 2 then
      join_game = False
    endif
    if started then
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

join_game = started
endif
game_dict = {"ready":join_game}
start_the_game = new serialize(encryption.encrypt(game_dict))
network.send(start_the_game)
if join_game OR started then
  join_game = True
  running = False
endif
except
  running = False
  logger.error("Something failed.")
  close()
endtry
redraw_window()
endwhile
endprocedure

if __name__ == "__main__":
  try
    client = new Client()
    client.run()
  except
    logger.error("Couldn't run main menu OR client.")
  endtry
endif

```

Main menu:

```

from Scripts.logger import *
try
  import pygame, sys, time
  from Scripts.settings import Config
  from Scripts.encryption import *
  from Scripts.debug import debug
  from Scripts.functions import *

except
  logger.critical("You do NOT have all the modules installed. Please install Pygame.")
endtry
logger.info("RealDL - Main Menu Code")
class MainMenu
  private background_music

  public procedure new()
    // Music
    try
      background_music = new pygame.mixer.Sound("Audio/main.ogg")
    except
      background_music = new pygame.mixer.Sound("../Audio/main.ogg")
    endtry
  endprocedure

  public procedure run()
    background_music.play(-1)
    while loop

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

if music_change == 0 then
  background_music.set_volume(0)
else
  background_music.set_volume(volume/100)
endif
events = new pygame.event.get()
for event in events
  if event.type == pygame.QUIT then
    close()
  endif
  if event.type == pygame.KEYDOWN then
    if event.key == pygame.K_ESCAPE then
      close()
    endif
  endif
next event
// Update the display and frame rate
redraw_window(events)
pygame.display.update()
clock.tick(FPS)
endwhile
endprocedure
  
```

Server:

```

from Scripts.logger import *
try
  from _thread import *
  from random import randint, choice
  import string, math, socket, time
  from Scripts.settings import Config
  from Scripts.encryption import *
  from Scripts.debug import debug

except
  logger.critical("You do NOT have all the modules installed. Please install Pygame, RSA AND Pycryptodome.")
endtry
logger.debug("RealDL Server Code.")

class Server inherits Config
  private //logger.info(f"There
  private ready_to_play
  private connections

  function end_game()
    player_count = 0
    for player in players.values()
      if player['player']['status'] == "alive" then
        player_count += 1
      endif
    next player
    if player_count <= 1 then
      return True
    else
      return False
    endif
  endfunction
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

endfunction

function validate_position(player)
  // Map boundaries, stops players from leaving the map.
  if player['x'] <= 180 OR player['x'] >= 3400 then
    return True
  endif
  elseif player['y'] <= -50 OR player['y'] >= 3150 then
    return True
  else
    return False
  endif
endfunction

public procedure handle_client_communication(conn, key_string, aes_encryption)
  running = True
  player_pos = []
  killed_players = []
  last_time = new time.time()
  end_time = 0
  reset_game_time = 15
  while running
    try
      //// ONE player left. -> after 30 seconds -> kick all players -> reset all key values
      // Recieve player data
      player_data = new aes_encryption.decrypt(unserialize(conn.recv(DATA_SIZE)))
      // check the player's status is correct (this doesn't result in a kick)
      data = player_data['player']
      player_data['player']['status'] = new validate_player_status(players[key_string]['player'], data)
      // Validate if the player has been killed.
      player_data['player']['kill_count'], killed_players = new kill_count(key_string,
player_data['player']['kill_count'], killed_players)
      players[key_string] = player_data
      if end_game() then
        if time.time() - end_time >= new reset_game_time then
          running = False
        else
          end_time = new time.time()
        endif
      endif
      // Calculate the speed of play (FPS).
      delta_time = new time.time() - last_time
      last_time = new time.time()
      // Validate the player's position.
      if validate_position(players[key_string]['player']) then
        running = False
        logger.info(f"Player {key_string} disconnected as they tried to leave the map.")
      endif
    endwhile
  endprocedure

  // Validate the player data to ensure they are not speed hacking. (this does result in a kick)
  player_pos.append([players[key_string]['player']['x'], players[key_string]['player']['y'], delta_time])
  if player_pos.length > 20 player_pos.pop(0)

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

endif
if NOT validate_movement(player_pos) then
  running = False
  logger.info(f"Player {key_string} disconnected because they were speed hacking.")
endif
// Validate player health to ensure they are not health hacking. (this does result in a kick)
if NOT validate_health(players[key_string]['player']) then
  running = False
  logger.info(f"Player {key_string} disconnected because they were health hacking.")
endif
//logger.info(f"Received Player Dict: {player_data}.")
if NOT data then
  logger.info(f"Player {key_string} disconnected.")
  running = False
else
  reply = players
  encrypted_reply = new serialize(aes_encryption.encrypt(reply))
endif
conn.sendall(encrypted_reply)
//logger.info(f"Sending All Player Dict: {reply}.")
except
  logger.info(f"Player {key_string} lost connection.")
  running = False
endtry
logger.info(f"Connection Closed for Player {key_string}.")
try
  del players[key_string]
  connections -= 1
except
  logger.info(f"No player with the ID: {key_string} exists.")
endtry
conn.close()
public procedure waiting_zone(conn, key_string, aes_encryption)
  running = True
  while running
    try
      // send over number of players connected
      reply = {"connections":connections,"started":ready_to_play}
      encrypted_reply = new serialize(aes_encryption.encrypt(reply))
      conn.sendall(encrypted_reply)
      // Receive start or not.
      start_dict = new aes_encryption.decrypt(unserialize(conn.recv(DATA_SIZE)))
      start = start_dict["ready"]
      if start AND connections >= 2 then
        ready_to_play = True
      endif
    except
      running = False
      logger.error("Something went wrong.")
    endtry
    if start then
      handle_client_communication(conn, key_string, aes_encryption)
    endif
  endwhile
  try

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

  del players[key_string]
  connections -= 1
except
  logger.info(f"No player with the ID: {key_string} exists.")
endtry
conn.close()
endprocedure

public procedure threaded_client(conn)
try
  // Send Public Key
  data_to_send = new serialize(public_key)
  conn.send(data_to_send)
  //logger.info(f"Sending Public Key: {public_key}")
  // Get AES Key and username
  dict_received_from_client = new
  rsa_encrypt.decrypt(unserialize(conn.recv(ENCRYPTION_DATA_SIZE)),private_key)
  aes_key = dict_received_from_client['aes_key']
  username = dict_received_from_client['username']
  aes_encryption = new AES_Encryption(aes_key)
  //logger.info(f"Received AES Key: {aes_key} and Username: {username}")
  // Create Player
  new_player, key_string = new create_new_player(username)
  players[key_string] = new_player
  //logger.info(f"Created New Player: {new_player}. ID: {key_string}")
  //Send player dict
  player_dict_send = {'player_data':new_player}
  encrypted_player = new serialize(aes_encryption.encrypt(player_dict_send))
  conn.send(encrypted_player)
  logger.info(f"Sending Player dict to client: {new_player}")
  waiting_zone(conn, key_string, aes_encryption)
  //handle_client_communication(conn, key_string, aes_encryption)
except
  //logger.error("An Error Occurred trying to setup Client-Server connection.")
try
  del players[key_string]
  connections -= 1
except
  logger.info(f"No player was deleted.")
endtry
conn.close()
endtry
endprocedure

public procedure run()
while running
  //// Limit on 5 players
  conn, addr = new s.accept()
  logger.debug(f"Connections: {str(connections)}")
  if ready_to_play AND connections == 0 then
    ready_to_play = False
  endif
  if ready_to_play OR connections >= 5 then
    conn.close()
  endif
endif

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

else
  connections += 1
endif
//logger.info(f"Connected to: {addr}")
//logger.info(f"There {'is' if connections== new = new 1 else 'are'} {connections} {'client' if connections== new = new 1 else 'clients'} connected to the server!")
  start_new_thread(threaded_client, (conn,))
endwhile
endprocedure

if __name__ == "__main__":
  try:
    server = new Server()
    server.run()
  except:
    logger.critical("Server has failed to launch.")
  endtry
endif
  
```

Python code:

Client:

```

class Client(Config):
  def __init__(self): ...
  def initialize_client(self, user_dict): ...
  def initialize_pygame(self, user_dict):
    try:
      # Sound and Music
      try:
        self.sword_sound = pygame.mixer.Sound("Audio/sword.wav")
        self.damage_sound = pygame.mixer.Sound("Audio/hit.wav")
        self.heal_sound = pygame.mixer.Sound("Audio/heal.wav")
        self.fire_sound = pygame.mixer.Sound("Audio/Fire.wav")
        self.death_sound = pygame.mixer.Sound("Audio/death.wav")
        self.background_music = pygame.mixer.Sound("Audio/main.ogg")
      except:
        self.sword_sound = pygame.mixer.Sound("../Audio/sword.wav")
        self.damage_sound = pygame.mixer.Sound("../Audio/hit.wav")
        self.heal_sound = pygame.mixer.Sound("../Audio/heal.wav")
        self.fire_sound = pygame.mixer.Sound("../Audio/Fire.wav")
        self.death_sound = pygame.mixer.Sound("../Audio/death.wav")
        self.background_music = pygame.mixer.Sound("../Audio/main.ogg")
    # Game Over
    self.game_over = Text(self.gameMenu.text_height, "Graphics/Fonts/Orbitron-ExtraBold.ttf", (247,155,16), None, None, None, self.gameMenu.base_text_size*10)
    self.game_over_board = Button((27,31,35), (27,31,35), self.gameMenu.settings.WIDTH/2, self.gameMenu.settings.HEIGHT/2*0.4,
                                 "Graphics/Fonts/Orbitron-Regular.ttf", (27,31,35), (27,31,35), self.gameMenu.settings.WIDTH*0.55,
                                 self.gameMenu.settings.HEIGHT*0.2,'Rectangle', None, self.gameMenu.big_text_size, int(self.gameMenu.curve*1.5))

    # Leaderboard
    self.leaderboard = Button((27,31,35), (27,31,35), self.gameMenu.settings.WIDTH/2*1.78, self.gameMenu.settings.HEIGHT/2, "Graphics/Fonts/Orbitron-Regular.ttf",
                             (27,31,35), (27,31,35), self.gameMenu.settings.WIDTH*0.2, self.gameMenu.settings.HEIGHT*0.7,'Rectangle', None,
                             self.gameMenu.big_text_size, int(self.gameMenu.curve*1.5))
    self.leaderboard_text = Text(self.gameMenu.text_height, "Graphics/Fonts/Orbitron-Bold.ttf", (240,240,240), None, None, None, self.gameMenu.base_text_size*3)
    self.leaderboard2 = Button((27,31,35), (27,31,35), self.gameMenu.settings.WIDTH/2*1.78, self.gameMenu.settings.HEIGHT/2*1.06,
                             "Graphics/Fonts/Orbitron-Regular.ttf", (27,31,35), (27,31,35), self.gameMenu.settings.WIDTH*0.18,
                             self.gameMenu.settings.HEIGHT*0.6,'Rectangle', None, self.gameMenu.big_text_size, int(self.gameMenu.curve*1.5))
    self.value_board = Text(self.gameMenu.text_height, "Graphics/Fonts/Orbitron-Medium.ttf", (240,240,240), None, None, None, self.gameMenu.base_text_size*3)

    self.game_finished = False
  except:
    logger.error("Couldn't correctly initialize pygame.")
    self.close()
  
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
def update_players(self, player_dict, dictionary_copy):
    # Update existing player instances and remove players that are not in player_dict
    try:
        ### Player Sounds ###

        for player_data, old_player_data in zip(player_dict.values(), dictionary_copy.values()):
            player_info = player_data['player']
            old_player_info = old_player_data['player']

            if player_info['id'] != self.player.id:
                ### Player Got Hit ###
                if player_info['health'] < old_player_info['health']:
                    if self.sound == 1:
                        self.damage_sound.play()

                ### Player dies ###
                if player_info['status'] == "dead" and old_player_info['status'] == "alive":
                    if self.sound == 1:
                        self.death_sound.play()

            ### Weapons ###
            weapon_ids = [weapon['player_id'] for weapon in player_data['weapon']]
            weapon_type = [weapon['type'] for weapon in player_data['weapon']]
            old_weapon_ids = [old_weapon['player_id'] for old_weapon in old_player_data['weapon']]
            new_weapon_ids = set(weapon_ids) - set(old_weapon_ids)
            if new_weapon_ids:
                for new_weapon_id in new_weapon_ids:
                    weapon_index = weapon_ids.index(new_weapon_id)
                    animation_type = weapon_type[weapon_index]
                    if self.sound == 1:
                        if animation_type == "melee":
                            self.sword_sound.play()

            ### Bullets ###
            bullet_ids = [bullet['id'] for bullet in player_data['bullets']]
            old_bullet_ids = [old_bullet['id'] for old_bullet in old_player_data['bullets']]
            new_bullet_ids = set(bullet_ids) - set(old_bullet_ids)
            if new_bullet_ids:
                for new_bullet_id in new_bullet_ids:
                    if self.sound == 1:
                        self.sword_sound.play()
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

    ### Fire Animation ###
magic_ids = [magic['player_id'] for magic in player_data['magic']]
magic_type = [magic['type'] for magic in player_data['magic']]
old_magic_ids = [old_magic['player_id'] for old_magic in old_player_data['magic']]
new_magic_ids = set(magic_ids) - set(old_magic_ids)
if new_magic_ids:
    for new_magic_id in new_magic_ids:
        magic_index = magic_ids.index(new_magic_id)
        animation_type = magic_type[magic_index]
        if self.sound == 1:
            if animation_type == "flame":
                self.fire_sound.play()
            if animation_type == "aura":
                self.heal_sound.play()

except:
    logger.error("Player disconnected.")
    self.close()

def players_number(self):
    player_count = 0
    player_list = [player.status_alive for player in self.players]
    for status in player_list:
        if status == "alive":
            player_count += 1
    return player_count

def redraw_window(self, all_players_dict, dictionary_copy):
    try:
        self.update_players(all_players_dict, dictionary_copy) # could be this
        self.level.run(self.player, self.dt)
        """self.player_attack_logic() # could be this"""
        self.player_attack_collisions()
        self.ui.player_count = self.players_number()
        self.ui.player_kill_count = self.kill_count
        self.ui.display(self.player)
        self.one_player_remaining()

        debug(f"Position: ({self.player.rect.x}, {self.player.rect.y})", self.WIDTH/2, 20)
        if self.dt != 0: debug(f"FPS: {int(1/self.dt)}", self.WIDTH/2, 50)
        debug(f"{self.player.direction}", self.WIDTH/2, 80)
        self.ui.draw_menu()
        if self.ui.draw_ui: self.player.paused = True
        else: self.player.paused = False
        pygame.display.update()
        # self.clock.tick(5)

    except:
        logger.error("Player has left the game.")
        self.close()

def create_attack(self, weapon_type="melee"):
    #logger.info("Create attack")
    try:
        if self.sound == 1 and weapon_type == "melee":
            self.sword_sound.play()
            self.current_attack = Melee(self.player, [self.level.visible_sprites], self.player.id)
    except:
        self.current_attack = None
        #logger.error("Failed to render attack image.")

def create_bullet(self):
    #logger.info("Bullet!")
    if self.sound == 1:
        self.sword_sound.play()
    self.bullet = Bullets(self.player, [self.level.visible_sprites], self.level.obstacle_sprites, self.player.id, self.level.player_sprites)

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

def create_magic(self, style, strength, cost):
    if style == 'heal':
        if self.sound == 1 and self.player.energy > cost:
            self.heal_sound.play()
            self.magic_player.heal(self.player, strength, cost, [self.level.visible_sprites], self.player.id)

    if style == 'flame':
        if self.sound == 1 and self.player.energy > cost:
            self.fire_sound.play()
            self.magic_player.flame(self.player, strength, cost, [self.level.visible_sprites], self.player.id)

def main(self):
    dictionary_copy = {}
    self.background_music.play(-1)
    if self.music == 1:
        self.background_music.set_volume(self.volume/100)
    else:
        self.background_music.set_volume(0)
    try:
        self.previous_time = time.time()
        while self.running:
            current_time = time.time()
            self.dt = current_time - self.previous_time
            self.previous_time = current_time

            self.handle_events()

        try:
            all_players_dict = self.send_player_data()

            #logger.debug(f"Sending player data: {all_players_dict}")
            #logger.debug(f"Received players dictionary: {all_players_dict}")
            self.redraw_window(all_players_dict, dictionary_copy)
            dictionary_copy = all_players_dict
        except Exception as e:
            logger.error(f"Failed to send/receive player data: {e}")
            self.close()

    except Exception as e:
        logger.error(f"Failed to run main loop: {e}")
        self.close()

    def one_player_remaining(self):
        if self.players_number() == 1 or self.game_finished:
            ## Player has won.
            self.game_finished = True

            ## Animation
            self.game_over_board.draw((27,31,35), None, None, True, None, None, None, "draw")
            self.game_over.draw("draw", "Game Over!", (self.gameMenu.settings.WIDTH/2), (self.gameMenu.settings.HEIGHT/2*0.4))

            ## Display kill counts for all players
            self.leaderboard.draw((27,31,35))
            self.leaderboard2.draw((240,240,240))
            self.leaderboard_text.draw("draw", "Leaderboard", (self.gameMenu.settings.WIDTH/2*1.78), (self.gameMenu.settings.HEIGHT/2*0.375))
            leaderboard_text = "\n".join([f'{player.username}: {player.kill_count}' for player in sorted(self.players, key=lambda x: x.kill_count, reverse=True)])
            leaderboard_list = leaderboard_text.split('\n')
            for index, leaderboard_index in enumerate(leaderboard_list):
                spacing_index = 0.52 + 0.1*index
                self.value_board.draw("draw", leaderboard_index, (self.gameMenu.settings.WIDTH/2*1.78), (self.gameMenu.settings.HEIGHT/2*spacing_index))

            ## Reset key player data
            ## Log out after a 30 second.
        else:
            self.game_over.draw("undraw", "Game Over!", (self.gameMenu.settings.WIDTH/2), (self.gameMenu.settings.HEIGHT/2*0.4))

    def handle_events(self):
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                self.quit()
            elif event.type == pygame.KEYDOWN and event.key == pygame.K_ESCAPE:
                self.ui.draw_ui = not self.ui.draw_ui

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

def send_player_data(self):
    player_dict = {
        'x': self.player.rect.x,
        'y': self.player.rect.y,
        'image': self.player.get_image_name(),
        'username': self.username,
        'status': self.player.return_alive_status(),
        'health': self.player.health,
        'id': self.id,
        'kill_count': self.kill_count,
        'killed_by': self.killed_by
    }
    player_total_dict = {
        'player': player_dict,
        'weapon': self.return_weapon(),
        'bullets': self.return_bullets(),
        'magic': self.return_magic_data()
    }
    player_encrypted_dict = self.serialize(self.encryption.encrypt(player_total_dict))
    self.network.send(player_encrypted_dict)
    all_players_dict = self.encryption.decrypt(self.unserialize(self.network.receive(self.DATA_SIZE)))
    return all_players_dict

def run(self):
    self.join_game = False
    self.gameMenu.run()
    user_dict = self.gameMenu.start_game()
    self.initialize_client(user_dict)
    self.loading_zone()
    self.main()

def loading_zone(self):
    def redraw_window():
        self.screen.fill((27,31,35))
        join_btn.draw((240,240,240),join_the_game)
        players.draw("draw", f"{connections} {'player' if connections==1 else 'players'} {'is' if connections==1 else 'are'} connected.", (self.WIDTH / 2), (self.HEIGHT / 2) * 0.4)
        join_hover = join_btn.is_hovered()
        join_click = join_btn.is_clicking()

        if join_hover:
            if not join_click:
                mouse.mode = 1
            else:
                mouse.mode = 2
        else:
            mouse.mode = 0
        mouse.draw()
        pygame.display.update()

    def join_the_game():
        self.join_game = True
    # Receive number of players
    running = True
    mouse = Mouse("Graphics/MainMenu/Mouse/mouse1.png", "Graphics/MainMenu/Mouse/mouse2.png", "Graphics/MainMenu/Mouse/mouse3.png")
    self.width_ratio = self.WIDTH / 1920
    self.height_ratio = self.HEIGHT / 1080

    # Variables
    players = Text(self.gameMenu.text_height, "Graphics/Fonts/Orbitron-Regular.ttf", (240, 240, 240), None, None, int(self.gameMenu.base_text_size * 3))
    join_btn = Button((39, 174, 96), (27,31,35), (self.gameMenu.settings.WIDTH/2), (self.gameMenu.settings.HEIGHT/2)*1.4-self.gameMenu.button_padding, "Graphics/Fonts/Orbitron-Medium.ttf", (27,31,35), (39, 174, 96), self.gameMenu.base_button_width, self.gameMenu.base_button_height, ["Start",'Rectangle', None, int(self.gameMenu.big_text_size/1.3), self.gameMenu.curve])
    connections = 0

    while running:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                self.quit()
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_ESCAPE:
                    self.quit()

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

try:
    connected_players = self.encryption.decrypt(self.unserialize(self.network.receive(self.DATA_SIZE)))
    connections = connected_players['connections']
    started = connected_players['started']
    if self.join_game and connections < 2:
        self.join_game = False
    if started:
        self.join_game = started
    game_dict = {"ready":self.join_game}
    start_the_game = self.serialize(self.encryption.encrypt(game_dict))
    self.network.send(start_the_game)
    if self.join_game or started:
        self.join_game = True
        running = False

except:
    running = False
    logger.error("Something failed.")
    self.close()

    redraw_window()

if __name__ == "__main__":
    try:
        client = Client()
        client.run()
    except:
        logger.error("Couldn't run main menu or client.")

```

Main menu

```

from Scripts.logger import *
try: ...
except:
    logger.critical("You do not have all the modules installed. Please install Pygame.")

logger.info("RealDL - Main Menu Code")

class MainMenu:
    def __init__(self):
        # Music
        try:
            self.background_music = pygame.mixer.Sound("Audio/main.ogg")
        except:
            self.background_music = pygame.mixer.Sound("../Audio/main.ogg")

    def run(self):
        self.background_music.play(-1)
        while self.loop:
            if self.music_change == 0:
                self.background_music.set_volume(0)
            else:
                self.background_music.set_volume(self.volume/100)
            events = pygame.event.get()
            for event in events:
                if event.type == pygame.QUIT:
                    self.close()
                if event.type == pygame.KEYDOWN:
                    if event.key == pygame.K_ESCAPE:
                        self.close()

            # Update the display and frame rate
            self.redraw_window(events)
            pygame.display.update()
            self.clock.tick(self.FPS)

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

Server:

```

from Scripts.logger import *
try:
    from _thread import *
    from random import randint, choice
    import string, math, socket, time
    from Scripts.settings import Config
    from Scripts.encryption import *
    from Scripts.debug import debug
except:
    logger.critical("You do not have all the modules installed. Please install Pygame, RSA and Pycryptodome.")

logger.debug("RealDL Server Code.")

class Server(Config):
    def __init__(self): ...

    def initialize_server(self): ...

    def create_random_string(self): ...

    def is_touching(self, player_x, player_y, threshold=100): ...

    def get_player_position(self): ...

    def validate_username(self, username): ...

    def create_new_player(self, username): ...

    def validate_movement(self, player_pos): ...

    def validate_health(self, player): ...

    def validate_player_status(self, player, data): ...

def kill_count(self, key_string, kill_count, killed_players):
    kills = []

    # gets a list of all ids of players that have killed someone
    for player in self.players.values():
        if player['player']['killed_by'] != "":
            if player['player']['id'] not in killed_players:
                kills.append(player['player']['killed_by'])
                killed_players.append(player['player']['id'])

    # checks if this player has killed anyone
    for kill_id in kills:
        # logger.debug(kill_count, kill_id, key_string)
        if kill_id == key_string:
            kill_count += 1
    return kill_count, killed_players

def end_game(self):
    player_count = 0
    for player in self.players.values():
        if player['player']['status'] == "alive":
            player_count += 1
    if player_count <= 1:
        return True
    else:
        return False

def validate_position(self, player):
    # Map boundaries, stops players from leaving the map.
    if player['x'] <= 180 or player['x'] >= 3400:
        return True
    elif player['y'] <= -50 or player['y'] >= 3150:
        return True
    else:
        return False
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

def handle_client_communication(self, conn, key_string, aes_encryption):
    running = True
    player_pos = []
    killed_players = []
    last_time = time.time()
    end_time = 0
    reset_game_time = 15
    while running:
        try:
            ## ONE player left. -> after 30 seconds -> kick all players -> reset all key values

            # Recieve player data
            player_data = aes_encryption.decrypt(self.deserialize(conn.recv(self.DATA_SIZE)))

            # check the player's status is correct (this doesn't result in a kick)
            data = player_data['player']
            player_data['player']['status'] = self.validate_player_status(self.players[key_string]['player'], data)

            # Validate if the player has been killed.
            player_data['player']['kill_count'], killed_players = self.kill_count(key_string, player_data['player']['kill_count'], killed_players)

            self.players[key_string] = player_data

            if self.end_game():
                if time.time() - end_time >= reset_game_time:
                    running = False
            else:
                end_time = time.time()

            # Calculate the speed of play (FPS).
            delta_time = time.time() - last_time
            last_time = time.time()

            # Validate the player's position.
            if self.validate_position(self.players[key_string]['player']):
                running = False
                logger.info(f"Player {key_string} disconnected as they tried to leave the map.")

            # Validate the player data to ensure they are not speed hacking. (this does result in a kick)
            player_pos.append([self.players[key_string]['player']['x'], self.players[key_string]['player']['y'], delta_time])
            if len(player_pos) > 20: player_pos.pop(0)
            if not self.validate_movement(player_pos):
                running = False
                logger.info(f"Player {key_string} disconnected because they were speed hacking.")

            # Validate player health to ensure they are not health hacking. (this does result in a kick)
            if not self.validate_health(self.players[key_string]['player']):
                running = False
                logger.info(f"Player {key_string} disconnected because they were health hacking.")

            #logger.info(f"Received Player Dict: {player_data}.")

            if not data:
                logger.info(f"Player {key_string} disconnected.")
                running = False
            else:
                reply = self.players
                encrypted_reply = self.serialize(aes_encryption.encrypt(reply))

            conn.sendall(encrypted_reply)
            #logger.info(f"Sending All Player Dict: {reply}.")

        except:
            logger.info(f"Player {key_string} lost connection.")
            running = False

        logger.info(f"Connection Closed for Player {key_string}.")
        try:
            del self.players[key_string]
            self.connections -= 1
        except:
            logger.info(f"No player with the ID: {key_string} exists.")
        conn.close()
    
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

def waiting_zone(self, conn, key_string, aes_encryption):
    running = True
    while running:
        try:
            # send over number of players connected
            reply = {"connections":self.connections,"started":self.ready_to_play}
            encrypted_reply = self.serialize(aes_encryption.encrypt(reply))

            conn.sendall(encrypted_reply)
            # Receive start or not.
            start_dict = aes_encryption.decrypt(self.deserialize(conn.recv(self.DATA_SIZE)))
            start = start_dict["ready"]
            if start and self.connections >= 2:
                self.ready_to_play = True
        except:
            running = False
            logger.error("Something went wrong.")

        if start:
            self.handle_client_communication(conn, key_string, aes_encryption)

    try:
        del self.players[key_string]
        self.connections -= 1
    except:
        logger.info(f"No player with the ID: {key_string} exists.")
    conn.close()

def threaded_client(self, conn):
    try:
        # Send Public Key
        data_to_send = self.serialize(self.public_key)
        conn.send(data_to_send)
        #logger.info(f"Sending Public Key: {self.public_key}")

        # Get AES Key and username
        dict_received_from_client = self.rsa_encrypt.decrypt(self.deserialize(conn.recv(self.ENCRYPTION_DATA_SIZE)),self.private_key)
        aes_key = dict_received_from_client['aes_key']
        username = dict_received_from_client['username']
        aes_encryption = AES_Encryption(aes_key)
        #logger.info(f"Received AES Key: {aes_key} and Username: {username}")

        # Create Player
        new_player, key_string = self.create_new_player(username)
        self.players[key_string] = new_player
        #logger.info(f"Created New Player: {new_player}. ID: {key_string}")

        #Send player dict
        player_dict_send = {'player_data':new_player}
        encrypted_player = self.serialize(aes_encryption.encrypt(player_dict_send))
        conn.send(encrypted_player)
        logger.info(f"Sending Player dict to client: {new_player}")
        self.waiting_zone(conn, key_string, aes_encryption)
        #self.handle_client_communication(conn, key_string, aes_encryption)
    except:
        #logger.error("An Error Occurred trying to setup Client-Server connection.")
        try:
            del self.players[key_string]
            self.connections -= 1
        except:
            logger.info(f"No player was deleted.")
    conn.close()
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

def run(self):
    while self.running:
        ## Limit on 5 players
        conn, addr = self.s.accept()
        logger.debug(f"Connections: {str(self.connections)}")

        if self.ready_to_play and self.connections == 0:
            self.ready_to_play = False

        if self.ready_to_play or self.connections >= 5:
            conn.close()
        else:
            self.connections += 1
        #logger.info(f"Connected to: {addr}")
        #logger.info(f"There {'is' if self.connections==1 else 'are'} {self.connections} connections")

        start_new_thread(self.threaded_client, (conn,))

if __name__ == "__main__":
    try:
        server = Server()
        server.run()
    except:
        logger.critical("Server has failed to launch.")
  
```

Tests proofs (Fig 6.1 - 6.7):

See MP4 videos.

Evaluation

Post-Development Testing

User requirement	Test data	Expected result	Actual result
1. The user must be able to interact with an aesthetically pleasing graphical user interface when either playing the game or navigating the main menu system. I will use Pygame to achieve this as well as my personal Pygame function module to create the buttons.	The buttons must smoothly change colour when hovered over. The mouse must change colour when clicking on a button.	Buttons colours should change slowly and smoothly and change back. The mouse should look like it is hovering with the colour slightly grey.	This worked as expected. The buttons changed colour. This worked as expected as well. The mouse was slightly grey when clicking.

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

	The background must continuously be should move from left moving slowly.	The background should move from left to right always.	This was a success. The background was moving.
2. The main menu system must have an option to quit the game. Another option to configure settings like controls and audio. Finally, an option to start the game by entering the IP Address of the server. In addition, players should be able to enter their name into another text box and have custom names in the game. The main menu system must also be able to connect players to the server when they choose to join a game.	The user should click on the quit button.	The client screen should close.	This worked as expected.
	Click on the settings button, change the key binds and audio settings. Set the volume to 50% and turn the sound off.	The user should have a settings UI appear in front of them then they can select different options to change the volume and turn the sound off.	This worked. They could set the volume to 50% and turn the sound off.
	The user should type into the username text box a name. They should type the IP into the server IP textbox and connect to the game pressing the connect button.	The user should be able to type into the two boxes. They should be able to choose their name and write the IP in and connect to the game.	This worked. The users could type in both text boxes and connect to the game.
3. When the server creates the player, they must have a unique ID so that they can be identified in the server. If a player leaves, that corresponding ID should be deleted from the player list ensuring players no longer show up on other clients' screen. Furthermore, players should leave seamlessly without affecting the server or any clients.	Connect a player to the game.	They should be given a unique ID.	User were given a unique ID as intended.
	Disconnect a player from the game.	That ID should no longer exist, and they should be deleted.	The user ID was deleted as expected.
	Seamless client disconnection when a player leaves the game.	Players should see one player has been removed from the map visually without any noticeable lag or disruption.	This worked. One player was removed from the map.
4. The game must be a locally hosted multiplayer game that must feature the last man standing game mode. In addition, there should be no player limit allowing as many players as possible to join during the allocate time before the game starts. Furthermore, the game should not start the counter until at least one player has join and should not start until there are at least two players. Finally, the game should end when there is only one left alive.	Connect to the game from another computer over the LAN.	A client should be able to connect to the server hosted by another computer.	This was a success. A client could connect to another computer running the client.
	Have 6 players join the game to test if the game has no player limit.	The server should allow 6 players to connect to the game.	This was a failure. Only 5 players could connect to the game due to the player limit imposed in sprint 6.
	Start the game with one player.	The game should start with a player.	This was a failure since at least 2 players must be in the game to start it.
	Have one player alive at the end of the game.	The game should end with an end game animation and a	This was a success. The game over animation

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

		leaderboard then it should disconnect all players.	appeared as well as the leaderboard.
5. When players' health bar reaches zero or below, they should become invisible. They should not be able to be seen by other players but should be able to see other players. Their character skin should change to a ghost when they die and should change back when the game is over when the waiting period starts.	Have a player die.	That player shouldn't be seen by alive players.	This worked. A player couldn't be seen by other players.
	Have another player die.	Their skin should become ghost like. Their character should be slightly transparent, and their username should be grey.	When the player died, they did look like a ghost.
	Have a player die then end the game.	The player should turn back into a normal player after the game is over and the waiting period starts again.	This didn't work. Players were kicked from the game instead due to the additions in sprint 6.
6. The players must be able to take damage from other players. The health bar should decrease when a player is hit. Furthermore, the server must be able to update that to all the other players who are playing the game.	Let a player get damaged by another player.	That player's health bar (the red bar) should decrease by the damage of the weapon.	This worked as intended. Players that were damaged had their health reduced.
	The server should send the health of that player to the other players.	The health of the damaged player should be the same for all clients.	This worked as intended. The health was sent to all clients.
7. The game must include a variety of different weapons for players to use to eliminate other players. The server must be able to register that a player has been damaged and update a player's health to all players.	Switch through all the weapons and use them visually as well as the magic abilities.	There should be 5 weapons, 3 guns and 2 magic abilities that should all work.	All 5 weapons, 3 guns and 2 magical abilities worked.
	The server should send the player dictionary to all clients.	All players should update the health of that player on their client.	This was already proved in the second proof of user requirement 6. The player dictionary is sent to all players.
8. The map must be aesthetically pleasing and feature some custom objects. In addition, there must be an invisible barrier which prevent players from escaping the map.	Explore around the map.	There should be custom objects and the map should be aesthetically pleasing.	The map was aesthetically pleasing. It had different custom objects.
	A player should try walk out of the designated map.	The player should not be able to walk outside of the map.	This was a success. The player could not walk outside the map.
9. The server must use asymmetric encryption to share the public key with	The server should send over the public key.	The client should be able to receive the public key.	The client was able to receive the public key.

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

each client. Clients should then encrypt their symmetric key and share that with the server to establish an encrypted connection. The server and clients should then encrypt and decrypt player data with the symmetric ensuring data security. Furthermore, no external threats should be able to do anything with the encrypted data.	The server should receive the symmetric key.	The client should be able to send the encrypted symmetric key to the server.	The client could send the encrypted symmetric key. The server could receive it.
	The server should encrypt the player dictionary and send it over to the client.	The client should decrypt the player dictionary from the server.	This was a success. We received the encrypted dictionary and were able to decrypt it.
10. The game should be a 2D top-down game with a player centred camera view. In addition, the game should feature some camera movement so that the player will never move off the screen allowing for a larger, more immersive map.	Walk around the map and collide into some obstacles.	The player should appear slightly over the object if they are lower than the object.	This worked as intended. The player was drawn under the objects when they were above them.
	The player should walk around the game map.	The game map should be 2D and the camera should always be centred on the player.	This worked as intended. The camera was always centred on the player and the game map was 2D.
11. The server must be robust and include an efficient server system that can handle multiple simultaneous connections whilst ensuring a smooth and lag free experience for players.	Have 5 players connect to the game and move around.	The game should be smooth and lag free for those connection.	This worked as intended although it was slightly laggy.
	Connect 3 players into the game.	The game experience should be smooth for all players.	This worked. It was a smooth experience for all clients. Their FPS was decent and above 60.
12. The game must have an in-game user interface that displays important information such as health, kill count and different abilities. Furthermore, players must be able to use power ups and other abilities.	There should be an in-game UI.	The in-game UI should feature the Exp, health bar, energy bar, kill count, players alive and 2 boxes displaying the current weapon and magic held.	There was in-game UI. This was a success.
	Kill 3 players.	The kill count should increase from 0 to 3 for that player. The number of players in the game should decrease by 3 for all clients.	This worked as intended. The player's kill count increased from 0 to 3 then the overall player count decreased from 4 to 1.
	Use the power ups.	All clients should be able to use the magic attack and heal animation with the corresponding sounds and graphics.	This worked as intended. Each client could use the heal and flame animation.

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

13. Players must be able to receive a small celebratory animation when a player wins the game. This must be a few seconds long and allow for any player statistics to be displayed on screen for that round before players join the waiting zone again.	Have one player win the game.	There should be a small win animation for a few seconds.	This was a success. There was a smooth win animation.
	Have a player get a kill count of 2 and another player get a kill count of 1.	The leaderboard should display the kill counts of each player in descending order.	This was a success. The player who had a kill count of 2 was first followed by the player with the kill count of 1.
	Change the key binds to arrow keys, left control and enter.	The player should only use those key binds in the game.	This was a success. Only Arrow keys, L-CTRL and Enter were able to be used as controls.
14. Players must be able to change their controls in the settings. Players must also be able to use either the WASD keys or the arrow keys to move around. In addition, players should be able to use the space key or left/right control key to attack other players.	Change the keys binds to WASD, right control and right shift.	The client should only be able to control the player with movement and both attacks with WASD, right control and right shift.	This was a success. Players could only use those intended controls.
	Leave the original controls as they are.	The player should use WASD for movement, space for attacking and left control for magic attacking only.	This was a success. Players could only use those controls.
	Switch through all the weapons and attack with them.	All the weapons should be visually different and look different attacking.	All the weapons were visually different. The guns and bullets look different. This was a success.
15. The game should have a variety of unique and visually pleasing weapons. Each should have its own characteristics and attributes, offering players a diverse gameplay experience.	Attack a player with all the weapons once.	Each weapon should have a different cooldown and attack damage to the player.	This was a success. The weapons have varying cooldowns and damage.

Stakeholder feedback

After the post development testing being mainly a success, I now need to ask my stakeholders in-depth questions about my project to determine their thoughts on the solution and how they feel about my project. I will ask Roland first then I will ask James his opinions on my project.

How did you feel about the overall visual appeal and user interface design of the game? Did the user interface of the game feel unique and enjoyable?

Roland: I believe that the game was very visually appealing. I do like the user interface and how the buttons change colour smoothly when hovered over instead of bluntly changing colour. I do think that the game was very unique and enjoyable to navigate the user interface without any difficulty.

James: To me the user interface was aesthetically pleasing. The moving background as well as the other visually pleasing buttons made the overall user interface experience great. I do feel that the user interface was unique but at the same time it was easy and fun to navigate it and change the user settings.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Were you personally satisfied with the responsiveness and intuitiveness of the button interactions? Did the transformations feel smooth and satisfying to click?

Roland: Yes, the responsiveness of the buttons was great. The overall button interactions were fine. I thought that they were good, and the overall responsiveness of the buttons was fast. I do feel that the buttons were satisfactory when clicking due to the level of the aesthetics that went into designing the user interface.

James: I am personally satisfied with the button responsiveness. I think that the button interactions were effective and efficient. I think that the buttons were fine. I do feel that the buttons did add to the overall user interface.

Did the background movement contribute positively to your immersion and enjoyment of the game?

Roland: Yes absolutely. The slow movement of the background in the main menu was great. A non-moving background would have worked with me, but a moving background did make the overall game and main menu look more appealing and immersive.

James: The background movement was great. It is a small effect, but it does add to the main menu and helps it look much better. Overall, the moving background added to my immersion into the game.

From your perspective, how easy was it to navigate through the main menu system and access different options?

Roland: The main menu was easy to navigate through. I found that it was quite easy to change the settings and to join the game. I like the ability to write IP into the game and join a specific server.

James: Yes. The main menu was easy to navigate through. I enjoyed navigating and finding what I was looking for as well as joining a game. The main menu system was easy to navigate and find something.

Were you able to configure settings like controls and audio to your liking, and did it enhance your gameplay experience?

Roland: Absolutely, I found the customisation options quite comprehensive. Adjusting controls and audio settings definitely enhanced my gameplay experience, allowing me to tailor the game to my preferences.

James: Yeah, I agree with Roland. Being able to tweak controls and audio settings was essential for me to optimise my gaming experience. It definitely added to the immersion and made gameplay more enjoyable.

How seamless was the process of joining a game via IP address and setting up your player profile?

Roland: Joining a game via IP address was surprisingly smooth. Setting up my player profile was straightforward as well. I didn't encounter any significant issues during the process.

James: I had a similar experience. Joining games via IP address was relatively seamless, and setting up my player profile was hassle-free. Overall, the process was user-friendly and didn't pose any major challenges.

Did you feel that the player identification system on the server provided a smooth and cohesive multiplayer experience?

Roland: Yes, I found the player identification system quite effective. It definitely contributed to a smooth and cohesive multiplayer experience by helping me keep track of other players and communicate more efficiently.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

James: Absolutely, the player identification system worked well for me too. It made it easier to coordinate with teammates and strategies during gameplay. Overall, it enhanced the multiplayer experience significantly.

How engaging did you find the last man standing game mode, and were there any aspects you particularly enjoyed or disliked?

Roland: The last man standing game mode was incredibly engaging. I enjoyed the adrenaline rush of being the sole survivor and the intense gameplay it offered. However, I did find that matches could sometimes drag on if players were too cautious.

James: I also found the last man standing mode very engaging. It kept me on the edge of my seat throughout the entire match. One aspect I particularly enjoyed was the strategic element of choosing when to engage with other players. However, sometimes the waiting periods between encounters could be a bit dull.

What were your thoughts on the visibility mechanics upon player death and rejoining the game after the waiting period?

Roland: I thought the visibility mechanics upon player death were well-implemented. It added an extra layer of challenge and suspense to the gameplay. Rejoining the game after the waiting period felt fair and balanced.

James: Yeah, I agree. The visibility mechanics after player death added to the tension of the game. It made each encounter feel significant. Rejoining the game after the waiting period was smooth and didn't disrupt the flow of gameplay.

Did you feel that the player damage system and health bar updates effectively communicated the intensity of gameplay?

Roland: Absolutely, the player damage system and health bar updates were crucial for understanding the intensity of gameplay. It provided clear feedback on the consequences of actions and helped me make better tactical decisions.

James: Yeah, I found the player damage system and health bar updates very informative. They gave me a good sense of how I was faring in combat and whether I needed to adjust my strategy. Overall, they effectively communicated the intensity of gameplay.

Which weapons did you find most enjoyable or effective to use, and were there any balance issues you encountered?

Roland: Personally, I found the guns to be the most enjoyable and effective weapon to use. Its versatility allowed me to adapt to various combat situations. I didn't encounter any significant balance issues with the weapons during my gameplay.

James: I preferred using the machine gun. It packed a punch at close range and was great for clearing out tight spaces. Overall, I felt the weapons were well-balanced, and each had its strengths and weaknesses that added depth to the gameplay.

How did you personally perceive the map design in terms of aesthetics and navigational challenges?

Roland: I thought the map design was visually appealing, with a good mix of environments and terrain features. Navigational challenges were present but not overwhelming, adding to the immersion without becoming frustrating.

James: Yeah, I enjoyed the aesthetics of the map design. It felt immersive and diverse, with plenty of interesting landmarks to explore. Navigational challenges were just right, providing a good balance between exploration and strategic positioning.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

How confident do you feel in the security measures implemented, particularly regarding encryption and data transmission?

Roland: I felt fairly confident in the security measures implemented. The encryption and data transmission protocols seemed robust, which reassured me about the safety of my personal information while gaming.

James: Yeah, I agree. The security measures seemed solid, and I didn't have any concerns about the safety of my data while playing. Overall, I felt confident in the platform's approach to security.

Did the camera movement and player-centred view contribute positively to your overall gaming experience?

Roland: Absolutely, the camera movement and player-centred view were crucial for maintaining immersion and situational awareness during gameplay. They contributed positively to my overall gaming experience by providing a smooth and intuitive interface.

James: Yeah, I found the camera movement and player-centred view very intuitive. They helped me stay focused during intense moments and added to the overall immersion of the game. Overall, they definitely contributed positively to my gaming experience.

From your perspective, how smoothly did the server perform, especially during moments of high player activity?

Roland: The server performance was generally solid, even during moments of high player activity. I experienced minimal lag or downtime, which allowed me to fully enjoy the gameplay without interruption.

James: Yeah, I was impressed with how smoothly the server performed, even when there were a lot of players online. I rarely experienced any lag or connectivity issues, which made for a seamless gaming experience overall.

Evaluation of User Requirements

User requirement 1

The user must be able to interact with an aesthetically pleasing graphical user interface when either playing the game or navigating the main menu system. I will use Pygame to achieve this as well as my personal Pygame function module to create the buttons.

I believe that I have created a user face that users can interact with. I also believe that the user interface that I developed was aesthetically pleasing as intended. My testing for the user requirement 1 does prove that my game is aesthetically pleasing as well as it being easy to navigate through.

User requirement 2

The main menu system must have an option to quit the game. Another option to configure settings like controls and audio. Finally, an option to start the game by entering the IP Address of the server. In addition, players should be able to enter their name into another text box and have custom names in the game. The main menu system must also be able to connect players to the server when they choose to join a game.

I personally think that I have completed user requirement 2. Firstly, my testing does prove that the main menu has an option to quit the game which does work. In addition, the other option was to configure settings which was possible to do in my game; and lastly, the user was able to enter their

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

name and the server IP address to join the game. Therefore, I believe that I have completed this user requirement.

User requirement 3

When the server creates the player, they must have a unique ID so that they can be identified in the server. If a player leaves, that corresponding ID should be deleted from the player list ensuring players no longer show up on other clients' screen. Furthermore, players should leave seamlessly without affecting the server or any clients.

This user requirement has been completed as well. The player does join with a unique ID, and it is able to be identified by the server. Furthermore, when a player leaves the game all their data is deleted from the game and their ID does not exist anymore. Finally, client do leave seamlessly as proven in the testing above. Therefore, this user requirement was completed successfully as I planned.

User requirement 4

The game must be a locally hosted multiplayer game that must feature the last man standing game mode. In addition, there should be no player limit allowing as many players as possible to join during the allocate time before the game starts. Furthermore, the game should not start the counter until at least one player has join and should not start until there are at least two players. Finally, the game should end when there is only one left alive.

This user requirement was partially met. The first part of the user requirement was met as the game does feature a last man standing game mode. However, due to the found limitations of the game in sprint 6, I decided to implement a player limit at 5 players. This does mean that my game does have a player limit now which means that this user requirement has not been met, however, the game in theory could have no player limit, but this would mean that each client and the server is very laggy. The second part of this user requirement was implemented differently to how I intended to implement it. There is no allocated time to join the game, players instead join the waiting zone which requires at least two players to join the game. That part is similar requiring two players to start the game, however, clients decide when the game starts instead of the server. That is a user requirement that was partially met but also implemented in a different way.

User requirement 5

When players' health bar reaches zero or below, they should become invisible. They should not be able to be seen by other players but should be able to see other players. Their character skin should change to a ghost when they die and should change back when the game is over when the waiting period starts.

This user requirement was mostly met however when implementing the game, I decided to kick all players from the game instead of turning dead players back into alive players and starting the game again. As proven by my testing above, when the health bar reaches 0, players become ghosts which are invisible and cannot interact with players. Players do not change back to regular players once they have died, they instead get kicked. This is because it is easier on the server and clients to have players rejoin the game again as a new player which makes it fair. It is also because implementing something difficult like that would be time consuming which is why I reverted to kicking players to reset the game. Overall, this requirement was mostly met except for players changing back to normal players after the game is over.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

User requirement 6

The players must be able to take damage from other players. The health bar should decrease when a player is hit. Furthermore, the server must be able to update that to all the other players who are playing the game.

The user requirement was met. As proven in my tests, players can take damage from other players and their health bar (red bar) does decrease. The server does in fact update the health to all players since all players do receive the player dictionary which does contain their health. Although their health is not necessarily used by the other clients, it is useful to render the player on the screen and is therefore met.

User requirement 7

The game must include a variety of different weapons for players to use to eliminate other players. The server must be able to register that a player has been damaged and update a player's health to all players.

This user requirement was met due to my tests above. The game does have a variety of weapons which all work as the player can eliminate players using any weapon. The client is also able to register that they have been damaged as their health bar goes down as well as their health being updated for all the clients. In conclusion, this user requirement has been met as is up to standard.

User requirement 8

The map must be aesthetically pleasing and feature some custom objects. In addition, there must be an invisible barrier which prevent players from escaping the map.

This user requirement was met. The map was aesthetically pleasing and does feature many different custom objects. Furthermore, as my testing proves, there is an invisible barrier that blocks players from escaping the map which therefore means that this user requirement has been met as planned.

User requirement 9

The server must use asymmetric encryption to share the public key with each client. Clients should then encrypt their symmetric key and share that with the server to establish an encrypted connection. The server and clients should then encrypt and decrypt player data with the symmetric ensuring data security. Furthermore, no external threats should be able to do anything with the encrypted data.

As proven by my tests, this user requirement has been met. The client can receive the public key from the server and the server can receive the personal symmetric key from the client. This key is personal to each client which means no client could hack or have unauthorised access to another client. The server has been proved to create a private connection between a client which means they were able to encrypt and decrypt player data with the symmetric key. Finally, no external threats could access the game due to the encryption. The server could have its information intercepted fairly easily; however, the encryption means that the information that is intercepted is not readable which means they could do nothing with the encrypted data. Overall, this user requirement was met which is good for the safety and security of the server.

User requirement 10

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

The game should be a 2D top-down game with a player centred camera view. In addition, the game should feature some camera movement so that the player will never move off the screen allowing for a larger, more immersive map.

The game is a 2D top-down style of game. As proven by most of my tests, the player is always in the centre of the screen as I designed. This does allow for a larger game map as the player is not limited to the size of their screen. Overall, this user requirement was met.

User requirement 11

The server must be robust and include an efficient server system that can handle multiple simultaneous connections whilst ensuring a smooth and lag free experience for players.

As proven in my tests, the server is robust as it can comfortably support multiple players. It can handle multiple simultaneous connections as well as remaining smooth and lag free, therefore, this user requirement has been proven as successfully completed.

User requirement 12

The game must have an in-game user interface that displays important information such as health, kill count and different abilities. Furthermore, players must be able to use power ups and other abilities.

User requirement 12 want completed successfully as proven by my tests. Firstly, the game does have an in-game UI which does display vital information that player should know. Furthermore, players can use the power ups and other abilities which are the magic animations. Overall, this user requirement was success as I proved that I have implemented what I stated.

User requirement 13

Players must be able to receive a small celebratory animation when a player wins the game. This must be a few seconds long and allow for any player statistics to be displayed on screen for that round before players join the waiting zone again.

This user requirement was mostly successful. There is a small celebratory animation when a player wins as proven by my tests as well as there is a leaderboard that is displayed rather than all the players statistic being displayed. The reason only the kill count is displayed and not everything, is due to the complexity and time required, but also because my stakeholders probably are not interested in all the stats at the end of the game, they just want to play the game. The celebration animation does last on the screen for 15 seconds before players are all kicked from the game. Overall, this user requirement was basically met apart from a few changes but in conclusion the main elements of this user requirement were completed.

User requirement 14

Players must be able to change their controls in the settings. Players must also be able to use either the WASD keys or the arrow keys to move around. In addition, players should be able to use the space key or left/right control key to attack other players.

This user requirement was met. Whatever controls are selected in the main menu are the exact control that are used in the game. This was proven by my tests and therefore, this user requirement was successful.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

User requirement 15

The game should have a variety of unique and visually pleasing weapons. Each should have its own characteristics and attributes, offering players a diverse gameplay experience.

Lastly, this user requirement has been met. The game includes multiple different visually pleasing weapons as well each weapon having its own characteristics like damage, cooldown, and speed for guns. Therefore, this user requirement was met.

Conclusion

After evaluating all my user requirements, I have successfully completed 12 out of the 15 which I think is good as well as almost completing or slightly changing the other 3 user requirements. In conclusion, I believe that the user requirements I set were reasonable and realistic to complete something in time and of a decent quality and size.

Evaluation of Usability Features

Buttons and Text

Every button and text used for the in-game UI, or the main menu UI was adequately sized, easy to click on and satisfying click. My stakeholders did agree that the buttons were good and satisfying to click on. All the buttons are intuitive to click on. The buttons lead to the corresponding destination. Overall, the button and text were well developed in my game.

Main Menu UI

From the information given by my stakeholders above, I can determine that the main menu UI was implemented effectively and successfully. The main menu UI was a success since all the settings options visually work but have also been proven to work in the game. Therefore, this was a successful usability feature.

In-game UI

As seen in my stakeholder interview, they think that the UI is generally good. The in-game UI is clean, and aesthetic but also it displays necessary information for users during the game. Overall, the in-game UI was completed successfully and is easy to understand the information.

Player movement

Player movement was completed successfully. This was developed in sprint 2 with the animations however it was tweaked slightly with the delta time in sprint 5 which slightly increased the speed. Overall, the player movement was fast enough to be engaging but not too slow that the game was boring. In conclusion, the player movement was completed successfully. My stakeholders didn't have a problem with the mechanics of the player movement.

Player attacks and collisions

The overall players attacking visuals worked and the collisions for the players worked. This was a successful usability feature as players can see the weapons visually as well as them working by reducing the players health. Overall, this was a successful usability feature.

Game Map and Camera

The game map was a successful feature. Although it is not a usability feature, it was still up to standard and was an aesthetic map according to my stakeholders.

Game over and leaderboard

The game over feature worked well. It worked as intended as proven by my tests. This was developed in the last sprint and is useful for any player who is interested in the stats of each player after each game.

Conclusion

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Overall, most if not all the features I created are aesthetic but also functional. The usability features are good. Most of the features described in my analysis have been completed. Overall, all my usability features are functional and were created effectively ensuring user satisfaction when playing.

Maintenace and Potential Issues

Server-client encrypted connection

The first thing that needs to be maintained is the encrypted connection, this is mainly due to me using external modules that may get outdated and possibly cause the game to break in the future. The encryption modules RSA and Pycryptodome will need to be monitored to ensure that the encryption remains solid as well as the version of python. Any version of python running a client script lower than 3.10 will not work. It is vital that the version of python is 3.10 to 3.12 otherwise the game will not work for that user. I would have to ensure that for my game to work, it must be running on a python version 3.10 to 3.12 with the same versions of those external modules that support those versions of python to ensure that my game can be maintained.

Python FPS lag

When there are more than 4 or 5 players connected to the game, it can get very laggy as the FPS will drop to sometimes less than 20. I found out that collisions don't exactly work on players that have an average FPS less than 60. This could be due to the time delta since when clients are running at the same speed with a lower FPS they could jump through obstacles. Another thing to mention is in server code I implemented a method that kicks players if they escape the map. This was only 2 sets of x-axis coordinates and 2 sets of y-axis coordinates. This does work as a temporary fix, but it could be more accurate, and it doesn't solve the issue. This is partly why I implemented the player limit of 5 as it will reduce the overall FPS dropping for players. This could also be because python is not meant to create a game like mine that sends data over the LAN. If I wanted to make this game be more responsive and have higher overall FPS, it would be better to run it with a more powerful software and programming language like C++. I have learnt that python is not made for graphical interfaces as much, but it is used for data analytics. C++ would have been the better programming language to use however it is very complex and programming in python was easiest to me.

Pygame file locations

Something that could be an issue is creating rendering images in using pygame. If you are running the client code like "py code/client.py" it is different to running "py client.py." This might seem odd, however is a potential issue as it now means pygame needs to get files from "../[file].png" or just "[file].png". This has and can be combatted by using a try except statement; however, I do believe that this is an issue with pygame. I had to use a try and except statement for every image rendered or sound effect created in the code otherwise it could fail.

Joining the game

Another potential issue that has been found was joining the game. The main issue with this is slow generation of the map in level. I couldn't fix this, but it can be dependent on the connection to the internet. What ends up happening is that clients are left with a screen that is trying to load but eventually times out. This is not a connection issue as all clients do connect to the game as the logs show that the player has joined the server and received their player dictionary. This is an issue with rendering the map. Sometimes the map generation takes a few seconds to other times it might take 100 times longer which is when the game times out. Overall, this could be a major issue however it sometimes doesn't connect. This feature would need updating and maintenance since this is one of the more difficult elements of the game as the player is connecting.

Collisions and guns; bullets

After testing the game on my own, I found that sometimes collisions between other players don't work. The reason might be due to the FPS difference since players detect whether they have been

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

hit by another player on their own client which is not the best practise since when a client who has a high FPS hits a player with a lower FPS. It could be because this is a multiplayer game, but it also may be due to another issue that I haven't figured out. Leading on to this, I couldn't find a way to delete the bullets when it hit a player. This is probably another issue with the way I set up the player and bullet collisions, but bullet is deleted when they hit the stationary trees, objects or the invisible barrier. I found another potential issue in the game with the guns, sometimes they don't shoot properly, although this could be an FPS issue or a connectivity issue. It is important to mention that this is only some of the time and not all the time. The guns and bullets usually work along with the collisions however the bullets never get deleted when they hit the player which is something I struggled to solve but ultimately could not solve.

Movement

The last potential issue is the movement. There was an issue where the player movement would jump, but this was because of me rounding the position of the player. When I removed the rounding, it resolved back to normal, however, movement and animations are affected by the FPS. The speed is the same but the just in position between each frame are not the same. This is why in older sprints, increasing your FPS would increase your speed. Overall, this is not a huge issue, but nonetheless is has been and could be a potential problem in the future.

Conclusion

The only other thing that should be maintained properly is the server-client connection. If this relationship is solid when the game runs, then the maintenance is minimal, and it is comparable to a single player game assume the encryption is not an issue. The only other issue is where we create the map and create the other classes for clients to render other players. This means that the version of pygame used must be the same as the final sprint so that the client and server script will continue to run. Overall, the main elements that need to be monitored and maintained will be the server-client connection, graphics rendering and the collisions. If these are maintained, then the game will run as intended and smoothly which creates a good game experience for my stakeholders.

Potential improvements

After completing my project to a decent standard and completing most of my user requirements here are a few things that I could have added if I had the time to do so.

Accounts

I could have implemented account into the game in if I longer. Firstly, my stakeholders were interested in accounts, but I ended up not creating them since they are very complicated, and I would not have enough time.

Deleting bullets

Something I touched up on earlier was deleting bullets when they hit the player. This is a feature that I would have like to add but due to the lack of time, I was unable to properly have a look at that since I had to move on and create the other elements of the game.

Player skins

Although this was not talked about in my sprints, it was an option in my analysis to add different player skins. This would have been a nice addition in the game since all players have the same skin when they play. This could have increased game diversity and player enjoyment when it comes to customisability in the game.

Database

Another thing that I could have added was a database to store usernames and passwords for accounts or to store player data from previous rounds. This is a potential improvement and a feature that could have been added if I had time, but I don't think that this would have change the game and the way it's played.

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

IP textbox

As seen in the main menu UI. There is a textbox for the IP. What I didn't realise was that you need the IP and the port to join a server. An improvement I could have added would have been to have another text box for the port or to combine the IP and the port to be "IP:port" then use string manipulation to get the IP and port separately. For my game this is not an imperative as I am mainly using the same port 5555 but if that port is taken it would be useful to be able to host the server on another port and to be able to access that port and IP instead of relying on the port being 5555.

Non-player enemies

I was considering having a third-party enemy in the game that was not controlled by players, but I decided not to since to establish a multiplayer connection is hard, and it would only make the game more complex and would increase the overall time complexity to do so. This could have been a nice feature to the last man standing game mode offering for a more unique gameplay, but it could be seen as unnecessary or taking away from the player vs player nature of the last man standing game mode.

Game rules

Another thing that could have been implemented to improve the game would have been a help page on the main menu GUI. It could have explained all the controls as well as the overall concept and goal in the game. Admittedly this is not a major feature that is not in the game but for a larger game it is necessary, however since my game is smaller, it is not as important as this is a prototype game rather than a proper professional game.

Upgrades

Something that I was planning to implement into my game but never had time was to add some upgrades. This would make the game more interesting however I ran out of time. If I had time, I would have added this feature to my game. The exp in the bottom right increases by 1000 when that player kills a player. I would have used that as a currency to upgrade the weapons, guns and their magical abilities. This could have increased player engagement as they would be able to become stronger than the other players.

Game over

The last thing that could have been improved was the game over animation. It could have been more exciting and have been effects rather than just fading in. Although that is an effect, it would have been good, if I had the time, to create a game over animation that was longer and in more depth.

Conclusion

In conclusion, there are some elements that I just didn't have time to implement that would have been interesting like a database or accounts, but there are also effects that could have been done better like the game over animation. In addition, there are also additional effects like the ports or help page that I was unaware of but that could have been an improvement. Overall, these improvements would make my game slightly better and more enjoyable for players, I don't think that they were necessarily going to make my game much better. I was able to implement everything I intended to do so with a few adjustments which is good. All in all, I believe that these additions would be good for my game however there wouldn't have been time to implement them into the game which is why they aren't in the game.

Appendix

Pseudo code:

Client:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
from Scripts.logger import *
try
    import pygame, sys, time
    from Scripts.network import Network
    from Scripts.settings import Config
    from Scripts.level import Level
    from Scripts.player import Player
    from Scripts.weapon import *
    from Scripts.encryption import *
    from Scripts.debug import debug
    from Scripts.functions import *
    from Scripts.ui import UI
    from Scripts.main_menu import MainMenu
    from Scripts.magic import MagicPlayer
    from Scripts.particles import AnimationPlayer

except
    logger.critical("You do NOT have all the modules installed. Please install Pygame, RSA AND Pycryptodome.")
endtry
logger.info("RealDL - Client Code")
pygame.init()

class Client inherits Config
    private height_ratio
    private width_ratio
    private ui.draw_ui
    private ui.player_kill_count
    private ui.player_count
    private player.kill_count
    private player.exp
    private player
    private killed_by
    private health
    private status
    private id
    private player_image
    private player_y
    private player_x
    private encryption
    private aes_key
    private rsa_encrypt
    private public_key
    private network
    private game_finished
    private value_board
    private leaderboard2
    private leaderboard_text
    private leaderboard
    private game_over_board
    private game_over
    private dt
    private previous_time
    private SERVER
    private server_ip
    private username
    private music
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
private sound
private volume
private magic
private offense
private movement
private attacked_sprites_ids
private bullet
private current_attack
private running
private ui
private magic_player
private animation_player
private background_music
private death_sound
private fire_sound
private heal_sound
private damage_sound
private sword_sound
private level
private kill_count
private player_count
private magic_animations
private weapons
private bullets
private players
private custom_mouse
private clock
private screen
private gameMenu
private join_game

public procedure new()
try
    Config.__init__()
    join_game = False
    gameMenu = new MainMenu()
except
    logger.error("Error, couldn't initialize the main menu.")
    close()
endtry
endprocedure

public procedure initialize_client(user_dict)
    logger.info("Client Connecting to Server")
    try
        initialize_pygame(user_dict)
        initialize_network()
    except
        logger.error("Error, couldn't initialize the client.")
        close()
    endtry
endprocedure

public procedure initialize_pygame(user_dict)
    try
        screen = new gameMenu.screen // pygame.display.get_surface()
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
clock = new gameMenu.clock // pygame.time.Clock()
custom_mouse = new Mouse("Graphics/MainMenu/Mouse/mouse1.png",
"Graphics/MainMenu/Mouse/mouse2.png", "Graphics/MainMenu/Mouse/mouse3.png")
players = []
bullets = []
weapons = []
magic_animations = []
player_count = players.length
kill_count = 0
level = new Level()
// Sound and Music
try
    sword_sound = new pygame.mixer.Sound("Audio/sword.wav")
    damage_sound = new pygame.mixer.Sound("Audio/hit.wav")
    heal_sound = new pygame.mixer.Sound("Audio/heal.wav")
    fire_sound = new pygame.mixer.Sound("Audio/Fire.wav")
    death_sound = new pygame.mixer.Sound("Audio/death.wav")
    background_music = new pygame.mixer.Sound("Audio/main.ogg")
except
    sword_sound = new pygame.mixer.Sound("../Audio/sword.wav")
    damage_sound = new pygame.mixer.Sound("../Audio/hit.wav")
    heal_sound = new pygame.mixer.Sound("../Audio/heal.wav")
    fire_sound = new pygame.mixer.Sound("../Audio/Fire.wav")
    death_sound = new pygame.mixer.Sound("../Audio/death.wav")
    background_music = new pygame.mixer.Sound("../Audio/main.ogg")
endtry
// particles
animation_player = new AnimationPlayer()
magic_player = new MagicPlayer(animation_player)
ui = new UI(custom_mouse, close, player_count, kill_count)
running = True
// attack sprites
current_attack = None
bullet = None
attacked_sprites_ids = []
// User settings
settings = user_dict['settings']
start = user_dict['start']
movement = settings['control']['movement']
offense = settings['control']['offense']
magic = settings['control']['magic']
volume = settings['audio']['volume']
sound = settings['audio']['sound']
music = settings['audio']['music']
username = start['username']
server_ip = start['server_ip']
SERVER = server_ip
previous_time = 0
dt = 0
// Game Over
game_over = new Text(gameMenu.text_height, "Graphics/Fonts/Orbitron-ExtraBold.ttf", (247,155,16), None, None,
None, gameMenu.base_text_size*10)
game_over_board = new Button((27,31,35), (27,31,35), gameMenu.settings.WIDTH/2,
gameMenu.settings.HEIGHT/2*0.4, "Graphics/Fonts/Orbitron-Regular.ttf", (27,31,35), (27,31,35),
gameMenu.settings.WIDTH*0.55, gameMenu.settings.HEIGHT*0.2,'Rectangle', None, gameMenu.big_text_size,
int(gameMenu.curve*1.5))
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

// Leaderboard
leaderboard = new Button((27,31,35), (27,31,35), gameMenu.settings.WIDTH/2*1.78,
gameMenu.settings.HEIGHT/2, "Graphics/Fonts/Orbitron-Regular.ttf", (27,31,35), (27,31,35),
gameMenu.settings.WIDTH*0.2, gameMenu.settings.HEIGHT*0.7,'Rectangle', None, gameMenu.big_text_size,
int(gameMenu.curve*1.5))
leaderboard_text = new Text(gameMenu.text_height, "Graphics/Fonts/Orbitron-Bold.ttf", (240,240,240), None,
None, None, gameMenu.base_text_size*3)
leaderboard2 = new Button((27,31,35), (27,31,35), gameMenu.settings.WIDTH/2*1.78,
gameMenu.settings.HEIGHT/2*1.06, "Graphics/Fonts/Orbitron-Regular.ttf", (27,31,35), (27,31,35),
gameMenu.settings.WIDTH*0.18, gameMenu.settings.HEIGHT*0.6,'Rectangle', None, gameMenu.big_text_size,
int(gameMenu.curve*1.5))
value_board = new Text(gameMenu.text_height, "Graphics/Fonts/Orbitron-Medium.ttf", (240,240,240), None,
None, None, gameMenu.base_text_size*3)
game_finished = False
except
  logger.error("Couldn't correctly initialize pygame.")
  close()
endtry
endprocedure

public procedure initialize_network()
try
  // Setup Network
  network = new Network(SERVER, PORT)
  network.connect()
  // Get Public Key
  public_key = new unserialize(network.receive(ENCRYPTION_DATA_SIZE))
  rsa_encrypt = new RSA_Encryption(public_key)
  logger.info(f"Received Public Key: {public_key}")
  // Setup and send AES Encryption
  aes_key = new AES_Keys(BITS)
  key = new aes_key.export_key()
  dict_to_send_to_server = {'aes_key':key,'username':username}
  encrypted_key_dict = new serialize(rsa_encrypt.encrypt(dict_to_send_to_server))
  network.send(encrypted_key_dict)
  logger.info(f"Sending AES KEY: {key}")
  // Receive Player
  encryption = new AES_Encryption(key)
  data = new encryption.decrypt(unserialize(network.receive(ENCRYPTION_DATA_SIZE)))
  player_info = data['player_data']['player']
  initialize_player(player_info)
  logger.info(f"Received player dict: {player_info}")
except
  logger.error("Error. Failed to connect to that Server IP Address.")
  close()
endtry
endprocedure

public procedure initialize_player(player_info)
player_x = player_info['x']
player_y = player_info['y']
player_image = player_info['image']
username = player_info['username']
id = player_info['id']
status = player_info['status']
health = player_info['health']

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

kill_count = player_info['kill_count']
killed_by = player_info['killed_by']
player = new Player([player_x, player_y], player_image, [level.visible_sprites], level.obstacle_sprites, username, id,
health, create_attack, destroy_attack, create_bullet, create_magic, movement, offense, magic, status)
endprocedure

public procedure update_players(player_dict, dictionary_copy)
  // Update existing player instances and remove players that are not in player_dict
  try
    ////// Player Sounds //////
    for player_data, old_player_data in zip(player_dict.values(), dictionary_copy.values())
      player_info = player_data['player']
      old_player_info = old_player_data['player']
      if player_info['id'] != player.id then
        ////// Player Got Hit //////
        if player_info['health'] < old_player_info['health'] then
          if sound == 1 then
            damage_sound.play()
          endif
        endif
        ////// Player dies //////
        if player_info['status'] == "dead" AND old_player_info['status'] == "alive" then
          if sound == 1 then
            death_sound.play()
          endif
        endif
        ////// Weapons //////
        weapon_ids = [weapon['player_id'] for weapon in player_data['weapon']]
        weapon_type = [weapon['type'] for weapon in player_data['weapon']]
        old_weapon_ids = [old_weapon['player_id'] for old_weapon in old_player_data['weapon']]
        new_weapon_ids = new set(weapon_ids) - set(old_weapon_ids)
        if new_weapon_ids then
          for new_weapon_id in new_weapon_ids
            weapon_index = new_weapon_ids.index(new_weapon_id)
            animation_type = weapon_type[weapon_index]
            if sound == 1 then
              if animation_type == "melee" then
                sword_sound.play()
              endif
            endif
            next new_weapon_id
          endif
        ////// Bullets //////
        bullet_ids = [bullet['id'] for bullet in player_data['bullets']]
        old_bullet_ids = [old_bullet['id'] for old_bullet in old_player_data['bullets']]
        new_bullet_ids = new set(bullet_ids) - set(old_bullet_ids)
        if new_bullet_ids then
          for new_bullet_id in new_bullet_ids
            if sound == 1 then
              sword_sound.play()
            endif
            next new_bullet_id
          endif
        ////// Fire Animation //////
        magic_ids = [magic['player_id'] for magic in player_data['magic']]
        magic_type = [magic['type'] for magic in player_data['magic']]

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

old_magic_ids = [old_magic['player_id'] for old_magic in old_player_data['magic']]
new_magic_ids = new set(magic_ids) - set(old_magic_ids)
if new_magic_ids then
  for new_magic_id in new_magic_ids
    magic_index = new magic_ids.index(new_magic_id)
    animation_type = magic_type[magic_index]
    if sound == 1 then
      if animation_type == "flame" then
        fire_sound.play()
      endif
      if animation_type == "aura" then
        heal_sound.play()
      endif
    endif
    next new_magic_id
  endif
endif
next player_data, old_player_data
endprocedure
  
```

```

////// Players //////
existing_player_ids = [player.id for player in players]
players_to_remove = []
for player_data in player_dict.values()
  player_info = player_data['player']
  player_id = player_info['id']
  if player_id == player.id AND kill_count < player_info['kill_count'] then
    player.exp += 1000
    kill_count = player_info['kill_count']
    player.kill_count = player_info['kill_count']
  endif
  if player_id in existing_player_ids then
    // Update existing players
    for player in players
      if player.id == player_id then
        player.kill_count = player_info['kill_count']
        player.rect.x = player_info['x']
        player.rect.y = player_info['y']
        player.health = player_info['health']
        player.status_alive = player_info['status']
        try player.image = new pygame.image.load(player_info['image']).convert_alpha()
        except player.image = new pygame.image.load(f"../{player_info['image']}").convert_alpha()
      else
        endtry
      endif
    next player
    //logger.debug(f"Creating new player instance with ID: {player_id}")
    new_player = Player(
      [player_info['x'], player_info['y']],
      player_info['image'],
      [level.visible_sprites, level.player_sprites],
      level.obstacle_sprites,
      player_info['username'],
      player_id,
      player_info['health'])
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

)
players.append(new_player)
endif
next player_data
// Remove players not found in player_dict
for player in players
  if player.id NOT in player_dict then
    //logger.debug(f"Removing player instance with id: {player.id}")
    players_to_remove.append(player)
  endif
next player
for player in players_to_remove
  players.remove(player)
  level.visible_sprites.remove(player)
  player.kill()
next player
////// Weapons ///////
// Update Weapons
weapons_to_remove = []
for weapon in weapons
  weapon_info = player_data['weapon']
  weapon_dict_ids = [weapon_dict['id'] for weapon_dict in weapon_info]
  weapon_dict_x = [weapon_dict['x'] for weapon_dict in weapon_info]
  weapon_dict_y = [weapon_dict['y'] for weapon_dict in weapon_info]
  weapon_dict_image = [weapon_dict['image'] for weapon_dict in weapon_info]
  if weapon.id in weapon_dict_ids then
    weapon_index = new_weapon_dict_ids.index(weapon.id)
    weapon.rect.x = weapon_dict_x[weapon_index]
    weapon.rect.y = weapon_dict_y[weapon_index]
    try weapon.image = new pygame.image.load(weapon_dict_image[weapon_index]).convert_alpha()
    except weapon.image = new pygame.image.load(f"../{weapon_dict_image[weapon_index]}").convert_alpha()
  else
    endtry
    //logger.debug(f"Removing weapon instance with id: {weapon.id}")
    weapons_to_remove.append(weapon)
  endif
next weapon
// Delete weapons.
for weapon in weapons_to_remove
  weapons.remove(weapon)
  level.visible_sprites.remove(weapon)
  weapon.kill()
next weapon
// Create weapons
for player_data in player_dict.values()
  weapon_info = player_data['weapon']
  weapon_ids = [weapon.id for weapon in weapons]
  weapon_dict_ids = [weapon_dict['id'] for weapon_dict in weapon_info]
  for weapon_id, weapon_data in zip(weapon_dict_ids, weapon_info)
    if weapon_id NOT in weapon_ids then
      //logger.debug(f"Creating a new weapon with the ID: {weapon_data['id']}")
      new_weapon = Weapon([level.visible_sprites, level.attack_sprites],
        weapon_data['x'],
        weapon_data['y'],
        weapon_data['id'],
        weapon_data['image'],

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

      weapon_data['player_id'],
      weapon_data['damage'])
    weapons.append(new_weapon)
  endif
next weapon_id, weapon_data
next player_data
///// Bullets /////
// Update Bullets
bullets_to_remove = []
for bullet in bullets
  bullet_info = player_data['bullets']
  bullet_dict_ids = [bullet_dict['id'] for bullet_dict in bullet_info]
  bullet_dict_x = [bullet_dict['x'] for bullet_dict in bullet_info]
  bullet_dict_y = [bullet_dict['y'] for bullet_dict in bullet_info]
  bullet_dict_image = [bullet_dict['image'] for bullet_dict in bullet_info]
  if bullet.id in bullet_dict_ids then
    bullet_index = new_bullet_dict_ids.index(bullet.id)
    bullet.rect.x = bullet_dict_x[bullet_index]
    bullet.rect.y = bullet_dict_y[bullet_index]
    try bullet.image = new pygame.image.load(bullet_dict_image[bullet_index]).convert_alpha()
    except bullet.image = new pygame.image.load(f"../{bullet_dict_image[bullet_index]}").convert_alpha()
  else
    endtry
    //logger.debug(f"Removing bullet instance with id: {bullet.id}")
    bullets_to_remove.append(bullet)
  endif
next bullet
// Delete Bullets
for bullet in bullets_to_remove
  bullets.remove(bullet)
  level.visible_sprites.remove(bullet)
  bullet.kill()
next bullet
// Create Bullets
for player_data in player_dict.values()
  bullet_info = player_data['bullets']
  bullet_ids = [bullet.id for bullet in bullets]
  bullet_dict_ids = [bullet_dict['id'] for bullet_dict in bullet_info]
  for bullet_id, bullet_data in zip(bullet_dict_ids, bullet_info)
    if bullet_id NOT in bullet_ids then
      //logger.debug(f"Creating a new weapon with the ID: {bullet_data['id']}")
      new_bullet = Weapon([level.visible_sprites, level.attack_sprites],
        bullet_data['x'],
        bullet_data['y'],
        bullet_data['id'],
        bullet_data['image'],
        bullet_data['player_id'],
        bullet_data['damage'],
        "bullet_copy")
      bullets.append(new_bullet)
    endif
  next bullet_id, bullet_data
next player_data
///// Magic /////
// Update Magic Animation
magic_to_remove = []

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

for magic in magic_animations
  magic_info = player_data['magic']
  magic_dict_ids = [magic_dict['id'] for magic_dict in magic_info]
  magic_dict_x = [magic_dict['x'] for magic_dict in magic_info]
  magic_dict_y = [magic_dict['y'] for magic_dict in magic_info]
  magic_dict_image = [magic_dict['image'] for magic_dict in magic_info]
  if magic.id in magic_dict_ids then
    magic_index = new magic_dict_ids.index(magic.id)
    magic.rect.x = magic_dict_x[magic_index]
    magic.rect.y = magic_dict_y[magic_index]
    magic.full_path = magic_dict_image[magic_index]
    try magic.image = new pygame.image.load(magic.full_path).convert_alpha()
    except magic.image = new pygame.image.load(f"../{magic.full_path}").convert_alpha()
  else
    endtry
    //logger.debug(f"Removing magic instance with id: {magic.id}")
    magic_to_remove.append(magic)
  endif
next magic
// Delete Magic
for magic in magic_to_remove
  magic_animations.remove(magic)
  level.visible_sprites.remove(magic)
  magic.kill()
next magic
// Create Magic Animation
for player_data in player_dict.values()
  magic_info = player_data['magic']
  magic_ids = [magic.id for magic in magic_animations]
  magic_dict_ids = [magic_dict['id'] for magic_dict in magic_info]
  for magic_id, magic_data in zip(magic_dict_ids, magic_info)
    if magic_id NOT in magic_ids then
      //logger.debug(f"Creating a new Magic Animation with the ID: {magic_data['id']}")

      if magic_data['type'] == "flame" then
        new_magic_animation = Weapon([level.visible_sprites, level.attack_sprites],
          magic_data['x'],
          magic_data['y'],
          magic_data['id'],
          magic_data['image'],
          magic_data['player_id'],
          magic_data['damage'],
          "magic_copy")
      else
        new_magic_animation = Weapon([level.visible_sprites],
          magic_data['x'],
          magic_data['y'],
          magic_data['id'],
          magic_data['image'],
          magic_data['player_id'],
          None,
          "magic_copy")
      endif
      magic_animations.append(new_magic_animation)
    endif
  next magic_id, magic_data
next player_data
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

except
  logger.error("Player disconnected.")
  close()
endtry
function players_number()
  player_count = 0
  player_list = [player.status_alive for player in players]
  for status in player_list
    if status == "alive" then
      player_count += 1
    endif
  next status
  return player_count
endfunction

public procedure redraw_window(all_players_dict, dictionary_copy)
  try
    update_players(all_players_dict, dictionary_copy) // could be this
    level.run(player, dt)
    //player_attack_logic() # could be this//
    player_attack_collisions()
    ui.player_count = new players_number()
    ui.player_kill_count = kill_count
    ui.display(player)
    one_player_remaining()
    debug(f"Position: {{player.rect.x}, {player.rect.y}}", WIDTH/2, 20)
    if dt != new 0 then debug(f"FPS then {int(1/dt)}", WIDTH/2, 50)
    endif
    debug(f"{player.direction}", WIDTH/2, 80)
    ui.draw_menu()
    if ui.draw_ui then player.paused = True
    else player.paused = False
    endif
    pygame.display.update()
    // clock.tick(5)
  except
    logger.error("Player has left the game.")
    close()
  endtry
endprocedure

public procedure create_attack(weapon_type= new "melee")
  //logger.info("Create attack")
  try
    if sound == 1 AND weapon_type == "melee" then
      sword_sound.play()
    endif
    current_attack = new Melee(player, [level.visible_sprites], player.id)
  except
    current_attack = None
    //logger.error("Failed to render attack image.")
  endtry
endprocedure

public procedure create_bullet()

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
//logger.info("Bullet!")  
if sound == 1 then  
    sword_sound.play()  
endif  
bullet = new Bullets(player, [level.visible_sprites], level.obstacle_sprites, player.id, level.player_sprites)  
endprocedure  
  
public procedure destroy_attack()  
    //logger.info("destroy attack.")  
try  
    if current_attack then  
        current_attack.kill()  
    endif  
    current_attack = None  
except  
    current_attack = None  
    //logger.error("Failed to destroy attack.")  
endtry  
endprocedure  
  
public procedure create_magic(style, strength, cost)  
if style == 'heal' then  
    if sound == 1 AND player.energy > cost then  
        heal_sound.play()  
    endif  
    magic_player.heal(player, strength, cost, [level.visible_sprites], player.id)  
endif  
if style == 'flame' then  
    if sound == 1 AND player.energy > cost then  
        fire_sound.play()  
    endif  
    magic_player.flame(player, strength, cost, [level.visible_sprites], player.id)  
endif  
endprocedure  
  
public procedure close()  
try  
    network.close()  
except  
    logger.exception("No server-client connection.")  
endtry  
background_music.stop()  
running = False  
player = None  
players = None  
gameMenu.restart_menu()  
run()  
endprocedure  
  
public procedure quit()  
    running = False  
    pygame.quit()  
    sys.exit()  
endprocedure
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

function return_weapon()
  weapon = []
  for sprite in level.visible_sprites
    if sprite.sprite_type == "weapon" then
      weapon_dict = {"x":sprite.rect.x,
                     "y":sprite.rect.y,
                     "image":sprite.full_path,
                     "damage":sprite.damage,
                     "type":sprite.type,
                     "id":sprite.id,
                     "player_id":player.id}
      weapon.append(weapon_dict)
    endif
  next sprite
  return weapon
endfunction

function return_bullets()
  bullets = []
  for sprite in level.visible_sprites
    if sprite.sprite_type == "bullet" then
      bullet_dict = {"x":sprite.rect.x,
                     "y":sprite.rect.y,
                     "image":sprite.full_path,
                     "damage":sprite.damage,
                     "id":sprite.id,
                     "player_id":player.id}
      bullets.append(bullet_dict)
    endif
  next sprite
  return bullets
endfunction

function return_magic_data()
  magic = []
  for sprite in level.visible_sprites
    if sprite.sprite_type == "magic" then
      magic_dict = {"x":sprite.rect.x,
                    "y":sprite.rect.y,
                    "image":sprite.return_image(),
                    "type":sprite.return_type(),
                    "damage":sprite.return_strength(),
                    "id":sprite.id,
                    "player_id":player.id}
      magic.append(magic_dict)
    endif
  next sprite
  return magic
endfunction

public procedure player_attack_collisions()
  // Removes attack sprite id that are no longer in the game
  sprite_ids = [sprite.id for sprite in level.visible_sprites]
  for sprite_id in attacked_sprites_ids
    if sprite_id NOT in sprite_ids then
      attacked_sprites_ids.remove(sprite_id)
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

  endif
next sprite_id
if level.attack_sprites then
  for attack_sprite in level.attack_sprites
    if "copy" in attack_sprite.sprite_type then
      // Checks if the attack is not from our player
      if attack_sprite.player_id != player.id then
        player_collisions = new pygame.sprite.spritecollide(attack_sprite,level.player_sprites,False)
        if player_collisions then
          for player in player_collisions
            if attack_sprite.damaged_player != True AND attack_sprite.id NOT in attacked_sprites_ids then
              if player.id == player.id AND player.health > 0 then
                attacked_sprites_ids.append(attack_sprite.id)
                attack_sprite.damaged_player = True
                old_player_health = player.health
                if sound == 1 then
                  damage_sound.play()
                endif
                player.damage_player(attack_sprite.damage)
                if player.health == 0 AND old_player_health > 0 AND killed_by == "" then
                  if sound == 1 then
                    death_sound.play()
                  endif
                  killed_by = attack_sprite.player_id
                endif
                if "bullet" in attack_sprite.sprite_type then
                  attack_sprite.kill()
                endif
              endif
            endif
          next player
        endif
      endif
    endif
  next attack_sprite
endif
endprocedure

```

```

public procedure main()
  dictionary_copy = {}
  background_music.play(-1)
  if music == 1 then
    background_music.set_volume(volume/100)
  else
    background_music.set_volume(0)
  endif
  try
    previous_time = new time.time()
  while running
    current_time = new time.time()
    dt = current_time - previous_time
    previous_time = current_time
    handle_events()
    try
      all_players_dict = new send_player_data()
      //logger.debug(f"Sending player data: {all_players_dict}")
    
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

//logger.debug(f"Received players dictionary: {all_players_dict}")
redraw_window(all_players_dict, dictionary_copy)
dictionary_copy = all_players_dict
except Exception as e
  logger.error(f"Failed to send/receive player data: {e}")
  close()
endtry
endwhile
except Exception as e
  logger.error(f"Failed to run main loop: {e}")
  close()
endtry
endprocedure

public procedure one_player_remaining()
if players_number() = new= new 1 OR game_finished then
  //// Player has won.
  game_finished = True
  //// Animation
  game_over_board.draw((27,31,35), None, None, True, None, None, None, "draw")
  game_over.draw("draw", "Game Over!",(gameMenu.settings.WIDTH/2), (gameMenu.settings.HEIGHT/2*0.4))
  //// Display kill counts for all players
  leaderboard.draw((27,31,35))
  leaderboard2.draw((240,240,240))
  leaderboard_text.draw("draw", "Leaderboard",(gameMenu.settings.WIDTH/2*1.78),
(gameMenu.settings.HEIGHT/2*0.375))
  leaderboard_text = new "\n".join([f"{player.username} {player.kill_count}" for player in sorted(players, key=
newlambda x.kill_count, reverse= newTrue)])
  next player
  leaderboard_list = new leaderboard_text.split('\n')
  for index, leaderboard_index in enumerate(leaderboard_list)
    spacing_index = 0.52 + 0.1*index
    value_board.draw("draw",leaderboard_index,(gameMenu.settings.WIDTH/2*1.78),
(gameMenu.settings.HEIGHT/2*spacing_index))
  next index, leaderboard_index
  //// Reset key player data
  //// Log out after a 30 second.
else
  game_over.draw("undraw", "Game Over!",(gameMenu.settings.WIDTH/2), (gameMenu.settings.HEIGHT/2*0.4))
endif
endprocedure

public procedure handle_events()
for event in pygame.event.get()
  if event.type == pygame.QUIT then
    quit()
  endif
  elseif event.type == pygame.KEYDOWN AND event.key == pygame.K_ESCAPE then
    ui.draw_ui = NOT ui.draw_ui
  endif
next event
endprocedure

function send_player_data()
  player_dict = {
    'x': player.rect.x,

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

'y': player.rect.y,
'image': player.get_image_name(),
'username': username,
'status': player.return_alive_status(),
'health': player.health,
'id': id,
'kill_count': kill_count,
'killed_by': killed_by
}
player_total_dict = {
  'player': player_dict,
  'weapon': return_weapon(),
  'bullets': return_bullets(),
  'magic': return_magic_data()
}
player_encrypted_dict = new serialize(encryption.encrypt(player_total_dict))
network.send(player_encrypted_dict)
all_players_dict = new encryption.decrypt(unserialize(network.receive(DATA_SIZE)))
return all_players_dict
endfunction

public procedure run()
  join_game = False
  gameMenu.run()
  user_dict = new gameMenu.start_game()
  initialize_client(user_dict)
  loading_zone()
  main()
endprocedure

public procedure loading_zone()
  procedure redraw_window()
    screen.fill((27,31,35))
    join_btn.draw((240,240,240),join_the_game)
    players.draw("draw", f'{connections} {"player" if connections= new = new 1 else "players"} {"is" if connections= new = new 1 else "are"} connected.', (WIDTH / 2), (HEIGHT / 2) * 0.4)
  endif
  join_hover = new join_btn.is_hovered()
  join_click = new join_btn.is_clicking()
  if join_hover then
    if NOT join_click then
      mouse.mode = 1
    else
      mouse.mode = 2
    else
      mouse.mode = 0
    endif
    mouse.draw()
    pygame.display.update()
  endprocedure

  procedure join_the_game()
    join_game = True
  endprocedure

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

// Receive number of players
running = True
mouse = new Mouse("Graphics/MainMenu/Mouse/mouse1.png", "Graphics/MainMenu/Mouse/mouse2.png",
"Graphics/MainMenu/Mouse/mouse3.png")
width_ratio = WIDTH / 1920
height_ratio = HEIGHT / 1080
// Variables
players = new Text(gameMenu.text_height, "Graphics/Fonts/Orbitron-Regular.ttf", (240, 240, 240), None, None, None,
int(gameMenu.base_text_size * 3))
join_btn = new Button((39, 174, 96), (27,31,35), (gameMenu.settings.WIDTH/2), (gameMenu.settings.HEIGHT/2)*1.4-
gameMenu.button_padding, "Graphics/Fonts/Orbitron-Medium.ttf", (27,31,35), (39, 174, 96),
gameMenu.base_button_width, gameMenu.base_button_height,'Start','Rectangle', None,
int(gameMenu.big_text_size/1.3), gameMenu.curve)
connections = 0
while running
  for event in pygame.event.get()
    if event.type == pygame.QUIT then
      quit()
    endif
    if event.type == pygame.KEYDOWN then
      if event.key == pygame.K_ESCAPE then
        quit()
      endif
    endif
  next event
  try
    connected_players = new encryption.decrypt(unserialize(network.receive(DATA_SIZE)))
    connections = connected_players['connections']
    started = connected_players['started']
    if join_game AND connections < 2 then
      join_game = False
    endif
    if started then
      join_game = started
    endif
    game_dict = {"ready":join_game}
    start_the_game = new serialize(encryption.encrypt(game_dict))
    network.send(start_the_game)
    if join_game OR started then
      join_game = True
      running = False
    endif
  except
    running = False
    logger.error("Something failed.")
    close()
  endtry
  redraw_window()
endwhile
endprocedure

if __name__ == "__main__":
  try
    client = new Client()
    client.run()
  except

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
logger.error("Couldn't run main menu OR client.")  
endtry  
endif  
  
Debug:  
from Scripts.logger import *  
try  
    import pygame  
  
except  
    logger.critical("You do NOT have all the modules installed. Please install Pygame.")  
endtry  
font = new pygame.font.Font(None,30)  
logger.debug("RealDL Debug Code.")  
procedure debug(info,x= new 10,y= new 10)  
    display_surface = new pygame.display.get_surface()  
    debug_surf = new font.render(str(info),True,'White')  
    debug_rect = new debug_surf.get_rect(center = new (x,y))  
    pygame.draw.rect(display_surface,'Black',debug_rect)  
    display_surface.blit(debug_surf,debug_rect)  
endprocedure
```

Encryption:

```
from Scripts.logger import *  
try  
    import rsa, pickle, os  
    from Crypto.Cipher import AES  
    from Crypto.Util.Padding import pad, unpad  
    from Crypto.Random import get_random_bytes  
  
except  
    logger.critical("You do NOT have all the modules installed. Please install RSA AND Pycryptodome.")  
endtry  
logger.info('RealDL Encryption Module : RSA')  
class RSA_Keys  
    private public_key,  
    private private_key  
    private public_key  
    private bits  
  
    public procedure new(bits)  
        bits = bits  
        key_dir = new "../Keys" if os.path.exists("../Keys") else "Keys"  
        endif  
        public_key_path = new os.path.join(key_dir, "public.pem")  
        private_key_path = new os.path.join(key_dir, "private.pem")  
        try  
            with open(public_key_path, "rb") as f:  
                public_key = new rsa.PublicKey.load_pkcs1(f.read())  
            with open(private_key_path, "rb") as f:  
                private_key = new rsa.PrivateKey.load_pkcs1(f.read())  
        except FileNotFoundError  
            public_key, private_key = new rsa.newkeys(bits)  
            with open(public_key_path, "wb") as f:
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
f.writeLine(public_key.save_pkcs1("PEM"))
with open(private_key_path, "wb") as f:
    f.writeLine(private_key.save_pkcs1("PEM"))
endtry
endprocedure

function export_keys()
    return public_key, private_key
endfunction

class RSA_Encryption
    private public_key

    public procedure new(public_key)
        try
            public_key = public_key
        except ValueError as e
            logging.error(f"Error: {e}")
        endtry
    endprocedure

    function encrypt(message)
        try
            return rsa.encrypt(serialise(message), public_key)
        except ValueError as e
            logging.error(f"Error: {e}")
        endtry
    endfunction

    function decrypt(encrypted_message, private_key)
        try
            encoded_message = new rsa.decrypt(encrypted_message, private_key)
            return unserialise(encoded_message)
        except ValueError as e
            logging.error(f"Error: {e}")
        endtry
    endfunction

    function serialise(data)
        try
            return pickle.dumps(data)
        except ValueError as e
            logging.error(f"Error: {e}")
        endtry
    endfunction

    function unserialise(data)
        try
            return pickle.loads(data)
        except ValueError as e
            logging.error(f"Error: {e}")
        endtry
    endfunction

class AES_Keys
    private key
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

private bytes

```
public procedure new(bits)
    bytes = bits DIV 8
    key = new get_random_bytes(bytes)
endprocedure

function export_key()
    return key
endfunction

class AES_Encryption
    private cipher
    private key

    public procedure new(key)
        try
            key = key
            cipher = new AES.new(key, AES.MODE_ECB)
        except ValueError as e
            logging.error(f"Error: {e}")
        endtry
    endprocedure

    function encrypt(message)
        try
            message_bytes = new serialise(message)
            padded_message = new pad(message_bytes, AES.block_size)
            return cipher.encrypt(padded_message)
        except ValueError as e
            logging.error(f"Error: {e}")
        endtry
    endfunction

    function decrypt(encrypted_message)
        try
            decrypted_padded_message = new cipher.decrypt(encrypted_message)
            decrypted_unpadded_bytes_message = new unpad(decrypted_padded_message, AES.block_size)
            return unserialise(decrypted_unpadded_bytes_message)
        except ValueError as e
            logging.error(f"Error: {e}")
        endtry
    endfunction

    function serialise(data)
        try
            return pickle.dumps(data)
        except ValueError as e
            logging.error(f"Error: {e}")
        endtry
    endfunction

    function unserialise(data)
        try
            return pickle.loads(data)
        except ValueError as e
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
logging.error(f"Error: {e}")
endtry
endfunction

Functions:
from Scripts.logger import *
try
    import pygame, webbrowser

except
    logger.critical("You do NOT have all the modules installed. Please install Pygame.")
endtry
logger.info("Pygame Functions 0.0.1 (Python 3.11.0)\nI am NOT affiliated with pygame.
https://github.com/TheRealDL1/pygame\_functions")

class Images inherits object
    private rect
    private load_image
    private screen
    private image

    public procedure new(image)
        image = image
        screen = new pygame.display.get_surface()
        load_image = new load_image_from_file()
        rect = new load_image.get_rect()
    endprocedure

    function load_image_from_file()
        try
            return pygame.image.load(image).convert_alpha()
        except
            return pygame.image.load(f"../{image}").convert_alpha()
        endtry
    endfunction

    public procedure display_icon()
        pygame.display.set_icon(load_image)
    endprocedure

    public procedure draw(x= new 0, y= new 0)
        screen.blit(load_image, (x,y))
    endprocedure

    public procedure resize(width, height)
        load_image = new pygame.transform.scale(load_image, (width, height))
    endprocedure

class Mouse inherits object
    private mouse_mode3
    private mouse_mode2
    private mouse_mode1
    private mode
    private screen
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
public procedure new(mouse_image1, mouse_image2, mouse_image3)
    pygame.mouse.set_visible(False)
    screen = new pygame.display.get_surface()
    mode = 0
    mouse_mode1 = new Images(mouse_image1)
    mouse_mode2 = new Images(mouse_image2)
    mouse_mode3 = new Images(mouse_image3)
endprocedure

public procedure draw()
    // Get the current mouse position
    mouse_x, mouse_y = new pygame.mouse.get_pos()
    // Draw a green square at the mouse cursor position
    if mode == 0 then
        mouse_mode1.draw(mouse_x, mouse_y)
    endif
    elseif mode == 1 then
        mouse_mode2.draw(mouse_x, mouse_y)
    else
        mouse_mode3.draw(mouse_x, mouse_y)
    endif
endprocedure

class WordButton inherits object
    private release
    private click
    private hover
    private large_font
    private base_font
    private largefont
    private basefont
    private largeSize
    private textSize
    private text
    private y
    private x
    private color2
    private color

    public procedure new(x, y, text, color1, color2, basefont, largefont, textSize= new 30,)
        //Sets the values for button/
        color = color1
        color2 = color2
        x = x
        y = y
        text = text
        textSize = textSize
        largeSize = new round(textSize * 1.25)
        basefont = basefont
        largefont = largefont
        try
            base_font = new pygame.font.Font(basefont, textSize)
            large_font = new pygame.font.Font(largefont, largeSize)
        except
            base_font = new pygame.font.Font(f"../{basefont}", textSize)
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

large_font = new pygame.font.Font(f"../{largefont}", largeSize)
endtry
hover = False
click = False
release = False
endprocedure

public procedure displayText(win, newText= new None, action= new None, link= new None)
if newText != None then
  text = newText
endif
mouse = new pygame.mouse.get_pos()
click = new pygame.mouse.get_pressed()
// Render text with both fonts
text = new large_font.render(text, 1, color2)
text2 = new base_font.render(text, 1, color)
// Check if mouse is over the text
if x + text.get_width()/2 > mouse[0] > x - text.get_width()/2 AND y + text.get_height()/2 > mouse[1] > y - text.get_height()/2 then
  hover = True
  // Render text with larger font and lighter color
  text = new large_font.render(text, 1, color2)
  // Center text vertically as well as horizontally
  win.blit(text, (x - text.get_width() / 2, y - text.get_height() / 2))
  // Check for mouse click and run action if specified
  if click[0] == 1 then
    click = True
  endif
  if click == True AND NOT click[0] then
    release = True
  endif
  if release then
    if action != None then
      action()
    endif
    if link != None then
      webbrowser.open(link)
    endif
    release = False
    click = False
  else
    endif
    hover = False
    click = False
    release = False
    // Render text with base font and color
    text2 = new base_font.render(text, 1, color)
    // Center text vertically as well as horizontally
    win.blit(text2, (x - text2.get_width() / 2, y - text2.get_height() / 2))
  endif
endprocedure
  
```

```

class Animation inherits object
  private monopoly_rect
  private screen_center_y
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

private screen_center_x
private scale_speed
private scale_max
private scale_min
private scale_direction
private image
private scale

public procedure new(image, scale, scale_direction, scale_min, scale_max, scale_speed, screen_y, screen_x)
  scale = scale //1.0
  image = image
  scale_direction = scale_direction //-1.35 // Start by zooming out
  scale_min = scale_min //0.9
  scale_max = scale_max //1.2
  scale_speed = scale_speed //0.01 // The rate at which the scale changes
  screen_center_x = screen_x
  screen_center_y = new screen_y //324//win.get_height() // 2
  monopoly_rect = new image.get_rect(center= new(screen_center_x, screen_center_y))
endprocedure

public procedure animation(win)
  scale += scale_direction * scale_speed
  if scale <= scale_min then
    if scale_direction < 0 then
      scale_direction = scale_direction * -1
    else
      scale_direction = scale_direction * 1// Start zooming in
    endif
  endif
  elseif scale >= scale_max then
    if scale_direction > 0 then
      scale_direction = scale_direction * -1
    else
      scale_direction = scale_direction * 1 // Start zooming out
    endif
  endif
  // Scale the image and get its rect
  scaled_monopoly = new pygame.transform.rotozoom(image, 0, scale)
  scaled_monopoly_rect = new scaled_monopoly.get_rect(center= new(screen_center_x, screen_center_y))
  // Blit the scaled image to the screen and update the display
  win.blit(scaled_monopoly, scaled_monopoly_rect)
endprocedure

class TextBox inherits object
  private show_cursor
  private click
  private hover
  private screen
  private thickness
  private curve
  private maxTextWidth
  private cursor
  private show_cursOR
  private last_update
  private color

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
private color_passive
private color_active
private y
private ogWidth
private x
private height
private width
private active
private base_font
private textfont
private textSize
private text

public procedure new(width, height, x, y, color1, color2, textfont, curve, thickness, maxTextWidth, text_size= new 50)
    text = ""
    textSize = text_size
    textfont = textfont
    try
        base_font = new pygame.font.Font(textfont, textSize)
    except
        base_font = new pygame.font.Font(f"../{textfont}", textSize)
    endtry
    active = False
    width = width
    height = height
    x = x - width / 2
    ogWidth = width
    y = y
    color_active = new pygame.Color(color1)
    color_passive = new pygame.Color(color2)
    color = color_passive
    last_update = 0 // time of last update
    show_cursOR = False // whether to show cursOR OR not
    cursor = 0
    maxTextWidth = maxTextWidth
    curve = curve
    thickness = thickness
    screen = new pygame.display.get_surface()
    hover = False
    click = False
endprocedure

public procedure draw()
    area = [x,y,width,height]
    pygame.draw.rect(screen, color, area, thickness, curve)
endprocedure

function is_hovered()
    mouse = new pygame.mouse.get_pos()
    return x + width > mouse[0] > x AND y + height > mouse[1] > y
endfunction

function is_clicking()
    mouse = new pygame.mouse.get_pos()
    click = new pygame.mouse.get_pressed()
    return x + width > mouse[0] > x AND y + height > mouse[1] > y AND click[0]
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

endfunction

```
public procedure checkTextBox()
mouse = new pygame.mouse.get_pos()
click = new pygame.mouse.get_pressed()
if x + width > mouse[0] > x AND y + height > mouse[1] > y then
    hover = True
    if click[0] then click = True
    endif
    if click AND NOT click[0] then click = False
    endif
    if active then active = False
    else active = True
else
    endif
    hover = False
    active = False
    click = False
endif
if active then
    color = color_active
else
    color = color_passive
endif
endprocedure

public procedure update(color= new (240,240,240),x_pos= new None, y_pos= new None)
middleOfX = new screen.get_width() DIV 2
if x_pos then x = x_pos
endif
if y_pos then y = y_pos
endif
if text then
    surface_area = new base_font.render(text, True, color)
    text_width = new surface_area.get_width() + 20
    if ogWidth > width then
        width = ogWidth
        x = middleOfX - width DIV 2
    endif
    if text_width > width then
        width = text_width
        x = middleOfX - width DIV 2
    endif
    elseif width > text_width then
        if ogWidth < width then
            width = text_width
            x = middleOfX - width DIV 2
        endif
        endif
    screen.blit(surface_area, (x + 5, y + (height DIV 2 - surface_area.get_height() DIV 2)))
else
    width = ogWidth
    x = middleOfX - width DIV 2
    surface_area = new base_font.render(text, True, color)
    screen.blit(surface_area, (x + 5, y + (height DIV 2 - surface_area.get_height() DIV 2)))
endif
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

time_since_last_update = new pygame.time.get_ticks() - last_update
if active AND time_since_last_update > 500 then
  show_cursor = NOT show_cursor
  last_update = new pygame.time.get_ticks()
endif
if show_cursor AND active == True then
  cursor_width = 2
  cursor_pos = new base_font.size(text[:cursor])[0] - 5
  text_to_show = text[:cursor] + '|' + text[cursor:]
  cursor_pos += new base_font.size(' ')[0] * 0.8
  surface_area = new base_font.render(text_to_show, True, color)
  cursor_area = new pygame.Surface((cursor_width, surface_area.get_height()))
  cursor_area.fill(color)
  screen.blit(cursor_area, (x + cursor_pos, y + (height DIV 2 - cursor_area.get_height() DIV 2)), (0, 0, cursor_width,
cursor_area.get_height()))
// else
//surface_area = new base_font.render(text, True, color)
//screen.blit(surface_area, (x + 5, y + (height // 2 - surface_area.get_height() // 2)))
endif
endprocedure

public procedure updateText(events, color= new (0,0,0), function= new None)
  surface_area = new base_font.render(text, True, color)
  text_width = new surface_area.get_width() + 20
  for event in events
    if event.type == pygame.MOUSEBUTTONDOWN then
      checkTextBox()
    endif
    if event.type == pygame.KEYDOWN then
      if active == True then
        if event.key == pygame.K_BACKSPACE then
          if cursor > 0 then
            text = text[:cursor-1] + text[cursor:]
            cursor -= 1
          endif
        endif
        elseif event.key == pygame.K_DELETE then
          text = text[:cursor] + text[cursor+1:]
        endif
        elseif event.key == pygame.K_RETURN then
          if function != None then
            function()
          endif
        endif
        elseif event.key == pygame.K_LEFT then
          if cursor > 0 then
            cursor -= 1
          endif
        endif
        elseif event.key == pygame.K_RIGHT then
          if cursor < text.length
            cursor += 1
          endif
        else
          endif
        if text_width < maxTextWidth AND event.key is NOT pygame.K_TAB then
          new_text = text[:cursor] + event.unicode + text[cursor:]
        endif
      endif
    endif
  endfor
endfunction
  
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
if new_text.length > len(text)
    text = new_text
    cursor += 1
endif
endif
endif
endif
next event
endprocedure

function return_text()
if text != "" then
    return text
else
    return None
endif
endfunction

class Text inherits object
    private opacity_color
    private opacity_color2
    private opacity_color1
    private opacityb
    private opacitya
    private time2
    private time
    private num_frames
    private animation_speed
    private animation_time
    private stop_start_animation
    private release
    private click
    private hover
    private base_font
    private screen
    private color
    private textSize
    private height
    private baseFont
    private y
    private x
    private text

    public procedure new(height, baseFont, color= new (0,0,0), x= new None, y= new None, text= new None, textSize= new
32)
        text = text
        x = x
        y = y
        baseFont = baseFont
        height = height
        textSize = textSize
        color = color
        screen = new pygame.display.get_surface()
        try
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

base_font = new pygame.font.Font(baseFont, textSize)
except
  base_font = new pygame.font.Font(f"../{baseFont}", textSize)
endtry
hover = False
click = False
release = False
// Animation
stop_start_animation = True
animation_time = 150
animation_speed = 13
num_frames = 60
time = 0
time2 = 0
// Opacity
opacitya = 0
opacityb = 255
opacity_color1 = 0
opacity_color2 = 0
opacity_color = 0
endprocedure

public procedure draw(mode= new"draw",new_text= newNone, new_x= newNone, new_y= newNone, function= new
None, link= newNone)
  if new_text then text = new_text
  endif
  if new_x then x = new_x
  endif
  if new_y then y = new_y
  endif
  if mode == "normal" then
    surface_area = new base_font.render(text, 1, color)
    screen.blit(surface_area, (x - surface_area.get_width() / 2, y - surface_area.get_height() / 2))
  endif
  if mode == "draw" then
    stop_start_animation = False
    time = new pygame.time.get_ticks()
    step = new (time - time2) / (animation_time / animation_speed)
    opacity_color1 = new change_opacity(opacity_color2, opacityb, step)
    opacity_color = opacity_color1
  endif
  if mode == "undraw" then
    time2 = new pygame.time.get_ticks()
    step = new (time2 - time) / (animation_time / animation_speed)
    opacity_color2 = new change_opacity(opacity_color1, opacitya, step)
    opacity_color = opacity_color2
  endif
  if NOT stop_start_animation then
    surface_area = new base_font.render(text, 1, color)
    surface_area.set_alpha(opacity_color)
    screen.blit(surface_area, (x - surface_area.get_width() / 2, y - surface_area.get_height() / 2))
  endif
  if is_hovered() then hover = new True
  endif
  if is_clicking() then click = new True
  endif

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
if click AND NOT is_clicking() then release = new True
endif
if release then
    if function then function()
    endif
    if link then webbrowser.open(link)
    endif
    release = False
    click = False
endif
if NOT is_hovered then
    hover = False
    click = False
    release = False
endif
endprocedure

function change_opacity(original_opacity, new_opacity, step_multiplier= new 0)
    new_opacity_return = None
    difference = new_opacity - original_opacity
    endif
    opacity_step = difference / num_frames
    endif
    if original_opacity > new_opacity then
        if int(original_opacity + opacity_step*step_multiplier) < new_opacity then
            new_opacity_return = new_opacity
        else
            new_opacity_return = new int(original_opacity + opacity_step*step_multiplier)
        endif
        if int(original_opacity + opacity_step*step_multiplier) > new_opacity then
            new_opacity_return = new_opacity
        else
            new_opacity_return = new int(original_opacity + opacity_step*step_multiplier)
        endif
    endif
    return new_opacity_return
endfunction

function is_hovered()
    mouse_pos = new pygame.mouse.get_pos()
    text_rect = new base_font.render(text, 1, color).get_rect()
    text_rect.center = new (x, y)
    return text_rect.collidepoint(mouse_pos)
endfunction

function is_clicking()
    mouse_pos = new pygame.mouse.get_pos()
    mouse_buttons = new pygame.mouse.get_pressed()
    text_rect = new base_font.render(text, 1, color).get_rect()
    text_rect.center = new (x, y)
    return text_rect.collidepoint(mouse_pos) AND mouse_buttons[0] = new= new 1
endfunction

class Button inherits object
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
private base_font
private basefont
private opacity2
private opacity1
private opacityb
private opacitya
private button_trans_color2
private text_trans_color2
private button_trans_color
private text_trans_color
private text_color2
private text_color
private color2
private color
private time2
private time
private num_frames
private animation_speed
private animation_time
private stop_start_animation
private release
private click
private hover
private curve
private textSize
private radius
private buttonType
private text
private image
private height
private width
private y
private x
private screen

//A class for all buttons//
public procedure new(color, color2, x, y, basefont, text_color= new (0,0,0), text_color2= new (0,0,0), width= new None,
height= new None, text= new "",buttonType= new 'Rectangle', radius= new None, textSize= new 30, curve= new 10, image=
new "")
    //Sets the values for buttton//
    screen = new pygame.display.get_surface()
    try
        x = x - width/2
        y = y - height/2
        width = width
        height = height
    except
        x = x
        y = y
    endif
    if image != "" then
        image = new Images(image)
        if height AND width then
            image.resize(width,height)
        endif
    endif
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
text = text
buttonType = buttonType
radius = radius
textSize = textSize
curve = curve
//Mouse clicking
hover = False
click = False
release = False
// Animation
stop_start_animation = True
animation_time = 150
animation_speed = 13
num_frames = 60
time = 0
time2 = 0
// Colors
color = color
color2 = color2
text_color = text_color
text_color2 = text_color2
text_trans_color = text_color
button_trans_color = color
text_trans_color2 = text_color
button_trans_color2 = color
// Opacity
opacitya = 255
opacityb = 180
opacity1 = 255
opacity2 = 180
// Fonts
basefont = basefont
try
    base_font = new pygame.font.Font(basefont, textSize)
except
    base_font = new pygame.font.Font(f"../{basefont}", textSize)
endtry
endprocedure

public procedure draw(outline= new None,action= new None, link= new None, colorChange= new True, newText= new
None, newX= new None, newY= new None, mode= new "normal")
//New X, Y and Text values//
if newText then text = newText
endif
if newX then x = newX
endif
if newY then y = newY
endif
//Draws the button. Variable for mouse detection//
mouse = new pygame.mouse.get_pos()
click = new pygame.mouse.get_pressed()
if buttonType == "Rectangle" then
    //If it is a rectangle it will draw it here//
    if mode == "normal" then
        if outline then
            //draws an outline//
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
    pygame.draw.rect(screen, outline, (x-2,y-2,width+4,height+4),0,curve)
endif
if x+width > mouse[0] > x AND y+height > mouse[1] > y AND colorChange then
    hover = True
    stop_start_animation = False
    time2 = new pygame.time.get_ticks()
    step = new (time2 - time) / (animation_time / animation_speed)
    //Draws a lighter version of the image//
    button_trans_color = new change_color(button_trans_color2, color2, step)
    pygame.draw.rect(screen, button_trans_color, (x,y,width,height), 0, curve)
if text != " then
    text_trans_color = new change_color(text_trans_color2, text_color2, step)
    text = new base_font.render(text, 1, text_trans_color)
    screen.blit(text, (x + (width/2 - text.get_width()/2), y + (height/2 - text.get_height()/2)))
endif
// Clicking the Button
if click[0] == 1 then
    click = True
endif
if click == True AND NOT click[0] then
    release = True
endif
if release then
    //If there is an action it is run//
    if action then
        action()
    endif
    if link then
        webbrowser.open(link)
    endif
    release = False
    click = False
else
endif
time = new pygame.time.get_ticks()
hover = False
click = False
release = False
step = new (time - time2) / (animation_time / animation_speed)
//A darker version of image when the player isn't hovering over//
if NOT stop_start_animation then
    button_trans_color2 = new change_color(button_trans_color, color, step)
    pygame.draw.rect(screen, button_trans_color2, (x,y,width,height),0,curve)
else
    pygame.draw.rect(screen, color, (x,y,width,height),0,curve)
endif
if text != " then
    //Text is blit here//
    if NOT stop_start_animation then
        text_trans_color2 = new change_color(text_trans_color, text_color, step)
        text = new base_font.render(text, 1, text_trans_color2)
    else
        text = new base_font.render(text, 1, text_color)
    endif
    screen.blit(text, (x + (width/2 - text.get_width()/2), y + (height/2 - text.get_height()/2)))
else
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
        endif
    endif
    if mode == "draw" then
        time2 = new pygame.time.get_ticks()
        step = new (time2 - time) / (animation_time / animation_speed)
        //Draws a lighter version of the image//
        button_trans_color = new change_color(button_trans_color2, color2, step)
        pygame.draw.rect(screen, button_trans_color, (x,y,width,height), 0, curve)
    endif
    if mode == "undraw" then
        time = new pygame.time.get_ticks()
        step = new (time - time2) / (animation_time / animation_speed)
        //A darker version of image when the player isn't hovering over//
        button_trans_color2 = new change_color(button_trans_color, color, step)
        pygame.draw.rect(screen, button_trans_color2, (x,y,width,height),0, curve)
    endif
    endif
endif
if buttonType == "Circle" then
    //If it is a circle it will draw it here//
    if outline then
        //draws an outline//
        pygame.draw.circle(screen, outline, (x, y),radius+2.5,0)
    endif
    //Trig for area//
    differenceInX = mouse[0] - x
    endif
    differenceInY = mouse[1] - y
    endif
    difference = new ( differenceInX**2 + differenceInY**2 )**0.5
endif
//Info on the circle//
if difference <= radius then
    hover = True
    stop_start_animation = False
    time2 = new pygame.time.get_ticks()
    step = new (time2 - time) / (animation_time / animation_speed)
    button_trans_color = new change_color(button_trans_color2, color2, step)
    //Draws a lighter version of the image//
    pygame.draw.circle(screen, button_trans_color, (x,y),radius,0)
    if text != " then
        //Text is blit here//
        text_trans_color = new change_color(text_trans_color2, text_color2, step)
        text = new base_font.render(text, 1, text_trans_color)
        screen.blit(text, (x - (text.get_width()/2), y - (text.get_height()/2)))
    endif
    // Clicking the Button
    if click[0] == 1 then
        click = True
    endif
    if click == True AND NOT click[0] then
        release = True
    endif
    if release then
        //If there is an action it is run//
        if action != None then
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
        action()  
    endif  
    if link != None then  
        webbrowser.open(link)  
    endif  
    release = False  
    click = False  
else  
endif  
time = new pygame.time.get_ticks()  
step = new (time - time2) / (animation_time / animation_speed)  
button_trans_color2 = new change_color(button_trans_color, color, step)  
hover = False  
click = False  
release = False  
//A darker version of image when the player isn't hovering over//  
pygame.draw.circle(screen, button_trans_color2, (x,y),radius,0)  
if text != " " then  
    //Text is blit here//  
    text_trans_color2 = new change_color(text_trans_color, text_color, step)  
    text = new base_font.render(text, 1, text_trans_color2)  
    screen.blit(text, (x - (text.get_width()/2), y - (text.get_height()/2)))  
endif  
endif  
endif  
if buttonType == "Image" then  
    if outline then  
        //draws an outline//  
        pygame.draw.rect(screen, outline, (x-3,y-3,width+6,height+6),0,curve)  
    endif  
    image.draw(x,y)  
    if x+width > mouse[0] > x AND y+height > mouse[1] > y then  
        time2 = new pygame.time.get_ticks()  
        step = new (time2 - time) / (animation_time / animation_speed)  
        opacity2 = new change_opacity(opacity1, opacityb, step)  
        image.load_image.set_alpha(opacity2)  
        hover = True  
        // Clicking the Button  
        if click[0] == 1 then  
            click = True  
        endif  
        if click == True AND NOT click[0] then  
            release = True  
        endif  
        if release then  
            //If there is an action it is run//  
            if action != None then  
                action()  
            endif  
            if link != None then  
                webbrowser.open(link)  
            endif  
            release = False  
            click = False  
        else  
            endif
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

time = new pygame.time.get_ticks()
hover = False
click = False
release = False
step = new (time - time2) / (animation_time / animation_speed)
opacity1 = new change_opacity(opacity2, opacitya, step)
image.load_image.set_alpha(opacity1)
endif
endif
endprocedure

function is_hovered()
  return hover
endfunction

function is_clicking()
  return click
endfunction

function change_color(original_color, new_color, step_multiplier= new 0)
  colors = []
  new_color_list = []
  for rgb1, rgb2 in zip(original_color, new_color)
    difference = rgb2 - rgb1
    endif
    colors.append(difference)
    endif
  next rgb1, rgb2
  color_step = [difference / num_frames for difference in colors]
  endif
  for og_color_rgb, new_color_rgb, step in zip(original_color, new_color, color_step)
    if og_color_rgb > new_color_rgb then
      if int(og_color_rgb + step_multiplier*step) < new_color_rgb then
        new_color_list.append(new_color_rgb)
      else
        new_color_list.append(int(og_color_rgb + step_multiplier*step))
      endif
    else
      if int(og_color_rgb + step_multiplier*step) > new_color_rgb then
        new_color_list.append(new_color_rgb)
      else
        new_color_list.append(int(og_color_rgb + step_multiplier*step))
      endif
    endif
  next og_color_rgb, new_color_rgb, step
  return tuple(new_color_list)
endfunction

function change_opacity(original_opacity, new_opacity, step_multiplier= new 0)
  new_opacity_return = None
  difference = new_opacity - original_opacity
  endif
  opacity_step = difference / num_frames
  endif
  if original_opacity > new_opacity then
    if int(original_opacity + opacity_step*step_multiplier) < new_opacity then

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

new_opacity_return = new_opacity
else
  new_opacity_return = new int(original_opacity + opacity_step*step_multiplier)
else
  endif
  if int(original_opacity + opacity_step*step_multiplier) > new_opacity then
    new_opacity_return = new_opacity
  else
    new_opacity_return = new int(original_opacity + opacity_step*step_multiplier)
  endif
endif
return new_opacity_return
endfunction

```

Level:

```

from Scripts.logger import *
try
  import pygame
  from Scripts.settings import Config
  from Scripts.tile import Tile
  from Scripts.support import *
except
  logger.critical("You do NOT have all the modules installed. Please install Pygame.")
endtry
logger.info("RealDL Level Code.")

class Level inherits Config
  private tile_id
  private player_sprites
  private attackable_sprites
  private attack_sprites
  private current_attack
  private obstacle_sprites
  private visible_sprites
  private display_surface

  public procedure new()
    Config.__init__()
    // get the display surface
    display_surface = new pygame.display.get_surface()
    // sprite group setup
    visible_sprites = new YSortCameraGroup()
    obstacle_sprites = new pygame.sprite.Group()
    // attack sprites
    current_attack = None
    attack_sprites = new pygame.sprite.Group()
    attackable_sprites = new pygame.sprite.Group()
    player_sprites = new pygame.sprite.Group()
    // sprite setup
    tile_id = 0
    create_map()
endprocedure

public procedure create_map()
  layouts =
endprocedure

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

'boundary': import_csv_layout('Map/map_FloorBlocks.csv'),
'grass': import_csv_layout('Map/map_Grass.csv'),
'object': import_csv_layout('Map/map_Objects.csv'),
}

graphics = {
  'grass': import_folder('Graphics/Game/grass'),
  'objects': import_folder('Graphics/Game/objects')

}

for style,layout in layouts.items()
  for row_index,row in enumerate(layout)
    for col_index, col in enumerate(row)
      if col != '-1' then
        x = col_index * TILESIZE
        y = row_index * TILESIZE
        if style == 'boundary' then
          Tile((x,y),[obstacle_sprites],'invisible',pygame.Surface((TILESIZE,TILESIZE)),TILE_ID)
        endif
        if style == 'grass' then
          tile_id = f"Tile{tile_id}"
          if col == '8' then
            grass_image = graphics['grass'][0]
          endif
          elseif col == '9' then
            grass_image = graphics['grass'][1]
          else
            grass_image = graphics['grass'][2]
          endif
          Tile((x,y),[visible_sprites,obstacle_sprites,attackable_sprites],'grass',grass_image,tile_id)
          tile_id += 1
        endif
        if style == 'object' then
          surf = new graphics['objects'][int(col)]
          Tile((x,y),[visible_sprites,obstacle_sprites],'object',surf,TILE_ID)
        endif
      endif
    next col_index, col
  next row_index, row
next style,layout

public procedure run(player, dt)
  // update and draw the game
  visible_sprites.custom_draw(player)
  visible_sprites.update(dt)
endprocedure

class YSortCameraGroup inherits pygame.sprite.Group
  private for
  private offset.y
  private offset.x
  private floor_rect
  private except
  endtry
  private try
  private offset

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

private half_height
private half_width
private display_surface
private settings

public procedure new()
  // general setup
  super.new(    )
  settings = new Config()
  display_surface = new pygame.display.get_surface()
  half_width = new display_surface.get_size()[0] DIV 2
  half_height = new display_surface.get_size()[1] DIV 2
  offset = new pygame.math.Vector2()
  // Floor
  try floor_surf = new pygame.image.load('Graphics/Game/tilemap/bg.png').convert()
  except floor_surf = new pygame.image.load('../Graphics/Game/tilemap/bg.png').convert()
  endtry
  floor_rect = new floor_surf.get_rect(topleft= new(0,0)) // in order to not see the white
endprocedure

public procedure custom_draw(player)
  // getting the offset
  offset.x = player.rect.centerx - half_width
  offset.y = player.rect.centery - half_height
  // drawing the floor
  display_surface.fill(settings.BG_COLOR)
  floor_offset_pos = floor_rect.topleft - offset
  display_surface.blit(floor_surf, floor_offset_pos)
  // for sprite in sprites():
  for sprite in sorted(sprites(), key= new lambda sprite sprite.rect.centery)
    offset_pos = sprite.rect.topleft - offset
    if sprite.sprite_type == "player" then
      if sprite.id == player.id then
        sprite.draw(offset_pos, (50, 190, 40))
      else
        if player.status_alive == "alive" then
          if NOT sprite.status_alive == "dead" then
            sprite.draw(offset_pos)
          else
            endif
            sprite.draw(offset_pos)
          endif
        else
          endif
        endif
        display_surface.blit(sprite.image, offset_pos)
      endif
    next sprite
  endprocedure

```

Logger

import logging

```

// Configure logging
logging.basicConfig(level= new logging.DEBUG) // Set the base logging level
// Create a logger instance

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

logger = new logging.getLogger(__name__)
logger.propagate = False // Disable propagation to parent logger
// Create a formatter for the log messages

class ColoredFormatter inherits logging.Formatter
  COLORS = {
    logging.DEBUG: "\033[1;34m", // Blue for DEBUG
    logging.INFO: "\033[1;29m", // Grey for INFO
    logging.WARNING: "\033[1;33m", // Yellow for WARNING
    logging.ERROR: "\033[1;91m", // Orange for ERROR
    logging.CRITICAL: "\033[1;31m" // Red for CRITICAL
  }
  function format(record)
    log_color = new COLORS.get(record.levelno, "\033[0m") // Default to no color
    log_level = record.levelname
    timestamp = new formatTime(record, datefmt)
    message = log_color + log_level + "\033[0m" + " - " + record.msg
    return f"{timestamp} - {message}"
  endfunction

// Create a handler for console output with the colored formatter
colored_console_handler = new logging.StreamHandler()
colored_console_handler.setFormatter(ColoredFormatter())
// Add the colored handler to the logger
logger.addHandler(colored_console_handler)
logger.info("RealDL Logger Code.")
  
```

Magic:

```

from Scripts.logger import *
try
  import pygame, random
  from Scripts.settings import *

except
  logger.critical("You do NOT have all the modules installed. Please install Pygame.")
endtry
logger.debug("RealDL Magic Code.")

class MagicPlayer inherits Config
  private animation_player

  public procedure new(animation_player)
    Config.__init__()
    animation_player = animation_player
  endprocedure

  public procedure heal(player, strength, cost, groups, player_id)
    if player.energy >= cost then
      player.health += strength
      player.energy -= cost
      if player.health >= player.stats['health'] then
        player.health = player.stats['health']
      endif
      animation_player.create_particles('aura', player.rect.center, strength, groups, player_id)
      animation_player.create_particles('heal', player.rect.center, strength, groups, player_id)
    endif
  endprocedure
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

  endif
endprocedure

public procedure flame(player,strength,cost,groups,player_id)
  if player.energy >= cost then
    player.energy -= cost
    if player.status.split('_')[0] = new= new 'right' then
      direction = new pygame.math.Vector2(1,0)
    endif
    elseif player.status.split('_')[0] = new= new 'left' then
      direction = new pygame.math.Vector2(-1,0)
    endif
    elseif player.status.split('_')[0] = new= new 'up' then
      direction = new pygame.math.Vector2(0,-1)
    else
      direction = new pygame.math.Vector2(0,1)
    endif
  for i in range(1,6)
    if direction.x then //horizontal
      offset_x = new (direction.x * i) * TILESIZE
      x = new player.rect.centerx + offset_x + random.randint(-TILESIZE DIV 3, TILESIZE DIV 3)
      y = new player.rect.centery + random.randint(-TILESIZE DIV 3, TILESIZE DIV 3)
      animation_player.create_particles('flame',(x,y),strength,groups, player_id)
    else // vertical
      offset_y = new (direction.y * i) * TILESIZE
      x = new player.rect.centerx + random.randint(-TILESIZE DIV 3, TILESIZE DIV 3)
      y = new player.rect.centery + offset_y + random.randint(-TILESIZE DIV 3, TILESIZE DIV 3)
      animation_player.create_particles('flame',(x,y),strength,groups, player_id)
    endif
  next i
  endif
endprocedure

```

Main menu:

```

from Scripts.logger import *
try
  import pygame, sys, time
  from Scripts.settings import Config
  from Scripts.encryption import *
  from Scripts.debug import debug
  from Scripts.functions import *

except
  logger.critical("You do NOT have all the modules installed. Please install Pygame.")
endtry
logger.info("RealDL - Main Menu Code")
class MainMenu
  private custom_mouse.mode
  private "music"
  private "sound"
  private starting_dict
  private volume
  private volume_button.x
  private music_change
  private music_box.color

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
private sound_change
private sound_box.color
private magic_keys
private attack_keys
private settings_screen
private keys
private current_x
private main_menu_pages
private loop
private background_music

public procedure new()
    // Music
    try
        background_music = new pygame.mixer.Sound("Audio/main.ogg")
    except
        background_music = new pygame.mixer.Sound("../Audio/main.ogg")
    endtry
endprocedure

public procedure close()
    loop = False
    pygame.quit()
    sys.exit()
endprocedure

public procedure options()
    main_menu_pages = "settings"
endprocedure

public procedure home()
    main_menu_pages = "home"
endprocedure

public procedure start_option()
    main_menu_pages = "start"
endprocedure

public procedure draw_moving_background()
    // Draw the moving image on the screen
    sunset_image.draw(current_x, 0)
    sunset_image.draw((current_x - sunset_image.rect.width), 0)
    current_x += 1
    // If the image has moved beyond its width, reset it to 0
    if current_x >= sunset_image.rect.width then
        current_x = 0
    endif
endprocedure

public procedure change_keys()
    if keys == 'WASD' then
        keys = 'Arrow Keys'
    else
        keys = 'WASD'
    endif
endprocedure
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
public procedure control_settings_change()
    settings_screen = "control"
endprocedure

public procedure audio_settings_change()
    settings_screen = "audio"
endprocedure

public procedure change_attack()
    // Changes the key binds.
    if attack_keys == "Space" then
        attack_keys = "L-Ctrl"
    endif
    elseif attack_keys == "L-Ctrl" then
        attack_keys = "R-Ctrl"
    else
        attack_keys = "Space"
    endif
endprocedure

public procedure change_magic()
    if magic_keys == "L-Shift" then
        magic_keys = "R-Shift"
    endif
    endif
    elseif magic_keys == "R-Shift" then
        magic_keys = "Enter"
    else
        magic_keys = "L-Shift"
    endif
    endif
endprocedure

public procedure sound_box_color()
    // Changes the sound box color
    if sound_change == 1 then
        sound_box.color = new (27,31,35)
        sound_change = 0
    else
        sound_box.color = new (39,174,96)
        sound_change = 1
    endif
endprocedure

public procedure music_box_color()
    // Changes the music box check color.
    if music_change == 1 then
        music_box.color = new (27,31,35)
        music_change = 0
    else
        music_box.color = new (39,174,96)
        music_change = 1
    endif
endprocedure
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

public procedure volume_settings()
  // Get the mouse and sets the volume to where the mouse is.
  mouse_pos = new pygame.mouse.get_pos()
  if mouse_pos[0] - volume_button.width/3 > volume_line.x then
    if mouse_pos[0] + (volume_button.width/3)*2 < volume_line.x + volume_line.width then
      volume_button.x = mouse_pos[0] - volume_button.width/3
      volume = new int(volume_ratio * (mouse_pos[0] - min_vol_x))
    endif
  endif
endprocedure

function create_dict()
  starting_dict = {
    "settings": {
      "control": {
        "movement": keys,
        "offense": attack_keys,
        "magic": magic_keys
      },
      "audio": {
        "volume": volume,
        "sound" then True if sound_change == 1 else False,
        "music" then True if music_change == 1 else False
      }
    },
    "start": {
      "username": name_text_box.return_text() OR "Player",
      "server_ip": ip_text_box.return_text() OR "192.168.0.223"
    }
  }
  background_music.stop()
  loop = False
  // Check the values
  //
  for category, subcategories in starting_dict.items()
    logger.info(f"{category}:")
    for subcategory, values in subcategories.items()
      logger.info(f" {subcategory}:")
      if isinstance(values, dict) then
        for key, value in values.items()
          logger.info(f"   {key}: {value}")
      else
        next key, value
        logger.info(f"   {values}"/)
      endif
    next subcategory, values
  next category, subcategories
endfunction

function start_game()
  if NOT loop then
    return starting_dict
  else
    return None
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

  endif
endfunction

public procedure restart_menu()
  loop = True
  home()
endprocedure

public procedure draw_objects(events)
  // Draw the buttons and check for hover
  if main_menu_pages == "home" then
    github_button.draw((27,31,35),None,"https://github.com/TheRealDL1/Simple-Client-Server")
    youtube_button.draw((27,31,35),None,"https://www.youtube.com/channel/UCf5op_Bt-OTWLQPtCxWnbfg")
    start_button.draw((27,31,35),start_option)
    options_button.draw((27,31,35),options)
    quit_button.draw((27,31,35),close,None,True)
    start_button_hover = new start_button.is_hovered()
    options_button_hover = new options_button.is_hovered()
    quit_button_hover = new quit_button.is_hovered()
    github_button_hover = new github_button.is_hovered()
    youtube_button_hover = new youtube_button.is_hovered()
    start_button_click = new start_button.is_clicking()
    options_button_click = new options_button.is_clicking()
    quit_button_click = new quit_button.is_clicking()
    github_button_click = new github_button.is_clicking()
    youtube_button_click = new youtube_button.is_clicking()
    // Check if mouse is hovering over buttons
    if start_button_hover OR options_button_hover OR quit_button_hover OR github_button_hover OR
    youtube_button_hover then
      // Draw hover text.
      if github_button_hover then github_name.draw("draw","Github", settings.WIDTH-(image_width/2)-
      image_padding, (image_width/2)+image_padding*3.1)
      else github_name.draw("undraw","Github", settings.WIDTH-(image_width/2)-image_padding,
      (image_width/2)+image_padding*3.1)
    endif
      if youtube_button_hover then youtube_name.draw("draw","Youtube", settings.WIDTH-(image_width*2)-
      (image_padding/4), (image_width/2)+image_padding*3.1)
      else youtube_name.draw("undraw","Youtube", settings.WIDTH-(image_width*2)-(image_padding/4),
      (image_width/2)+image_padding*3.1)
    endif
      if NOT start_button_click AND NOT options_button_click AND NOT quit_button_click AND NOT
    github_button_click AND NOT youtube_button_click then
      custom_mouse.mode = 1
    else
      custom_mouse.mode = 2
    else
    endif
      custom_mouse.mode = 0
      github_name.draw("undraw","Github", settings.WIDTH-(image_width/2)-image_padding,
      (image_width/2)+image_padding*3.1)
      youtube_name.draw("undraw","Youtube", settings.WIDTH-(image_width*2)-(image_padding/4),
      (image_width/2)+image_padding*3.1)
    endif
  endif
  if main_menu_pages == "settings" then
    options_board.draw((27,31,35),None, None, False)
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

back.draw((240,240,240),home)
control_settings.draw("draw","Control Settings", (settings.WIDTH/2)*0.70,
(settings.HEIGHT/2)*0.42,control_settings_change)
audio_settings.draw("draw","Audio Settings", (settings.WIDTH/2)*1.3,
(settings.HEIGHT/2)*0.42,audio_settings_change)
github_button.draw((27,31,35),None,"https://github.com/TheRealDL1/Simple-Client-Server")
youtube_button.draw((27,31,35),None,"https://www.youtube.com/@dominicpike")
back_hover = new back.is_hovered()
github_button_hover = new github_button.is_hovered()
youtube_button_hover = new youtube_button.is_hovered()
key_binds_hover = new control_settings.is_hovered()
audio_hover = new audio_settings.is_hovered()
key_hover = new key_binds.is_hovered()
attack_hover = new attack_btn.is_hovered()
volume_btn_hover = new volume_button.is_hovered()
sound_box_hover = new sound_box.is_hovered()
music_box_hover = new music_box.is_hovered()
magic_btn_hover = new magic_btn.is_hovered()
back_click = new back.is_clicking()
github_button_click = new github_button.is_clicking()
youtube_button_click = new youtube_button.is_clicking()
key_binds_click = new control_settings.is_clicking()
audio_click = new audio_settings.is_clicking()
key_click = new key_binds.is_clicking()
attack_click = new attack_btn.is_clicking()
volume_btn_click = new volume_button.is_clicking()
sound_box_click = new sound_box.is_clicking()
music_box_click = new music_box.is_clicking()
magic_btn_click = new magic_btn.is_clicking()

if back_hover OR github_button_hover OR youtube_button_hover OR key_binds_hover OR audio_hover OR
key_hover OR attack_hover OR volume_btn_hover OR sound_box_hover OR music_box_hover OR magic_btn_hover then
  if github_button_hover then github_name.draw("draw","Github", settings.WIDTH-(image_width/2)-
image_padding, (image_width/2)+image_padding*3.1)
  else github_name.draw("undraw","Github", settings.WIDTH-(image_width/2)-image_padding,
(image_width/2)+image_padding*3.1)
  endif
  if youtube_button_hover then youtube_name.draw("draw","Youtube", settings.WIDTH-(image_width*2)-
(image_padding/4), (image_width/2)+image_padding*3.1)
  else youtube_name.draw("undraw","Youtube", settings.WIDTH-(image_width*2)-(image_padding/4),
(image_width/2)+image_padding*3.1)
  endif
  if NOT back_click AND NOT github_button_click AND NOT youtube_button_click AND NOT key_binds_click AND
NOT audio_click AND NOT key_click AND NOT attack_click AND NOT volume_btn_click AND NOT sound_box_click AND
NOT music_box_click AND NOT magic_btn_click then
    custom_mouse.mode = 1
  else
    if volume_btn_click then
      volume_settings()
    endif
    custom_mouse.mode = 2
  else
    endif
    custom_mouse.mode = 0
    github_name.draw("undraw","Github", settings.WIDTH-(image_width/2)-image_padding,
(image_width/2)+image_padding*3.1)
  endif
endif

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
youtube_name.draw("undraw","Youtube", settings.WIDTH-(image_width*2)-(image_padding/4),
(image_width/2)+image_padding*3.1)
endif
if settings_screen == "control" then
    if audio_hover then underline2.draw(None,None,None,True,None,None,None,"draw")
    else underline2.draw(None,None,None,True,None,None,None,"undraw")
endif
underline.draw(None,None,None,True,None,None,None,"draw")
box_control.draw((240,240,240))
movement.draw("draw","Movement", (settings.WIDTH/2)*0.70, (settings.HEIGHT/2)*0.57)
key_binds.draw((240,240,240),change_keys, None, True, keys)
attack_text.draw("draw","Offense", (settings.WIDTH/2)*0.70, (settings.HEIGHT/2)*0.82)
attack_btn.draw((240,240,240),change_attack, None, True, attack_keys)
magic_text.draw("draw","Magic", (settings.WIDTH/2)*0.70, (settings.HEIGHT/2)*1.07)
magic_btn.draw((240,240,240),change_magic, None, True, magic_keys)
endif
elseif settings_screen == "audio" then
    if key_binds_hover then underline.draw(None,None,None,True,None,None,None,"draw")
    else underline.draw(None,None,None,True,None,None,None,"undraw")
endif
underline2.draw(None,None,None,True,None,None,None,"draw")
box_audio.draw((240,240,240))
volume_text.draw("draw","Volume", (settings.WIDTH/2)*1.3, (settings.HEIGHT/2)*0.57)
volume_bar.draw((240,240,240))
volume_line.draw((240,240,240))
volume_button.draw((240,240,240))
volume_indicator.draw("draw",str(volume), (settings.WIDTH/2)*1.435, (settings.HEIGHT/2)*0.685)
audio_text.draw("draw","Audio", (settings.WIDTH/2)*1.3, (settings.HEIGHT/2)*0.82)
audio_box.draw((240,240,240))
sound_text.draw("draw","Sound", (settings.WIDTH/2)*1.185, (settings.HEIGHT/2)*0.97)
music_text.draw("draw","Music", (settings.WIDTH/2)*1.18, (settings.HEIGHT/2)*1.08)
sound_box.draw((240,240,240),sound_box_color)
music_box.draw((240,240,240),music_box_color)
endif
endif
if main_menu_pages == "start" then
    options_board.draw((27,31,35),None, None, False)
    start_back.draw((240,240,240),home)
    github_button.draw((27,31,35),None,"https://github.com/TheRealDL1/Simple-Client-Server")
    youtube_button.draw((27,31,35),None,"https://www.youtube.com/@dominicpike")
    join_boarder.draw((240,240,240))
    bullet_assault.draw("draw","Bullet Assault", (settings.WIDTH/2), (settings.HEIGHT/2)*0.43)
    server_ip_text.draw("draw","Server IP Address", (settings.WIDTH/2), (settings.HEIGHT/2)*0.955)
    message = "Enter the Server's IP Address that you want to join!"
    join_message.draw("draw",message, (settings.WIDTH/2), (settings.HEIGHT/2)*0.52)
    ip_text_box.draw()
    ip_text_box.updateText(events)
    ip_text_box.update()
    join_btn.draw((240,240,240),create_dict)
// Name Text Box
    name_box_text.draw("draw","Username", (settings.WIDTH/2), (settings.HEIGHT/2)*0.665)
    name_text_box.draw()
    name_text_box.updateText(events)
    name_text_box.update()
    github_button_hover = new github_button.is_hovered()
    youtube_button_hover = new youtube_button.is_hovered()
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

start_back_hover = new start_back.is_hovered()
join_hover = new join_btn.is_hovered()
text_box_hover = new ip_text_box.is_hovered()
name_hover = new name_text_box.is_hovered()
github_button_click = new github_button.is_clicking()
youtube_button_click = new youtube_button.is_clicking()
start_back_click = new start_back.is_clicking()
join_click = new join_btn.is_clicking()
text_box_click = new ip_text_box.is_clicking()
name_click = new name_text_box.is_clicking()
if start_back_hover OR github_button_hover OR youtube_button_hover OR join_hover OR text_box_hover OR
name_hover then
  // Draw hover text.
  if github_button_hover then github_name.draw("draw","Github", settings.WIDTH-(image_width/2)-
image_padding, (image_width/2)+image_padding*3.1)
    else github_name.draw("undraw","Github", settings.WIDTH-(image_width/2)-image_padding,
(image_width/2)+image_padding*3.1)
  endif
  if youtube_button_hover then youtube_name.draw("draw","Youtube", settings.WIDTH-(image_width*2)-
(image_padding/4), (image_width/2)+image_padding*3.1)
    else youtube_name.draw("undraw","Youtube", settings.WIDTH-(image_width*2)-(image_padding/4),
(image_width/2)+image_padding*3.1)
  endif
  if NOT start_back_click AND NOT github_button_click AND NOT youtube_button_click AND NOT join_click AND
NOT text_box_click AND NOT name_click then
    custom_mouse.mode = 1
  else
    custom_mouse.mode = 2
  else
  endif
  custom_mouse.mode = 0
  github_name.draw("undraw","Github", settings.WIDTH-(image_width/2)-image_padding,
(image_width/2)+image_padding*3.1)
  youtube_name.draw("undraw","Youtube", settings.WIDTH-(image_width*2)-(image_padding/4),
(image_width/2)+image_padding*3.1)
  endif
  endif
  // Draw the mouse
  custom_mouse.draw()
endprocedure

public procedure redraw_window(events)
  draw_moving_background()
  draw_objects(events)
endprocedure

public procedure run()
  background_music.play(-1)
  while loop
    if music_change == 0 then
      background_music.set_volume(0)
    else
      background_music.set_volume(volume/100)
    endif
    events = new pygame.event.get()
    for event in events
  endwhile
endprocedure
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

if event.type == pygame.QUIT then
  close()
endif
if event.type == pygame.KEYDOWN then
  if event.key == pygame.K_ESCAPE then
    close()
  endif
endif
next event
// Update the display and frame rate
redraw_window(events)
pygame.display.update()
clock.tick(FPS)
endwhile
endprocedure
  
```

Network:

```

import socket
from Scripts.logger import *

logger.debug("RealDL Network Code.")

class Network
  private addr
  private port
  private server
  private client

  public procedure new(server, port)
    try
      client = new socket.socket(socket.AF_INET, socket.SOCK_STREAM)
      server = server
      port = port
      addr = new (server, port)
    except
      logger.critical("Failed to setup Server-Client connection.")
    endtry
  endprocedure

  public procedure connect()
    try
      client.connect(addr)
    except socket.error as Error
      logger.error(f"Socket failed trying to connect {Error}")
    endtry
  endprocedure

  public procedure close()
    client.close()
  endprocedure

  function receive(data_size)
    try
      return client.recv(data_size)
    except socket.error as Error
      
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

logger.error(f"Socket failed trying to receive {Error}")
endtry
endfunction

public procedure send(data)
try
  client.send(data)
except socket.error as Error
  logger.error(f"Socket failed trying to send {Error}")
endtry
endprocedure

public procedure sendall(data)
try
  client.sendall(data)
except socket.error as Error
  logger.error(f"Socket failed trying to sendall {Error}")
endtry
endprocedure
  
```

Particles:

```

from Scripts.logger import *
try
  import pygame, string, random
  from Scripts.settings import Config
  from Scripts.support import import_folder
except
  logger.critical("You do NOT have all the modules installed. Please install Pygame.")
endtry
logger.debug("RealDL Magic Code.")
class AnimationPlayer
  private images
  private frames
  private particles

  public procedure new()
    particles = None
    frames = {
      // magic
      'flame': import_folder('Graphics/Game/particles/flame/frames'),
      'aura': import_folder('Graphics/Game/particles/aura'),
      'heal': import_folder('Graphics/Game/particles/heal/frames'),
    }

    images = {
      // magic
      'flame': import_folder('Graphics/Game/particles/flame/frames',None),
      'aura': import_folder('Graphics/Game/particles/aura',None),
      'heal': import_folder('Graphics/Game/particles/heal/frames',None),
    }
  endprocedure

  public procedure create_particles(animation_type,pos,strength,groups,player_id)
    animation_frames = frames[animation_type]
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

animation_images = images[animation_type]
particles = new ParticleEffect(pos,animation_frames,animation_images,animation_type,groups,strength,player_id)
endprocedure

class ParticleEffect inherits pygame.sprite.Sprite
  private full_path
  private rect
  private image
  private images
  private frames
  private animation_speed
  private frame_index
  private player_id
  private id
  private animation_type
  private damaged_player
  private strength
  private sprite_type
  private settings

  public procedure new(pos,animation_frames,animation_images,animation_type,groups,strength= new None,player_id= new None)
    super.new(groups)
    settings = new Config()
    sprite_type = 'magic'
    strength = strength
    damaged_player = False
    animation_type = animation_type
    id = new create_id()
    player_id = player_id
    frame_index = 0
    animation_speed = new 0.15*(settings.frame_increase_rate/1.5)
    frames = animation_frames
    images = animation_images
    image = frames[frame_index]
    rect = new image.get_rect(center = new pos)
  endprocedure

  public procedure animate(dt)
    frame_index += animation_speed * dt
    if frame_index >= frames.length
      kill()
    else
      image = new frames[int(frame_index)]
    endif
  endprocedure

  function return_type()
    return animation_type
  endfunction

  function return_strength()
    return strength
  endfunction

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

function return_image()
  full_path = new images[int(frame_index)]
  if "../" in full_path then
    full_path = new full_path.replace("../","");
  endif
  return full_path
endfunction

function create_id()
try
  characters = string.ascii_letters + string.digits
  return ".join(random.choice(characters) for _ in range(settings.ID_STRING_LENGTH))"
except
  logger.error("Something went wrong with creating a unique ID.")
endtry
endfunction

public procedure update(dt)
  animate(dt)
endprocedure
  
```

Player

```

from Scripts.logger import *
try
  import pygame, math
  from Scripts.support import import_folder
  from Scripts.settings import Config
except
  logger.critical("You do NOT have all the modules installed. Please install Pygame.")
endtry
logger.debug("RealDL Player Code.")

class Player inherits pygame.sprite.Sprite
  private hitbox.top
  private hitbox.bottom
  private hitbox.left
  private hitbox.right
  private rect.center
  private hitbox.y
  private pos.y
  private hitbox.x
  private pos.x
  private direction.x
  private direction.y
  private animation_images
  private animations
  private obstacle_sprites
  private sprite_type
  private large_font
  private font
  private textSize
  private largefont
  private basefont
  private id
  private username
  
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
private kill_count
private image_alpha
private alpha
private invulnerability_duration
private hurt_time
private vulnerable
private status_alive
private speed
private exp
private attack_strength
private energy
private health
private upgrade_cost
private //max_stats
private stats
private finish_attacking
private attacking_reset
private magic_switch_time
private can_switch_magic
private magic
private create_magic
private magic_index
private switch_duration_cooldown
private weapon_switch_time
private can_switch_weapon
private weapon_type
private weapon
private weapon_index
private create_bullet
private destroy_attack
private create_attack
private paused
private attack_time
private attack_cooldown
private holding_magic_btn
private holding_attack_btn
private magic_attacking
private attacking
private direction
private animation_speed
private frame_index
private status
private 'magic'
private 'attack'
private 'move_right'
private 'move_left'
private 'move_down'
private 'move_up'
private key_binds
private magic_key
private offense
private movement
private pos
private old_hitbox
private hitbox
private rect
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
private image_name
private image
private screen
private settings

public procedure new(pos, image, groups, obstacle_sprites, username, id, health, create_attack= new None,
destroy_attack= new None, create_bullet= new None, create_magic= new None, movement= new "WASD", offense= new
"Space", magic= new "L-Shift",status_alive= new "alive") then
    // Setting up player
    try
        super.new(groups)
        settings = new Config()
        screen = new pygame.display.get_surface()
        try
            image = new pygame.image.load(image).convert_alpha()
            image_name = image
        except
            image = new pygame.image.load(f"../{image}").convert_alpha()
            image_name = f"../{image}"
        endtry
        rect = new image.get_rect(topleft = new pos)
        hitbox = new rect.inflate(0,-26)
        old_hitbox = new hitbox.copy()
        pos = new pygame.math.Vector2(hitbox.topleft)
        movement = movement
        offense = offense
        magic_key = magic
        key_binds = {
            'move_up' then pygame.K_w if movement == "WASD" else pygame.K_UP,
            'move_down' then pygame.K_s if movement == "WASD" else pygame.K_DOWN,
            'move_left' then pygame.K_a if movement == "WASD" else pygame.K_LEFT,
            'move_right' then pygame.K_d if movement == "WASD" else pygame.K_RIGHT,
            'attack' then pygame.K_SPACE if offense == "Space" else pygame.K_LCTRL if offense == "L-Ctrl" else
pygame.K_RCTRL,
            'magic' then pygame.K_LSHIFT if magic_key == "L-Shift" else pygame.K_RSHIFT if magic_key == "R-Shift" else
pygame.K_RETURN
        endif
        endif
        endif
        endif
        endif
        endif
    }
    // graphics setup
    import_player_assets()
    status = 'down'
    frame_index = 0
    animation_speed = new 0.15*(settings.frame_increase_rate)*0.6
    // Movement
    direction = new pygame.math.Vector2()
    attacking = False
    magic_attacking = False
    holding_attack_btn = False
    holding_magic_btn = False
    attack_cooldown = 400
    attack_time = None
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
paused = False
// weapon
create_attack = create_attack
destroy_attack = destroy_attack
create_bullet = create_bullet
weapon_index = 0
weapon = new list(settings.weapon_data.keys())[weapon_index]
weapon_type = settings.weapon_data[weapon]['type']
can_switch_weapon = True
weapon_switch_time = None
switch_duration_cooldown = 200
// magic
magic_index = 0
create_magic = create_magic
magic = new list(settings.magic_data.keys())[magic_index]
can_switch_magic = True
magic_switch_time = None
// magic and weapons
attacking_reset = False
finish_attacking = 0
// stats
stats = {'health': 100,'energy':60,'attack': 10,'magic': 4,'speed': 6}
//max_stats = {'health': 300, 'energy': 140, 'attack': 20, 'magic' : 10, 'speed': 10}
upgrade_cost = {'health': 100, 'energy': 100, 'attack': 100, 'magic' : 100, 'speed': 100}
health = health
energy = stats['energy'] * 0.8
attack_strength = stats['attack']
exp = 500
speed = new stats['speed']*(settings.frame_increase_rate/1.55)
status_alive = status_alive
// damage timer
vulnerable = True
hurt_time = None
invulnerability_duration = 500
alpha = 255
image_alpha = 255
//Other stats
kill_count = 0
username = username
id = id
basefont = "Graphics/Fonts/Orbitron-Medium.ttf"
largefont = "Graphics/Fonts/Orbitron-Bold.ttf"
textSize = 20
try
    font = new pygame.font.Font(basefont, textSize)
    large_font = new pygame.font.Font(largefont, textSize)
except
    font = new pygame.font.Font(f"../{basefont}", textSize)
    large_font = new pygame.font.Font(f"../{largefont}", textSize)
endtry
sprite_type = "player"
obstacle_sprites = obstacle_sprites
except
    logger.error("Failed to create Player Class")
endtry
endprocedure
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

endif
public procedure import_player_assets()
  character_path = 'Graphics/Game/player/'
  animations = {'up': [], 'down': [], 'left': [], 'right': []}
endprocedure

  'right_idle':[],'left_idle':[],'up_idle':[],'down_idle':[],
  'right_attack':[],'left_attack':[],'up_attack':[],'down_attack':[]
animation_images = {'up': [], 'down': [], 'left': [], 'right': [],
  'right_idle':[],'left_idle':[],'up_idle':[],'down_idle':[],
  'right_attack':[],'left_attack':[],'up_attack':[],'down_attack':[]}
for animation in animations.keys()
  full_path = character_path + animation
  animations[animation],animation_images[animation] = new import_folder(full_path,True)

next animation
public procedure input()
  keys = new pygame.key.get_pressed()
  if NOT paused then
    if NOT attacking AND NOT magic_attacking then
      if keys[key_binds['move_up']] then
        direction.y = -1
        status = 'up'
      endif
      elseif keys[key_binds['move_down']] then
        direction.y = 1
        status = 'down'
      else
        direction.y = 0
      endif
      if keys[key_binds['move_right']] then
        direction.x = 1
        status = 'right'
      endif
      elseif keys[key_binds['move_left']] then
        direction.x = -1
        status = 'left'
      else
        direction.x = 0
      endif
      if status_alive == "alive" then
        // attack input
        if keys[key_binds['attack']] AND NOT magic_attacking AND NOT attacking_reset then
          attacking = True
          holding_attack_btn = True
          attack_time = new pygame.time.get_ticks()
          if create_attack != None then
            create_attack(weapon_type)
            if weapon_type == "gun" then
              if create_bullet then
                create_bullet()
              endif
            endif
          endif
        endif
      endif
    endif
  endif
endif

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

// magic input
if keys[key_binds['magic']] AND NOT attacking AND NOT attacking_reset then
  magic_attacking = True
  holding_magic_btn = True
  attack_time = new pygame.time.get_ticks()
  style = new list(settings.magic_data.keys())[magic_index]
  strength = new list(settings.magic_data.values())[magic_index]['strength'] + stats['magic']
  cost = new list(settings.magic_data.values())[magic_index]['cost']
  if create_magic then
    create_magic(style,strength,cost)
  endif
endif
endif
if keys[pygame.K_q] AND can_switch_weapon then
  can_switch_weapon = False
  weapon_switch_time = new pygame.time.get_ticks()
  if weapon_index < list(settings.weapon_data.keys().length)) - 1
    weapon_index += 1
  else
    weapon_index = 0
  endif
  weapon = new list(settings.weapon_data.keys())[weapon_index]
  weapon_type = settings.weapon_data[weapon]['type']
endif
if keys[pygame.K_e] AND can_switch_magic then
  can_switch_magic = False
  magic_switch_time = new pygame.time.get_ticks()
  if magic_index < list(settings.magic_data.keys().length)) - 1
    magic_index += 1
  else
    magic_index = 0
  endif
  magic = new list(settings.magic_data.keys())[magic_index]
else
endif
if NOT keys[key_binds['attack']] then
  holding_attack_btn = False
endif
if NOT keys[key_binds['magic']] then
  holding_magic_btn = False
endif
endif
endif
endprocedure

public procedure get_status()
endprocedure

// idle status
if direction.x == 0 AND direction.y == 0 then
  if NOT 'idle' in status AND NOT 'attack' in status then
    status = status + '_idle'
  endif
endif
if attacking OR magic_attacking then
  direction.x = 0

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

direction.y = 0
if NOT 'attack' in status then
  if 'idle' in status then
    status = new status.replace('_idle','_attack')
  else
    status = status + '_attack'
  else
    endif
  endif
  if 'attack' in status then
    status = new status.replace('_attack','')
  endif
endif
public procedure move(speed, dt)
if NOT paused then
  if hitbox.x != old_hitbox.x OR hitbox.y != old_hitbox.y then
    if NOT 'idle' in status then
      old_hitbox = new hitbox.copy()
    endif
  endif
  if direction.magnitude() != new 0 then
    direction = new direction.normalize()
  endif
  pos.x += direction.x * speed * dt
  hitbox.x = pos.x
  collision('horizontal')
  pos.y += direction.y * speed * dt
  hitbox.y = pos.y
  collision('vertical')
  rect.center = hitbox.center
  endif
endprocedure

public procedure collision(direction)
collision_sprites = new pygame.sprite.spritecollide(obstacle_sprites,False)
if collision_sprites then
  if direction == 'horizontal' then
    for sprite in collision_sprites
      // Collision on the right
      if hitbox.right >= sprite.hitbox.left AND old_hitbox.right <= sprite.old_hitbox.left then
        hitbox.right = sprite.hitbox.left
        pos.x = hitbox.x
      endif
      // Collision on the left
      if hitbox.left <= sprite.hitbox.right AND old_hitbox.left >= sprite.old_hitbox.right then
        hitbox.left = sprite.hitbox.right
        pos.x = hitbox.x
      endif
    next sprite
  endif
  if direction == 'vertical' then
    for sprite in collision_sprites
      // Collision on the bottom
      if hitbox.bottom >= sprite.hitbox.top AND old_hitbox.bottom <= sprite.old_hitbox.top then
        hitbox.bottom = sprite.hitbox.top
        pos.y = hitbox.y
      endif
    next sprite
  endif
endif
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

      endif
      // Collision on the top
      if hitbox.top <= sprite.hitbox.bottom AND old_hitbox.top >= sprite.old_hitbox.bottom then
        hitbox.top = sprite.hitbox.bottom
        pos.y = hitbox.y
      endif
    next sprite
  endif
endif
endprocedure

public procedure cooldowns()
  current_time = new pygame.time.get_ticks()
if attacking then
  if current_time - attack_time >= attack_cooldown + settings.weapon_data[weapon]['cooldown'] then
    if NOT holding_attack_btn then
      attacking = False
      attacking_reset = True
      finish_attacking = new pygame.time.get_ticks()
      if destroy_attack != None then
        destroy_attack()
      endif
    endif
  endif
  if current_time - attack_time >= settings.weapon_data[weapon]['cooldown'] then
    if holding_attack_btn AND weapon_type == "gun" then
      if create_bullet then
        create_bullet()
        attack_time = current_time
      endif
    endif
  endif
endif
if magic_attacking then
  if current_time - attack_time >= settings.magic_data[magic]['cooldown'] then
    if NOT holding_magic_btn then
      magic_attacking = False
      attacking_reset = True
      finish_attacking = new pygame.time.get_ticks()
    endif
  endif
endif
if attacking_reset then
  if current_time - finish_attacking >= settings.attack_or_magic_cooldown then
    attacking_reset = False
  endif
endif
if NOT can_switch_weapon then
  if current_time - weapon_switch_time >= switch_duration_cooldown then
    can_switch_weapon = True
  endif
endif
if NOT can_switch_magic then
  if current_time - magic_switch_time >= switch_duration_cooldown then
    can_switch_magic = True
  endif
endif

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
endif
if NOT vulnerable then
    if current_time - hurt_time >= invulnerability_duration then
        vulnerable = True
    endif
endif
endprocedure

function get_image_name()
if "../" in image_name then
    image_name = new image_name.replace("../","");
endif
return image_name
endfunction

public procedure animate(dt)
if NOT paused then
    animation = animations[status]
    image = animation_images[status]
    // loop over the frame index
    frame_index += animation_speed * dt
    if frame_index >= animation.length
        frame_index = 0
    endif
    // set the image
    image = new animation[int(frame_index)]
    image_name = new image[int(frame_index)]
    rect = new image.get_rect(center = new hitbox.center)
    //flicker
    flicker()
endif
endprocedure

public procedure set_opacity()
    image.set_alpha(alpha)
endprocedure

function return_alpha()
    return image_alpha
endfunction

public procedure flicker()
// flicker
if NOT vulnerable then
    image_alpha = new wave_value()
    image.set_alpha(image_alpha)
else
    image.set_alpha(255)
endif
endprocedure

function get_full_weapon_damage()
    base_damage = stats['attack']
    weapon_damage = settings.weapon_data[weapon]['damage']
    return base_damage + weapon_damage
endfunction
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
function get_full_magic_damage()
    base_damage = stats['magic']
    spell_damage = settings.magic_data[magic]['strength']
    return base_damage + spell_damage
endfunction

function get_value_by_index(index)
    return list(stats.values())[index]
endfunction

function get_cost_by_index(index)
    return list(upgrade_cost.values())[index]
endfunction

public procedure energy_recovery()
    if energy < stats['energy'] then
        energy += 0.002 * stats['magic']
    else
        energy = stats['energy']
    endif
endprocedure

public procedure damage_player(damage)
    if vulnerable then
        vulnerable = False
        hurt_time = new pygame.time.get_ticks()
        if health - damage <= 0 then
            health = 0
            status_alive = "dead"
        else
            health -= damage
        endif
    endif
endprocedure

function return_alive_status()
    return status_alive
endfunction

function wave_value()
    value = new math.sin(pygame.time.get_ticks())
    if value >= 0 then
        return 255
    else
        return 0
    endif
endfunction

public procedure draw(offset_pos, color= new (190, 40, 50))
    // Draw the player.
    alpha = 255
    if status_alive == "dead" then
        color = new (128, 128, 128)
        alpha = 135
    endif
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

draw_image = new image.copy()
draw_image.set_alpha(alpha)
text_surface = new large_font.render(username, True, color)
text_rect = new text_surface.get_rect()
username_pos = new (offset_pos[0] + rect.width DIV 2 - text_surface.get_width() DIV 2, offset_pos[1] - 35)
// Define the dimensions and position of the black box
box_width = text_rect.width + 6 // Adjust the width as needed
box_height = text_rect.height + 6 // Adjust the height as needed
box_pos = new (username_pos[0]-3, username_pos[1]-3) // Adjust the position as needed
// Draw the black box
pygame.draw.rect(screen, (27,31,35), ((box_pos[0]-2,box_pos[1]-2), (box_width+4, box_height+4)),3,10)
//pygame.draw.rect(screen, (27,31,35), (box_pos, (box_width, box_height)),0,10)
screen.blit(draw_image, offset_pos)
screen.blit(text_surface, username_pos)
endprocedure

public procedure update(dt)
  input()
  cooldowns()
  get_status()
  animate(dt)
  move(speed, dt)
  energy_recovery()
endprocedure
  
```

Server:

```

from Scripts.logger import *
try
  from _thread import *
  from random import randint, choice
  import string, math, socket, time
  from Scripts.settings import Config
  from Scripts.encryption import *
  from Scripts.debug import debug

except
  logger.critical("You do NOT have all the modules installed. Please install Pygame, RSA AND Pycryptodome.")
endtry
logger.debug("RealDL Server Code.")

class Server inherits Config
  private //logger.info(f"There
  private health_leeway
  private allowed_health
  private allowed_speed
  private speed_leeway
  private coordinates
  private rsa_encrypt
  private public_key,
  private rsa_keys
  private s
  private running
  private connections
  private players
  private port
  
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
private server
private ready_to_play

public procedure new()
try
    Config.__init__()
    initialize_server()
except
    logger.error(f"Couldn't initialize server.")
endtry
endprocedure

public procedure initialize_server()
    ready_to_play = False
    server = SERVER
    port = PORT
    players = {}
    connections = 0
    running = True
    s = new socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    rsa_keys = new RSA_Keys(ENCRYPTION_DATA_SIZE)
    public_key, private_key = new rsa_keys.export_keys()
    rsa_encrypt = new RSA_Encryption(public_key)
    coordinates = [
        [[1030, 1270],[1278, 1616]],
        [[1100, 1660],[1278, 2180]],
        [[1420, 580],[2494, 432]],
        [[1670, 245],[1865, 592]],
        [[2250, 2485],[2622, 2896]],
        [[3134, 2485],[2818, 2832]]
    ]
    // Speed
    speed_leeway = 100
    allowed_speed = new 10*(frame_increase_rate/1.55)
    // Health
    allowed_health = 300
    health_leeway = 15
    try
        s.bind((server, port))
    except socket.error as e
        logger.error(f"Socket Error: {e}")
    endtry
    s.listen()
    logger.info("Waiting for connections, Server Started")
    logger.info(f"Server IP Address: {SERVER}")
endprocedure

function create_random_string()
    characters = string.ascii_letters + string.digits
    return ".join(choice(characters) for _ in range(ID_STRING_LENGTH))"
endfunction

public procedure is_touching(player_x, player_y, threshold= new 100)
    touching = False
    for player in players.values()
        // If two players are taking both an x and y position. We want to seperate player spawns.
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

// The threshold is default at 100. We dont want players to close to each other.
if abs(player['player']['x']-player_x) <= new SQUARE_SIZE + threshold AND abs(player['player']['y']-player_y) <= new
SQUARE_SIZE + threshold then
  touching = True
endif
next player
return touching
endprocedure

public procedure get_player_position()
function create_coords()
  random_area = new choice(coordinates)
  if random_area == coordinates[2] then
    x = new randint(random_area[0][0], random_area[1][0])
    y = new randint(random_area[1][1], random_area[0][1])
  endif
  elseif random_area == coordinates[5] then
    x = new randint(random_area[1][0], random_area[0][0])
    y = new randint(random_area[0][1], random_area[1][1])
  else
    x = new randint(random_area[0][0], random_area[1][0])
    y = new randint(random_area[0][1], random_area[1][1])
  endif
  return x,y
endfunction

valid_coords = False
while NOT valid_coords
  x, y = new create_coords()
  if NOT is_touching(x, y) then
    valid_coords = True
  endif
endwhile
return x, y
endprocedure

function validate_username(username)
  username_valid = False
  indent = 0
  usernames = new [player['player']['username'] for player in players.values()]
  new_username = username
  while NOT username_valid
    if indent > 0 then
      new_username = new_username + f"_{str(indent)}"
    endif
    if new_username in usernames then
      indent += 1
    else
      username_valid = True
    endif
  endwhile
  return new_username
endfunction

function create_new_player(username)
  key_string = new create_random_string()

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

player_x, player_y = new get_player_position()
username = new validate_username(username)
return {
  "player": {
    "x": player_x,
    "y": player_y,
    "image": "Graphics/Game/player/down_idle/idle_down.png",
    "username": username,
    "status": "alive",
    "health": 100,
    "id": key_string,
    "kill_count": 0,
    "killed_by": ""
  },
  "weapon": [],
  "bullets": [],
  "magic": []
}, key_string
endfunction

public procedure validate_movement(player_pos)
  function calculate_speed(pos1, pos2, delta_time)
    // Calculate distance between two positions using Euclidean distance formula
    try
      delta_x = pos2[0] - pos1[0]
      delta_y = pos2[1] - pos1[1]
      distance = new math.sqrt(delta_x**2 + delta_y**2)
      // Calculate speed (distance / time)
      speed = distance / delta_time
    except
      speed = allowed_speed
    endtry
    return speed
  endfunction

  // Calculate the distance moved between updates
  first_pos = player_pos[0][:2]
  last_pos = player_pos[-1][:2]
  total_time = new sum(values[2] for values in player_pos)
  // Calculate speed between the first and last positions
  speed = new calculate_speed(first_pos, last_pos, total_time)
  if speed > allowed_speed + speed_leeway then
    return False
  endif
  return True
endprocedure

function validate_health(player)
  if player['health'] > allowed_health + health_leeway then
    return False
  else
    return True
  endif
endfunction

function validate_player_status(player, data)
  if player['status'] == "dead" AND data['status'] == 'alive' then

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

    return player['status']
  endif
  elseif player['status'] == "alive" AND data['status'] == 'dead' then
    return data['status']
  else
    return player['status']
  endif
endfunction

function kill_count(key_string, kill_count, killed_players)
  kills = []
  // gets a list of all ids of players that have killed someone
  for player in players.values()
    if player['player']['killed_by'] != "" then
      if player['player']['id'] NOT in killed_players then
        kills.append(player['player']['killed_by'])
        killed_players.append(player['player']['id'])
      endif
    endif
  next player
  // checks if this player has killed anyone
  for kill_id in kills
    // //logger.debug(kill_count, kill_id, key_string)
    if kill_id == key_string then
      kill_count += 1
    endif
  next kill_id
  return kill_count, killed_players
endfunction

function end_game()
  player_count = 0
  for player in players.values()
    if player['player']['status'] == "alive" then
      player_count += 1
    endif
  next player
  if player_count <= 1 then
    return True
  else
    return False
  endif
endfunction

function validate_position(player)
  // Map boundaries, stops players from leaving the map.
  if player['x'] <= 180 OR player['x'] >= 3400 then
    return True
  endif
  elseif player['y'] <= -50 OR player['y'] >= 3150 then
    return True
  else
    return False
  endif
endfunction
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

public procedure handle_client_communication(conn, key_string, aes_encryption)
  running = True
  player_pos = []
  killed_players = []
  last_time = new time.time()
  end_time = 0
  reset_game_time = 15
  while running
    try
      //// ONE player left. -> after 30 seconds -> kick all players -> reset all key values
      // Recieve player data
      player_data = new aes_encryption.decrypt(unserialize(conn.recv(DATA_SIZE)))
      // check the player's status is correct (this doesn't result in a kick)
      data = player_data['player']
      player_data['player']['status'] = new validate_player_status(players[key_string]['player'], data)
      // Validate if the player has been killed.
      player_data['player']['kill_count'], killed_players = new kill_count(key_string, player_data['player']['kill_count'],
      killed_players)
      players[key_string] = player_data
      if end_game() then
        if time.time() - end_time >= new reset_game_time then
          running = False
        else
          end_time = new time.time()
        endif
      // Calculate the speed of play (FPS).
      delta_time = new time.time() - last_time
      last_time = new time.time()
      // Validate the player's position.
      if validate_position(players[key_string]['player']) then
        running = False
        logger.info(f"Player {key_string} disconnected as they tried to leave the map.")
      endif
    endwhile
  endprocedure

  // Validate the player data to ensure they are not speed hacking. (this does result in a kick)
  player_pos.append([players[key_string]['player']['x'], players[key_string]['player']['y'], delta_time])
  if player_pos.length > 20 player_pos.pop(0)
  endif
  if NOT validate_movement(player_pos) then
    running = False
    logger.info(f"Player {key_string} disconnected because they were speed hacking.")
  endif
  // Validate player health to ensure they are not health hacking. (this does result in a kick)
  if NOT validate_health(players[key_string]['player']) then
    running = False
    logger.info(f"Player {key_string} disconnected because they were health hacking.")
  endif
  //logger.info(f"Received Player Dict: {player_data}.")
  if NOT data then
    logger.info(f"Player {key_string} disconnected.")
    running = False
  else

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

    reply = players
    encrypted_reply = new serialize(aes_encryption.encrypt(reply))
  endif
  conn.sendall(encrypted_reply)
  //logger.info(f"Sending All Player Dict: {reply}.")
except
  logger.info(f"Player {key_string} lost connection.")
  running = False
endtry
logger.info(f"Connection Closed for Player {key_string}.")
try
  del players[key_string]
  connections -= 1
except
  logger.info(f"No player with the ID: {key_string} exists.")
endtry
conn.close()
public procedure waiting_zone(conn, key_string, aes_encryption)
running = True
while running
  try
    // send over number of players connected
    reply = {"connections":connections,"started":ready_to_play}
    encrypted_reply = new serialize(aes_encryption.encrypt(reply))
    conn.sendall(encrypted_reply)
    // Receive start or not.
    start_dict = new aes_encryption.decrypt(unserialize(conn.recv(DATA_SIZE)))
    start = start_dict["ready"]
    if start AND connections >= 2 then
      ready_to_play = True
    endif
  except
    running = False
    logger.error("Something went wrong.")
  endtry
  if start then
    handle_client_communication(conn, key_string, aes_encryption)
  endif
endwhile
try
  del players[key_string]
  connections -= 1
except
  logger.info(f"No player with the ID: {key_string} exists.")
endtry
conn.close()
endprocedure

public procedure threaded_client(conn)
try
  // Send Public Key
  data_to_send = new serialize(public_key)
  conn.send(data_to_send)
  //logger.info(f"Sending Public Key: {public_key}")
  // Get AES Key and username

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

dict_received_from_client = new rsa_encrypt.decrypt(unserialize(conn.recv(ENCRYPTION_DATA_SIZE)),private_key)
aes_key = dict_received_from_client['aes_key']
username = dict_received_from_client['username']
aes_encryption = new AES_Encryption(aes_key)
//logger.info(f"Received AES Key: {aes_key} and Username: {username}")
// Create Player
new_player, key_string = new create_new_player(username)
players[key_string] = new_player
//logger.info(f"Created New Player: {new_player}. ID: {key_string}")
//Send player dict
player_dict_send = {'player_data':new_player}
encrypted_player = new serialize(aes_encryption.encrypt(player_dict_send))
conn.send(encrypted_player)
logger.info(f"Sending Player dict to client: {new_player}")
waiting_zone(conn, key_string, aes_encryption)
//handle_client_communication(conn, key_string, aes_encryption)
except
//logger.error("An Error Occurred trying to setup Client-Server connection.")
try
  del players[key_string]
  connections -= 1
except
  logger.info(f"No player was deleted.")
endtry
conn.close()
endtry
endprocedure

public procedure run()
  while running
    //// Limit on 5 players
    conn, addr = new s.accept()
    logger.debug(f"Connections: {str(connections)}")
    if ready_to_play AND connections == 0 then
      ready_to_play = False
    endif
    if ready_to_play OR connections >= 5 then
      conn.close()
    else
      connections += 1
    endif
    //logger.info(f"Connected to: {addr}")
    //logger.info(f"There {'is' if connections= new = new 1 else 'are'} {connections} {'client' if connections= new = new 1 else 'clients'} connected to the server!")
    start_new_thread(threaded_client, (conn,))
  endwhile
endprocedure

if __name__ == "__main__":
  try
    server = new Server()
    server.run()
  except
    logger.critical("Server has failed to launch.")
  endif
endif
  
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Settings:

```
from Scripts.logger import *
try
    from socket import gethostname, gethostbyname
    import pickle, pygame

except
    logger.critical("You do NOT have all the modules installed. Please install Pygame.")
endtry
pygame.init()
logger.info("Github: https://github.com/TheRealDL1/Simple-Client-Server")
logger.debug("RealDL Settings Code.")

class Config
    private UPGRADE_BG_COLOR_SELECTED
    private BAR_COLOR_SELECTED
    private BAR_COLOR
    private TEXT_COLOR_SELECTED
    private UI_BORDER_COLOR_ACTIVE
    private ENERGY_COLOR
    private HEALTH_COLOR
    private TEXT_COLOR
    private UI_BORDER_COLOR
    private UI_BG_COLOR
    private WATER_COLOR
    private UI_FONT_SIZE
    private UI_FONT
    private ITEM_BOX_SIZE
    private ENERGY_BAR_WIDTH
    private HEALTH_BAR_WIDTH
    private BAR_HEIGHT
    private magic_data
    private bullet_type
    private weapon_data
    private attack_or_magic_cooldown
    private TILE_ID
    private TILESIZE
    private ENCRYPTION_DATA_SIZE
    private SMALL_DATA
    private DATA_SIZE
    private BG_COLOR
    private ID_STRING_LENGTH
    private PORT
    private SERVER
    private HOST_NAME
    private FPS
    private BITS
    private SQUARE_SIZE
    private frame_increase_rate
    private WIDTH
    private HEIGHT

    public procedure new()
        // Screen Width and Height
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
info = new pygame.display.Info()
max_width = info.current_w
max_height = info.current_h
HEIGHT = max_height // 720
WIDTH = max_width // 1280
frame_increase_rate = 100
// Other Server/ Client Settings
SQUARE_SIZE = 64
BITS = 256
FPS = 60
HOST_NAME = new gethostname()
SERVER = new gethostbyname(HOST_NAME)
PORT = 5555
ID_STRING_LENGTH = 30
BG_COLOR = new (113, 221, 238)
DATA_SIZE = 8192
SMALL_DATA = 32
ENCRYPTION_DATA_SIZE = 1024
TILESIZE = 64
TILE_ID = 'Tile'
// weapons
attack_or_magic_cooldown = 250
weapon_data = {
    'sword': {'cooldown': 100, 'speed': 0, 'damage': 15,'graphic':'Graphics/Game/weapons/sword/full.png','type':'melee'},
    'lance': {'cooldown': 400, 'speed': 0, 'damage': 30,'graphic':'Graphics/Game/weapons/lance/full.png','type':'melee'},
    'axe': {'cooldown': 300, 'speed': 0, 'damage': 20,'graphic':'Graphics/Game/weapons/axe/full.png','type':'melee'},
    'rapier': {'cooldown': 50, 'speed': 0, 'damage': 8,'graphic':'Graphics/Game/weapons/rapier/full.png','type':'melee'},
    'sai': {'cooldown': 80, 'speed': 0, 'damage': 10,'graphic':'Graphics/Game/weapons/sai/full.png','type':'melee'},
    'revolver': {'cooldown': 1000, 'speed': 20, 'damage': 25,'graphic':'Graphics/Game/weapons/revolver/full.png','type':'gun'},
    'msg': {'cooldown': 175, 'speed': 10, 'damage': 5,'graphic':'Graphics/Game/weapons/msg/full.png','type':'gun'},
    'pistol': {'cooldown': 600, 'speed': 15, 'damage': 18,'graphic':'Graphics/Game/weapons/pistol/full.png','type':'gun'}}}
// Bullets
bullet_type = {
    'msg': {
        'down': 'Graphics/Game/bullets/msg/down.png',
        'left': 'Graphics/Game/bullets/msg/left.png',
        'right': 'Graphics/Game/bullets/msg/right.png',
        'up': 'Graphics/Game/bullets/msg/up.png',
    },
    'pistol': {
        'down': 'Graphics/Game/bullets/pistol/down.png',
        'left': 'Graphics/Game/bullets/pistol/left.png',
        'right': 'Graphics/Game/bullets/pistol/right.png',
        'up': 'Graphics/Game/bullets/pistol/up.png',
    },
    'revolver': {
        'down': 'Graphics/Game/bullets/revolver/down.png',
        'left': 'Graphics/Game/bullets/revolver/left.png',
        'right': 'Graphics/Game/bullets/revolver/right.png',
        'up': 'Graphics/Game/bullets/revolver/up.png',
    }
}
// magic
magic_data = {
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

'flame': {'strength': 5,'cost': 20,'graphic':'Graphics/Game/particles/flame/fire.png','type':'magic','cooldown':1250},  

'heal' : {'strength': 20,'cost': 10,'graphic':'Graphics/Game/particles/heal/heal.png','type':'magic','cooldown':650}}  

// ui  

BAR_HEIGHT = 20  

HEALTH_BAR_WIDTH = 200  

ENERGY_BAR_WIDTH = 140  

ITEM_BOX_SIZE = 90  

UI_FONT = 'Graphics/Fonts/Orbitron-Medium.ttf'  

UI_FONT_SIZE = 18  

// general colors  

WATER_COLOR = new (113, 221, 238)  

UI_BG_COLOR = new (34, 34, 34)  

UI_BORDER_COLOR = new (17, 17, 17)  

TEXT_COLOR = new (238, 238, 238)  

// ui colors  

HEALTH_COLOR = new (255, 0, 0)  

ENERGY_COLOR = new (23, 108, 235)  

UI_BORDER_COLOR_ACTIVE = new (255, 215, 0)  

// upgrade menu  

TEXT_COLOR_SELECTED = new (17, 17, 17)  

BAR_COLOR = new (238, 238, 238)  

BAR_COLOR_SELECTED = new (17, 17, 17)  

UPGRADE_BG_COLOR_SELECTED = new (238, 238, 238)  

endprocedure

function serialize(data)
try
  return pickle.dumps(data)
except pickle.PicklingError as Error
  logger.error(f"Failed to pickle data: {Error}")
endtry
endfunction

function unserialize(data)
try
  return pickle.loads(data)
except pickle.UnpicklingError as Error
  logger.error(f"Failed to unpickle data: {Error}")
endtry
endfunction

Support:  

from Scripts.logger import *
try
  from csv import reader
  from os import walk, path
  import pygame
except
  logger.critical("You do NOT have all the modules installed. Please install Pygame.")
endtry
logger.debug("RealDL Support Code.")
function import_csv_layout(path)
  terrain_map = []
  try
    with open(path) as level_map:
      layout = new reader(level_map, delimiter = new ',')

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

for row in layout
  terrain_map.append(list(row))
next row
except
  with open(f"../{path}") as level_map:
    layout = new reader(level_map,delimiter = new ',')
for row in layout
  terrain_map.append(list(row))
next row
endtry
return terrain_map
endfunction

function import_folder(folder_path,return_image_list= new False)

surface_list = []
image_list = []
values = False
while NOT values
  for root, dirs, img_files in walk(folder_path)
    for image in img_files
      full_path = new path.join(root, image)
      try
        image_surf = new pygame.image.load(full_path).convert_alpha()
      except
        image_surf = new pygame.image.load(f"../{full_path}").convert_alpha()
      endtry
      surface_list.append(image_surf)
      image_list.append(full_path)
    next image
  next root, dirs, img_files
if surface_list AND image_list then
  values = True
else
  if "../" in folder_path then
    values = True
  else
    folder_path = "../" + folder_path
  endif
endif
endwhile
if return_image_list == True then
  return surface_list, image_list
endif
elseif return_image_list == None then
  return image_list
else
  return surface_list
endif
endfunction
  
```

Tile:

```

from Scripts.logger import *
try
  import pygame
  
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
from Scripts.settings import *

except
    logger.critical("You do NOT have all the modules installed. Please install Pygame.")
endtry
logger.debug("RealDL Tile Code.")

class Tile inherits pygame.sprite.Sprite
    private old_hitbox
    private hitbox
    private rect
    private image
    private id
    private sprite_type
    private settings

    public procedure new(pos,groups,sprite_type,surface,id)
        try
            super.new(groups)
            settings = new Config()
            sprite_type = sprite_type
            id = id
            image = surface
            if sprite_type == 'object' then
                rect = new image.get_rect(topleft = new (pos[0],pos[1] - settings.TILESIZE))
            else
                rect = new image.get_rect(topleft = new pos)
            endif
            hitbox = new rect.inflate(-10,-10)
            old_hitbox = new hitbox.copy()
        except
            logger.error(f"Failed to create Tile Class.")
        endtry
    endprocedure
```

UI:

```
from Scripts.logger import *
try
    from Scripts.functions import *
    from Scripts.settings import Config
    import pygame

except
    logger.critical("You do NOT have all the modules installed. Please install Pygame.")
endtry
logger.debug("RealDL UI Code.")

class UI inherits Config
    private custom_mouse.mode
    private magic_graphics
    private weapon_graphics
    private energy_bar_rect
    private health_bar_rect
    private players_alive
    private kill_count
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
private boarder_stats
private player_kill_count
private player_count
private font
private continue_btn
private bullet_assault
private quit
private info2
private info
private join_boarder
private border
private base_button_height
private base_button_width
private thickness
private curve
private big_text_size
private base_text_size
private text_height
private button_padding
private image_padding
private image_height
private image_width
private BASE_BUTTON_HEIGHT
private BASE_BUTTON_WIDTH
private THICKNESS
private CURVE
private BUTTON_PADDING
private IMAGE_PADDING
private IMAGE_HEIGHT
private IMAGE_WIDTH
private TEXT_HEIGHT
private BASE_TEXT_SIZE
private BIG_TEXT_SIZE
private quit_function
private draw_ui
private custom_mouse
private height_ratio
private width_ratio
private DEFAULT_HEIGHT
private DEFAULT_WIDTH
private screen_height
private screen_width
private screen

public procedure new(custom_mouse, quit_function, player_count, kill_count)
    // Setup Pygame Variables
    Config.__init__()
    screen = new pygame.display.get_surface()
    info = new pygame.display.Info()
    screen_width = info.current_w
    screen_height = info.current_h
    DEFAULT_WIDTH = 1920
    DEFAULT_HEIGHT = 1080
    width_ratio = screen_width / DEFAULT_WIDTH
    height_ratio = screen_height / DEFAULT_HEIGHT
    custom_mouse = custom_mouse
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

draw_ui = False
quit_function = quit_function
// Constants setup
BIG_TEXT_SIZE = 50
BASE_TEXT_SIZE = 15
TEXT_HEIGHT = 20
IMAGE_WIDTH = 64
IMAGE_HEIGHT = 64
IMAGE_PADDING = 20
BUTTON_PADDING = 85
CURVE = 10
THICKNESS = 2
BASE_BUTTON_WIDTH = 250
BASE_BUTTON_HEIGHT = 70
// Variable setup
image_width = new int(IMAGE_WIDTH * width_ratio)
image_height = new int(IMAGE_HEIGHT * height_ratio)
image_padding = new int(IMAGE_PADDING * width_ratio)
button_padding = new int(BUTTON_PADDING * height_ratio)
text_height = new int(TEXT_HEIGHT * height_ratio)
base_text_size = new int(BASE_TEXT_SIZE * height_ratio)
big_text_size = new int(BIG_TEXT_SIZE * height_ratio)
curve = new int(CURVE * height_ratio)
thickness = new int(THICKNESS * height_ratio)
base_button_width = new int(BASE_BUTTON_WIDTH * width_ratio)
base_button_height = new int(BASE_BUTTON_HEIGHT * height_ratio)
// UI Board
border = new Button((27, 31, 35), (27, 31, 35), screen_width / 2, screen_height / 2, "Graphics/Fonts/Orbitron-Regular.ttf", (27, 31, 35), (27, 31, 35), screen_width * 0.7, screen_height * 0.7, 'Rectangle', None, big_text_size, int(curve * 1.5))
join_boarder = new Button((27, 31, 35), (27, 31, 35), (screen_width / 2), (screen_height / 2), "Graphics/Fonts/Orbitron-Regular.ttf", (240, 240, 240), (136, 173, 227), base_button_width * 3, base_button_height * 6, 'Rectangle', None, big_text_size, curve)
info = new Text(text_height, "Graphics/Fonts/Orbitron-Regular.ttf", (240, 240, 240), None, None, None, int(base_text_size * 1.7))
info2 = new Text(text_height, "Graphics/Fonts/Orbitron-Regular.ttf", (240, 240, 240), None, None, None, int(base_text_size * 1.7))
quit = new Button((174, 39, 96), (27, 31, 35), screen_width / 2, screen_height * 0.78, "Graphics/Fonts/Orbitron-Medium.ttf", (27, 31, 35), (174, 39, 96), base_button_width, base_button_height, 'Quit', 'Rectangle', None, int(big_text_size / 1.3), curve)
bullet_assault = new Text(text_height, "Graphics/Fonts/Orbitron-Bold.ttf", (240, 240, 240), None, None, None, int(base_text_size * 3.5))
continue_btn = new Button((39, 174, 96), (27, 31, 35), (screen_width / 2), (screen_height / 2) * 1.445 - button_padding, "Graphics/Fonts/Orbitron-Medium.ttf", (27, 31, 35), (39, 174, 96), base_button_width, base_button_height, 'Continue', 'Rectangle', None, int(big_text_size / 1.3), curve)
// In-game UI. kill count,
// general
try
  font = new pygame.font.Font(UI_FONT, UI_FONT_SIZE)
except
  font = new pygame.font.Font(f"../{UI_FONT}", UI_FONT_SIZE)
endtry
player_count = player_count
player_kill_count = kill_count
// Kill count and player count
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

boarder_stats = new Button(UI_BG_COLOR, UI_BG_COLOR, WIDTH-(HEALTH_BAR_WIDTH)/2-10-2,
10+(BAR_HEIGHT*1.25)+2, "Graphics/Fonts/Orbitron-Medium.ttf", TEXT_COLOR, TEXT_COLOR, (HEALTH_BAR_WIDTH),
BAR_HEIGHT*2.5, "", 'Rectangle', None, UI_FONT_SIZE, 0)
kill_count = new Button(UI_BG_COLOR, UI_BG_COLOR, WIDTH-(HEALTH_BAR_WIDTH)/2-10-2,
13.5+(BAR_HEIGHT/2)+2, "Graphics/Fonts/Orbitron-Medium.ttf", TEXT_COLOR, TEXT_COLOR, (HEALTH_BAR_WIDTH),
BAR_HEIGHT, f'Kill count: {player_kill_count}', 'Rectangle', None, UI_FONT_SIZE, 0)
players_alive = new Button(UI_BG_COLOR, UI_BG_COLOR, WIDTH-(HEALTH_BAR_WIDTH)/2-10-2,
BAR_HEIGHT+16.5+(BAR_HEIGHT/2)+2, "Graphics/Fonts/Orbitron-Medium.ttf", TEXT_COLOR, TEXT_COLOR,
(HEALTH_BAR_WIDTH), BAR_HEIGHT, f'Players alive: {player_count}', 'Rectangle', None, UI_FONT_SIZE, 0)
// bar setup
health_bar_rect = new pygame.Rect(10, 10, HEALTH_BAR_WIDTH, BAR_HEIGHT)
energy_bar_rect = new pygame.Rect(10, 34, ENERGY_BAR_WIDTH, BAR_HEIGHT)
// convert weapon dictionary
weapon_graphics = []
for weapon in weapon_data.values()
  weapon_path = weapon['graphic']
  try
    weapon = new pygame.image.load(weapon_path).convert_alpha()
  except
    weapon = new pygame.image.load(f"../{weapon_path}").convert_alpha()
  endtry
  weapon_graphics.append(weapon)
next weapon
// convert magic dictionary
magic_graphics = []
for magic in magic_data.values()
  magic_path = magic['graphic']
  try
    magic = new pygame.image.load(magic_path).convert_alpha()
  except
    magic = new pygame.image.load(f"../{magic_path}").convert_alpha()
  endtry
  magic_graphics.append(magic)
next magic
endprocedure

public procedure stop_drawing()
  draw_ui = False
endprocedure

public procedure draw_menu()
if draw_ui then
  border.draw((27, 31, 35), None, None, False)
  join_boarder.draw((240, 240, 240))
  info.draw("draw", "Game Paused: You have entered the main menu.", (screen_width / 2), (screen_height / 2) *
0.665)
  info2.draw("draw", "Movement is currently disabled.", (screen_width / 2), (screen_height / 2) * 0.72)
  quit.draw((240, 240, 240), quit_function)
  continue_btn.draw((240, 240, 240), stop_drawing)
  bullet_assault.draw("draw", "Bullet Assault", (screen_width / 2), (screen_height / 2) * 0.43)
  start_back_hover = new quit.is_hovered()
  join_hover = new continue_btn.is_hovered()
  start_back_click = new quit.is_clicking()
  join_click = new continue_btn.is_clicking()
  if start_back_hover OR join_hover then
    if NOT start_back_click AND NOT join_click then

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

    custom_mouse.mode = 1
else
  custom_mouse.mode = 2
endif
custom_mouse.mode = 0
endif
// Draw the mouse
custom_mouse.draw()
endif
endprocedure

public procedure show_info()
boarder_stats.draw(UI_BORDER_COLOR,None,None,False)
kill_count.draw(None,None,None,False,f"Kill count: {player_kill_count}")
players_alive.draw(None,None,None,False,f"Players alive: {player_count}")
endprocedure

public procedure show_bar(current, max_amount, bg_rect, color)
// draw bg
pygame.draw.rect(screen, UI_BG_COLOR, bg_rect)
// converting stat to pixel
ratio = current / max_amount
current_width = bg_rect.width * ratio
current_rect = new bg_rect.copy()
current_rect.width = current_width
// drawing the bar
pygame.draw.rect(screen, color, current_rect)
pygame.draw.rect(screen, UI_BORDER_COLOR, bg_rect, 3)
endprocedure

public procedure show_exp(exp)
text_surf = new font.render(str(int(exp)), False, TEXT_COLOR)
x = new screen.get_size()[0] - 20
y = new screen.get_size()[1] - 20
text_rect = new text_surf.get_rect(bottomright= new(x, y))
pygame.draw.rect(screen, UI_BG_COLOR, text_rect.inflate(20, 20))
screen.blit(text_surf, text_rect)
pygame.draw.rect(screen, UI_BORDER_COLOR, text_rect.inflate(20, 20), 3)
endprocedure

function selection_box(left, top, has_switted)
bg_rect = new pygame.Rect(left, top, ITEM_BOX_SIZE, ITEM_BOX_SIZE)
pygame.draw.rect(screen, UI_BG_COLOR, bg_rect)
if has_switted then
  pygame.draw.rect(screen, UI_BORDER_COLOR_ACTIVE, bg_rect, 3)
else
  pygame.draw.rect(screen, UI_BORDER_COLOR, bg_rect, 3)
endif
return bg_rect
endfunction

public procedure weapon_overlay(weapon_index, has_switted)
bg_rect = new selection_box(10, HEIGHT-ITEM_BOX_SIZE-10, has_switted)
weapon_surf = weapon_graphics[weapon_index]
weapon_rect = new weapon_surf.get_rect(center= newbg_rect.center)

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

screen.blit(weapon_surf, weapon_rect)
endprocedure

public procedure magic_overlay(magic_index, has_swapped)
  bg_rect = new selection_box(10+ITEM_BOX_SIZE+10, HEIGHT-ITEM_BOX_SIZE-10, has_swapped)
  magic_surf = magic_graphics[magic_index]
  magic_rect = new magic_surf.get_rect(center= newbg_rect.center)
  screen.blit(magic_surf, magic_rect)
endprocedure

public procedure display(player)
  show_bar(player.health, player.stats['health'], health_bar_rect, HEALTH_COLOR)
  show_bar(player.energy, player.stats['energy'], energy_bar_rect, ENERGY_COLOR)
  show_exp(player.exp)
  show_info()
  weapon_overlay(player.weapon_index, NOT player.can_switch_weapon)
  magic_overlay(player.magic_index, NOT player.can_switch_magic)
endprocedure
  
```

Weapon:

```

from Scripts.logger import *
try
  import pygame, string, random
  from Scripts.settings import *

except
  logger.critical("You do NOT have all the modules installed. Please install Pygame.")
endtry
logger.debug("RealDL Weapon Code.")

class Melee inherits pygame.sprite.Sprite
  private rect
  private image
  private full_path
  private damaged_player
  private damage
  private last_shot_time
  private type
  private time
  private player_id
  private id
  private settings
  private sprite_type

  public procedure new(player, groups, player_id)
    try
      super.new(groups)
      sprite_type = "weapon"
      settings = new Config()
      id = new create_id()
      player_id = player_id
      time = new pygame.time.get_ticks()
      type = player.weapon_type
      last_shot_time = 0 // Initialize the last shot time to 0
      damage = player.attack_strength + settings.weapon_data[player.weapon]['damage']
    
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

damaged_player = False
direction = new player.status.split('_')[0]
// Load the image
full_path = f'Graphics/Game/weapons/{player.weapon}/{direction}.png'
try
  image = new pygame.image.load(full_path).convert_alpha()
except
  image = new pygame.image.load(f"../{full_path}").convert_alpha()
endtry
// Set the placement of the sprite
if direction == 'right' then
  rect = new image.get_rect(midleft= newplayer.rect.midright + pygame.math.Vector2(0, 16))
endif
elseif direction == 'left' then
  rect = new image.get_rect(midright= newplayer.rect.midleft + pygame.math.Vector2(0, 16))
endif
elseif direction == 'down' then
  rect = new image.get_rect(midtop= newplayer.rect.midbottom + pygame.math.Vector2(-10, 0))
else
  rect = new image.get_rect(midbottom= newplayer.rect.midtop + pygame.math.Vector2(-10, 0))
endif
except
  logger.error("Failed to create Weapon")
endtry
endprocedure

function create_id()
try
  characters = string.ascii_letters + string.digits
  return ".join(random.choice(characters) for _ in range(settings.ID_STRING_LENGTH))"
except
  logger.error("Something went wrong with creating a unique ID.")
endtry
endfunction

class Weapon inherits pygame.sprite.Sprite
  private rect.y
  private rect.x
  private rect
  private image
  private damaged_player
  private damage
  private full_path
  private y
  private x
  private player_id
  private id
  private sprite_type

  public procedure new(groups, x, y, id, image, player_id, damage= new None, sprite_type= new "weapon_copy")
    super.new(groups)
    sprite_type = sprite_type
    id = id
    player_id = player_id
    x = x

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

y = y
full_path = image
damage = damage
damaged_player = False
try
  image = new pygame.image.load(full_path).convert_alpha()
except
  image = new pygame.image.load(f"../{full_path}").convert_alpha()
endtry
rect = new image.get_rect()
rect.x = x
rect.y = y
endprocedure

function return_image()
  if "../" in full_path then
    full_path = new full_path.replace("../", "")
  endif
  return full_path
endfunction

class Bullets inherits pygame.sprite.Sprite
  private rect.y
  private rect.x
  private rect
  private image
  private full_path
  private collided
  private cooldown
  private damaged_player
  private damage
  private speed
  private direction
  private player
  private player_group
  private player_id
  private id
  private sprite_type
  private obstacle_sprites
  private settings

  public procedure new(player, groups, obstacle_sprites, player_id, player_group)
    super.new(groups)
    settings = new Config()
    obstacle_sprites = obstacle_sprites
    sprite_type = "bullet"
    id = new create_id()
    player_id = player_id
    player_group = player_group
    player = player
    direction = new player.status.split('_')[0]
    gun = player.weapon
    speed = settings.weapon_data[gun]['speed']*settings.frame_increase_rate/1.5
    damage = player.attack_strength + settings.weapon_data[gun]['damage']
    damaged_player = False
  endprocedure
endclass
  
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

cooldown = settings.weapon_data[gun]['cooldown']
collided = False
full_path = settings.bullet_type[gun][direction]
try
  image = new pygame.image.load(full_path).convert_alpha()
except
  image = new pygame.image.load(f"../{full_path}").convert_alpha()
endtry
// Set the placement of the sprite
if direction == 'right' then
  rect = new image.get_rect(midleft= newplayer.rect.midright + pygame.math.Vector2(20, 8))
endif
elseif direction == 'left' then
  rect = new image.get_rect(midright= newplayer.rect.midleft + pygame.math.Vector2(-20, 8))
endif
elseif direction == 'down' then
  rect = new image.get_rect(midtop= newplayer.rect.midbottom + pygame.math.Vector2(-18, 21))
else
  rect = new image.get_rect(midbottom= newplayer.rect.midtop + pygame.math.Vector2(-18, -21))
endif
endprocedure

public procedure update(dt)
  collision('obstacle')
  collision('player')
  if NOT collided then
    if direction == 'right' then
      rect.x += speed * dt
    endif
    elseif direction == 'left' then
      rect.x -= speed * dt
    endif
    elseif direction == 'down' then
      rect.y += speed * dt
    else
      rect.y -= speed * dt
    endif
  endif
  if collided then
    kill()
  endif
endprocedure

function create_id()
  try
    characters = string.ascii_letters + string.digits
    return ".join(random.choice(characters) for _ in range(settings.ID_STRING_LENGTH))"
  except
    logger.error("Something went wrong with creating a unique ID.")
  endtry
endfunction

public procedure collision(collision_type)
  if collision_type == "obstacle" then
    collision_sprites = new pygame.sprite.spritecollide(obstacle_sprites,False)
    if collision_sprites then

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
    collided = True
  endif
endif
if collision_type == "player" then
  collision_sprites = new pygame.sprite.spritecollide(player_group,False)
  if collision_sprites then
    collided = True
  endif
endif
//if direction == 'horizontal':
for sprite in obstacle_sprites
  if sprite.hitbox.colliderect(rect) then
    collided = True
  endif
next sprite
for sprite in player_group
  if sprite.hitbox.colliderect(rect) then
    collided = True
  endif
next sprite
if direction == 'vertical' then
  for sprite in obstacle_sprites
    if sprite.hitbox.colliderect(rect) then
      collided = True
    endif
  next sprite
  for sprite in player_group
    if sprite.hitbox.colliderect(rect) then
      collided = True///
    endif
  next sprite
endif
// You may want to set its initial position and direction here
endprocedure
```

Python code:

Client:

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

from Scripts.logger import *
try:
    import pygame, sys, time
    from Scripts.network import Network
    from Scripts.settings import Config
    from Scripts.level import Level
    from Scripts.player import Player
    from Scripts.weapon import *
    from Scripts.encryption import *
    from Scripts.debug import debug
    from Scripts.functions import *
    from Scripts.ui import UI
    from Scripts.main_menu import MainMenu
    from Scripts.magic import MagicPlayer
    from Scripts.particles import AnimationPlayer
except:
    logger.critical("You do not have all the modules installed. Please install Pygame, RSA and Pycryptodome.")

logger.info("RealDL - Client Code")
pygame.init()

class Client(Config):
    def __init__(self):
        try:
            Config.__init__(self)
            self.join_game = False
            self.gameMenu = MainMenu()
        except:
            logger.error("Error, couldn't initialize the main menu.")
            self.close()

    def initialize_client(self, user_dict):
        logger.info("Client Connecting to Server")
        try:
            self.initialize_pygame(user_dict)
            self.initialize_network()
        except:
            logger.error("Error, couldn't initialize the client.")
            self.close()

    def initialize_pygame(self, user_dict):
        try:
            self.screen = self.gameMenu.screen # pygame.display.get_surface()
            self.clock = self.gameMenu.clock # pygame.time.Clock()
            self.custom_mouse = Mouse("Graphics/MainMenu/Mouse/mouse1.png", "Graphics/MainMenu/Mouse/mouse2.png", "Graphics/MainMenu/Mouse/mouse3.png")
            self.players = []
            self.bullets = []
            self.weapons = []
            self.magic_animations = []
            self.player_count = len(self.players)
            self.kill_count = 0
            self.level = Level()
        except:
            logger.error("Error, couldn't initialize the pygame module.")



```

```

# Sound and Music
try:
    self.sword_sound = pygame.mixer.Sound("Audio/sword.wav")
    self.damage_sound = pygame.mixer.Sound("Audio/hit.wav")
    self.heal_sound = pygame.mixer.Sound("Audio/heal.wav")
    self.fire_sound = pygame.mixer.Sound("Audio/fire.wav")
    self.death_sound = pygame.mixer.Sound("Audio/death.wav")
    self.background_music = pygame.mixer.Sound("Audio/main.ogg")
except:
    self.sword_sound = pygame.mixer.Sound("../Audio/sword.wav")
    self.damage_sound = pygame.mixer.Sound("../Audio/hit.wav")
    self.heal_sound = pygame.mixer.Sound("../Audio/heal.wav")
    self.fire_sound = pygame.mixer.Sound("../Audio/Fire.wav")
    self.death_sound = pygame.mixer.Sound("../Audio/death.wav")
    self.background_music = pygame.mixer.Sound("../Audio/main.ogg")

# particles
self.animation_player = AnimationPlayer()
self.magic_player = MagicPlayer(self.animation_player)
self.ui = UI(self.custom_mouse, self.close, self.player_count, self.kill_count)
self.running = True

# attack sprites
self.current_attack = None
self.bullet = None
self.attacked_sprites_ids = []

# User settings
settings = user_dict['settings']
start = user_dict['start']
self.movement = settings['control']['movement']
self.offense = settings['control']['offense']
self.magic = settings['control']['magic']
self.volume = settings['audio']['volume']
self.sound = settings['audio']['sound']
self.music = settings['audio']['music']
self.username = start['username']
self.server_ip = start['server_ip']
self.SERVER = self.server_ip
self.previous_time = 0
self.dt = 0

# Game Over
self.game_over = Text(self.gameMenu.text_height, "Graphics/Fonts/Orbitron-ExtraBold.ttf", (247,155,16), None, None, None, self.gameMenu.base_text_size*10)
self.game_over_board = Button((27,31,35), (27,31,35), self.gameMenu.settings.WIDTH/2, self.gameMenu.settings.HEIGHT/2*0.4, "Graphics/Fonts/Orbitron-Regular.ttf", (27,31,35), (27,31,35),
                             self.gameMenu.settings.WIDTH*0.55, self.gameMenu.settings.HEIGHT*0.2,'Rectangle', None, self.gameMenu.big_text_size, int(self.gameMenu.curve*1.5))

# Leaderboard
self.leaderboard = Button((27,31,35), (27,31,35), self.gameMenu.settings.WIDTH/2*1.78, self.gameMenu.settings.HEIGHT/2, "Graphics/Fonts/Orbitron-Regular.ttf", (27,31,35), (27,31,35),
                          self.gameMenu.settings.WIDTH*0.2, self.gameMenu.settings.HEIGHT*0.7,'Rectangle', None, self.gameMenu.big_text_size, int(self.gameMenu.curve*1.5))
self.leaderboard_text = Text(self.gameMenu.text_height, "Graphics/Fonts/Orbitron-Bold.ttf", (240,240,240), None, None, None, self.gameMenu.base_text_size*3)
self.leaderboard2 = Button((27,31,35), (27,31,35), self.gameMenu.settings.WIDTH/2*1.78, self.gameMenu.settings.HEIGHT/2*1.06, "Graphics/Fonts/Orbitron-Regular.ttf", (27,31,35), (27,31,35),
                           self.gameMenu.settings.WIDTH*0.18, self.gameMenu.settings.HEIGHT*0.6,'Rectangle', None, self.gameMenu.big_text_size, int(self.gameMenu.curve*1.5))

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
        self.gameMenu.settings.WIDTH*0.2, self.gameMenu.settings.HEIGHT*0.7,'','Rectangle', None, self.gameMenu.big_text_size, int(self.gameMenu.curve*1.5))
self.leaderboard_text = Text(self.gameMenu.text_height, "Graphics/FONTS/Orbitron-Bold.ttf", (240,240,240), None, None, None, self.gameMenu.base_text_size*3)
self.leaderboard2 = Button((27,31,35), (27,31,35), self.gameMenu.settings.WIDTH*1.78, self.gameMenu.settings.HEIGHT*2*1.06, "Graphics/FONTS/Orbitron-Regular.ttf", (27,31,35), (27,31,35),
                           self.gameMenu.settings.WIDTH*0.18, self.gameMenu.settings.HEIGHT*0.6,'','Rectangle', None, self.gameMenu.big_text_size, int(self.gameMenu.curve*1.5))
self.value_board = Text(self.gameMenu.text_height, "Graphics/FONTS/Orbitron-Medium.ttf", (240,240,240), None, None, None, self.gameMenu.base_text_size*3)

self.game_finished = False
except:
    logger.error("Couldn't correctly initialize pygame.")
    self.close()

def initialize_network(self):
    try:
        # Setup Network
        self.network = Network(self.SERVER, self.PORT)
        self.network.connect()

        # Get Public Key
        self.public_key = self.unserialize(self.network.receive(self.ENCRYPTION_DATA_SIZE))
        self.rsa_encrypt = RSA_Encryption(self.public_key)
        logger.info("Received Public Key: " + str(self.public_key))

        # Setup and send AES Encryption
        self.aes_key = AES_Keys(self.BITS)
        key = self.aes_key.export_key()
        dict_to_send_to_server = {'aes_key':key,'username':self.username}
        encrypted_key_dict = self.serialize(self.rsa_encrypt.encrypt(dict_to_send_to_server))
        self.network.send(encrypted_key_dict)
        logger.info("Sending AES KEY: " + str(key))

        # Receive Player
        self.encryption = AES_Encryption(key)
        data = self.encryption.decrypt(self.unserialize(self.network.receive(self.ENCRYPTION_DATA_SIZE)))
        player_info = data['player_data']['player']
        self.initialize_player(player_info)
        logger.info("Received player dict: " + str(player_info))
    except:
        logger.error("Error. Failed to connect to that Server IP Address.")
        self.close()

def initialize_player(self, player_info):
    self.player_x = player_info['x']
    self.player_y = player_info['y']
    self.player_image = player_info['image']
    self.username = player_info['username']
    self.id = player_info['id']
    self.status = player_info['status']
    self.health = player_info['health']
    self.kill_count = player_info['kill_count']
    self.killed_by = player_info['killed_by']
    self.player = Player([self.player_x, self.player_y], self.player_image, [self.level.visible_sprites], self.level.obstacle_sprites, self.username, self.id, self.health,
                        self.create_attack, self.destroy_attack, self.create_bullet, self.create_magic, self.movement, self.offense, self.magic, self.status)
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
def update_players(self, player_dict, dictionary_copy):
    # Update existing player instances and remove players that are not in player_dict
    try:
        ### Player Sounds ###

        for player_data, old_player_data in zip(player_dict.values(), dictionary_copy.values()):
            player_info = player_data['player']
            old_player_info = old_player_data['player']

            if player_info['id'] != self.player.id:
                ### Player Got Hit ###
                if player_info['health'] < old_player_info['health']:
                    if self.sound == 1:
                        self.damage_sound.play()

                ### Player dies ###
                if player_info['status'] == "dead" and old_player_info['status'] == "alive":
                    if self.sound == 1:
                        self.death_sound.play()

                ### Weapons ###
                weapon_ids = [weapon['player_id'] for weapon in player_data['weapon']]
                weapon_type = [weapon['type'] for weapon in player_data['weapon']]
                old_weapon_ids = [old_weapon['player_id'] for old_weapon in old_player_data['weapon']]
                new_weapon_ids = set(weapon_ids) - set(old_weapon_ids)
                if new_weapon_ids:
                    for new_weapon_id in new_weapon_ids:
                        weapon_index = weapon_ids.index(new_weapon_id)
                        animation_type = weapon_type[weapon_index]
                        if self.sound == 1:
                            if animation_type == "melee":
                                self.sword_sound.play()

                ### Bullets ###
                bullet_ids = [bullet['id'] for bullet in player_data['bullets']]
                old_bullet_ids = [old_bullet['id'] for old_bullet in old_player_data['bullets']]
                new_bullet_ids = set(bullet_ids) - set(old_bullet_ids)
                if new_bullet_ids:
                    for new_bullet_id in new_bullet_ids:
                        if self.sound == 1:
                            self.sword_sound.play()

                ### Fire Animation ###
                magic_ids = [magic['player_id'] for magic in player_data['magic']]
                magic_type = [magic['type'] for magic in player_data['magic']]
                old_magic_ids = [old_magic['player_id'] for old_magic in old_player_data['magic']]
                new_magic_ids = set(magic_ids) - set(old_magic_ids)
                if new_magic_ids:
                    for new_magic_id in new_magic_ids:
                        magic_index = magic_ids.index(new_magic_id)
                        animation_type = magic_type[magic_index]
                        if self.sound == 1:
                            if animation_type == "flame":
                                self.fire_sound.play()
                            if animation_type == "aura":
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
if animation_type == "aura":
    self.heal_sound.play()

### Players ###

existing_player_ids = [player.id for player in self.players]
players_to_remove = []

for player_data in player_dict.values():
    player_info = player_data['player']
    player_id = player_info['id']

    if player_id == self.player.id and self.kill_count < player_info['kill_count']:
        self.player.exp += 1000
        self.kill_count = player_info['kill_count']
        self.player.kill_count = player_info['kill_count']

    if player_id in existing_player_ids:
        # Update existing players
        for player in self.players:
            if player.id == player_id:
                player.kill_count = player_info['kill_count']
                player.rect.x = player_info['x']
                player.rect.y = player_info['y']
                player.health = player_info['health']
                player.status_alive = player_info['status']
                try: player.image = pygame.image.load(player_info['image']).convert_alpha()
                except: player.image = pygame.image.load(f"../{player_info['image']}").convert_alpha()

    else:
        #logger.debug(f"Creating new player instance with ID: {player_id}")
        new_player = Player(
            [player_info['x'], player_info['y']],
            player_info['image'],
            [self.level.visible_sprites, self.level.player_sprites],
            self.level.obstacle_sprites,
            player_info['username'],
            player_id,
            player_info['health']
        )
        self.players.append(new_player)

# Remove players not found in player_dict
for player in self.players:
    if player.id not in player_dict:
        #logger.debug(f"Removing player instance with id: {player.id}")
        players_to_remove.append(player)

for player in players_to_remove:
    self.players.remove(player)
    self.level.visible_sprites.remove(player)
    player.kill()
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
### Weapons ###

# Update Weapons
weapons_to_remove = []
for weapon in self.weapons:
    weapon_info = player_data['weapon']
    weapon_dict_ids = [weapon_dict['id'] for weapon_dict in weapon_info]
    weapon_dict_x = [weapon_dict['x'] for weapon_dict in weapon_info]
    weapon_dict_y = [weapon_dict['y'] for weapon_dict in weapon_info]
    weapon_dict_image = [weapon_dict['image'] for weapon_dict in weapon_info]
    if weapon.id in weapon_dict_ids:
        weapon_index = weapon_dict_ids.index(weapon.id)
        weapon.rect.x = weapon_dict_x[weapon_index]
        weapon.rect.y = weapon_dict_y[weapon_index]
        try: weapon.image = pygame.image.load(weapon_dict_image[weapon_index]).convert_alpha()
        except: weapon.image = pygame.image.load(f"../{weapon_dict_image[weapon_index]}").convert_alpha()
    else:
        #logger.debug(f"Removing weapon instance with id: {weapon.id}")
        weapons_to_remove.append(weapon)

# Delete weapons.
for weapon in weapons_to_remove:
    self.weapons.remove(weapon)
    self.level.visible_sprites.remove(weapon)
    weapon.kill()

# Create weapons
for player_data in player_dict.values():
    weapon_info = player_data['weapon']
    weapon_ids = [weapon.id for weapon in self.weapons]
    weapon_dict_ids = [weapon_dict['id'] for weapon_dict in weapon_info]
    for weapon_id, weapon_data in zip(weapon_dict_ids, weapon_info):
        if weapon_id not in weapon_ids:
            #logger.debug(f"Creating a new weapon with the ID: {weapon_data['id']}")
            new_weapon = Weapon([self.level.visible_sprites, self.level.attack_sprites],
                                weapon_data['x'],
                                weapon_data['y'],
                                weapon_data['id'],
                                weapon_data['image'],
                                weapon_data['player_id'],
                                weapon_data['damage'])
            self.weapons.append(new_weapon)
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
### Bullets ###

# Update Bullets
bullets_to_remove = []
for bullet in self.bullets:
    bullet_info = player_data['bullets']
    bullet_dict_ids = [bullet_dict['id'] for bullet_dict in bullet_info]
    bullet_dict_x = [bullet_dict['x'] for bullet_dict in bullet_info]
    bullet_dict_y = [bullet_dict['y'] for bullet_dict in bullet_info]
    bullet_dict_image = [bullet_dict['image'] for bullet_dict in bullet_info]
    if bullet.id in bullet_dict_ids:
        bullet_index = bullet_dict_ids.index(bullet.id)
        bullet.rect.x = bullet_dict_x[bullet_index]
        bullet.rect.y = bullet_dict_y[bullet_index]
        try: bullet.image = pygame.image.load(bullet_dict_image[bullet_index]).convert_alpha()
        except: bullet.image = pygame.image.load(f"../{bullet_dict_image[bullet_index]}").convert_alpha()
    else:
        #logger.debug(f"Removing bullet instance with id: {bullet.id}")
        bullets_to_remove.append(bullet)

# Delete Bullets
for bullet in bullets_to_remove:
    self.bullets.remove(bullet)
    self.level.visible_sprites.remove(bullet)
    bullet.kill()

# Create Bullets
for player_data in player_dict.values():
    bullet_info = player_data['bullets']
    bullet_ids = [bullet.id for bullet in self.bullets]
    bullet_dict_ids = [bullet_dict['id'] for bullet_dict in bullet_info]
    for bullet_id, bullet_data in zip(bullet_dict_ids, bullet_info):
        if bullet_id not in bullet_ids:
            #logger.debug(f"Creating a new weapon with the ID: {bullet_data['id']}")
            new_bullet = Weapon([self.level.visible_sprites, self.level.attack_sprites],
                                bullet_data['x'],
                                bullet_data['y'],
                                bullet_data['id'],
                                bullet_data['image'],
                                bullet_data['player_id'],
                                bullet_data['damage'],
                                "bullet_copy")
            self.bullets.append(new_bullet)
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
def update_players(self, player_dict, dictionary_copy):
    """ Magic """
    # Update Magic Animation

    magic_to_remove = []
    for magic in self.magic_animations:
        magic_info = player_data['magic']
        magic_dict_ids = [magic_dict['id'] for magic_dict in magic_info]
        magic_dict_x = [magic_dict['x'] for magic_dict in magic_info]
        magic_dict_y = [magic_dict['y'] for magic_dict in magic_info]
        magic_dict_image = [magic_dict['image'] for magic_dict in magic_info]
        if magic.id in magic_dict_ids:
            magic_index = magic_dict_ids.index(magic.id)
            magic.rect.x = magic_dict_x[magic_index]
            magic.rect.y = magic_dict_y[magic_index]
            magic.full_path = magic_dict_image[magic_index]

            try: magic.image = pygame.image.load(magic.full_path).convert_alpha()
            except: magic.image = pygame.image.load(f"../{magic.full_path}").convert_alpha()

        else:
            #logger.debug(f"Removing magic instance with id: {magic.id}")
            magic_to_remove.append(magic)

    # Delete Magic
    for magic in magic_to_remove:
        self.magic_animations.remove(magic)
        self.level.visible_sprites.remove(magic)
        magic.kill()

    # Create Magic Animation
    for player_data in player_dict.values():
        magic_info = player_data['magic']
        magic_ids = [magic.id for magic in self.magic_animations]
        magic_dict_ids = [magic_dict['id'] for magic_dict in magic_info]
        for magic_id, magic_data in zip(magic_dict_ids, magic_info):
            if magic_id not in magic_ids:
                #logger.debug(f"Creating a new Magic Animation with the ID: {magic_data['id']}")
                if magic_data['type'] == "flame":
                    new_magic_animation = Weapon([self.level.visible_sprites, self.level.attack_sprites],
                                                magic_data['x'],
                                                magic_data['y'],
                                                magic_data['id'],
                                                magic_data['image'],
                                                magic_data['player_id'],
                                                magic_data['damage'],
                                                "magic_copy")
                else:
                    new_magic_animation = Weapon([self.level.visible_sprites],
                                                magic_data['x'],
                                                magic_data['y'],
                                                magic_data['id'],
                                                magic_data['image'],
                                                magic_data['player_id'],
                                                None,
                                                "magic_copy")
                self.magic_animations.append(new_magic_animation)
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
except:  
    logger.error("Player disconnected.")  
    self.close()  
  
def players_number(self):  
    player_count = 0  
    player_list = [player.status_alive for player in self.players]  
    for status in player_list:  
        if status == "alive":  
            player_count += 1  
    return player_count  
  
def redraw_window(self, all_players_dict, dictionary_copy):  
    try:  
        self.update_players(all_players_dict, dictionary_copy) # could be this  
        self.level.run(self.player, self.dt)  
        """self.player_attack_logic() # could be this"""  
        self.player_attack_collisions()  
        self.ui.player_count = self.players_number()  
        self.ui.player_kill_count = self.kill_count  
        self.ui.display(self.player)  
        self.one_player_remaining()  
  
        debug(f"Position: ({self.player.rect.x}, {self.player.rect.y})", self.WIDTH/2, 20)  
        if self.dt != 0: debug(f"FPS: {int(1/self.dt)}", self.WIDTH/2, 50)  
        debug(f"{self.player.direction}", self.WIDTH/2, 80)  
        self.ui.draw_menu()  
        if self.ui.draw_ui: self.player.paused = True  
        else: self.player.paused = False  
        pygame.display.update()  
        # self.clock.tick(5)  
    except:  
        logger.error("Player has left the game.")  
        self.close()  
  
def create_attack(self, weapon_type="melee"):  
    #logger.info("Create attack")  
    try:  
        if self.sound == 1 and weapon_type == "melee":  
            self.sword_sound.play()  
            self.current_attack = Melee(self.player, [self.level.visible_sprites], self.player.id)  
    except:  
        self.current_attack = None  
        #logger.error("Failed to render attack image.")  
  
def create_bullet(self):  
    #logger.info("Bullet!")  
    if self.sound == 1:  
        self.sword_sound.play()  
    self.bullet = Bullets(self.player, [self.level.visible_sprites], self.level.obstacle_sprites, self.player.id, self.level.player_sprites)
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
def destroy_attack(self):
    #logger.info("destroy attack.")
    try:
        if self.current_attack:
            self.current_attack.kill()
        self.current_attack = None
    except:
        self.current_attack = None
        #logger.error("Failed to destroy attack.")

def create_magic(self,style,strength,cost):
    if style == 'heal':
        if self.sound == 1 and self.player.energy > cost:
            self.heal_sound.play()
            self.magic_player.heal(self.player,strength,cost,[self.level.visible_sprites], self.player.id)

    if style == 'flame':
        if self.sound == 1 and self.player.energy > cost:
            self.fire_sound.play()
            self.magic_player.flame(self.player,strength,cost,[self.level.visible_sprites], self.player.id)

def close(self):
    try:
        self.network.close()
    except:
        logger.exception("No server-client connection.")
    self.background_music.stop()
    self.running = False
    self.player = None
    self.players = None
    self.gameMenu.restart_menu()
    self.run()

def quit(self):
    self.running = False
    pygame.quit()
    sys.exit()

def return_weapon(self):
    weapon = []
    for sprite in self.level.visible_sprites:
        if sprite.sprite_type == "weapon":
            weapon_dict = {"x":sprite.rect.x,
                           "y":sprite.rect.y,
                           "image":sprite.full_path,
                           "damage":sprite.damage,
                           "type":sprite.type,
                           "id":sprite.id,
                           "player_id":self.player.id}
            weapon.append(weapon_dict)
    return weapon
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
def return_bullets(self):
    bullets = []

    for sprite in self.level.visible_sprites:
        if sprite.sprite_type == "bullet":
            bullet_dict = {"x":sprite.rect.x,
                           "y":sprite.rect.y,
                           "image":sprite.full_path,
                           "damage":sprite.damage,
                           "id":sprite.id,
                           "player_id":self.player.id}
            bullets.append(bullet_dict)
    return bullets

def return_magic_data(self):
    magic = []
    for sprite in self.level.visible_sprites:
        if sprite.sprite_type == "magic":
            magic_dict = {"x":sprite.rect.x,
                           "y":sprite.rect.y,
                           "image":sprite.return_image(),
                           "type":sprite.return_type(),
                           "damage":sprite.return_strength(),
                           "id":sprite.id,
                           "player_id":self.player.id}
            magic.append(magic_dict)
    return magic

def player_attack_collisions(self):
    # Removes attack sprite id that are no longer in the game
    sprite_ids = [sprite.id for sprite in self.level.visible_sprites]
    for sprite_id in self.attacked_sprites_ids:
        if sprite_id not in sprite_ids:
            self.attacked_sprites_ids.remove(sprite_id)

    if self.level.attack_sprites:
        for attack_sprite in self.level.attack_sprites:
            if "copy" in attack_sprite.sprite_type:
                # Checks if the attack is not from our player
                if attack_sprite.player_id != self.player.id:
                    player_collisions = pygame.sprite.spritecollide(attack_sprite,self.level.player_sprites,False)
                    if player_collisions:
                        for player in player_collisions:
                            if attack_sprite.damaged_player != True and attack_sprite.id not in self.attacked_sprites_ids:
                                if player.id == self.player.id and self.player.health > 0:
                                    self.attacked_sprites_ids.append(attack_sprite.id)
                                    attack_sprite.damaged_player = True
                                    old_player_health = self.player.health
                                    if self.sound == 1:
                                        self.damage_sound.play()
                                    self.player.damage_player(attack_sprite.damage)
                                    if self.player.health == 0 and old_player_health > 0 and self.killed_by == '':
                                        if self.sound == 1:
                                            self.death_sound.play()
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

if self.sound == 1:
    self.death_sound.play()
self.killed_by = attack_sprite.player_id
if "bullet" in attack_sprite.sprite_type:
    attack_sprite.kill()

def main(self):
    dictionary_copy = {}
    self.background_music.play(-1)
    if self.music == 1:
        self.background_music.set_volume(self.volume/100)
    else:
        self.background_music.set_volume(0)
    try:
        self.previous_time = time.time()
        while self.running:
            current_time = time.time()
            self.dt = current_time - self.previous_time
            self.previous_time = current_time

            self.handle_events()

        try:
            all_players_dict = self.send_player_data()

            #logger.debug(f"Sending player data: {all_players_dict}")
            #logger.debug(f"Received players dictionary: {all_players_dict}")
            self.redraw_window(all_players_dict, dictionary_copy)
            dictionary_copy = all_players_dict
        except Exception as e:
            logger.error(f"Failed to send/receive player data: {e}")
            self.close()

    except Exception as e:
        logger.error(f"Failed to run main loop: {e}")
        self.close()
    
```

```

def one_player_remaining(self):
    if self.players_number() == 1 or self.game_finished:
        ## Player has won.
        self.game_finished = True

    ## Animation
    self.game_over_board.draw((27,31,35), None, None, True, None, None, None, "draw")
    self.game_over.draw("draw", "Game Over!", (self.gameMenu.settings.WIDTH/2), (self.gameMenu.settings.HEIGHT/2*0.4))

    ## Display kill counts for all players
    self.leaderboard.draw((27,31,35))
    self.leaderboard2.draw((240,240,240))
    self.leaderboard_text = "\n".join(f"({player.username}): {player.kill_count}" for player in sorted(self.players, key=lambda x: x.kill_count, reverse=True))
    leaderboard_list = leaderboard_text.split("\n")
    for index, leaderboard_index in enumerate(leaderboard_list):
        spacing_index = 0.52 - 0.1*index
        self.value_board.draw("draw", leaderboard_index, (self.gameMenu.settings.WIDTH/2*1.78), (self.gameMenu.settings.HEIGHT/2*0.375))

    ## Reset key player data
    ## Log out after a 30 second.
else:
    self.game_over.draw("undraw", "Game Over!", (self.gameMenu.settings.WIDTH/2), (self.gameMenu.settings.HEIGHT/2*0.4))

def handle_events(self):
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            self.quit()
        elif event.type == pygame.KEYDOWN and event.key == pygame.K_ESCAPE:
            self.ui.draw_ui = not self.ui.draw_ui

def send_player_data(self):
    player_dict = {
        'x': self.player.rect.x,
        'y': self.player.rect.y,
        'image': self.player.get_image_name(),
        'username': self.username,
        'status': self.player.return_alive_status(),
        'health': self.player.health,
        'id': self.id,
        'kill_count': self.kill_count,
        'killed_by': self.killed_by
    }
    player_total_dict = {
        'player': player_dict,
        'weapon': self.return_weapon(),
        'bullets': self.return_bullets(),
        'magic': self.return_magic_data()
    }
    player_encrypted_dict = self.serialize(self.verification.encrypt(player_total_dict))
    self.network.send(player_encrypted_dict)
    all_players_dict = self.verification.decrypt(self.unserialize(self.network.receive(self.DATA_SIZE)))
    return all_players_dict
    
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

def run(self):
    self.join_game = False
    self.gameMenu.run()
    user_dict = self.gameMenu.start_game()
    self.initialize_client(user_dict)
    self.loading_zone()
    self.main()

def loading_zone(self):
    def redraw_window():
        self.screen.fill((27,31,35))
        join_btn.draw((240,240,240),join_the_game)
        players.draw("draw", f'{connections} {"player" if connections==1 else "players"} {"is" if connections==1 else "are"} connected.', (self.WIDTH / 2), (self.HEIGHT / 2) * 0.4)
        join_hover = join_btn.is_hovered()
        join_click = join_btn.is_clicking()

        if join_hover:
            if not join_click:
                mouse.mode = 1
            else:
                mouse.mode = 2
        else:
            mouse.mode = 0
        mouse.draw()
        pygame.display.update()

    def join_the_game():
        self.join_game = True
        # Receive number of players
        running = True
        mouse = Mouse("Graphics/MainMenu/Mouse/mouse1.png", "Graphics/MainMenu/Mouse/mouse2.png", "Graphics/MainMenu/Mouse/mouse3.png")
        self.width_ratio = self.WIDTH / 1920
        self.height_ratio = self.HEIGHT / 1080

        # Variables
        players = Text(self.gameMenu.text_height, "Graphics/Fonts/Orbitron-Regular.ttf", (240, 240, 240), None, None, None, int(self.gameMenu.base_text_size * 3))
        join_btn = Button((99, 174, 96), (27,31,35), (self.gameMenu.settings.WIDTH/2), (self.gameMenu.settings.HEIGHT/2)*1.4-self.gameMenu.button_padding, "Graphics/Fonts/Orbitron-Medium.ttf",
                          [(27,31,35), (39, 174, 96), self.gameMenu.base_button_width, self.gameMenu.base_button_height,'Start','Rectangle', None, int(self.gameMenu.big_text_size/1.3), self.gameMenu.curve])
        connections = 0

    while running:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                self.quit()
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_ESCAPE:
                    self.quit()

        try:
            connected_players = self.verification.decrypt(self.unserialize(self.network.receive(self.DATA_SIZE)))
            connections = connected_players['connections']
            started = connected_players['started']
            if self.join_game and connections < 2:
                self.join_game = False
            if started:
                self.join_game = started
            .....
        
```



```

while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            self.quit()
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
                self.quit()

    try:
        connected_players = self.verification.decrypt(self.unserialize(self.network.receive(self.DATA_SIZE)))
        connections = connected_players['connections']
        started = connected_players['started']
        if self.join_game and connections < 2:
            self.join_game = False
        if started:
            self.join_game = started
        game_dict = {"ready":self.join_game}
        start_the_game = self.serialize(self.verification.encrypt(game_dict))
        self.network.send(start_the_game)
        if self.join_game or started:
            self.join_game = True
            running = False

    except:
        running = False
        logger.error("Something failed.")
        self.close()

    redraw_window()

if __name__ == "__main__":
    try:
        client = Client()
        client.run()
    except:
        logger.error("Couldn't run main menu or client.")

```

Debug:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
from Scripts.logger import *
try:
    import pygame
except:
    logger.critical("You do not have all the modules installed. Please install Pygame.")
font = pygame.font.Font(None, 30)

logger.debug("RealDL Debug Code.")

def debug(info, x=10, y=10):
    display_surface = pygame.display.get_surface()
    debug_surf = font.render(str(info), True, 'White')
    debug_rect = debug_surf.get_rect(center=(x, y))
    pygame.draw.rect(display_surface, 'Black', debug_rect)
    display_surface.blit(debug_surf, debug_rect)
```

Encryption:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
from Scripts.logger import *
try:
    import rsa, pickle, os
    from Crypto.Cipher import AES
    from Crypto.Util.Padding import pad, unpad
    from Crypto.Random import get_random_bytes
except:
    logger.critical("You do not have all the modules installed. Please install RSA and Pycryptodome.")

logger.info('RealDL Encryption Module : RSA')

class RSA_Keys:
    def __init__(self, bits):
        self.bits = bits
        key_dir = "../Keys" if os.path.exists("../Keys") else "Keys"
        public_key_path = os.path.join(key_dir, "public.pem")
        private_key_path = os.path.join(key_dir, "private.pem")

    try:
        with open(public_key_path, "rb") as f:
            self.public_key = rsa.PublicKey.load_pkcs1(f.read())

        with open(private_key_path, "rb") as f:
            self.private_key = rsa.PrivateKey.load_pkcs1(f.read())
    except FileNotFoundError:
        self.public_key, self.private_key = rsa.newkeys(self.bits)
        with open(public_key_path, "wb") as f:
            f.write(self.public_key.save_pkcs1("PEM"))

        with open(private_key_path, "wb") as f:
            f.write(self.private_key.save_pkcs1("PEM"))

    def export_keys(self):
        return self.public_key, self.private_key

class RSA_Encryption:
    def __init__(self, public_key):
        try:
            self.public_key = public_key
        except ValueError as e:
            logging.error(f"Error: {e}")

    def encrypt(self, message):
        try:
            return rsa.encrypt(self.serialise(message), self.public_key)
        except ValueError as e:
            logging.error(f"Error: {e}")

    def decrypt(self, encrypted_message, private_key):
        try:
            encoded_message = rsa.decrypt(encrypted_message, private_key)
            return self.unserialise(encoded_message)
        except ValueError as e:
            logging.error(f"Error: {e}")
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
def serialise(self, data):
    try:
        return pickle.dumps(data)
    except ValueError as e:
        logging.error(f"Error: {e}")

def unserialise(self, data):
    try:
        return pickle.loads(data)
    except ValueError as e:
        logging.error(f"Error: {e}")

class AES_Keys:
    def __init__(self, bits):
        self.bytes = bits // 8
        self.key = get_random_bytes(self.bytes)

    def export_key(self):
        return self.key

class AES_Encryption:
    def __init__(self, key):
        try:
            self.key = key
            self.cipher = AES.new(self.key, AES.MODE_ECB)
        except ValueError as e:
            logging.error(f"Error: {e}")

    def encrypt(self, message):
        try:
            message_bytes = self.serialise(message)
            padded_message = pad(message_bytes, AES.block_size)
            return self.cipher.encrypt(padded_message)
        except ValueError as e:
            logging.error(f"Error: {e}")

    def decrypt(self, encrypted_message):
        try:
            decrypted_padded_message = self.cipher.decrypt(encrypted_message)
            decrypted_unpadded_bytes_message = unpad(decrypted_padded_message, AES.block_size)
            return self.unserialise(decrypted_unpadded_bytes_message)
        except ValueError as e:
            logging.error(f"Error: {e}")

    def serialise(self, data):
        try:
            return pickle.dumps(data)
        except ValueError as e:
            logging.error(f"Error: {e}")

    def unserialise(self, data):
        try:
            return pickle.loads(data)
        except ValueError as e:
            logging.error(f"Error: {e}")
```

Functions:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
from Scripts.logger import *
try:
    import pygame, webbrowser
except:
    logger.critical("You do not have all the modules installed. Please install Pygame.")

logger.info("Pygame Functions 0.0.1 (Python 3.11.0)\nI am not affiliated with pygame. https://github.com/TheRealDL1/pygame\_functions")

class Images(object):
    def __init__(self, image):
        self.image = image
        self.screen = pygame.display.get_surface()
        self.load_image = self.load_image_from_file()
        self.rect = self.load_image.get_rect()

    def load_image_from_file(self):
        try:
            return pygame.image.load(self.image).convert_alpha()
        except:
            return pygame.image.load(f"../{self.image}").convert_alpha()

    def display_icon(self):
        pygame.display.set_icon(self.load_image)

    def draw(self, x=0, y=0):
        self.screen.blit(self.load_image, (x,y))

    def resize(self, width, height):
        self.load_image = pygame.transform.scale(self.load_image, (width, height))

class Mouse(object):
    def __init__(self,mouse_image1, mouse_image2, mouse_image3):
        pygame.mouse.set_visible(False)
        self.screen = pygame.display.get_surface()
        self.mode = 0
        self.mouse_mode1 = Images(mouse_image1)
        self.mouse_mode2 = Images(mouse_image2)
        self.mouse_mode3 = Images(mouse_image3)

    def draw(self):
        # Get the current mouse position
        mouse_x, mouse_y = pygame.mouse.get_pos()

        # Draw a green square at the mouse cursor position
        if self.mode == 0:
            self.mouse_mode1.draw(mouse_x , mouse_y)
        elif self.mode == 1:
            self.mouse_mode2.draw(mouse_x , mouse_y)
        else:
            self.mouse_mode3.draw(mouse_x , mouse_y)
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
class WordButton(object):
    def __init__(self, x, y, text, color1, color2, basefont, largefont, textSize=30,):
        """Sets the values for button"""
        self.color = color1
        self.color2 = color2
        self.x = x
        self.y = y
        self.text = text
        self.textSize = textSize
        self.largeSize = round(self.textSize * 1.25)
        self.basefont = basefont
        self.largefont = largefont
        try:
            self.base_font = pygame.font.Font(self.basefont, self.textSize)
            self.large_font = pygame.font.Font(self.largefont, self.largeSize)
        except:
            self.base_font = pygame.font.Font(f'../{self.basefont}', self.textSize)
            self.large_font = pygame.font.Font(f'../{self.largefont}', self.largeSize)
        self.hover = False
        self.click = False
        self.release = False

    def displayText(self, win, newText=None, action=None, link=None):
        if newText != None:
            self.text = newText
        mouse = pygame.mouse.get_pos()
        click = pygame.mouse.get_pressed()

        # Render text with both fonts
        text = self.large_font.render(self.text, 1, self.color2)
        text2 = self.base_font.render(self.text, 1, self.color)

        # Check if mouse is over the text
        if self.x + text.get_width() / 2 > mouse[0] > self.x - text.get_width() / 2 and self.y + text.get_height() / 2 > mouse[1] > self.y - text.get_height() / 2:
            self.hover = True
            # Render text with larger font and lighter color
            text = self.large_font.render(self.text, 1, self.color2)

        # Center text vertically as well as horizontally
        win.blit(text, (self.x - text.get_width() / 2, self.y - text.get_height() / 2))

        # Check for mouse click and run action if specified
        if click[0] == 1:
            self.click = True

        if self.click == True and not click[0]:
            self.release = True

        if self.release:
            if action != None:
                action()
            if link != None:
                webbrowser.open(link)
            self.release = False
            self.click = False
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
        else:
            self.hover = False
            self.click = False
            self.release = False
            # Render text with base font and color
            text2 = self.base_font.render(self.text, 1, self.color)

            # Center text vertically as well as horizontally
            win.blit(text2, (self.x - text2.get_width() / 2, self.y - text2.get_height() / 2))

class Animation(object):
    def __init__(self, image, scale, scale_direction, scale_min, scale_max, scale_speed, screen_y, screen_x):
        self.scale = scale #1.0
        self.image = image
        self.scale_direction = scale_direction #-1.35 # Start by zooming out
        self.scale_min = scale_min #0.9
        self.scale_max = scale_max #1.2
        self.scale_speed = scale_speed #0.01 # The rate at which the scale changes
        self.screen_center_x = screen_x
        self.screen_center_y = screen_y #324#win.get_height() // 2
        self.monopoly_rect = self.image.get_rect(center=(self.screen_center_x, self.screen_center_y))

    def animation(self, win):
        self.scale += self.scale_direction * self.scale_speed
        if self.scale <= self.scale_min:
            if self.scale_direction < 0:
                self.scale_direction = self.scale_direction * -1
            else:
                self.scale_direction = self.scale_direction * 1# Start zooming in
        elif self.scale >= self.scale_max:
            if self.scale_direction > 0:
                self.scale_direction = self.scale_direction * -1
            else:
                self.scale_direction = self.scale_direction * 1 # Start zooming out

        # Scale the image and get its rect
        scaled_monopoly = pygame.transform.rotozoom(self.image, 0, self.scale)
        scaled_monopoly_rect = scaled_monopoly.get_rect(center=(self.screen_center_x, self.screen_center_y))

        # Blit the scaled image to the screen and update the display
        win.blit(scaled_monopoly, scaled_monopoly_rect)

class TextBox(object):
    def __init__(self, width, height, x, y, color1, color2, textfont, curve, thickness, maxTextWidth, text_size=50):
        self.text = ''
        self.textSize = text_size
        self.textfont = textfont
        try:
            self.base_font = pygame.font.Font(self.textfont, self.textSize)
        except:
            self.base_font = pygame.font.Font(f"../{self.textfont}", self.textSize)
        self.active = False
        self.width = width
        self.height = height
        self.x = x - self.width / 2
        self.oldWidth = width
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
self.y = y
self.color_active = pygame.Color(color1)
self.color_passive = pygame.Color(color2)
self.color = self.color_passive
self.last_update = 0 # time of last update
self.show_cursor = False # whether to show cursor or not
self.cursor = 0
self.maxTextWidth = maxTextWidth
self.curve = curve
self.thickness = thickness
self.screen = pygame.display.get_surface()
self.hover = False
self.click = False

def draw(self):
    area = [self.x, self.y, self.width, self.height]
    pygame.draw.rect(self.screen, self.color, area, self.thickness, self.curve)

def is_hovered(self):
    mouse = pygame.mouse.get_pos()
    return self.x + self.width > mouse[0] > self.x and self.y + self.height > mouse[1] > self.y

def is_clicking(self):
    mouse = pygame.mouse.get_pos()
    click = pygame.mouse.get_pressed()
    return self.x + self.width > mouse[0] > self.x and self.y + self.height > mouse[1] > self.y and click[0]

def checkTextBox(self):
    mouse = pygame.mouse.get_pos()
    click = pygame.mouse.get_pressed()

    if self.x + self.width > mouse[0] > self.x and self.y + self.height > mouse[1] > self.y:
        self.hover = True
        if click[0]: self.click = True
        if self.click and not click[0]: self.click = False
        if self.active: self.active = False
        else: self.active = True

    else:
        self.hover = False
        self.active = False
        self.click = False

    if self.active:
        self.color = self.color_active
    else:
        self.color = self.color_passive

def update(self, color=(240,240,240),x_pos=None, y_pos=None):
    middleOfX = self.screen.get_width() // 2
    if x_pos: self.x = x_pos
    if y_pos: self.y = y_pos

    if self.text:
        surface_area = self.base_font.render(self.text, True, color)
        text_width = surface_area.get_width() + 20
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
if self.ogWidth > self.width:
    self.width = self.ogWidth
    self.x = middleOfX - self.width // 2
if text_width > self.width:
    self.width = text_width
    self.x = middleOfX - self.width // 2
elif self.width > text_width:
    if self.ogWidth < self.width:
        self.width = text_width
        self.x = middleOfX - self.width // 2
    self.screen.blit(surface_area, (self.x + 5, self.y + (self.height // 2 - surface_area.get_height() // 2)))
else:
    self.width = self.ogWidth
    self.x = middleOfX - self.width // 2
surface_area = self.base_font.render(self.text, True, color)
self.screen.blit(surface_area, (self.x + 5, self.y + (self.height // 2 - surface_area.get_height() // 2)))

time_since_last_update = pygame.time.get_ticks() - self.last_update
if self.active and time_since_last_update > 500:
    self.show_cursor = not self.show_cursor
    self.last_update = pygame.time.get_ticks()

if self.show_cursor and self.active == True:
    cursor_width = 2
    cursor_pos = self.base_font.size(self.text[:self.cursor])[0] - 5
    text_to_show = self.text[:self.cursor] + '|' + self.text[self.cursor:]
    cursor_pos += self.base_font.size(' ')[0] * 0.8
    surface_area = self.base_font.render(text_to_show, True, color)
    cursor_area = pygame.Surface((cursor_width, surface_area.get_height()))
    cursor_area.fill(color)
    self.screen.blit(cursor_area, (self.x + cursor_pos, self.y + (self.height // 2 - cursor_area.get_height() // 2)), (0, 0, cursor_width, cursor_area.get_height()))
else:
    #surface_area = self.base_font.render(self.text, True, color)
    #self.screen.blit(surface_area, (self.x + 5, self.y + (self.height // 2 - surface_area.get_height() // 2)))

def updateText(self, events, color=(0,0,0), function=None):
    surface_area = self.base_font.render(self.text, True, color)
    text_width = surface_area.get_width() + 20
    for event in events:
        if event.type == pygame.MOUSEBUTTONDOWN:
            self.checkTextBox()
        if event.type == pygame.KEYDOWN:
            if self.active == True:
                if event.key == pygame.K_BACKSPACE:
                    if self.cursor > 0:
                        self.text = self.text[:self.cursor-1] + self.text[self.cursor:]
                        self.cursor -= 1
                elif event.key == pygame.K_DELETE:
                    self.text = self.text[:self.cursor] + self.text[self.cursor+1:]
                elif event.key == pygame.K_RETURN:
                    if function != None:
                        function()

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
        elif event.key == pygame.K_LEFT:
            if self.cursor > 0:
                self.cursor -= 1
        elif event.key == pygame.K_RIGHT:
            if self.cursor < len(self.text):
                self.cursor += 1
        else:
            if text_width < self.maxTextWidth and event.key is not pygame.K_TAB:
                new_text = self.text[:self.cursor] + event.unicode + self.text[self.cursor:]
                if len(new_text) > len(self.text):
                    self.text = new_text
                self.cursor += 1

    def return_text(self):
        if self.text != "":
            return self.text
        else:
            return None

class Text(object):
    def __init__(self, height, baseFont, color=(0,0,0), x=None, y=None, text=None, textSize=32):
        self.text = text
        self.x = x
        self.y = y
        self.baseFont = baseFont
        self.height = height
        self.textSize = textSize
        self.color = color
        self.screen = pygame.display.get_surface()
        try:
            self.base_font = pygame.font.Font(self.baseFont, self.textSize)
        except:
            self.base_font = pygame.font.Font(f"../{self.baseFont}", self.textSize)
        self.hover = False
        self.click = False
        self.release = False

        # Animation
        self.stop_start_animation = True
        self.animation_time = 150
        self.animation_speed = 13
        self.num_frames = 60
        self.time = 0
        self.time2 = 0

        # Opacity
        self.opacitya = 0
        self.opacityb = 255
        self.opacity_color1 = 0
        self.opacity_color2 = 0
        self.opacity_color = 0
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
def draw(self, mode="draw", new_text=None, new_x=None, new_y=None, function= None, link=None):
    if new_text: self.text = new_text
    if new_x: self.x = new_x
    if new_y: self.y = new_y

    if mode == "normal":
        surface_area = self.base_font.render(self.text, 1, self.color)
        self.screen.blit(surface_area, (self.x - surface_area.get_width() / 2, self.y - surface_area.get_height() / 2))

    if mode == "draw":
        self.stop_start_animation = False
        self.time = pygame.time.get_ticks()
        step = (self.time - self.time2) / (self.animation_time / self.animation_speed)
        self.opacity_color1 = self.change_opacity(self.opacity_color2, self.opacityb, step)
        self.opacity_color = self.opacity_color1

    if mode == "undraw":
        self.time2 = pygame.time.get_ticks()
        step = (self.time2 - self.time) / (self.animation_time / self.animation_speed)
        self.opacity_color2 = self.change_opacity(self.opacity_color1, self.opacitya, step)
        self.opacity_color = self.opacity_color2

    if not self.stop_start_animation:
        surface_area = self.base_font.render(self.text, 1, self.color)
        surface_area.set_alpha(self.opacity_color)
        self.screen.blit(surface_area, (self.x - surface_area.get_width() / 2, self.y - surface_area.get_height() / 2))

    if self.is_hovered(): self.hover = True
    if self.is_clicking(): self.click = True
    if self.click and not self.is_clicking(): self.release = True

    if self.release:
        if function: function()
        if link: webbrowser.open(link)

        self.release = False
        self.click = False

    if not self.is_hovered:
        self.hover = False
        self.click = False
        self.release = False

def change_opacity(self, original_opacity, new_opacity, step_multiplier=0):
    new_opacity_return = None
    difference = new_opacity - original_opacity
    opacity_step = difference / self.num_frames
    if original_opacity > new_opacity:
        if int(original_opacity + opacity_step*step_multiplier) < new_opacity:
            new_opacity_return = new_opacity
        else:
            new_opacity_return = int(original_opacity + opacity_step*step_multiplier)
    else:
        if int(original_opacity + opacity_step*step_multiplier) > new_opacity:
            new_opacity_return = new_opacity
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
        else:
            new_opacity_return = int(original_opacity + opacity_step*step_multiplier)

        return new_opacity_return

    def is_hovered(self):
        mouse_pos = pygame.mouse.get_pos()
        text_rect = self.base_font.render(self.text, 1, self.color).get_rect()
        text_rect.center = (self.x, self.y)
        return text_rect.collidepoint(mouse_pos)

    def is_clicking(self):
        mouse_pos = pygame.mouse.get_pos()
        mouse_buttons = pygame.mouse.get_pressed()
        text_rect = self.base_font.render(self.text, 1, self.color).get_rect()
        text_rect.center = (self.x, self.y)
        return text_rect.collidepoint(mouse_pos) and mouse_buttons[0] == 1

class Button(object):
    """A class for all buttons"""
    def __init__(self, color, color2, x, y, basefont, text_color=(0,0,0), text_color2=(0,0,0), width=None,
                 height=None, text='', buttonType='Rectangle', radius=None, textSize=30, curve=10, image=""):
        """Sets the values for button"""
        self.screen = pygame.display.get_surface()
        try:
            self.x = x - width/2
            self.y = y - height/2
            self.width = width
            self.height = height
        except:
            self.x = x
            self.y = y

        if image != "":
            self.image = Images(image)
            if height and width:
                self.image.resize(width,height)
        self.text = text
        self.buttonType = buttonType
        self.radius = radius
        self.textSize = textSize
        self.curve = curve
        #Mouse clicking
        self.hover = False
        self.click = False
        self.release = False
        # Animation
        self.stop_start_animation = True
        self.animation_time = 150
        self.animation_speed = 13
        self.num_frames = 60
        self.time = 0
        self.time2 = 0
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
# Colors
self.color = color
self.color2 = color2
self.text_color = text_color
self.text_color2 = text_color2
self.text_trans_color = self.text_color
self.button_trans_color = self.color
self.text_trans_color2 = self.text_color
self.button_trans_color2 = self.color
# Opacity
self.opacitya = 255
self.opacityb = 180
self.opacity1 = 255
self.opacity2 = 180
# Fonts
self.basefont = basefont
try:
    self.base_font = pygame.font.Font(self.basefont, self.textSize)
except:
    self.base_font = pygame.font.Font(f"../{self.basefont}", self.textSize)

def draw(self,outline=None,action=None, link=None, colorChange=True, newText=None, newX=None, newY=None, mode="normal"):
    """New X, Y and Text values"""
    if newText: self.text = newText
    if newX: self.x = newX
    if newY: self.y = newY

    """Draws the button. Variable for mouse detection"""
    mouse = pygame.mouse.get_pos()
    click = pygame.mouse.get_pressed()

    if self.buttonType == "Rectangle":
        """If it is a rectangle it will draw it here"""
        if mode == "normal":
            if outline:
                """draws an outline"""
                pygame.draw.rect(self.screen, outline, (self.x-2,self.y-2,self.width+4,self.height+4),0,self.curve)
            if self.x+self.width > mouse[0] > self.x and self.y+self.height > mouse[1] > self.y and colorChange:
                self.hover = True
                self.stop_start_animation = False
                self.time2 = pygame.time.get_ticks()
                step = (self.time2 - self.time) / (self.animation_time / self.animation_speed)
                """Draws a lighter version of the image"""
                self.button_trans_color = self.change_color(self.button_trans_color2, self.color2, step)
                pygame.draw.rect(self.screen, self.button_trans_color, (self.x,self.y,self.width,self.height), 0, self.curve)
            if self.text != '':
                self.text_trans_color = self.change_color(self.text_trans_color2, self.text_color2, step)
                text = self.base_font.render(self.text, 1, self.text_trans_color)
                self.screen.blit(text, (self.x + (self.width/2 - text.get_width()/2), self.y + (self.height/2 - text.get_height()/2)))

        # Clicking the Button
        if click[0] == 1:
            self.click = True
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
if self.click == True and not click[0]:
    self.release = True

if self.release:
    """If there is an action it is run"""
    if action:
        action()
    if link:
        webbrowser.open(link)
    self.release = False
    self.click = False

else:
    self.time = pygame.time.get_ticks()
    self.hover = False
    self.click = False
    self.release = False
    step = (self.time - self.time2) / (self.animation_time / self.animation_speed)
    """A darker version of image when the player isn't hovering over"""
    if not self.stop_start_animation:
        self.button_trans_color2 = self.change_color(self.button_trans_color, self.color, step)
        pygame.draw.rect(self.screen, self.button_trans_color2, (self.x,self.y,self.width,self.height),0, self.curve)
    else:
        pygame.draw.rect(self.screen, self.color, (self.x,self.y,self.width,self.height),0, self.curve)
    if self.text != '':
        """Text is blit here"""
        if not self.stop_start_animation:
            self.text_trans_color2 = self.change_color(self.text_trans_color, self.text_color, step)
            text = self.base_font.render(self.text, 1, self.text_trans_color2)
        else:
            text = self.base_font.render(self.text, 1, self.text_color)
        self.screen.blit(text, (self.x + (self.width/2 - text.get_width()/2), self.y + (self.height/2 - text.get_height()/2)))

else:
    if mode == "draw":
        self.time2 = pygame.time.get_ticks()
        step = (self.time2 - self.time) / (self.animation_time / self.animation_speed)
        """Draws a lighter version of the image"""
        self.button_trans_color = self.change_color(self.button_trans_color2, self.color2, step)
        pygame.draw.rect(self.screen, self.button_trans_color, (self.x,self.y,self.width,self.height), 0, self.curve)

    if mode == "undraw":
        self.time = pygame.time.get_ticks()
        step = (self.time - self.time2) / (self.animation_time / self.animation_speed)
        """A darker version of image when the player isn't hovering over"""
        self.button_trans_color2 = self.change_color(self.button_trans_color, self.color, step)
        pygame.draw.rect(self.screen, self.button_trans_color2, (self.x,self.y,self.width,self.height),0, self.curve)

if self.buttonType == "Circle":
    """If it is a circle it will draw it here"""
    if outline:
        """draws an outline"""


```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
        pygame.draw.circle(self.screen, outline, (self.x, self.y),self.radius+2.5,0)
    """Trig for area"""
    differenceInX = mouse[0] - self.x
    differenceInY = mouse[1] - self.y
    difference = ( differenceInX**2 + differenceInY**2 )**0.5
    """Info on the circle"""
    if difference <= self.radius:
        self.hover = True
        self.stop_start_animation = False
        self.time2 = pygame.time.get_ticks()
        step = (self.time2 - self.time) / (self.animation_time / self.animation_speed)
        self.button_trans_color = self.change_color(self.button_trans_color2, self.color2, step)
    """Draws a lighter version of the image"""
        pygame.draw.circle(self.screen, self.button_trans_color, (self.x,self.y),self.radius,0)

    if self.text != '':
        """Text is blit here"""
        self.text_trans_color = self.change_color(self.text_trans_color2, self.text_color2, step)
        text = self.base_font.render(self.text, 1, self.text_trans_color)
        self.screen.blit(text, (self.x - (text.get_width()/2), self.y - (text.get_height()/2)))

    # Clicking the Button
    if click[0] == 1:
        self.click = True

    if self.click == True and not click[0]:
        self.release = True

    if self.release:
        """If there is an action it is run"""
        if action != None:
            action()
        if link != None:
            webbrowser.open(link)
        self.release = False
        self.click = False
    else:
        self.time = pygame.time.get_ticks()
        step = (self.time - self.time2) / (self.animation_time / self.animation_speed)
        self.button_trans_color2 = self.change_color(self.button_trans_color, self.color, step)
        self.hover = False
        self.click = False
        self.release = False
    """A darker version of image when the player isn't hovering over"""
    pygame.draw.circle(self.screen, self.button_trans_color2, (self.x,self.y),self.radius,0)

    if self.text != '':
        """Text is blit here"""
        self.text_trans_color2 = self.change_color(self.text_trans_color, self.text_color, step)
        text = self.base_font.render(self.text, 1, self.text_trans_color2)
        self.screen.blit(text, (self.x - (text.get_width()/2), self.y - (text.get_height()/2)))
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
| if self.buttonType == "Image":  
|     if outline:  
|         """draws an outline"""  
|         pygame.draw.rect(self.screen, outline, (self.x-3,self.y-3,self.width+6,self.height+6),0,self.curve)  
|  
|     self.image.draw(self.x,self.y)  
|  
| if self.x+self.width > mouse[0] > self.x and self.y+self.height > mouse[1] > self.y:  
|     self.time2 = pygame.time.get_ticks()  
|     step = (self.time2 - self.time) / (self.animation_time / self.animation_speed)  
|     self.opacity2 = self.change_opacity(self.opacity1, self.opacityb, step)  
|     self.image.load_image.set_alpha(self.opacity2)  
|     self.hover = True  
|     # Clicking the Button  
|     if click[0] == 1:  
|         self.click = True  
|  
| if self.click == True and not click[0]:  
|     self.release = True  
|  
| if self.release:  
|     """If there is an action it is run"""  
|     if action != None:  
|         action()  
|     if link != None:  
|         webbrowser.open(link)  
|     self.release = False  
|     self.click = False  
| else:  
|     self.time = pygame.time.get_ticks()  
|     self.hover = False  
|     self.click = False  
|     self.release = False  
|     step = (self.time - self.time2) / (self.animation_time / self.animation_speed)  
|     self.opacity1 = self.change_opacity(self.opacity2, self.opacitya, step)  
|     self.image.load_image.set_alpha(self.opacity1)  
|  
def is_hovered(self):  
    return self.hover  
|  
def is_clicking(self):  
    return self.click  
|  
def change_color(self, original_color, new_color, step_multiplier=0):  
    colors = []  
    new_color_list = []  
    for rgb1, rgb2 in zip(original_color, new_color):  
        difference = rgb2 - rgb1  
        colors.append(difference)  
|  
    color_step = [difference / self.num_frames for difference in colors]  
|  
    for og_color_rgb, new_color_rgb, step in zip(original_color, new_color, color_step):  
        if og_color_rgb > new_color_rgb:  
|
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
if int(og_color_rgb + step_multiplier*step) < new_color_rgb:  
    new_color_list.append(new_color_rgb)  
else:  
    new_color_list.append(int(og_color_rgb + step_multiplier*step))  
else:  
    if int(og_color_rgb + step_multiplier*step) > new_color_rgb:  
        new_color_list.append(new_color_rgb)  
    else:  
        new_color_list.append(int(og_color_rgb + step_multiplier*step))  
  
return tuple(new_color_list)  
  
def change_opacity(self, original_opacity, new_opacity, step_multiplier=0):  
    new_opacity_return = None  
    difference = new_opacity - original_opacity  
    opacity_step = difference / self.num_frames  
    if original_opacity > new_opacity:  
        if int(original_opacity + opacity_step*step_multiplier) < new_opacity:  
            new_opacity_return = new_opacity  
        else:  
            new_opacity_return = int(original_opacity + opacity_step*step_multiplier)  
    else:  
        if int(original_opacity + opacity_step*step_multiplier) > new_opacity:  
            new_opacity_return = new_opacity  
        else:  
            new_opacity_return = int(original_opacity + opacity_step*step_multiplier)  
  
    return new_opacity_return
```

Level:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
from Scripts.logger import *
try:
    import pygame
    from Scripts.settings import Config
    from Scripts.tile import Tile
    from Scripts.support import *
except:
    logger.critical("You do not have all the modules installed. Please install Pygame.")

logger.info("RealDL Level Code.")

class Level(Config):
    def __init__(self):
        Config.__init__(self)

    # get the display surface
    self.display_surface = pygame.display.get_surface()

    # sprite group setup
    self.visible_sprites = YSortCameraGroup()
    self.obstacle_sprites = pygame.sprite.Group()

    # attack sprites
    self.current_attack = None
    self.attack_sprites = pygame.sprite.Group()
    self.attachable_sprites = pygame.sprite.Group()
    self.player_sprites = pygame.sprite.Group()

    # sprite setup
    self.tile_id = 0
    self.create_map()

    def create_map(self):
        layouts = {
            'boundary': import_csv_layout('Map/map_FloorBlocks.csv'),
            'grass': import_csv_layout('Map/map_Grass.csv'),
            'object': import_csv_layout('Map/map_Objects.csv'),
        }

        graphics = {
            'grass': import_folder('Graphics/Game/grass'),
            'objects': import_folder('Graphics/Game/objects')
        }
        for style,layout in layouts.items():
            for row_index, row in enumerate(layout):
                for col_index, col in enumerate(row):
                    if col != '-1':
                        x = col_index * self.TILESIZE
                        y = row_index * self.TILESIZE
                        if style == 'boundary':
                            Tile((x,y),[self.obstacle_sprites],'invisible',pygame.Surface((self.TILESIZE,self.TILESIZE)),self.TILE_ID)
                        if style == 'grass':
                            tile_id = f"Tile{self.tile_id}"
                            if col == '8':
                                grass_image = graphics['grass'][0]
                            elif col == '9':
                                grass_image = graphics['grass'][1]
                            else:
                                grass_image = graphics['grass'][2]
                            self.visible_sprites.add(Tile((x,y),[self.obstacle_sprites],tile_id,grass_image))
                            self.attachable_sprites.add(Tile((x,y),[self.obstacle_sprites],tile_id,grass_image))
                            self.player_sprites.add(Tile((x,y),[self.obstacle_sprites],tile_id,grass_image))
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
        Tile((x,y),[self.visible_sprites,self.obstacle_sprites,self.attackable_sprites],'grass',grass_image,tile_id)
        self.tile_id += 1
    if style == 'object':
        surf = graphics['objects'][int(col)]
        Tile((x,y),[self.visible_sprites,self.obstacle_sprites],'object',surf,self.TILE_ID)

def run(self, player, dt):
    # update and draw the game
    self.visible_sprites.custom_draw(player)
    self.visible_sprites.update(dt)

class YSortCameraGroup(pygame.sprite.Group):
    def __init__(self):
        # general setup
        super().__init__()
        self.settings = Config()
        self.display_surface = pygame.display.get_surface()
        self.half_width = self.display_surface.get_size()[0] // 2
        self.half_height = self.display_surface.get_size()[1] // 2
        self.offset = pygame.math.Vector2()

        # Floor
        try: self.floor_surf = pygame.image.load('Graphics/Game/tilemap/bg.png').convert()
        except: self.floor_surf = pygame.image.load('../Graphics/Game/tilemap/bg.png').convert()

        self.floor_rect = self.floor_surf.get_rect(topleft=(0,0)) # in order to not see the white

    def custom_draw(self, player):
        # getting the offset
        self.offset.x = player.rect.centerx - self.half_width
        self.offset.y = player.rect.centery - self.half_height

        # drawing the floor
        self.display_surface.fill(self.settings.BG_COLOR)
        floor_offset_pos = self.floor_rect.topleft - self.offset
        self.display_surface.blit(self.floor_surf, floor_offset_pos)

        # for sprite in self.sprites():
        for sprite in sorted(self.sprites(), key=lambda sprite: sprite.rect.centery):
            offset_pos = sprite.rect.topleft - self.offset
            if sprite.sprite_type == "player":
                if sprite.id == player.id:
                    sprite.draw(offset_pos, (50, 190, 40))
                else:
                    if player.status_alive == "alive":
                        if not sprite.status_alive == "dead":
                            sprite.draw(offset_pos)
                    else:
                        sprite.draw(offset_pos)
            else:
                self.display_surface.blit(sprite.image, offset_pos)
```

Logger:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
import logging

# Configure logging
logging.basicConfig(level=logging.DEBUG) # Set the base logging level

# Create a logger instance
logger = logging.getLogger(__name__)
logger.propagate = False # Disable propagation to parent logger

# Create a formatter for the log messages
class ColoredFormatter(logging.Formatter):
    COLORS = {
        logging.DEBUG: "\033[1;34m",      # Blue for DEBUG
        logging.INFO: "\033[1;29m",       # Grey for INFO
        logging.WARNING: "\033[1;33m",    # Yellow for WARNING
        logging.ERROR: "\033[1;91m",      # Orange for ERROR
        logging.CRITICAL: "\033[1;31m"   # Red for CRITICAL
    }

    def format(self, record):
        log_color = self.COLORS.get(record.levelno, "\033[0m") # Default to no color
        log_level = record.levelname
        timestamp = self.formatTime(record, self.datefmt)
        message = log_color + log_level + "\033[0m" + " - " + record.msg
        return f"{timestamp} - {message}"

# Create a handler for console output with the colored formatter
colored_console_handler = logging.StreamHandler()
colored_console_handler.setFormatter(ColoredFormatter())

# Add the colored handler to the logger
logger.addHandler(colored_console_handler)

logger.info("RealDL Logger Code.")
```

Magic:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
from Scripts.logger import *
try:
    import pygame, random
    from Scripts.settings import *
except:
    logger.critical("You do not have all the modules installed. Please install Pygame.")

logger.debug("RealDL Magic Code.")

class MagicPlayer(Config):
    def __init__(self,animation_player):
        Config.__init__(self)
        self.animation_player = animation_player

    def heal(self,player,strength,cost,groups, player_id):
        if player.energy >= cost:
            player.health += strength
            player.energy -= cost
            if player.health >= player.stats['health']:
                player.health = player.stats['health']
            self.animation_player.create_particles('aura',player.rect.center,strength,groups, player_id)
            self.animation_player.create_particles('heal',player.rect.center,strength,groups, player_id)

    def flame(self,player,strength,cost,groups,player_id):
        if player.energy >= cost:
            player.energy -= cost

            if player.status.split('_')[0] == 'right':
                direction = pygame.math.Vector2(1,0)
            elif player.status.split('_')[0] == 'left':
                direction = pygame.math.Vector2(-1,0)
            elif player.status.split('_')[0] == 'up':
                direction = pygame.math.Vector2(0,-1)
            else:
                direction = pygame.math.Vector2(0,1)

            for i in range(1,6):
                if direction.x: #horizontal
                    offset_x = (direction.x * i) * self.TILESIZE
                    x = player.rect.centerx + offset_x + random.randint(-self.TILESIZE // 3, self.TILESIZE // 3)
                    y = player.rect.centery + random.randint(-self.TILESIZE // 3, self.TILESIZE // 3)
                    self.animation_player.create_particles('flame',(x,y),strength,groups, player_id)
                else: # vertical
                    offset_y = (direction.y * i) * self.TILESIZE
                    x = player.rect.centerx + random.randint(-self.TILESIZE // 3, self.TILESIZE // 3)
                    y = player.rect.centery + offset_y + random.randint(-self.TILESIZE // 3, self.TILESIZE // 3)
                    self.animation_player.create_particles('flame',(x,y),strength,groups, player_id)
```

Main menu:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
from Scripts.logger import *
try:
    import pygame, sys, time
    from Scripts.settings import Config
    from Scripts.encryption import *
    from Scripts.debug import debug
    from Scripts.functions import *
except:
    logger.critical("You do not have all the modules installed. Please install Pygame.")

logger.info("RealDL - Main Menu Code")

class MainMenu:
    def __init__(self):
        # Music
        try:
            self.background_music = pygame.mixer.Sound("Audio/main.ogg")
        except:
            self.background_music = pygame.mixer.Sound("../Audio/main.ogg")

    def close(self):
        self.loop = False
        pygame.quit()
        sys.exit()

    def options(self):
        self.main_menu_pages = "settings"

    def home(self):
        self.main_menu_pages = "home"

    def start_option(self):
        self.main_menu_pages = "start"

    def draw_moving_background(self):
        # Draw the moving image on the screen
        self.sunset_image.draw(self.current_x, 0)
        self.sunset_image.draw((self.current_x - self.sunset_image.rect.width), 0)
        self.current_x += 1

        # If the image has moved beyond its width, reset it to 0
        if self.current_x >= self.sunset_image.rect.width:
            self.current_x = 0

    def change_keys(self):
        if self.keys == 'WASD':
            self.keys = 'Arrow Keys'
        else:
            self.keys = 'WASD'

    def control_settings_change(self):
        self.settings_screen = "control"

    def audio_settings_change(self):
        self.settings_screen = "audio"
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
def change_attack(self):
    # Changes the key binds.
    if self.attack_keys == "Space":
        self.attack_keys = "L-Ctrl"
    elif self.attack_keys == "L-Ctrl":
        self.attack_keys = "R-Ctrl"
    else:
        self.attack_keys = "Space"

def change_magic(self):
    if self.magic_keys == "L-Shift":
        self.magic_keys = "R-Shift"
    elif self.magic_keys == "R-Shift":
        self.magic_keys = "Enter"
    else:
        self.magic_keys = "L-Shift"

def sound_box_color(self):
    # Changes the sound box color
    if self.sound_change == 1:
        self.sound_box.color = (27,31,35)
        self.sound_change = 0
    else:
        self.sound_box.color = (39,174,96)
        self.sound_change = 1

def music_box_color(self):
    # Changes the music box check color.
    if self.music_change == 1:
        self.music_box.color = (27,31,35)
        self.music_change = 0
    else:
        self.music_box.color = (39,174,96)
        self.music_change = 1

def volume_settings(self):
    # Get the mouse and sets the volume to where the mouse is.
    mouse_pos = pygame.mouse.get_pos()
    if mouse_pos[0] - self.volume_button.width/3 > self.volume_line.x:
        if mouse_pos[0] + (self.volume_button.width/3)*2 < self.volume_line.x + self.volume_line.width:
            self.volume_button.x = mouse_pos[0] - self.volume_button.width/3
            self.volume = int(self.volume_ratio * (mouse_pos[0] - self.min_vol_x))

def create_dict(self):
    self.starting_dict = {
        "settings": {
            "control": {
                "movement": self.keys,
                "offense": self.attack_keys,
                "magic": self.magic_keys
            },
            "audio": {
                "volume": self.volume,
                "sound": True if self.sound_change == 1 else False,
                "music": True if self.music_change == 1 else False
            }
        },
        "start": {
    
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
        "username": self.name_text_box.return_text() or "Player",
        "server_ip": self.ip_text_box.return_text() or "192.168.0.223"
    }
}
self.background_music.stop()
self.loop = False

# Check the values
"""
for category, subcategories in self.starting_dict.items():
    logger.info(f"{category}:")
    for subcategory, values in subcategories.items():
        logger.info(f"  {subcategory}:")
        if isinstance(values, dict):
            for key, value in values.items():
                logger.info(f"    {key}: {value}")
        else:
            logger.info(f"    {values}")
"""

def start_game(self):
    if not self.loop:
        return self.starting_dict
    else:
        return None

def restart_menu(self):
    self.loop = True
    self.home()

def draw_objects(self, events):
    # Draw the buttons and check for hover
    if self.main_menu_pages == "home":
        self.github_button.draw((27,31,35),None,"https://github.com/TheRealDL1/Simple-Client-Server")
        self.youtube_button.draw((27,31,35),None,"https://www.youtube.com/channel/UCf5op_Bt-OTWLQPtCxWnbfg")
        self.start_button.draw((27,31,35),self.start_option)
        self.options_button.draw((27,31,35),self.options)
        self.quit_button.draw((27,31,35),self.close,None,True)

        start_button_hover = self.start_button.is_hovered()
        options_button_hover = self.options_button.is_hovered()
        quit_button_hover = self.quit_button.is_hovered()
        github_button_hover = self.github_button.is_hovered()
        youtube_button_hover = self.youtube_button.is_hovered()

        start_button_click = self.start_button.is_clicking()
        options_button_click = self.options_button.is_clicking()
        quit_button_click = self.quit_button.is_clicking()
        github_button_click = self.github_button.is_clicking()
        youtube_button_click = self.youtube_button.is_clicking()
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

if start_button_hover or options_button_hover or quit_button_hover or github_button_hover or youtube_button_hover:
    # Draw hover text.
    if github_button_hover: self.github_name.draw("draw","Github", self.settings.WIDTH-(self.image_width/2)-self.image_padding, (self.image_width/2)+self.image_padding*3.1)
    else: self.github_name.draw("undraw","Github", self.settings.WIDTH-(self.image_width/2)-self.image_padding, (self.image_width/2)+self.image_padding*3.1)

    if youtube_button_hover: self.youtube_name.draw("draw","Youtube", self.settings.WIDTH-(self.image_width/2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)
    else: self.youtube_name.draw("undraw","Youtube", self.settings.WIDTH-(self.image_width/2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)

    if not start_button_click and not options_button_click and not quit_button_click and not github_button_click and not youtube_button_click:
        self.custom_mouse.mode = 1
    else:
        self.custom_mouse.mode = 2
else:
    self.custom_mouse.mode = 0
    self.github_name.draw("undraw","Github", self.settings.WIDTH-(self.image_width/2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)
    self.youtube_name.draw("undraw","Youtube", self.settings.WIDTH-(self.image_width/2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)

if self.main_menu_pages == "settings":
    self.options_board.draw((27,31,35),None, None, False)
    self.back.draw((240,240,240),self.home)
    self.control_settings.draw("draw","Control Settings", (self.settings.WIDTH/2)*0.70, (self.settings.HEIGHT/2)*0.42,self.control_settings_change)
    self.audio_settings.draw("draw","Audio Settings", (self.settings.WIDTH/2)*1.3, (self.settings.HEIGHT/2)*0.42,self.audio_settings_change)
    self.github_button.draw((27,31,35),None,"https://github.com/TheRealDL1/Simple-Client-Server")
    self.youtube_button.draw((27,31,35),None,"https://www.youtube.com/dominicpike")

back_hover = self.back.is_hovered()
github_button_hover = self.github_button.is_hovered()
youtube_button_hover = self.youtube_button.is_hovered()
key_binds_hover = self.control_settings.is_hovered()
audio_hover = self.audio_settings.is_hovered()
key_hover = self.key.binds.is_hovered()
attack_hover = self.attack_btn.is_hovered()
volume_btn_hover = self.volume_button.is_hovered()
sound_box_hover = self.sound_box.is_hovered()
music_box_hover = self.music_box.is_hovered()
magic_btn_hover = self.magic_BTN.is_hovered()

back_click = self.back.is_clicking()
github_button_click = self.github_button.is_clicking()
youtube_button_click = self.youtube_button.is_clicking()
key_binds_click = self.control_settings.is_clicking()
audio_click = self.audio_settings.is_clicking()
key_click = self.key.binds.is_clicking()
attack_click = self.attack_btn.is_clicking()
volume_btn_click = self.volume_button.is_clicking()
sound_box_click = self.sound_box.is_clicking()
music_box_click = self.music_box.is_clicking()
magic_btn_click = self.magic_BTN.is_clicking()

if back_hover or github_button_hover or youtube_button_hover or key_binds_hover or audio_hover or key_hover or attack_hover or volume_btn_hover or sound_box_hover or music_box_hover or magic_btn_hover:
    if github_button_hover: self.github_name.draw("draw","Github", self.settings.WIDTH-(self.image_width/2)-self.image_padding, (self.image_width/2)+self.image_padding*3.1)
    else: self.github_name.draw("undraw","Github", self.settings.WIDTH-(self.image_width/2)-self.image_padding, (self.image_width/2)+self.image_padding*3.1)

    if youtube_button_hover: self.youtube_name.draw("draw","Youtube", self.settings.WIDTH-(self.image_width/2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)
    else: self.youtube_name.draw("undraw","Youtube", self.settings.WIDTH-(self.image_width/2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)

if not back_click and not github_button_click and not youtube_button_click and not key_binds_click and not audio_click and not key_click and not attack_click and not volume_btn_click and not sound_box_click and not music_box_click and not magic_btn_click:
    self.custom_mouse.mode = 1
else:
    if volume_btn_click:
        self.volume_settings()
        self.custom_mouse.mode = 2
    else:
        self.custom_mouse.mode = 0
    self.github_name.draw("undraw","Github", self.settings.WIDTH-(self.image_width/2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)
    self.youtube_name.draw("undraw","Youtube", self.settings.WIDTH-(self.image_width/2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)

if self.settings_screen == "control":
    if audio_hover: self.underline2.draw(None,None,None,True,None,None,"draw")
    else: self.underline2.draw(None,None,None,True,None,None,"undraw")
    self.underline1.draw(None,None,None,True,None,None,"draw")
    self.box_control.draw((240,240,240))
    self.movement.draw("draw","Movement", (self.settings.WIDTH/2)*0.70, (self.settings.HEIGHT/2)*0.57)
    self.key_binds.draw((240,240,240),self.change_keys, None, True, self.keys)
    self.attack_text.draw("draw","Offense", (self.settings.WIDTH/2)*0.70, (self.settings.HEIGHT/2)*0.82)
    self.attack_btn.draw((240,240,240),self.change_attack, None, True, self.attack_keys)
    self.magic_text.draw("draw","Magic", (self.settings.WIDTH/2)*0.70, (self.settings.HEIGHT/2)*1.07)
    self.magic_BTN.draw((240,240,240),self.change_mag, None, True, self.magic_keys)

elif self.settings_screen == "audio":
    if key_binds_hover: self.underline1.draw(None,None,None,True,None,None,"draw")
    else: self.underline1.draw(None,None,None,True,None,None,"undraw")
    self.underline2.draw((240,240,240))
    self.volume_audio.draw((240,240,240))
    self.volume_text.draw("draw","Volume", (self.settings.WIDTH/2)*1.3, (self.settings.HEIGHT/2)*0.57)
    self.volume_bar.draw((240,240,240))
    self.volume_line.draw((240,240,240))
    self.volume_button.draw((240,240,240))
    self.volume_indicator.draw("draw",str(self.volume), (self.settings.WIDTH/2)*1.435, (self.settings.HEIGHT/2)*0.685)
    self.audio_text.draw("draw","Audio", (self.settings.WIDTH/2)*1.3, (self.settings.HEIGHT/2)*0.82)
    self.audio_box.draw((240,240,240))
    self.sound_text.draw("draw","Sound", (self.settings.WIDTH/2)*1.185, (self.settings.HEIGHT/2)*0.97)
    self.music_text.draw("draw","Music", (self.settings.WIDTH/2)*1.18, (self.settings.HEIGHT/2)*1.08)
    self.sound_box.draw((240,240,240),self.sound_box_color)
    self.music_box.draw((240,240,240),self.music_box_color)

if self.main_menu_pages == "start":
    self.options_board.draw((27,31,35),None, None, False)
    self.start_back.draw((240,240,240),self.home)
    self.github_button.draw((27,31,35),None,"https://github.com/TheRealDL1/Simple-Client-Server")
    self.youtube_button.draw((27,31,35),None,"https://www.youtube.com/dominicpike")
    self.bullet_assault.draw("draw","Bullet Assault", (self.settings.WIDTH/2), (self.settings.HEIGHT/2)*0.43)
    self.server_ip_text.draw("draw","Server IP Address", (self.settings.WIDTH/2), (self.settings.HEIGHT/2)*0.955)
    message = "Enter the Server's IP address that you want to join!"
    self.join_message.draw("draw",message, (self.settings.WIDTH/2), (self.settings.HEIGHT/2)*0.52)
    self.ip_text_box.draw()
    self.ip_text_box.updateText(events)
    self.ip_text_box.update()
    self.join_btn.draw((240,240,240),self.create_dict)

```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

# Name Text Box
self.name_box.text.draw("draw","Username", (self.settings.WIDTH/2), (self.settings.HEIGHT/2)*0.665)
self.name_text_box.draw()
self.name_text_box.updateText(events)
self.name_text_box.update()

github_button_hover = self.github_button.is_hovered()
youtube_button_hover = self.youtube_button.is_hovered()
start_back_hover = self.start_back.is_hovered()
join_hover = self.join_btn.is_hovered()
text_box_hover = self.ip_text_box.is_hovered()
name_hover = self.name_text_box.is_hovered()

github_button_click = self.github_button.is_clicking()
youtube_button_click = self.youtube_button.is_clicking()
start_back_click = self.start_back.is_clicking()
join_click = self.join_btn.is_clicking()
text_box_click = self.ip_text_box.is_clicking()
name_click = self.name_text_box.is_clicking()

if start_back_hover or github_button_hover or youtube_button_hover or join_hover or text_box_hover or name_hover:
    # Draw hover text.
    if github_button_hover: self.github_name.draw("draw","Github", self.settings.WIDTH-(self.image_width/2)-self.image_padding, (self.image_width/2)+self.image_padding*3.1)
    else: self.github_name.draw("undraw","Github", self.settings.WIDTH-(self.image_width/2)-self.image_padding, (self.image_width/2)+self.image_padding*3.1)

    if youtube_button_hover: self.youtube_name.draw("draw","Youtube", self.settings.WIDTH-(self.image_width*2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)
    else: self.youtube_name.draw("undraw","Youtube", self.settings.WIDTH-(self.image_width*2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)

    if not start_back_click and not github_button_click and not youtube_button_click and not join_click and not text_box_click and not name_click:
        self.custom_mouse.mode = 1
    else:
        self.custom_mouse.mode = 2
else:
    self.custom_mouse.mode = 0
    self.github_name.draw("undraw","Github", self.settings.WIDTH-(self.image_width/2)-self.image_padding, (self.image_width/2)+self.image_padding*3.1)
    self.youtube_name.draw("undraw","Youtube", self.settings.WIDTH-(self.image_width*2)-(self.image_padding/4), (self.image_width/2)+self.image_padding*3.1)

# Draw the mouse
self.custom_mouse.draw()

def redraw_window(self, events):
    self.draw_moving_background()
    self.draw_objects(events)

def run(self):
    self.background_music.play(-1)
    while self.loop:
        if self.music_change == 0:
            self.background_music.set_volume(0)
        else:
            self.background_music.set_volume(self.volume/100)
        events = pygame.event.get()
        for event in events:
            if event.type == pygame.QUIT:
                self.close()
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_ESCAPE:
                    self.close()

```

```

    if event.key == pygame.K_ESCAPE:
        self.close()

    # Update the display and frame rate
    self.redraw_window(events)
    pygame.display.update()
    self.clock.tick(self.FPS)

```

Network:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
Copy > Network > Schedule
import socket
from Scripts.logger import *

logger.debug("RealDL Network Code.")

class Network():
    def __init__(self, server, port):
        try:
            self.client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            self.server = server
            self.port = port
            self.addr = (self.server, self.port)
        except:
            logger.critical("Failed to setup Server-Client connection.")

    def connect(self):
        try:
            self.client.connect(self.addr)
        except socket.error as Error:
            logger.error(f"Socket failed trying to connect: {Error}")

    def close(self):
        self.client.close()

    def receive(self, data_size):
        try:
            return self.client.recv(data_size)
        except socket.error as Error:
            logger.error(f"Socket failed trying to receive: {Error}")

    def send(self, data):
        try:
            self.client.send(data)
        except socket.error as Error:
            logger.error(f"Socket failed trying to send: {Error}")

    def sendall(self, data):
        try:
            self.client.sendall(data)
        except socket.error as Error:
            logger.error(f"Socket failed trying to sendall: {Error}")
```

Particles:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
from Scripts.logger import *
try:
    import pygame, string, random
    from Scripts.settings import Config
    from Scripts.support import import_folder
except:
    logger.critical("You do not have all the modules installed. Please install Pygame.")

logger.debug("RealDL Magic Code.")

class AnimationPlayer:
    def __init__(self):
        self.particles = None
        self.frames = {
            # magic
            'flame': import_folder('Graphics/Game/particles/flame/frames'),
            'aura': import_folder('Graphics/Game/particles/aura'),
            'heal': import_folder('Graphics/Game/particles/heal/frames'),
        }

        self.images = {
            # magic
            'flame': import_folder('Graphics/Game/particles/flame/frames',None),
            'aura': import_folder('Graphics/Game/particles/aura',None),
            'heal': import_folder('Graphics/Game/particles/heal/frames',None),
        }

    def create_particles(self,animation_type,pos,strength,groups,player_id):
        animation_frames = self.frames[animation_type]
        animation_images = self.images[animation_type]
        self.particles = ParticleEffect(pos,animation_frames,animation_images,animation_type,groups,strength,player_id)

class ParticleEffect(pygame.sprite.Sprite):
    def __init__(self,pos,animation_frames,animation_images,animation_type,groups,strength=None,player_id=None):
        super().__init__(groups)
        self.settings = Config()
        self.sprite_type = 'magic'
        self.strength = strength
        self.damaged_player = False
        self.animation_type = animation_type
        self.id = self.create_id()
        self.player_id = player_id
        self.frame_index = 0
        self.animation_speed = 0.15*(self.settings.frame_increase_rate/1.5)
        self.frames = animation_frames
        self.images = animation_images
        self.image = self.frames[self.frame_index]
        self.rect = self.image.get_rect(center = pos)

    def animate(self, dt):
        self.frame_index += self.animation_speed * dt
        if self.frame_index >= len(self.frames):
            self.kill()
        else:
            self.image = self.frames[int(self.frame_index)]

    def return_type(self):
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
def return_type(self):
    return self.animation_type

def return_strength(self):
    return self.strength

def return_image(self):
    self.full_path = self.images[int(self.frame_index)]
    if "../" in self.full_path:
        self.full_path = self.full_path.replace("../", "")
    return self.full_path

def create_id(self):
    try:
        characters = string.ascii_letters + string.digits
        return ''.join(random.choice(characters) for _ in range(self.settings.ID_STRING_LENGTH))
    except:
        logger.error("Something went wrong with creating a unique ID.")

def update(self, dt):
    self.animate(dt)
```

Player:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
from Scripts.logger import *
try:
    import pygame, math
    from Scripts.support import import_folder
    from Scripts.settings import Config
except:
    logger.critical("You do not have all the modules installed. Please install Pygame.")

logger.debug("RealDL Player Code.")

class Player(pygame.sprite.Sprite):
    def __init__(self, pos, image, groups, obstacle_sprites, username, id, health, create_attack=None, destroy_attack=None, create_bullet=None, create_magic=None,
                 movement="WASD", offense="Space", magic="L-Shift", status_alive="alive"):
        # Setting up player
        try:
            super().__init__(groups)
            self.settings = Config()
            self.screen = pygame.display.get_surface()
            try:
                self.image = pygame.image.load(image).convert_alpha()
                self.image_name = image
            except:
                self.image = pygame.image.load(f"../{image}").convert_alpha()
                self.image_name = f"../{image}"
            self.rect = self.image.get_rect(topleft = pos)
            self.hitbox = self.rect.inflate(0,-26)
            self.old_hitbox = self.hitbox.copy()
            self.pos = pygame.math.Vector2(self.hitbox.topleft)
            self.movement = movement
            self.offense = offense
            self.magic_key = magic
            self.key_binds = {
                'move_up': pygame.K_w if self.movement == "WASD" else pygame.K_UP,
                'move_down': pygame.K_s if self.movement == "WASD" else pygame.K_DOWN,
                'move_left': pygame.K_a if self.movement == "WASD" else pygame.K_LEFT,
                'move_right': pygame.K_d if self.movement == "WASD" else pygame.K_RIGHT,
                'attack': pygame.K_SPACE if self.offense == "Space" else pygame.K_LCTRL if self.offense == "L-Shift" else pygame.K_RCTRL,
                'magic': pygame.K_LSHIFT if self.magic_key == "L-Shift" else pygame.K_RSHIFT if self.magic_key == "R-Shift" else pygame.K_RETURN
            }

            # graphics setup
            self.import_player_assets()
            self.status = 'down'
            self.frame_index = 0
            self.animation_speed = 0.15*(self.settings.frame_increase_rate)*0.6

            # Movement
            self.direction = pygame.math.Vector2()
            self.attacking = False
            self.magic_attacking = False
            self.holding_attack_btn = False
            self.holding_magic_btn = False
            self.attack_cooldown = 400
            self.attack_time = None
            self.paused = False

            # weapon
            self.create_attack = create_attack
            self.destroy_attack = destroy_attack
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
self.destroy_attack = destroy_attack
self.create_bullet = create_bullet
self.weapon_index = 0
self.weapon = list(self.settings.weapon_data.keys())[self.weapon_index]
self.weapon_type = self.settings.weapon_data[self.weapon]['type']
self.can_switch_weapon = True
self.weapon_switch_time = None
self.switch_duration_cooldown = 200

# magic
self.magic_index = 0
self.create_magic = create_magic
self.magic = list(self.settings.magic_data.keys())[self.magic_index]
self.can_switch_magic = True
self.magic_switch_time = None

# magic and weapons
self.attacking_reset = False
self.finish_attacking = 0

# stats
self.stats = {'health': 100, 'energy': 60, 'attack': 10, 'magic': 4, 'speed': 6}
#self.max_stats = {'health': 300, 'energy': 140, 'attack': 20, 'magic': 10, 'speed': 10}
self.upgrade_cost = {'health': 100, 'energy': 100, 'attack': 100, 'magic': 100, 'speed': 100}
self.health = health
self.energy = self.stats['energy'] * 0.8
self.attack_strength = self.stats['attack']
self.exp = 500
self.speed = self.stats['speed']*(self.settings.frame_increase_rate/1.55)
self.status_alive = status_alive

# damage timer
self.vulnerable = True
self.hurt_time = None
self.invulnerability_duration = 500
self.alpha = 255
self.image_alpha = 255

#Other stats
self.kill_count = 0
self.username = username
self.id = id
self.basefont = "Graphics/Fonts/Orbitron-Medium.ttf"
self.largefont = "Graphics/Fonts/Orbitron-Bold.ttf"
self.textSize = 20
try:
    self.font = pygame.font.Font(self.basefont, self.textSize)
    self.large_font = pygame.font.Font(self.largefont, self.textSize)
except:
    self.font = pygame.font.Font(f"../{self.basefont}", self.textSize)
    self.large_font = pygame.font.Font(f"../{self.largefont}", self.textSize)

self.sprite_type = "player"

self.obstacle_sprites = obstacle_sprites
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
except:  
    logger.error("Failed to create Player Class")  
  
def import_player_assets(self):  
    character_path = 'Graphics/Game/player/'  
    self.animations = {'up': [], 'down': [], 'left': [], 'right': [],  
                      'right_idle': [], 'left_idle': [], 'up_idle': [], 'down_idle': [],  
                      'right_attack': [], 'left_attack': [], 'up_attack': [], 'down_attack': []}  
  
    self.animation_images = {'up': [], 'down': [], 'left': [], 'right': [],  
                            'right_idle': [], 'left_idle': [], 'up_idle': [], 'down_idle': [],  
                            'right_attack': [], 'left_attack': [], 'up_attack': [], 'down_attack': []}  
  
    for animation in self.animations.keys():  
        full_path = character_path + animation  
        self.animations[animation], self.animation_images[animation] = import_folder(full_path, True)  
  
def input(self):  
    keys = pygame.key.get_pressed()  
    if not self.paused:  
        if not self.attacking and not self.magic_attacking:  
            if keys[self.key_binds['move_up']]:  
                self.direction.y = -1  
                self.status = 'up'  
            elif keys[self.key_binds['move_down']]:  
                self.direction.y = 1  
                self.status = 'down'  
            else:  
                self.direction.y = 0  
  
            if keys[self.key_binds['move_right']]:  
                self.direction.x = 1  
                self.status = 'right'  
            elif keys[self.key_binds['move_left']]:  
                self.direction.x = -1  
                self.status = 'left'  
            else:  
                self.direction.x = 0  
  
            if self.status_alive == "alive":  
                # attack input  
                if keys[self.key_binds['attack']] and not self.magic_attacking and not self.attacking_reset:  
                    self.attacking = True  
                    self.holding_attack_btn = True  
                    self.attack_time = pygame.time.get_ticks()  
                    if self.create_attack != None:  
                        self.create_attack(self.weapon_type)  
                        if self.weapon_type == "gun":  
                            if self.create_bullet:  
                                self.create_bullet()
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
# magic input
if keys[self.key_binds['magic']] and not self.attacking and not self.attacking_reset:
    self.magic_attacking = True
    self.holding_magic_btn = True
    self.attack_time = pygame.time.get_ticks()
    style = list(self.settings.magic_data.keys())[self.magic_index]
    strength = list(self.settings.magic_data.values())[self.magic_index]['strength'] + self.stats['magic']
    cost = list(self.settings.magic_data.values())[self.magic_index]['cost']
    if self.create_magic:
        self.create_magic(style, strength, cost)

if keys[pygame.K_q] and self.can_switch_weapon:
    self.can_switch_weapon = False
    self.weapon_switch_time = pygame.time.get_ticks()

    if self.weapon_index < len(list(self.settings.weapon_data.keys())) - 1:
        self.weapon_index += 1
    else:
        self.weapon_index = 0

    self.weapon = list(self.settings.weapon_data.keys())[self.weapon_index]
    self.weapon_type = self.settings.weapon_data[self.weapon]['type']

if keys[pygame.K_e] and self.can_switch_magic:
    self.can_switch_magic = False
    self.magic_switch_time = pygame.time.get_ticks()

    if self.magic_index < len(list(self.settings.magic_data.keys())) - 1:
        self.magic_index += 1
    else:
        self.magic_index = 0

    self.magic = list(self.settings.magic_data.keys())[self.magic_index]

else:
    if not keys[self.key_binds['attack']]:
        self.holding_attack_btn = False
    if not keys[self.key_binds['magic']]:
        self.holding_magic_btn = False

def get_status(self):
    # idle status
    if self.direction.x == 0 and self.direction.y == 0:
        if not 'idle' in self.status and not 'attack' in self.status:
            self.status = self.status + '_idle'

    if self.attacking or self.magic_attacking:
        self.direction.x = 0
        self.direction.y = 0
        if not 'attack' in self.status:
            if 'idle' in self.status:
                self.status = self.status.replace('_idle','_attack')
            else:
                self.status = self.status + '_attack'
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
else:
    if 'attack' in self.status:
        self.status = self.status.replace('_attack','')

def move(self,speed, dt):
    if not self.paused:
        if self.hitbox.x != self.old_hitbox.x or self.hitbox.y != self.old_hitbox.y:
            if not 'idle' in self.status:
                self.old_hitbox = self.hitbox.copy()

        if self.direction.magnitude() != 0:
            self.direction = self.direction.normalize()

        self.pos.x += self.direction.x * speed * dt
        self.hitbox.x = self.pos.x
        self.collision('horizontal')

        self.pos.y += self.direction.y * speed * dt
        self.hitbox.y = self.pos.y
        self.collision('vertical')
        self.rect.center = self.hitbox.center

def collision(self,direction):
    collision_sprites = pygme.sprite.spritecollide(self,self.obstacle_sprites,False)
    if collision_sprites:
        if direction == 'horizontal':
            for sprite in collision_sprites:

                # Collision on the right
                if self.hitbox.right >= sprite.hitbox.left and self.old_hitbox.right <= sprite.old_hitbox.left:
                    self.hitbox.right = sprite.hitbox.left
                    self.pos.x = self.hitbox.x

                # Collision on the left
                if self.hitbox.left <= sprite.hitbox.right and self.old_hitbox.left >= sprite.old_hitbox.right:
                    self.hitbox.left = sprite.hitbox.right
                    self.pos.x = self.hitbox.x

        if direction == 'vertical':
            for sprite in collision_sprites:
                # Collision on the bottom
                if self.hitbox.bottom >= sprite.hitbox.top and self.old_hitbox.bottom <= sprite.old_hitbox.top:
                    self.hitbox.bottom = sprite.hitbox.top
                    self.pos.y = self.hitbox.y

                # Collision on the top
                if self.hitbox.top <= sprite.hitbox.bottom and self.old_hitbox.top >= sprite.old_hitbox.bottom:
                    self.hitbox.top = sprite.hitbox.bottom
                    self.pos.y = self.hitbox.y
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
def cooldowns(self):
    current_time = pygame.time.get_ticks()

    if self.attacking:
        if current_time - self.attack_time >= self.attack_cooldown + self.settings.weapon_data[self.weapon]['cooldown']:
            if not self.holding_attack_btn:
                self.attacking = False
                self.attacking_reset = True
                self.finish_attacking = pygame.time.get_ticks()
                if self.destroy_attack != None:
                    self.destroy_attack()
        if current_time - self.attack_time >= self.settings.weapon_data[self.weapon]['cooldown']:
            if self.holding_attack_btn and self.weapon_type == "gun":
                if self.create_bullet:
                    self.create_bullet()
                self.attack_time = current_time

    if self.magic_attacking:
        if current_time - self.attack_time >= self.settings.magic_data[self.magic]['cooldown']:
            if not self.holding_magic_btn:
                self.magic_attacking = False
                self.attacking_reset = True
                self.finish_attacking = pygame.time.get_ticks()

    if self.attacking_reset:
        if current_time - self.finish_attacking >= self.settings.attack_or_magic_cooldown:
            self.attacking_reset = False

    if not self.can_switch_weapon:
        if current_time - self.weapon_switch_time >= self.switch_duration_cooldown:
            self.can_switch_weapon = True

    if not self.can_switch_magic:
        if current_time - self.magic_switch_time >= self.switch_duration_cooldown:
            self.can_switch_magic = True

    if not self.vulnerable:
        if current_time - self.hurt_time >= self.invulnerability_duration:
            self.vulnerable = True

def get_image_name(self):
    if "../" in self.image_name:
        self.image_name = self.image_name.replace("../", "")
    return self.image_name

def animate(self, dt):
    if not self.paused:
        animation = self.animations[self.status]
        image = self.animation_images[self.status]

        # loop over the frame index
        self.frame_index += self.animation_speed * dt
        if self.frame_index >= len(animation):
            self.frame_index = 0
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
if self.frame_index >= len(animation):
    self.frame_index = 0

    # set the image
    self.image = animation[int(self.frame_index)]
    self.image_name = image[int(self.frame_index)]
    self.rect = self.image.get_rect(center = self.hitbox.center)

    #flicker
    self.flicker()

def set_opacity(self):
    self.image.set_alpha(self.alpha)

def return_alpha(self):
    return self.image_alpha

def flicker(self):
    # flicker
    if not self.vulnerable:
        self.image_alpha = self.wave_value()
        self.image.set_alpha(self.image_alpha)
    else:
        self.image.set_alpha(255)

def get_full_weapon_damage(self):
    base_damage = self.stats['attack']
    weapon_damage = self.settings.weapon_data[self.weapon]['damage']
    return base_damage + weapon_damage

def get_full_magic_damage(self):
    base_damage = self.stats['magic']
    spell_damage = self.settings.magic_data[self.magic]['strength']
    return base_damage + spell_damage

def get_value_by_index(self,index):
    return list(self.stats.values())[index]

def get_cost_by_index(self,index):
    return list(self.upgrade_cost.values())[index]

def energy_recovery(self):
    if self.energy < self.stats['energy']:
        self.energy += 0.002 * self.stats['magic']
    else:
        self.energy = self.stats['energy']

def damage_player(self,damage):
    if self.vulnerable:
        self.vulnerable = False
        self.hurt_time = pygame.time.get_ticks()
        if self.health - damage <= 0:
            self.health = 0
            self.status_alive = "dead"
        else:
            self.health -= damage
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
def return_alive_status(self):
    return self.status_alive

def wave_value(self):
    value = math.sin(pygame.time.get_ticks())
    if value >= 0:
        return 255
    else:
        return 0

def draw(self, offset_pos, color=(190, 40, 50)):
    # Draw the player.
    alpha = 255
    if self.status_alive == "dead":
        color = (128, 128, 128)
        alpha = 135
    draw_image = self.image.copy()
    draw_image.set_alpha(alpha)
    text_surface = self.large_font.render(self.username, True, color)
    text_rect = text_surface.get_rect()
    username_pos = (offset_pos[0] + self.rect.width // 2 - text_surface.get_width() // 2, offset_pos[1] - 35)

    # Define the dimensions and position of the black box
    box_width = text_rect.width + 6 # Adjust the width as needed
    box_height = text_rect.height + 6 # Adjust the height as needed
    box_pos = (username_pos[0]-3, username_pos[1]-3) # Adjust the position as needed

    # Draw the black box
    pygame.draw.rect(self.screen, (27,31,35), ((box_pos[0]-2,box_pos[1]-2), (box_width+4, box_height+4)),3,10)
    #pygame.draw.rect(self.screen, (27,31,35), (box_pos, (box_width, box_height)),0,10)
    self.screen.blit(draw_image, offset_pos)
    self.screen.blit(text_surface, username_pos)

def update(self, dt):
    self.input()
    self.cooldowns()
    self.get_status()
    self.animate(dt)
    self.move(self.speed, dt)
    self.energy_recovery()
```

Server:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
from Scripts.logger import *
try:
    from _thread import *
    from random import randint, choice
    import string, math, socket, time
    from Scripts.settings import Config
    from Scripts.encryption import *
    from Scripts.debug import debug
except:
    logger.critical("You do not have all the modules installed. Please install Pygame, RSA and Pycryptodome.")

logger.debug("RealDL Server Code.")

class Server(Config):
    def __init__(self):
        try:
            Config.__init__(self)
            self.initialize_server()
        except:
            logger.error(f"Couldn't initialize server.")

    def initialize_server(self):
        self.ready_to_play = False
        self.server = self.SERVER
        self.port = self.PORT
        self.players = {}
        self.connections = 0
        self.running = True
        self.s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.rsa_keys = RSA_Keys(self.ENCRYPTION_DATA_SIZE)
        self.public_key, self.private_key = self.rsa_keys.export_keys()
        self.rsa_encrypt = RSA_Encryption(self.public_key)
        self.coordinates = [
            [[1030, 1270],[1278, 1616]],
            [[1100, 1660],[1278, 2180]],
            [[1420, 580],[2494, 432]],
            [[1670, 245],[1865, 592]],
            [[2250, 2485],[2622, 2896]],
            [[3134, 2485],[2818, 2832]]
        ]

        # Speed
        self.speed_leeway = 100
        self.allowed_speed = 10*(self.frame_increase_rate/1.55)
        # Health
        self.allowed_health = 300
        self.health_leeway = 15

        try:
            self.s.bind((self.server, self.port))
        except socket.error as e:
            logger.error(f"Socket Error: {e}")
        self.s.listen()
    logger.info("Waiting for connections, Server Started")
    logger.info(f"Server IP Address: {self.SERVER}")
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

def create_random_string(self):
  characters = string.ascii_letters + string.digits
  return ''.join(choice(characters) for _ in range(self.ID_STRING_LENGTH))

def is_touching(self, player_x, player_y, threshold=100):
  touching = False
  for player in self.players.values():
    # If two players are taking both an x and y position. We want to seperate player spawns.
    # The threshold is default at 100. We dont want players to close to each other.
    if abs(player['player']['x']-player_x) <= self.SQUARE_SIZE + threshold and abs(player['player']['y']-player_y) <= self.SQUARE_SIZE + threshold:
      touching = True
  return touching

def get_player_position(self):
  def create_coords():
    random_area = choice(self.coordinates)
    if random_area == self.coordinates[2]:
      x = randint(random_area[0][0], random_area[1][0])
      y = randint(random_area[1][1], random_area[0][1])
    elif random_area == self.coordinates[5]:
      x = randint(random_area[1][0], random_area[0][0])
      y = randint(random_area[0][1], random_area[1][1])
    else:
      x = randint(random_area[0][0], random_area[1][0])
      y = randint(random_area[0][1], random_area[1][1])
    return x,y

    valid_coords = False
    while not valid_coords:
      x, y = create_coords()
      if not self.is_touching(x, y):
        valid_coords = True
    return x, y

  def validate_username(self, username):
    username_valid = False
    indent = 0
    usernames = [player['player']['username'] for player in self.players.values()]
    new_username = username
    while not username_valid:
      if indent > 0:
        new_username = username + f"_{str(indent)}"
      if new_username in usernames:
        indent += 1
      else:
        username_valid = True
    return new_username

  def create_new_player(self, username):
    key_string = self.create_random_string()
    player_x, player_y = self.get_player_position()
    username = self.validate_username(username)
    return {
      "player":{
        "x": player_x,
        "y": player_y,
        "image": "Graphics/Game/player/down_idle/idle_down.png",
        "username": username,
      }
    }
  
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
"status": "alive",
"health": 100,
"id": key_string,
"kill_count":0,
"killed_by":"",
],
"weapon":[],
"bullets":[],
"magic":[]}, key_string

def validate_movement(self, player_pos):
def calculate_speed(pos1, pos2, delta_time):
    # Calculate distance between two positions using Euclidean distance formula
    try:
        delta_x = pos2[0] - pos1[0]
        delta_y = pos2[1] - pos1[1]
        distance = math.sqrt(delta_x**2 + delta_y**2)

        # Calculate speed (distance / time)
        speed = distance / delta_time
    except:
        speed = self.allowed_speed
    return speed

    # Calculate the distance moved between updates
first_pos = player_pos[0][:2]
last_pos = player_pos[-1][:2]
total_time = sum(values[2] for values in player_pos)

    # Calculate speed between the first and last positions
speed = calculate_speed(first_pos, last_pos, total_time)
if speed > self.allowed_speed + self.speed_leeway:
    return False
return True

def validate_health(self, player):
    if player['health'] > self.allowed_health + self.health_leeway:
        return False
    else:
        return True

def validate_player_status(self, player, data):
    if player['status'] == "dead" and data['status'] == 'alive':
        return player['status']
    elif player['status'] == "alive" and data['status'] == 'dead':
        return data['status']
    else:
        return player['status']

def kill_count(self, key_string, kill_count, killed_players):
    kills = []

    # gets a list of all ids of players that have killed someone
    for player in self.players.values():
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
if player['player']['killed_by'] != "":
    if player['player']['id'] not in killed_players:
        kills.append(player['player']['killed_by'])
        killed_players.append(player['player']['id'])

# checks if this player has killed anyone
for kill_id in kills:
    # logger.debug(kill_count, kill_id, key_string)
    if kill_id == key_string:
        kill_count += 1
return kill_count, killed_players

def end_game(self):
    player_count = 0
    for player in self.players.values():
        if player['player']['status'] == "alive":
            player_count += 1
    if player_count <= 1:
        return True
    else:
        return False

def validate_position(self, player):
    # Map boundaries, stops players from leaving the map.
    if player['x'] <= 180 or player['x'] >= 3400:
        return True
    elif player['y'] <= -50 or player['y'] >= 3150:
        return True
    else:
        return False

def handle_client_communication(self, conn, key_string, aes_encryption):
    running = True
    player_pos = []
    killed_players = []
    last_time = time.time()
    end_time = 0
    reset_game_time = 15
    while running:
        try:
            ## ONE player left. -> after 30 seconds -> kick all players -> reset all key values

            # Recieve player data
            player_data = aes_encryption.decrypt(self.unserialize(conn.recv(self.DATA_SIZE)))

            # check the player's status is correct (this doesn't result in a kick)
            data = player_data['player']
            player_data['player']['status'] = self.validate_player_status(self.players[key_string]['player'], data)

            # Validate if the player has been killed.
            player_data['player']['kill_count'], killed_players = self.kill_count(key_string, player_data['player']['kill_count'], killed_players)

            self.players[key_string] = player_data

            if self.end_game():
                break
        except:
            break
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
    if time.time() - end_time >= reset_game_time:
        running = False
    else:
        end_time = time.time()

    # Calculate the speed of play (FPS).
    delta_time = time.time() - last_time
    last_time = time.time()

    # Validate the player's position.
    if self.validate_position(self.players[key_string]['player']):
        running = False
        logger.info(f"Player {key_string} disconnected as they tried to leave the map.")

    # Validate the player data to ensure they are not speed hacking. (this does result in a kick)
    player_pos.append([self.players[key_string]['player']['x'], self.players[key_string]['player']['y'], delta_time])
    if len(player_pos) > 20: player_pos.pop(0)
    if not self.validate_movement(player_pos):
        running = False
        logger.info(f"Player {key_string} disconnected because they were speed hacking.")

    # Validate player health to ensure they are not health hacking. (this does result in a kick)
    if not self.validate_health(self.players[key_string]['player']):
        running = False
        logger.info(f"Player {key_string} disconnected because they were health hacking.")

    #logger.info(f"Received Player Dict: {player_data}.")
```

if not data:
 logger.info(f"Player {key_string} disconnected.")
 running = False
else:
 reply = self.players
 encrypted_reply = self.serialize(aes_encryption.encrypt(reply))

 conn.sendall(encrypted_reply)
 #logger.info(f"Sending All Player Dict: {reply}.")
except:
 logger.info(f"Player {key_string} lost connection.")
 running = False

logger.info(f"Connection Closed for Player {key_string}.")
try:
 del self.players[key_string]
 self.connections -= 1
except:
 logger.info(f"No player with the ID: {key_string} exists.")
conn.close()

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
def waiting_zone(self, conn, key_string, aes_encryption):
    running = True
    while running:
        try:
            # send over number of players connected
            reply = {"connections":self.connections,"started":self.ready_to_play}
            encrypted_reply = self.serialize(aes_encryption.encrypt(reply))

            conn.sendall(encrypted_reply)
            # Receive start or not.
            start_dict = aes_encryption.decrypt(self.deserialize(conn.recv(self.DATA_SIZE)))
            start = start_dict["ready"]
            if start and self.connections >= 2:
                self.ready_to_play = True
        except:
            running = False
            logger.error("Something went wrong.")

        if start:
            self.handle_client_communication(conn, key_string, aes_encryption)

    try:
        del self.players[key_string]
        self.connections -= 1
    except:
        logger.info(f"No player with the ID: {key_string} exists.")
    conn.close()

def threaded_client(self, conn):
    try:
        # Send Public Key
        data_to_send = self.serialize(self.public_key)
        conn.send(data_to_send)
        #logger.info(f"Sending Public Key: {self.public_key}")

        # Get AES Key and username
        dict_received_from_client = self.rsa_encrypt.decrypt(self.deserialize(conn.recv(self.ENCRYPTION_DATA_SIZE)),self.private_key)
        aes_key = dict_received_from_client['aes_key']
        username = dict_received_from_client['username']
        aes_encryption = AES_Encryption(aes_key)
        #logger.info(f"Received AES Key: {aes_key} and Username: {username}")

        # Create Player
        new_player, key_string = self.create_new_player(username)
        self.players[key_string] = new_player
        #logger.info(f"Created New Player: {new_player}. ID: {key_string}")

        #Send player dict
        player_dict_send = {'player_data':new_player}
        encrypted_player = self.serialize(aes_encryption.encrypt(player_dict_send))
        conn.send(encrypted_player)
        logger.info(f"Sending Player dict to client: {new_player}")
        self.waiting_zone(conn, key_string, aes_encryption)
        #self.handle_client_communication(conn, key_string, aes_encryption)
    except:
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
#logger.error("An Error Occurred trying to setup Client-Server co
try:
    del self.players[key_string]
    self.connections -= 1
except:
    logger.info(f"No player was deleted.")
conn.close()

def run(self):
    while self.running:
        ## Limit on 5 players
        conn, addr = self.s.accept()
        logger.debug(f"Connections: {str(self.connections)}")

        if self.ready_to_play and self.connections == 0:
            self.ready_to_play = False

        if self.ready_to_play or self.connections >= 5:
            conn.close()
        else:
            self.connections += 1
#logger.info(f"Connected to: {addr}")
#logger.info(f"There {'is' if self.connections==1 else 'are'} {se
            start_new_thread(self.threaded_client, (conn,))

if __name__ == "__main__":
    try:
        server = Server()
        server.run()
    except:
        logger.critical("Server has failed to launch.")
```

Settings:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
from Scripts.logger import *
try:
    from socket import gethostname, gethostbyname
    import pickle, pygame
except:
    logger.critical("You do not have all the modules installed. Please install Pygame.")
pygame.init()

logger.info("Github: https://github.com/TheRealDL1/Simple-Client-Server")
logger.debug("RealDL Settings Code.")

class Config():
    def __init__(self):
        # Screen Width and Height
        info = pygame.display.Info()
        max_width = info.current_w
        max_height = info.current_h
        self.HEIGHT = max_height # 720
        self.WIDTH = max_width # 1280
        self.frame_increase_rate = 100

        # Other Server/ Client Settings
        self.SQUARE_SIZE = 64
        self.BITS = 256
        self.FPS = 60
        self.HOST_NAME = gethostname()
        self.SERVER = gethostbyname(self.HOST_NAME)
        self.PORT = 5555
        self.ID_STRING_LENGTH = 30
        self.BG_COLOR = (113, 221, 238)
        self.DATA_SIZE = 8192
        self.SMALL_DATA = 32
        self.ENCRYPTION_DATA_SIZE = 1024
        self.TILESIZE = 64
        self.TITLE_ID = 'Tile'

        # weapons
        self.attack_or_magic_cooldown = 250
        self.weapon_data = {
            'sword': {'cooldown': 100, 'speed': 0, 'damage': 15, 'graphic': 'Graphics/Game/weapons/sword/full.png', 'type': 'melee'},
            'lance': {'cooldown': 400, 'speed': 0, 'damage': 30, 'graphic': 'Graphics/Game/weapons/lance/full.png', 'type': 'melee'},
            'axe': {'cooldown': 300, 'speed': 0, 'damage': 20, 'graphic': 'Graphics/Game/weapons/axe/full.png', 'type': 'melee'},
            'rapier': {'cooldown': 50, 'speed': 0, 'damage': 8, 'graphic': 'Graphics/Game/weapons/rapier/full.png', 'type': 'melee'},
            'sai': {'cooldown': 80, 'speed': 0, 'damage': 10, 'graphic': 'Graphics/Game/weapons/sai/full.png', 'type': 'melee'},
            'revolver': {'cooldown': 1000, 'speed': 20, 'damage': 25, 'graphic': 'Graphics/Game/weapons/revolver/full.png', 'type': 'gun'},
            'msg': {'cooldown': 175, 'speed': 10, 'damage': 5, 'graphic': 'Graphics/Game/weapons/msg/full.png', 'type': 'gun'},
            'pistol': {'cooldown': 600, 'speed': 15, 'damage': 18, 'graphic': 'Graphics/Game/weapons/pistol/full.png', 'type': 'gun'}}
        }

        # Bullets

        self.bullet_type = {
            'msg': {
                'down': 'Graphics/Game/bullets/msg/down.png',
                'left': 'Graphics/Game/bullets/msg/left.png',
                'right': 'Graphics/Game/bullets/msg/right.png',
                'up': 'Graphics/Game/bullets/msg/up.png',
            },
        }
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

'pistol': {
    'down': 'Graphics/Game/bullets/pistol/down.png',
    'left': 'Graphics/Game/bullets/pistol/left.png',
    'right': 'Graphics/Game/bullets/pistol/right.png',
    'up': 'Graphics/Game/bullets/pistol/up.png',
},
'revolver': {
    'down': 'Graphics/Game/bullets/revolver/down.png',
    'left': 'Graphics/Game/bullets/revolver/left.png',
    'right': 'Graphics/Game/bullets/revolver/right.png',
    'up': 'Graphics/Game/bullets/revolver/up.png',
}
}

# magic
self.magic_data = {
    'flame': {'strength': 5, 'cost': 20, 'graphic': 'Graphics/Game/particles/flame/fire.png', 'type': 'magic', 'cooldown': 1250},
    'heal': {'strength': 20, 'cost': 10, 'graphic': 'Graphics/Game/particles/heal/heal.png', 'type': 'magic', 'cooldown': 650}
}

# ui
self.BAR_HEIGHT = 20
self.HEALTH_BAR_WIDTH = 200
self.ENERGY_BAR_WIDTH = 140
self.ITEM_BOX_SIZE = 90
self.UI_FONT = 'Graphics/Fonts/Orbitron-Medium.ttf'
self.UI_FONT_SIZE = 18

# general colors
self.WATER_COLOR = (113, 221, 238)
self.UI_BG_COLOR = (34, 34, 34)
self.UI_BORDER_COLOR = (17, 17, 17)
self.TEXT_COLOR = (238, 238, 238)

# ui colors
self.HEALTH_COLOR = (255, 0, 0)
self.ENERGY_COLOR = (23, 108, 235)
self.UI_BORDER_COLOR_ACTIVE = (255, 215, 0)

# upgrade menu
self.TEXT_COLOR_SELECTED = (17, 17, 17)
self.BAR_COLOR = (238, 238, 238)
self.BAR_COLOR_SELECTED = (17, 17, 17)
self.UPGRADE_BG_COLOR_SELECTED = (238, 238, 238)

def serialize(self, data):
    try:
        return pickle.dumps(data)
    except pickle.PicklingError as Error:
        logger.error(f"Failed to pickle data: {Error}")
    return None

def unserialize(self, data):
    try:
        return pickle.loads(data)
    except pickle.UnpicklingError as Error:
        logger.error(f"Failed to unpickle data: {Error}")

This class is used for the client and the server.
  
```

Support:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
from Scripts.logger import *
try:
    from csv import reader
    from os import walk, path
    import pygame
except:
    logger.critical("You do not have all the modules installed. Please install Pygame.")

logger.debug("RealDL Support Code.")

def import_csv_layout(path):
    terrain_map = []

    try:
        with open(path) as level_map:
            layout = reader(level_map, delimiter = ',')
            for row in layout:
                terrain_map.append(list(row))
    except:
        with open(f"../{path}") as level_map:
            layout = reader(level_map, delimiter = ',')
            for row in layout:
                terrain_map.append(list(row))
    return terrain_map

def import_folder(folder_path, return_image_list=False):
    surface_list = []
    image_list = []
    values = False

    while not values:
        for root, dirs, img_files in walk(folder_path):
            for image in img_files:
                full_path = path.join(root, image)
                try:
                    image_surf = pygame.image.load(full_path).convert_alpha()
                except:
                    image_surf = pygame.image.load(f"../{full_path}").convert_alpha()
                surface_list.append(image_surf)
                image_list.append(full_path)
        if surface_list and image_list:
            values = True
        else:
            if "../" in folder_path:
                values = True
            else:
                folder_path = "../" + folder_path

    if return_image_list == True:
        return surface_list, image_list
    elif return_image_list == None:
        return image_list
    else:
        return surface_list
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

Tile:

```
from Scripts.logger import *
try:
    import pygame
    from Scripts.settings import *
except:
    logger.critical("You do not have all the modules installed. Please install Pygame.")

logger.debug("RealDL Tile Code.")

class Tile(pygame.sprite.Sprite):
    def __init__(self, pos, groups, sprite_type, surface, id):
        try:
            super().__init__(groups)
            self.settings = Config()
            self.sprite_type = sprite_type
            self.id = id
            self.image = surface
            if sprite_type == 'object':
                self.rect = self.image.get_rect(topleft = (pos[0], pos[1] - self.settings.TILESIZE))
            else:
                self.rect = self.image.get_rect(topleft = pos)
            self.hitbox = self.rect.inflate(-10,-10)
            self.old_hitbox = self.hitbox.copy()

        except:
            logger.error(f"Failed to create Tile Class.")
```

UI:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
from Scripts.logger import *
try:
    from Scripts.functions import *
    from Scripts.settings import Config
    import pygame
except:
    logger.critical("You do not have all the modules installed. Please install Pygame.")

logger.debug("RealDL UI Code.")

class UI(Config):
    def __init__(self, custom_mouse, quit_function, player_count, kill_count):
        # Setup Pygame Variables
        Config.__init__(self)
        self.screen = pygame.display.get_surface()
        info = pygame.display.Info()
        self.screen_width = info.current_w
        self.screen_height = info.current_h
        self.DEFAULT_WIDTH = 1920
        self.DEFAULT_HEIGHT = 1080
        self.width_ratio = self.screen_width / self.DEFAULT_WIDTH
        self.height_ratio = self.screen_height / self.DEFAULT_HEIGHT
        self.custom_mouse = custom_mouse
        self.draw_ui = False
        self.quit_function = quit_function

        # Constants setup
        self.BIG_TEXT_SIZE = 50
        self.BASE_TEXT_SIZE = 15
        self.TEXT_HEIGHT = 20
        self.IMAGE_WIDTH = 64
        self.IMAGE_HEIGHT = 64
        self.IMAGE_PADDING = 20
        self.BUTTON_PADDING = 85
        self.CURVE = 10
        self.THICKNESS = 2
        self.BASE_BUTTON_WIDTH = 250
        self.BASE_BUTTON_HEIGHT = 70

        # Variable setup
        self.image_width = int(self.IMAGE_WIDTH * self.width_ratio)
        self.image_height = int(self.IMAGE_HEIGHT * self.height_ratio)
        self.image_padding = int(self.IMAGE_PADDING * self.width_ratio)
        self.button_padding = int(self.BUTTON_PADDING * self.height_ratio)
        self.text_height = int(self.TEXT_HEIGHT * self.height_ratio)
        self.base_text_size = int(self.BASE_TEXT_SIZE * self.height_ratio)
        self.big_text_size = int(self.BIG_TEXT_SIZE * self.height_ratio)
        self.curve = int(self.CURVE * self.height_ratio)
        self.thickness = int(self.THICKNESS * self.height_ratio)
        self.base_button_width = int(self.BASE_BUTTON_WIDTH * self.width_ratio)
        self.base_button_height = int(self.BASE_BUTTON_HEIGHT * self.height_ratio)
```

Candidate name: Dominic Pike
 Candidate number: 4172
 Centre name: Howard of Effingham
 Centre number: 64335

```

# UI Board
self.border = Button((27, 31, 35), (27, 31, 35), self.screen_width / 2, self.screen_height / 2, "Graphics/Fonts/Orbitron-Regular.ttf", (27, 31, 35), (27, 31, 35), self.screen_width * 0.7,
                     self.screen_height * 0.7, '', 'Rectangle', None, self.big_text_size, int(self.curve * 1.5))
self.join_boarder = Button((27, 31, 35), (27, 31, 35), (self.screen_width / 2), (self.screen_height / 2), "Graphics/Fonts/Orbitron-Regular.ttf", (240, 240, 240), (136, 173, 227),
                           self.base_button_width * 3, self.base_button_height * 6, '', 'Rectangle', None, self.big_text_size, self.curve)
self.info = Text(self.text_height, "Graphics/Fonts/Orbitron-Regular.ttf", (240, 240, 240), None, None, None, int(self.base_text_size * 1.7))
self.info2 = Text(self.text_height, "Graphics/Fonts/Orbitron-Regular.ttf", (240, 240, 240), None, None, None, int(self.base_text_size * 1.7))
self.quit = Button((174, 39, 96), (27, 31, 35), self.screen_width / 2, self.screen_height / 2, "Graphics/Fonts/Orbitron-Medium.ttf", (27, 31, 35), (174, 39, 96), self.base_button_width,
                  self.base_button_height, 'Quit', 'Rectangle', None, int(self.big_text_size / 1.3), self.curve)
self.bullet_assault = Text(self.text_height, "Graphics/Fonts/Orbitron-Bold.ttf", (240, 240, 240), None, None, None, int(self.base_text_size * 3.5))
self.continue_btn = Button((39, 174, 96), (27, 31, 35), (self.screen_width / 2), (self.screen_height / 2) * 1.445 - self.button_padding, "Graphics/Fonts/Orbitron-Medium.ttf", (27, 31, 35),
                           (39, 174, 96), self.base_button_width, self.base_button_height, 'Continue', 'Rectangle', None, int(self.big_text_size / 1.3), self.curve)

# In-game UI. Kill count,
# general
try:
    self.font = pygame.font.Font(self.UI_FONT, self.UI_FONT_SIZE)
except:
    self.font = pygame.font.Font(f"../{self.UI_FONT}", self.UI_FONT_SIZE)
self.player_count = player_count
self.player_kill_count = kill_count

# Kill count and player count
self.boarder_stats = Button(self.UI_BG_COLOR, self.UI_BG_COLOR, self.WIDTH-(self.HEALTH_BAR_WIDTH)/2-10-2, 10+(self.BAR_HEIGHT*1.25)+2, "Graphics/Fonts/Orbitron-Medium.ttf", self.TEXT_COLOR,
                            self.TEXT_COLOR, (self.HEALTH_BAR_WIDTH), self.BAR_HEIGHT*2.5, "", 'Rectangle', None, self.UI_FONT_SIZE, 0)
self.kill_count = Button(self.UI_BG_COLOR, self.UI_BG_COLOR, self.WIDTH-(self.HEALTH_BAR_WIDTH)/2-10-2, 13.5+(self.BAR_HEIGHT/2)+2, "Graphics/Fonts/Orbitron-Medium.ttf", self.TEXT_COLOR,
                        self.TEXT_COLOR, (self.HEALTH_BAR_WIDTH), self.BAR_HEIGHT, f"Kill count: {self.player_kill_count}", 'Rectangle', None, self.UI_FONT_SIZE, 0)
self.players_alive = Button(self.UI_BG_COLOR, self.UI_BG_COLOR, self.WIDTH-(self.HEALTH_BAR_WIDTH)/2-10-2, self.BAR_HEIGHT+16.5+(self.BAR_HEIGHT/2)+2, "Graphics/Fonts/Orbitron-Medium.ttf",
                           self.TEXT_COLOR, self.TEXT_COLOR, (self.HEALTH_BAR_WIDTH), self.BAR_HEIGHT, f"Players alive: {self.player_count}", 'Rectangle', None, self.UI_FONT_SIZE, 0)

# bar setup
self.health_bar_rect = pygame.Rect(10, 10, self.HEALTH_BAR_WIDTH, self.BAR_HEIGHT)
self.energy_bar_rect = pygame.Rect(10, 34, self.ENERGY_BAR_WIDTH, self.BAR_HEIGHT)

# convert weapon dictionary
self.weapon_graphics = []
for weapon in self.weapon_data.values():
    weapon_path = weapon['graphic']
    try:
        weapon = pygame.image.load(weapon_path).convert_alpha()
    except:
        weapon = pygame.image.load(f"../{weapon_path}").convert_alpha()
    self.weapon_graphics.append(weapon)

# convert magic dictionary
self.magic_graphics = []
for magic in self.magic_data.values():
    magic_path = magic['graphic']
    try:
        magic = pygame.image.load(magic_path).convert_alpha()
    except:
        magic = pygame.image.load(f"../{magic_path}").convert_alpha()
    self.magic_graphics.append(magic)

```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
def stop_drawing(self):
    self.draw_ui = False

def draw_menu(self):
    if self.draw_ui:
        self.border.draw((27, 31, 35), None, None, False)
        self.join_boarder.draw((240, 240, 240))
        self.info.draw("draw", "Game Paused: You have entered the main menu.", (self.screen_width / 2), (self.screen_height / 2) * 0.665)
        self.info2.draw("draw", "Movement is currently disabled.", (self.screen_width / 2), (self.screen_height / 2) * 0.72)
        self.quit.draw((240, 240, 240), self.quit_function)
        self.continue_btn.draw((240, 240, 240), self.stop_drawing)
        self.bullet_assault.draw("draw", "Bullet Assault", (self.screen_width / 2), (self.screen_height / 2) * 0.43)

        start_back_hover = self.quit.is_hovered()
        join_hover = self.continue_btn.is_hovered()

        start_back_click = self.quit.is_clicking()
        join_click = self.continue_btn.is_clicking()

        if start_back_hover or join_hover:
            if not start_back_click and not join_click:
                self.custom_mouse.mode = 1
            else:
                self.custom_mouse.mode = 2
        else:
            self.custom_mouse.mode = 0

        # Draw the mouse
        self.custom_mouse.draw()

def show_info(self):
    self.boarder_stats.draw(self.UI_BORDER_COLOR,None,None,False)
    self.kill_count.draw(None,None,None,False,f"Kill count: {self.player_kill_count}")
    self.players_alive.draw(None,None,None,False,f"Players alive: {self.player_count}")

def show_bar(self, current, max_amount, bg_rect, color):
    # draw bg
    pygame.draw.rect(self.screen, self.UI_BG_COLOR, bg_rect)

    # converting stat to pixel
    ratio = current / max_amount
    current_width = bg_rect.width * ratio
    current_rect = bg_rect.copy()
    current_rect.width = current_width

    # drawing the bar
    pygame.draw.rect(self.screen, color, current_rect)
    pygame.draw.rect(self.screen, self.UI_BORDER_COLOR, bg_rect, 3)

def show_exp(self, exp):
    text_surf = self.font.render(str(int(exp)), False, self.TEXT_COLOR)
    x = self.screen.get_size()[0] - 20
    y = self.screen.get_size()[1] - 20
    text_rect = text_surf.get_rect(bottomright=(x, y))
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
pygame.draw.rect(self.screen, self.UI_BG_COLOR, text_rect.inflate(20, 20))
self.screen.blit(text_surf, text_rect)

def selection_box(self, left, top, has_swapped):
    bg_rect = pygame.Rect(left, top, self.ITEM_BOX_SIZE, self.ITEM_BOX_SIZE)
    pygame.draw.rect(self.screen, self.UI_BORDER_COLOR, bg_rect)
    if has_swapped:
        pygame.draw.rect(self.screen, self.UI_BORDER_COLOR_ACTIVE, bg_rect, 3)
    else:
        pygame.draw.rect(self.screen, self.UI_BORDER_COLOR, bg_rect, 3)
    return bg_rect

def weapon_overlay(self, weapon_index, has_swapped):
    bg_rect = self.selection_box(10, self.HEIGHT-self.ITEM_BOX_SIZE-10, has_swapped)
    weapon_surf = self.weapon_graphics[weapon_index]
    weapon_rect = weapon_surf.get_rect(center=bg_rect.center)

    self.screen.blit(weapon_surf, weapon_rect)

def magic_overlay(self, magic_index, has_swapped):
    bg_rect = self.selection_box(10+self.ITEM_BOX_SIZE+10, self.HEIGHT-self.ITEM_BOX_SIZE-10, has_swapped)
    magic_surf = self.magic_graphics[magic_index]
    magic_rect = magic_surf.get_rect(center=bg_rect.center)

    self.screen.blit(magic_surf, magic_rect)

def display(self, player):
    self.show_bar(player.health, player.stats['health'], self.health_bar_rect, self.HEALTH_COLOR)
    self.show_bar(player.energy, player.stats['energy'], self.energy_bar_rect, self.ENERGY_COLOR)

    self.show_exp(player.exp)
    self.show_info()

    self.weapon_overlay(player.weapon_index, not player.can_switch_weapon)
    self.magic_overlay(player.magic_index, not player.can_switch_magic)
```

Weapon:

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
from Scripts.logger import *
try:
    import pygame, string, random
    from Scripts.settings import *
except:
    logger.critical("You do not have all the modules installed. Please install Pygame.")

logger.debug("RealDL Weapon Code.")

class Melee(pygame.sprite.Sprite):
    def __init__(self, player, groups, player_id):
        try:
            super().__init__(groups)
            self.sprite_type = "weapon"
            self.settings = Config()
            self.id = self.create_id()
            self.player_id = player_id
            self.time = pygame.time.get_ticks()
            self.type = player.weapon_type
            self.last_shot_time = 0 # Initialize the last shot time to 0
            self.damage = player.attack_strength + self.settings.weapon_data[player.weapon]['damage']
            self.damaged_player = False
            direction = player.status.split('_')[0]

            # Load the image
            self.full_path = f'Graphics/Game/weapons/{player.weapon}/{direction}.png'
            try:
                self.image = pygame.image.load(self.full_path).convert_alpha()
            except:
                self.image = pygame.image.load(f"../{self.full_path}").convert_alpha()

            # Set the placement of the sprite
            if direction == 'right':
                self.rect = self.image.get_rect(midleft=player.rect.midright + pygame.math.Vector2(0, 16))
            elif direction == 'left':
                self.rect = self.image.get_rect(midright=player.rect.midleft + pygame.math.Vector2(0, 16))
            elif direction == 'down':
                self.rect = self.image.get_rect(midtop=player.rect.midbottom + pygame.math.Vector2(-10, 0))
            else:
                self.rect = self.image.get_rect(midbottom=player.rect.midtop + pygame.math.Vector2(-10, 0))
        except:
            logger.error("Failed to create Weapon")

    def create_id(self):
        try:
            characters = string.ascii_letters + string.digits
            return ''.join(random.choice(characters) for _ in range(self.settings.ID_STRING_LENGTH))
        except:
            logger.error("Something went wrong with creating a unique ID.")
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
class Weapon(pygame.sprite.Sprite):
    def __init__(self, groups, x, y, id, image, player_id, damage=None, sprite_type="weapon_copy"):
        super().__init__(groups)
        self.sprite_type = sprite_type
        self.id = id
        self.player_id = player_id
        self.x = x
        self.y = y
        self.full_path = image
        self.damage = damage
        self.damaged_player = False
        try:
            self.image = pygame.image.load(self.full_path).convert_alpha()
        except:
            self.image = pygame.image.load(f"../{self.full_path}").convert_alpha()

        self.rect = self.image.get_rect()
        self.rect.x = self.x
        self.rect.y = self.y

    def return_image(self):
        if "../" in self.full_path:
            self.full_path = self.full_path.replace("../", "")
        return self.full_path

class Bullets(pygame.sprite.Sprite):
    def __init__(self, player, groups, obstacle_sprites, player_id, player_group):
        super().__init__(groups)
        self.settings = Config()
        self.obstacle_sprites = obstacle_sprites
        self.sprite_type = "bullet"
        self.id = self.create_id()
        self.player_id = player_id
        self.player_group = player_group
        self.player = player
        self.direction = player.status.split('_')[0]
        gun = player.weapon
        self.speed = self.settings.weapon_data[gun]['speed']*self.settings.frame_increase_rate/1.5
        self.damage = player.attack_strength + self.settings.weapon_data[gun]['damage']
        self.damaged_player = False
        self.cooldown = self.settings.weapon_data[gun]['cooldown']
        self.collided = False

        self.full_path = self.settings.bullet_type[gun][self.direction]
        try:
            self.image = pygame.image.load(self.full_path).convert_alpha()
        except:
            self.image = pygame.image.load(f"../{self.full_path}").convert_alpha()

    # Set the placement of the sprite
```

Candidate name: Dominic Pike
Candidate number: 4172
Centre name: Howard of Effingham
Centre number: 64335

```
# Set the placement of the sprite
if self.direction == 'right':
    self.rect = self.image.get_rect(midleft=player.rect.midright + pygame.math.Vector2(20, 8))
elif self.direction == 'left':
    self.rect = self.image.get_rect(midright=player.rect.midleft + pygame.math.Vector2(-20, 8))
elif self.direction == 'down':
    self.rect = self.image.get_rect(midtop=player.rect.midbottom + pygame.math.Vector2(-18, 21))
else:
    self.rect = self.image.get_rect(midbottom=player.rect.midtop + pygame.math.Vector2(-18, -21))

def update(self, dt):
    self.collision('obstacle')
    self.collision('player')
    if not self.collided:
        if self.direction == 'right':
            self.rect.x += self.speed * dt
        elif self.direction == 'left':
            self.rect.x -= self.speed * dt
        elif self.direction == 'down':
            self.rect.y += self.speed * dt
        else:
            self.rect.y -= self.speed * dt

    if self.collided:
        self.kill()

def create_id(self):
    try:
        characters = string.ascii_letters + string.digits
        return ''.join(random.choice(characters) for _ in range(self.settings.ID_STRING_LENGTH))
    except:
        logger.error("Something went wrong with creating a unique ID.")

def collision(self, collision_type):
    if collision_type == "obstacle":
        collision_sprites = pygame.sprite.spritecollide(self, self.obstacle_sprites, False)
        if collision_sprites:
            self.collided = True

    if collision_type == "player":
        collision_sprites = pygame.sprite.spritecollide(self, self.player_group, False)
        if collision_sprites:
            self.collided = True
```

Proof: See MP4 videos attached.