Design Documentation

This document will serve as the design basis of the software we are developing. Included are two sets of interaction models, Use Case diagrams and Sequence diagrams; a Structural Class diagram; a Behavioural State diagram and a Context Diagram.

For a visual representation of the final system see the mock UI images at the end.

# Usage Scenarios

We are modelling our usage scenarios from a behavioural perspective as our system uses an event based architecture in part (see System Architecture below).

We have decided the five main scenarios in our system, the first one is being able to **buy items from the shop**. The shop is one of the features in our game that is integral for the user experience, the shop may be viewed and the user may purchase items. The second usage scenarios is to allow the user to **control the character’s** movements, which is important for the user. The third usage scenario is to **view and enter high scores** in the game, which allows the game to be competitive. The fourth usage scenario is to **allow the user to attack the enemies** on command, and for the enemy AI to move and return attacks. The fifth and final usage scenario, is **starting and closing the game**, vital to any application. Below are our use case diagrams for each scenario:

Untitled Diagram.png

Figure 0.1 *Use Case diagram for using the in game shop.*

The user is able to interact with an in game shop to purchase items that affect gameplay. It is possible within the shop to check the currently available inventory as well as purchase items. Purchasing an item updates the player's character data.Untitled Diagram.png

Figure 0.2 *Use Case diagram for controlling the user character.*

An integral part of the game is being able to make the on-screen character move within its environment. The user can use keyboard inputs to make the character move multiple directions. It is necessary to make sure that no collisions occur, if it is found that an object collides with the user’s character then the move will not be executed.

Scores.png

Figure 0.3 *Use Case diagram for inputting new high scores.*

The user is able to view the high scores in the game, which will be displayed as a list. After playing the game, the game will take the name the user has entered to enter a record into the high scores, so the high score list gets updated after each game.

Untitled Diagram.png

Figure 0.4 *Use Case diagram for attacking enemies.*

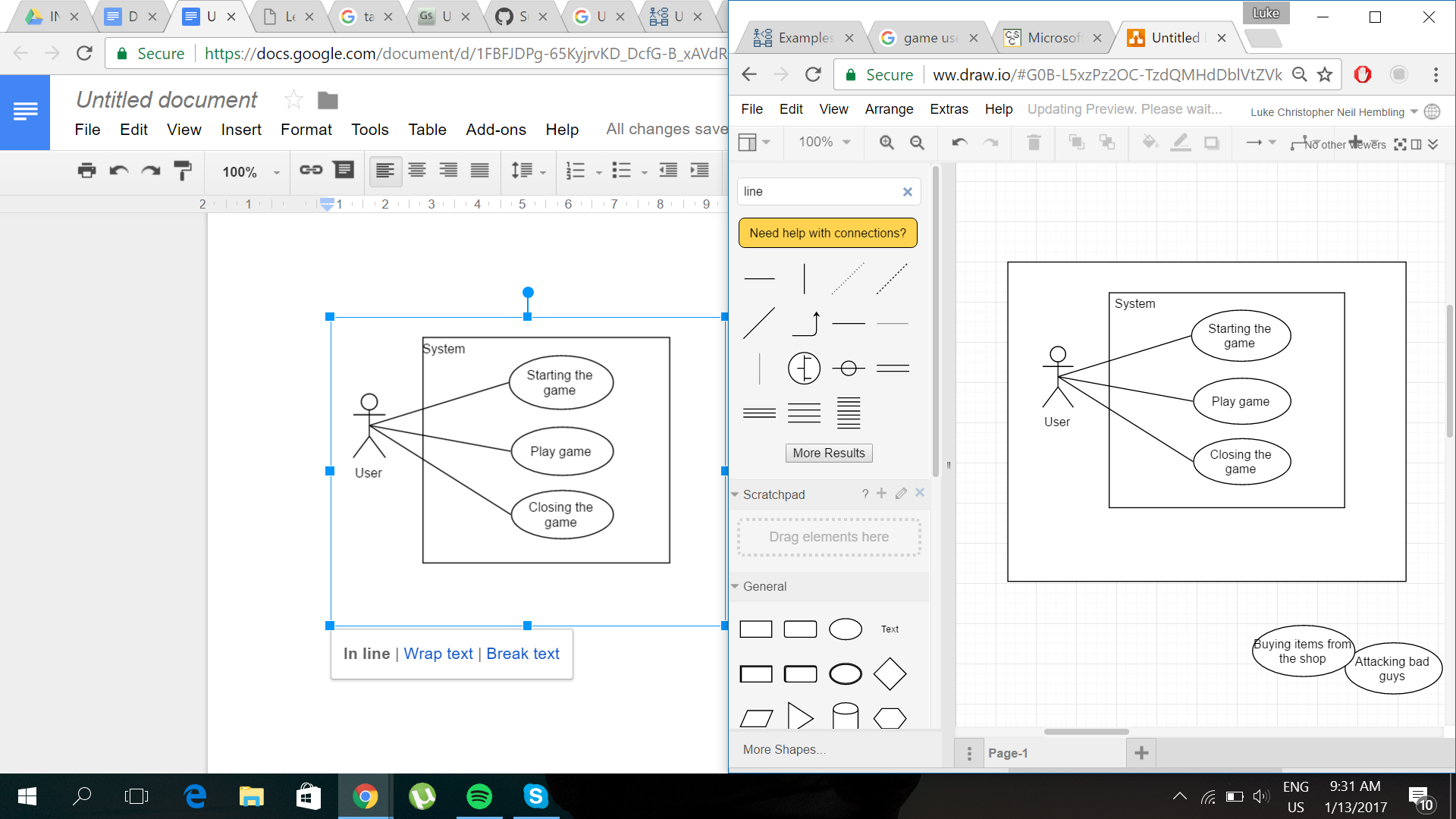
Firing at enemies is a key aspect of the game, and is how the user progresses. Firing at enemies with incite a points increase. It is also important that enemies can be destroyed by the user. Conversely the enemy artificial intelligence(AI) must be capable advancing upon the user character, as well as attacking him/her when within range; this will reduce the user character health.

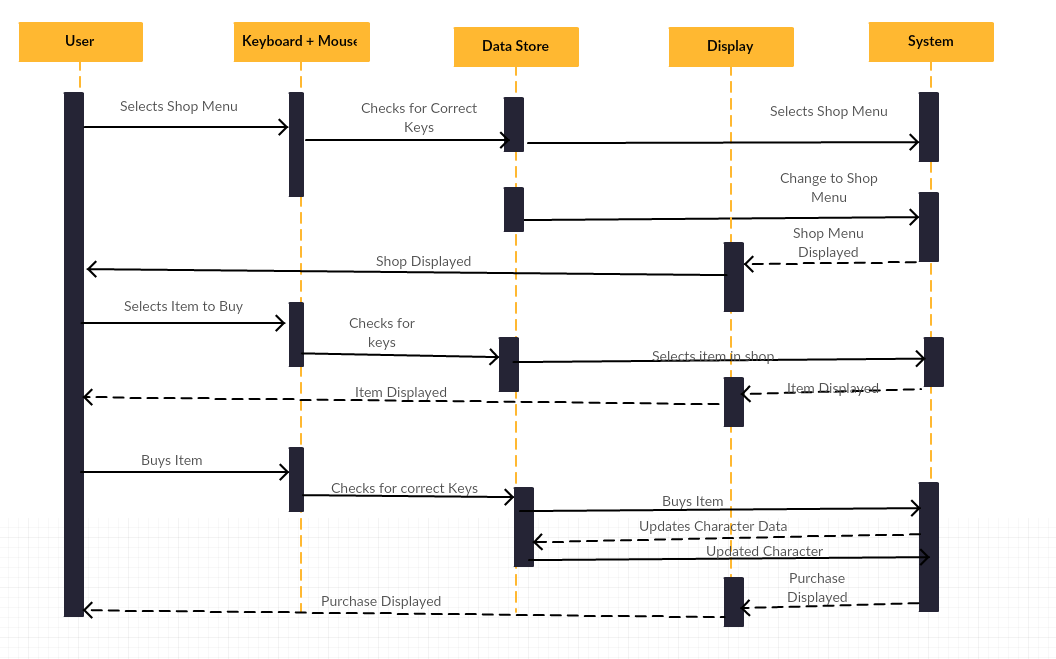
Figure 0.5 *Use Case diagram for starting the game.*

The user is able to start and close the game, just like any other application. While it may seem simple, it is important that the user is able to open and close the game on command. In addition to this the user is able to play the game, after starting the game, and only after the game has started.

## Game Transfer Data

|  |  |
| --- | --- |
| Actors | User, Enemy AI |
| Description | The user may view the high scores list, they may also view and purchase from the in-game shop. The user may control their character, its reaction based on user command. The user may also shoot at the enemy, to which the enemy AI will move and attack back at the user character. |
| Data | High score information and user name storage, characters data (such as health). |
| Stimulus | User viewing scores or shop, entering data (such as name), and user command within gameplay. |
| Response | Views scores or shop, data gets entered to system, and the character shoots on user command. |
| Comments | User must beat enemy AI to progress through game and gain in-game currency. One round of the game must be completed before a record is entered into the system for high scores. |

# Sequence Diagrams



*Figure 1.1 - A sequence diagram for buying an item from the game’s shop.*

*This Diagram shows the sequences that will be used between the main parts of the System when the user in buying something from the in game store. It shows that connections between the User, Keyboard and Mouse, Data Store, Display and System.*



*Figure 1.2 - Character Movement Sequence Diagram*

*This Sequence diagram shows the different parts of the System working together for our in game character to move. It shows the links between the KeyListen function and the Character and also the Collision check. It shows us how the different functions operate with one another.*



*Figure 1.3 - Attacking Enemy Sequence diagram*

*This diagram shows the process of the character attacking an Enemy, it shows all of the processes working together in order for the actions to be completed. This diagram shows the user’s input which is the Keystroke and the system’s functions Character, Ammunition and Enemy. The box highlighted shows multiple processes/scenarios which are dependent on the situation of the game.*



*Figure 1.4 - Start Game Sequence diagram*

*This sequence diagram shows how the user will start the game. We have shown the KeyListen function which waits for the user to press the correct key. This message is then passed to the main menu which ultimately starts the Game UI ready for the user to play.*



*Figure 1. 5 - HighScore input Sequence Diagram*

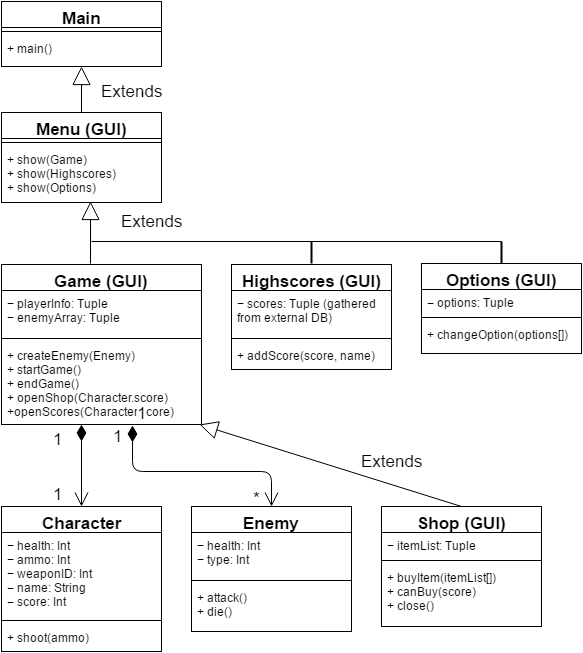
*This sequence Diagram shows how a highscore is input into the system. This follows on from the Starting the Game diagram and shows what happens once the game is finished and how the scores are updated. This diagram shows that the score is input and can then be relayed back to the user. A Separate User Interface is then opened and the ScoreUpdate process is then run.*

# 

# 

# System Architecture

The system we are going to develop will function on both an **event and object oriented basis (pattern)**; different objects in the system will require input in order to produce some kind of output, without user interaction the system will produce little to no output. Each element in the system is also an object with its own properties and functions too.



*Figure 2.1 The class diagram showing how each of the main components connect together.*

We based our decision on which pattern/style to use on the recommendations of Sommerville (2011, p. 152), we found that performance was an important factor considering that the system needs to function quickly at run time (a key requirement we discussed in the System Requirements Specification) as it's considered a very important part of video game systems in general.

Safety and Security marked low on our priorities as there are no features that have any impact on human risk or sensitive data. Availability and maintainability are other aspects that the system will unlikely need, each object could be swapped out without affecting the others but with the system being so close-knit any important changes would be to the system as a whole.

StateDiagram.png*Figure 2.2*

Event-driven being a key pattern in our system, our behavioural model can be seen above as a state diagram in figure 2.2 demonstrating the states of objects / events.

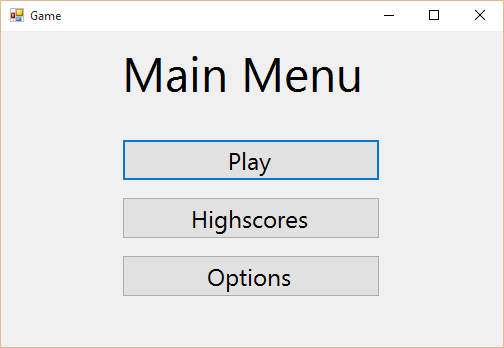
ContextDiagram.png

*Figure 2.3 Context diagram for describing the data relationships into and out of the system*

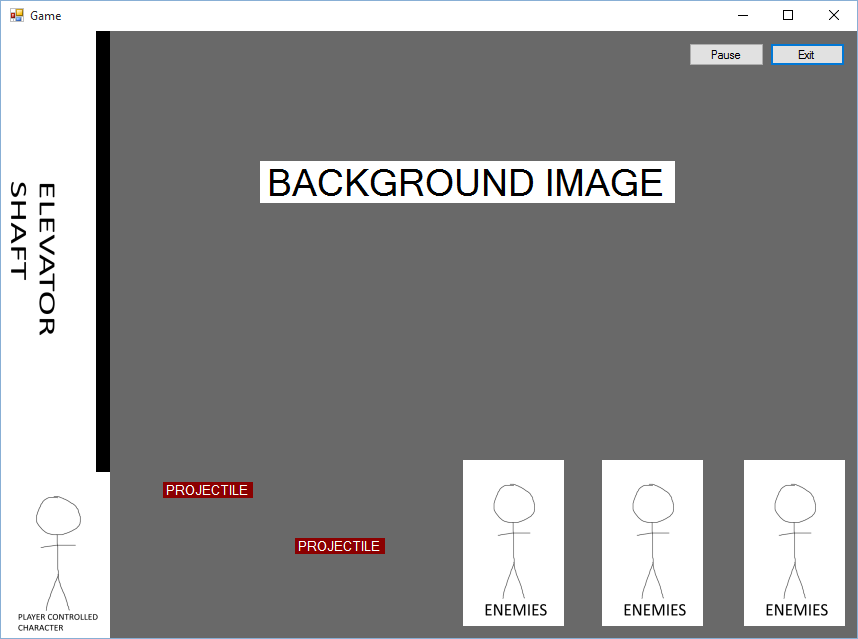
Above (figure 2.3) is the context diagram for our system, it describes the basic components and their relationships. Despite there being few large separable elements, the core system will contain the majority of the functionality and processing.

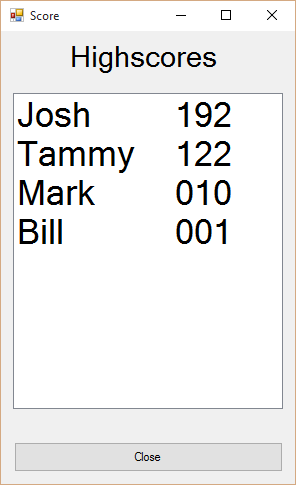
# GUI Mocks

Taking the models shown in this document, we have produced some graphical mock-up examples of what the final system will look like, they show the possible functionality and interaction components.



*Figure 3.1 - The main menu GUI, here users are presented with the option to start the game, view highscores and change settings. It is made to look simple to avoid user Usability issues.*



*Figure 3.2 - The game GUI, this is where the user can take control of their character (Seen in the bottom left) whilst enemies progressively move towards the character. The user can also fire projectiles at the enemies to reduce their health and eventually remove them from the game. For more details see SRS / Above.*

*Figure 3.3 - Highscores GUI showing previous scores. Scores may be added by the system upon completion of the game, and possibly (with enough development time) editable from the highscores GUI.*

# 

# References

Sommerville, I. (2011). *Software engineering*. Boston: Pearson.

Stevens, P., & Pooley, R. J. (2006). *Using UML: Software engineering with objects and components*. Harlow, England: Addison-Wesley.

# Declaration

|  |  |  |
| --- | --- | --- |
| **Name** | **Work Done** | **Signature** |
| Benjamin Rose | **I have contributed the following to the submission:** All mock up GUI images, the second sequence diagram, use case *scenarios*, all of System Architecture and its diagrams, referencing. |  |
| Bruno Bernardo | **I have contributed the following to the submission:** The sequence From figure 3 to 5 and contributed a bit to the Use case scenarios. |  |
| Robert Barry | No longer in unit |  |
| Danny Letts | Contributed largely to the Sequence diagram section of the report and general touches to the rest of the document. |  |
| Ceiran Sims-Glover | **I have contributed the following to the submission:** Use case diagrams of user interactions. Descriptions of use cases. | IMG_20170115_214056539.jpg |
| Luke Hembling | Contributed primarily to Usage Scenarios section, diagrams, the Use Case table and descriptions. |  |