Graded Unit Project

HND Software Development

Table of Contents

[Planning 4](#_Toc412752054)

[Project Proposal/Brief 4](#_Toc412752055)

[Requirements 5](#_Toc412752056)

[Interface 5](#_Toc412752057)

[Hardware Requirements 5](#_Toc412752058)

[Software Requirements 5](#_Toc412752059)

[Functional Requirements 5](#_Toc412752060)

[Non-functional Requirements 5](#_Toc412752061)

[Research 6](#_Toc412752062)

[Similar Games 7](#_Toc412752063)

[Mechanics and rules of Space Invaders 10](#_Toc412752064)

[Game Elements 10](#_Toc412752065)

[Game Screen 10](#_Toc412752066)

[Score 10](#_Toc412752067)

[End 10](#_Toc412752068)

[Features of a Fixed Shooter game 11](#_Toc412752069)

[Survey Results 12](#_Toc412752070)

[Project Plan 13](#_Toc412752071)

[Legal Note 13](#_Toc412752072)

[Development Cycle 13](#_Toc412752073)

[Resources Required 14](#_Toc412752074)

[Language Choice 14](#_Toc412752075)

[Work Breakdown Structure (WBS) 15](#_Toc412752076)

[Gantt Charts 17](#_Toc412752077)

[Planning 17](#_Toc412752078)

[Development 17](#_Toc412752079)

[Evaluation 17](#_Toc412752080)

[Design 18](#_Toc412752081)

[Initial top-level Use Case 18](#_Toc412752082)

[Use Cases 19](#_Toc412752083)

[Class List 31](#_Toc412752084)

[Class Diagram 33](#_Toc412752085)

[Sequence Diagrams 34](#_Toc412752086)

[Start Game 34](#_Toc412752087)

[Move 34](#_Toc412752088)

[End Game 35](#_Toc412752089)

[Data Dictionary 36](#_Toc412752090)

[Methods / Functions 38](#_Toc412752091)

[High Score Database Design 40](#_Toc412752092)

[ERD 40](#_Toc412752093)

[Score 40](#_Toc412752094)

[Player 40](#_Toc412752095)

[Class Responsibility Collaboration Cards (CRC Cards) 41](#_Toc412752096)

[Screen Designs 43](#_Toc412752097)

[Game Elements 43](#_Toc412752098)

[Game Screen 45](#_Toc412752099)

[Options Screen 46](#_Toc412752100)

[High Score Screen 47](#_Toc412752101)

[Development 48](#_Toc412752102)

[Test Plan 48](#_Toc412752103)

[Full Pseudo Code 48](#_Toc412752104)

[Final Game Design 48](#_Toc412752105)

[Evaluation 48](#_Toc412752106)

[Testing 48](#_Toc412752107)

[Self-Assessment 48](#_Toc412752108)

[Appendix 48](#_Toc412752109)

[References 48](#_Toc412752110)

# Planning

## Project Proposal/Brief

For this project I propose to create a 2D arcade game, based off the 1978 classic ‘Space Invaders’, in which the user, herein referred to as the ‘Player’, may defeat waves of enemies by controlling a turret. This game will be written in the C# programming language, using MS Visual Studio as the development environment, with the addition of XNA framework.

The player will initially be presented with the title and a menu detailing their gameplay options; Arcade Mode, Co-op Mode, Vs Mode and Endless Mode, in addition to the customisation menu and high score. Upon selection of their choice, the player will be placed into the game environment and pitted against waves of enemies to defeat in order to progress. Enemies are defeated by shooting them, whilst avoiding defeat (the player may take three direct ‘hits’ from the enemies before game over is declared).

Once game-over is declared, or the end reached, the player will be presented with their high score, and the option to enter their name and save the score if they so wish, before being returned to the main menu. Co-op and Vs mode tracks the scores of each player individually, and additionally displays the score as a running tally during these modes. Each game mode has its own high-score table for the player to populate with their scores.

In Arcade mode, the player must defeat 11 waves of normal enemies, and a final ‘boss’ encounter, in order to unlock Endless mode, which pits the player against never-ending waves of enemies, with a boss every 12th wave, until they forfeit or reach game over. Co-op mode has 2 players each controlling an individual turret working together to defeat the waves of enemies, whilst Vs mode allows one player to play the role of the enemies (attacker), and one to be the turret (defender)

The player will interact with the game world through use of the control scheme, which is customisable, although the controls will default to A: move left, B: move right, Spacebar: shoot

Each enemy can take one direct hit, with the exception of the boss and will return fire to the player’s last position. The boss itself requires between 10 and 20 hits (boss ‘health’ increases by 2 in Endless mode after every 12 waves), with boss health being tracked and displayed on a ‘health bar’.

The player may customise their turret and laser colours in the main menu by either selecting a colour from a preset, or by entering a hex value.

## Requirements

### Interface

The game UI will be designed around keyboard input, and all graphical elements should be reasonably comfortable to use.

### Hardware Requirements

The game has a simple design and as such should run on any modern pc. A keyboard and mouse is required in order to fully interact with the game.

### Software Requirements

As the game will be designed using Windows forms, the user will require a Windows 7 or Windows 8 operating system.

### Functional Requirements

* The Player will be shown game menu on application start
* The Player may exit the application at any time
* The Player may quit the game at any time
* The Player may pause the game at any time
* The Player may exit to the main menu at any time
* When the Player chooses to exit, a confirmation window will appear
* The Player will “die” if hit 3 times during the game
* The Player must defeat 11 ‘waves’ of enemies
* The Boss will spawn after the 10th ‘wave’ of enemies
* The Player will be able to toggle game sound on/off
* Music will be playing when Game is running
* Visual and Auditory effects will play when the Player is hit.
* The Player will be prompted to enter their name on Game Over

### Non-functional Requirements

* The Game will run on the Windows platform, with support primarily being aimed at Windows 7 and 8.
* Game background will be black.
* The High Score will only display top 10 entries
* The Player will be awarded points for each normal enemy eliminated
* The Player will be awarded 100 points for eliminating the Boss
* Each enemy will come in one of the following colours (red, purple, yellow, orange, green)

## Research

For the purposes of this task, and to get a better understanding of the subject matter, research will be conducted, and any results will be analysed in order to further the project.

The following research methods will be employed;

* Close analysis and research on the various similar projects, including the original Space Invaders.
* An in-depth look at the specific features of this genre (Fixed Shooter)
* A survey aimed at gamers and those with an interest in this kind of project, to determine player expectations and likes.

### Similar Games

**Space Invaders (Original)**

In 1978, Japanese Developer Tomohiro Nishikado created and designed a 2D fixed shooter for the Arcade, which he named ‘Space Invaders’.

In this classic game, the goal of the player is to eliminate waves of aliens with a laser cannon, in order to gain as many points as possible, without letting the aliens reach the bottom of the screen. Each wave contains 5 rows of 11 aliens that, when eliminated, spawns another wave of aliens that move faster. This effect is cumulative; each new wave is progressively faster than the last. Game Over is reached when the player is hit by an alien laser, or if the player fails to stop the alien invasion.

Fig 1. – Gameplay Example. Source: Wikipedia

Due to the major success of Space Invaders, Tomohiro Nishikado is also credited with inventing the genre of “fixed shooter” and starting the “Golden Age” of Arcade games, where gaming gained its position in the global industry. Space Invaders was hugely successful, and its impact on the way games were designed was huge, so much so that it has become a pop culture icon.

Space Invaders remains a successful franchise to this day, with sequels, reboots and console ports being made.

**Galaxian**

Galaxian was developed and released in 1979 by Namco, another Japanese company most known for the arcade hit “Pac-Man”. Galaxian was meant as a competitor to Tomohiro Nishikado’s hugely successful “Space Invaders”.

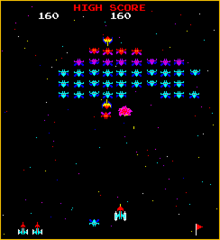
Following a similar style to “Space Invaders”, Galaxian has the player fending off endless masses of aliens, whilst attempting to eliminate as many as possible. Galaxian differs in how the aliens behave, however. Instead of being restricted to simply moving vertically and horizontally, the aliens in Galaxian will periodically break formation and ‘dive’ at the player. This ‘diving’ action made Galaxian the first game to feature foes with separate personalities.

Fig 2. – Galaxian Gameplay. Source: Wikipedia

**Breakout**

The arcade game Breakout was developed and published by Atari in 1976, and is cited by Tomohiro Nishikado as his main inspiration for “Space Invaders”. In Breakout, bricks cover the top third of the screen, with the goal being to clear the screen. To accomplish this, a ball bounces off the edges of the screen. The player controls a bat in order to keep the ball in play, and if the player fails to do so, they lose a turn. If the player fails to keep the ball in play three times, game over is initiated.

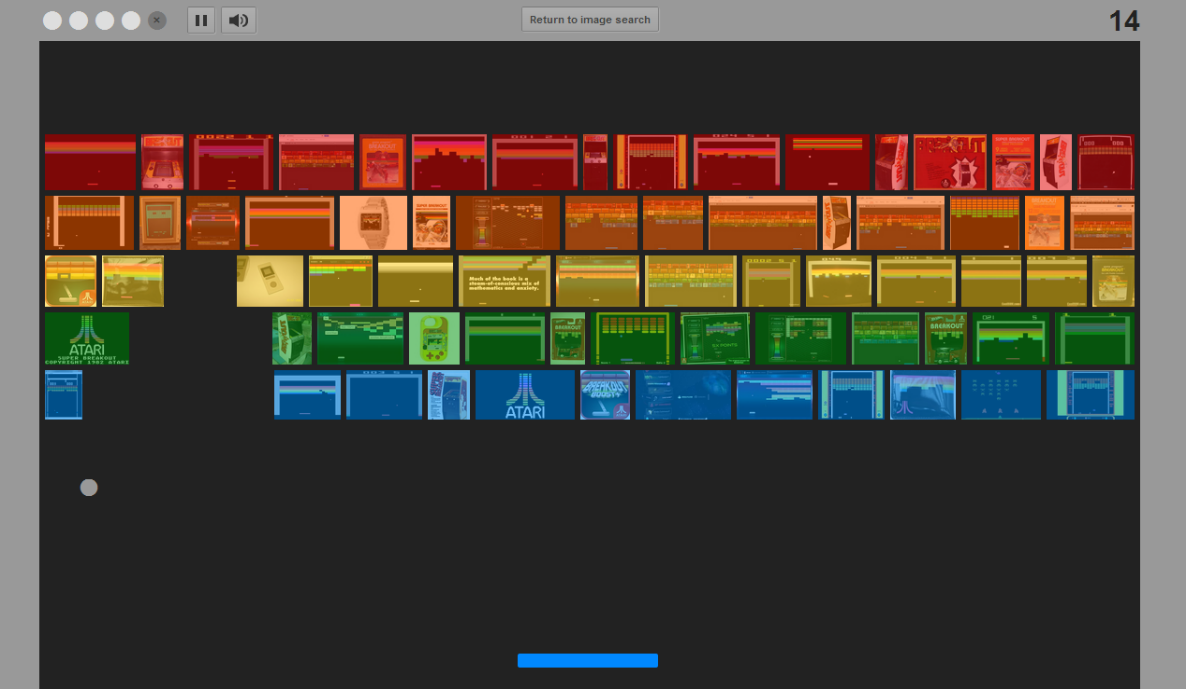


Fig 3. – Breakout Gameplay (Google Version) Source: Google

Breakout has seen many clones and reboots since its initial release, including a commemorative version by Google.

## Mechanics and rules of Space Invaders

### Game Elements

The enemies, also referred to as Aliens. Apart from having different looks, each aliens offers increasing points, with the spaceship giving the player a bonus 100 points. The player interacts with the aliens by shooting them for points, and avoiding the returning fire.



The static shield, and the player controlled turret. The player starts with three static shields, which take damage from both the aliens and the player. The shields remain in game until fully destroyed, and can be used by the player to hide from damage. The player controls their turret with the arrow keys.

I:\Graded Unit\turret.PNG

### Game Screen

Upon game start, the player is presented with the game screen, which is filled with rows of aliens. Each row has a different alien type, which grants the player an increasing number of points when defeated. Once every row of alien is defeated, a new wave is spawned, with increased speed.

The number of rows and columns of aliens differs from each variant of game.

### Score

Score is tracked at the top of the screen, and the number of points granted to the player depends on the enemy defeated. The spaceship enemy grants the player 100 points.

### End

If the player gets hit three times, they run out of lives and it is game over.

### Features of a Fixed Shooter game

* **Short playing sessions** – The game is designed to allow the player to quickly start and finish games, with no repercussions, although it is not usually encouraged by the game itself (see competitive)
* **Fixed Axis of movement** – Gameplay usually only allows one axis of movement (left to right, or up/down)
* **Gameplay takes place on one screen** – Instead of the more common ‘side-scroller’ gameplay, the game instead takes place on one fixed screen.
* **Involves dodging** –In order to survive, the player must avoid being hit by the enemies
* **Usually lone assault** – In a fixed shooter game, it is usually a solo player, with no help, against the AI enemy.
* **Simple gameplay, simple rules** – The player can quickly learn the rules and controls of the game
* **Limited Lives –** If the player fails to avoid being hit, they have (usually) three lives, with a life being lost upon hit.
* **Competitive –** The player is encouraged to improve their skill at the game, and compete with their friends for the best high score

### Survey Results

As part of the research process, an online survey was made using the services offered by Survey Monkey. (<http://www.surveymonkey.com>)

This survey was aimed at gamers and those with an interest in fixed shooter games, which was established before access to the survey was granted.

A total of 9 people answered the following;

1. **Have you ever played Space Invaders or a similar game?**

Yes – 100%

No – 0%

1. **Do you consider yourself to be a “gamer”?**

Yes – 88.89%

No – 11.1%

1. **What is your age range?**

Under 13 – 0%

14-17 – 0%

18–24 – 77.78%

Over 25 – 22.2%

1. **How many hours per week do you spend playing video games?**

Less than 1 hour – 0%

1 to 5 hours – 11.1%

5 to 20 hours – 11.1%

More than 20 hours – 77.78%

I don’t stop playing video games – 0%

1. **Which of the following is your music preference whilst playing games?**

Game’s own music – 28.57%

Own music – 57.14%

No music – 14.29%

1. **Do you view Space Invaders as an "iconic" game?**

Yes – 100%

No – 0%

1. **How did you find your experience with a fixed shooter game?**

Very Enjoyable – 11.1%

Enjoyable – 66.67%

Was ok – 22.2%

Didn’t like it – 0%

Hated it – 0%

N/A- 0%

1. **Do you find graphics or gameplay to be more important?**

Graphics – 11.1%

Gameplay – 88.89%

1. **Which of the following do you prefer?**

Customisable colours (hex) – 71.43%

Preset colour choices – 28.57%

No colour options – 0%

The survey itself can be found at: <https://www.surveymonkey.com/s/6PLF2KT>

# Project Plan

## Legal Note

Due to the nature of the project, research was done on UK law regarding game clones and copyright. This research involved thorough examination of the following acts; The Copyright, Designs and Patents Act 1988, and its sub-act, The Copyright (Computer Programs) Regulations 1992.

The Copyright, Designs and Patents Act 1988 covers and protects, in relation to computer programs, the assets (sounds, graphical elements and the code itself), but does not cover the specific idea behind the program. This means the original idea behind a fixed shooter game such as Space Invaders is not copyrighted, merely the expression of the idea, the game itself.

The legislation put forward in The Copyright, Designs and Patents Act 1988 details that *“Fair dealing with a work for the purposes of private study does not infringe any copyright in the work”* (section 29, Research and private study). Additionally, *“sole purpose of illustration for instruction does not infringe copyright in the work”* (Section 321b).

This means that some usage of copyrighted materials and examination of the code is also legal, under the Act.

Fair dealing is the limitation and exemption to the *“exclusive right granted by copyright law to the author of a creative work”*.

## Development Cycle

After much deliberation and examination of the various software development models available, the Waterfall model has been identified as the most appropriate model to use in this project.

The waterfall model was chosen as it allows for a structured development, as progress “flows” from one section to the next, and it is unlikely that any significant changes will occur to the requirements or purpose of the project.

The three major sections of the project are as follows;

* Planning (further sectioned into research, requirements and design)
* Development (broken down into coding, documentation and testing)
* Evaluation

## Resources Required

Hardware

* PC running Windows 7 and all its peripherals will be required. The PC must have enough resources (processing ability, RAM, storage, etc.) in order to effectively run the software.

Software

* MS Visual Studio 2013 will be used as the programming environment for this project
* MS Word 2013 will be used for the write-up of this report and all other documentation
* MS Access 2013 will be utilised in order to create the database needed
* MS Project 2013 for the WBS and other project management related diagrams

Other

* Internet access is required for the research phase of this project, and for use as knowledge base during the development phase.
* Time – development of the program, write-ups and all other project elements will require a considerable amount of time

## Language Choice

The programming language chosen for this project is C#, which was chosen as it is an object oriented language that functions well with XNA framework. It is a common language choice for those wishing to create games, and additionally, it is a language which the developer of the project is fairly experienced with.

An alternate choice for this project could be java, another object oriented language that could be used in the Eclipse environment.

## Work Breakdown Structure (WBS)

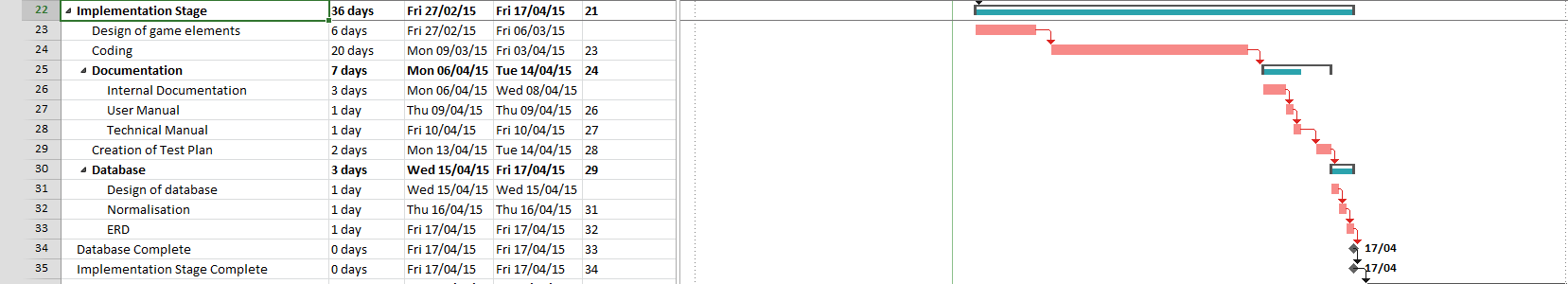
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Task Name | Duration | Start | Finish | Predecessors | Resource Names |
| **Planning Stage** | **27 days** | **Wed 21/01/15** | **Thu 26/02/15** |  |  |
| Product Brief | 1 day | Wed 21/01/15 | Wed 21/01/15 |  | MS Word, PC, Siobhan Wallace |
| **Research** | **9 days** | **Wed 21/01/15** | **Tue 03/02/15** | **2** | **Internet, MS Word, PC, Siobhan Wallace** |
| Research of similar games | 3 days | Wed 21/01/15 | Mon 26/01/15 |  |  |
| Research of features | 2 days | Mon 26/01/15 | Wed 28/01/15 | 4 |  |
| Research of Fixed Shooter Mechanics | 2 days | Wed 28/01/15 | Fri 30/01/15 | 5 |  |
| Survey | 1 day | Fri 30/01/15 | Mon 02/02/15 | 6 |  |
| Required Resources | 1 day | Mon 02/02/15 | Tue 03/02/15 | 7 |  |
| Research Complete | 0 days | Tue 03/02/15 | Tue 03/02/15 | 8 |  |
| Project Plan | 1 day | Tue 03/02/15 | Wed 04/02/15 | 9 | MS Project, PC, Siobhan Wallace |
| **Design** | **16 days** | **Thu 05/02/15** | **Thu 26/02/15** | **10** | **PC, Siobhan Wallace, MS Word, Visual Paradigm** |
| Top Level Use Case Diagram | 1 day | Thu 05/02/15 | Thu 05/02/15 |  |  |
| Use Cases | 1 day | Fri 06/02/15 | Fri 06/02/15 | 12 |  |
| CRC (Class Responsibility Collaboration) Cards | 2 days | Mon 09/02/15 | Tue 10/02/15 | 13 |  |
| Sequence diagrams | 3 days | Wed 11/02/15 | Fri 13/02/15 | 14 |  |
| Detailed Class List | 2 days | Mon 16/02/15 | Tue 17/02/15 | 15 |  |
| Class Diagrams | 2 days | Wed 18/02/15 | Thu 19/02/15 | 16 |  |
| Data Dictionary | 2 days | Fri 20/02/15 | Mon 23/02/15 | 17 |  |
| Design of GUI | 3 days | Tue 24/02/15 | Thu 26/02/15 | 18 | GIMP |
| Design Complete | 0 days | Thu 26/02/15 | Thu 26/02/15 | 19 |  |
| Planning Stage Complete | 0 days | Thu 26/02/15 | Thu 26/02/15 | 20 |  |
| **Implementation Stage** | **36 days** | **Fri 27/02/15** | **Fri 17/04/15** | **21** | **Internet, MS Visual Studio, PC, Siobhan Wallace** |
| Design of game elements | 6 days | Fri 27/02/15 | Fri 06/03/15 |  |  |
| Coding | 20 days | Mon 09/03/15 | Fri 03/04/15 | 23 |  |
| **Documentation** | **7 days** | **Mon 06/04/15** | **Tue 14/04/15** | **24** | **MS Word** |
| Internal Documentation | 3 days | Mon 06/04/15 | Wed 08/04/15 |  |  |
| User Manual | 1 day | Thu 09/04/15 | Thu 09/04/15 | 26 |  |
| Technical Manual | 1 day | Fri 10/04/15 | Fri 10/04/15 | 27 |  |
| Creation of Test Plan | 2 days | Mon 13/04/15 | Tue 14/04/15 | 28 |  |
| **Database** | **3 days** | **Wed 15/04/15** | **Fri 17/04/15** | **29** | **MS Access** |
| Design of database | 1 day | Wed 15/04/15 | Wed 15/04/15 |  |  |
| Normalisation | 1 day | Thu 16/04/15 | Thu 16/04/15 | 31 |  |
| ERD | 1 day | Fri 17/04/15 | Fri 17/04/15 | 32 |  |
| Database Complete | 0 days | Fri 17/04/15 | Fri 17/04/15 | 33 |  |
| Implementation Stage Complete | 0 days | Fri 17/04/15 | Fri 17/04/15 | 34 |  |
| **Evaluation Stage** | **20 days** | **Mon 20/04/15** | **Fri 15/05/15** | **35** | **MS Word, PC, Siobhan Wallace** |
| Establish criteria needed for evaluation | 5 days | Mon 20/04/15 | Fri 24/04/15 |  |  |
| Summarise changes to original design | 3 days | Mon 27/04/15 | Wed 29/04/15 | 37 |  |
| Evaluate effectiveness of game | 7 days | Thu 30/04/15 | Fri 08/05/15 | 38 |  |
| Determine future recommendations | 3 days | Mon 11/05/15 | Wed 13/05/15 | 39 |  |
| Individual Analysis | 2 days | Thu 14/05/15 | Fri 15/05/15 | 40 |  |
| Evaluation Stage Complete | 0 days | Fri 15/05/15 | Fri 15/05/15 | 41 |  |

## Gantt Charts

Gantt Charts are a core element of any project, used to determine a time scale for the project, and break it down into individual tasks.

### I:\Graded Unit\planning gantt.PNGPlanning

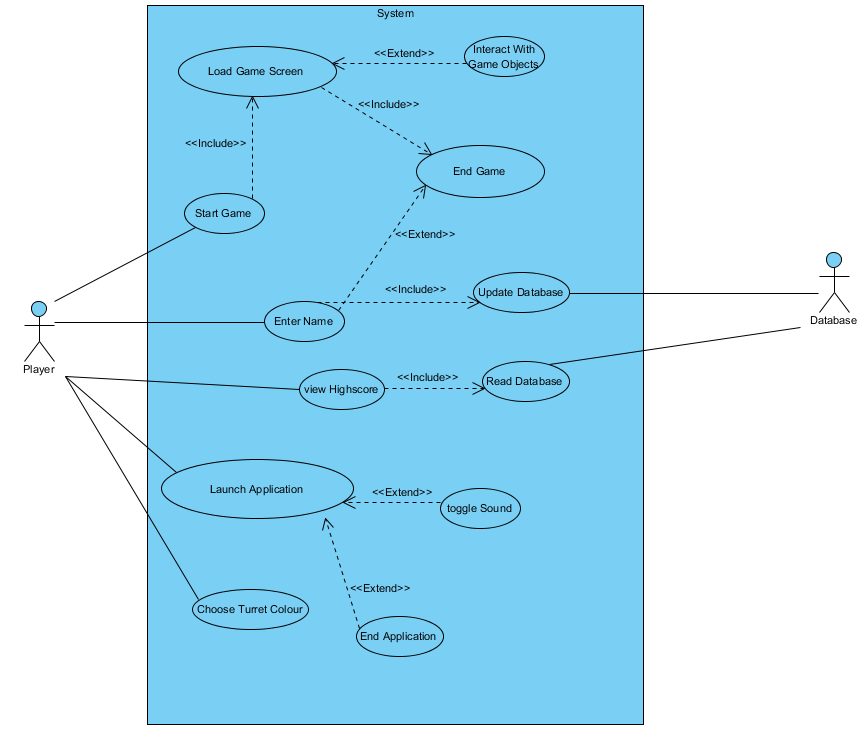
### Development



### I:\Graded Unit\evaluation gantt.PNGEvaluation

# Design

### Initial top-level Use Case

One of the many UML tools available, a use case diagram assists with actor identification, and their interaction with the system. Analysis of a use case diagram allow developers to initialise classes and determine the functionality required by each.

## Use Cases

A more in-depth look at each individual use case.

|  |  |
| --- | --- |
| Use Case: | Launch Application |
| Prerequisites: | Application is not running |
| Post Conditions: | Application will Launch  GUI will be initialised |
| Pseudo Code: | 1. BEGIN 2. Initialise GUI 3. Declare sound 4. Declare player\_Score 5. Declare game\_Screen 6. Set sound to TRUE 7. END |
| Activity Diagram | I:\Graded Unit\Launch App Activity.PNG |

|  |  |
| --- | --- |
| Use Case: | Start Game |
| Prerequisites: | Application is running  Player Clicked ‘Start’ Button |
| Post Conditions: | Game Will Start  Game Screen will launch |
| Pseudo Code: | 1. BEGIN 2. Set player\_Score to 0 3. Initialise enemies 4. Initialise player\_Shield 5. Initialise player 6. Initialise lasers 7. Create new game\_Screen 8. Draw game\_Screen 9. END |
| Activity Diagram | I:\Graded Unit\Activity Diagrams\start game activity.PNG |

|  |  |
| --- | --- |
| Use Case: | Load Game Screen |
| Prerequisites: | Application is running  Game is starting |
| Post Conditions: | Game Screen is loaded |
| Pseudo Code: | 1. BEGIN 2. Declare game\_Screen to be a 2d object array 3. Draw player 4. Draw player\_Shield 5. For each array of objects in game\_Screen 6. LOOP    1. For each object in enemyArray    2. LOOP       1. Draw new enemy       2. END LOOP   6.3 END LOOP   1. END |
| Activity Diagram | I:\Graded Unit\Activity Diagrams\load game screen.PNG |

|  |  |
| --- | --- |
| Use Case: | Interact With Game Objects |
| Prerequisites: | Application is running  Game Screen is loaded |
| Post Conditions: | Game object will move |
| Pseudo Code: | 1. BEGIN 2. Wait for user input 3. Read user input 4. Change playerCoords to user input    1. IF player input = fire THEN       1. Create new laser   4.2 END IF   1. END |
| Activity Diagram | I:\Graded Unit\Activity Diagrams\interact with game.PNG |

|  |  |
| --- | --- |
| Use Case: | End Game |
| Prerequisites: | Application is running  Game Screen is loaded |
| Post Conditions: | Game Screen will close |
| Pseudo Code: | 1. BEGIN 2. Check player\_Shield    1. IF player\_Shield = 0 THEN       1. Display gameover       2. Display player\_Score       3. END IF    2. ELSE no gameover   2.2.1 Return to 2   1. END |
| Activity Diagram | I:\Graded Unit\Activity Diagrams\end game.PNG |

|  |  |
| --- | --- |
| Use Case: | Enter Name |
| Prerequisites: | End Game |
| Post Conditions: | Player name and score is sent to database |
| Pseudo Code:Initialise enemies | 1. BEGIN 2. Declare player\_Name 3. Declare high\_Score\_List 4. Display Name Entry Window 5. Check user input    1. IF user input has spaces THEN    2. Display error    3. Return to 3    4. END IF    5. ELSE IF user input has numeric value       1. Display error       2. Return to 3       3. END IF    6. ELSE       1. set player\_Name to user input       2. END IF 6. Read scores from database 7. Calculate position for new entry 8. Write new entry to database 9. END |
| Activity Diagram | I:\Graded Unit\Activity Diagrams\enter name.PNG |

|  |  |
| --- | --- |
| Use Case: | Update Database |
| Prerequisites: | Player has entered name  Game has ended |
| Post Conditions: | Application sends data to database |
| Pseudo Code: | 1. BEGIN 2. Declare Data 3. Set Data to value passed in 4. Establish a connection with the high score database 5. Update database with Data 6. END |
| Activity Diagram | I:\Graded Unit\update database activity.PNG |

|  |  |
| --- | --- |
| Use Case: | Read Database |
| Prerequisites: | View High-Score |
| Post Conditions: | Database sends data to application |
| Pseudo Code: | 1. BEGIN 2. Declare score 3. Establish a connection with high score database 4. Retrieve data from database 5. Set Data to retrieved value 6. Return Data 7. END |
| Activity Diagram | I:\Graded Unit\Activity Diagrams\read database.PNG |

|  |  |
| --- | --- |
| Use Case: | Choose Turret Colour |
| Prerequisites: | Application is running  Player has clicked options menu |
| Post Conditions: | Turret will change colour |
| Pseudo Code: | 1. BEGIN 2. Prompt user for hex code    1. IF user input is NOT valid hex THEN   2.11 Display Error  2.12 Return to 2  2.13 END IF  2.2 ELSE IF user inputs valid hex THEN  2.2.1 set turretColour to user input  2.2.2 END IF  2.3 ELSE IF user cancels hex entry  2.3.1 Display presets  2.3.2 Wait for user input  2.3.3 Read user input  2.3.4 set turretColour to user input  2.3.5 END IF  2.4 ELSE  2.4.1 Set turretColourDefault = TRUE  2.4.2 END IF   1. END |
| Activity Diagram | I:\Graded Unit\Activity Diagrams\turret colour.PNG |

|  |  |
| --- | --- |
| Use Case: | Toggle Sound |
| Prerequisites: | Application is running  Player Clicked ‘Sound’ Button |
| Post Conditions: | Sound is toggled ON or OFF |
| Pseudo Code: | 1. BEGIN 2. Check Sound   2.1 IF sound equals true THEN  2.11 Set sound to FALSE  2.12 Turn sound OFF  2.13 END IF  2.2 ELSE  2.21 Set sound to TRUE  2.22 Turn sound ON  2.3 END IF   1. END |
| Activity Diagram | I:\Graded Unit\Activity Diagrams\toggle sound.PNG |

|  |  |
| --- | --- |
| Use Case: | View High-Score |
| Prerequisites: | Application is running  Player Clicked ‘View High Score’ Button |
| Post Conditions: | High Score list is displayed |
| Pseudo Code: | 1. BEGIN 2. Declare high\_Score\_List 3. Declare database\_Score object 4. Read from database using interface 5. Set database\_Score to Database output 6. Convert database\_Score to String and set as high\_Score\_List 7. Display high\_Score\_List in window 8. END |
| Activity Diagram | I:\Graded Unit\Activity Diagrams\view high score.PNG |

|  |  |
| --- | --- |
| Use Case: | End Application |
| Prerequisites: | Application is running |
| Post Conditions: | Application will end |
| Pseudo Code: | 1. BEGIN 2. End Application 3. END |
| Activity Diagram | I:\Graded Unit\end application activity.PNG |

## Class List

An in-depth look at the classes, their attributes and methods that the project will be using.

|  |  |
| --- | --- |
| Class Name: OptionsScreen() | |
| Attributes | Private string colour  Private Boolean turretColourDefault  Private String hexColour |
| Methods | getColour()  getHexColour()  getturretColourDefault()  setColour()  setHexColour(String hexColour)  setturretColourDefault(Boolean turretColourDefault)  convertHexToColour() |

|  |  |
| --- | --- |
| Class Name: Game() | |
| Attributes | Private Bool Sound |
| Methods | toggleSound()  Screen() |

|  |  |
| --- | --- |
| Class Name: Game\_Screen() | |
| Attributes | Private int Height  Private int Width  Private object gameScreen |
| Methods | getHeight()  getWidth()  setHeight()  setWidth()  drawGameScreen() |

|  |  |
| --- | --- |
| Class Name: Enemies() | |
| Attributes | Private Int size  Private Int hp  Private String enemyColour  Private point enemyCoordinates |
| Methods | getSize()  getHP()  getEnemyColour()  getEnemyCoordinates()  setEnemyCoordinates(point enemyCoordinates)  moveEnemies()  setHp(int hp)  setSize(int size)  setEnemyColour(string enemyColour) |

|  |  |
| --- | --- |
| Class Name: Player() | |
| Attributes | Private int playerShield  Private string playerColour  Private point playerCoords  Private int playerScore |
| Methods | getPlayerShield()  getPlayerScore()  getPlayerColour()  getPlayerCoords()  setPlayerColour(String playerColour)  setPlayerCoords(point playerCoords)  setPlayerScore(int playerScore)  setPlayerShield(int playerShield)  movePlayer() |

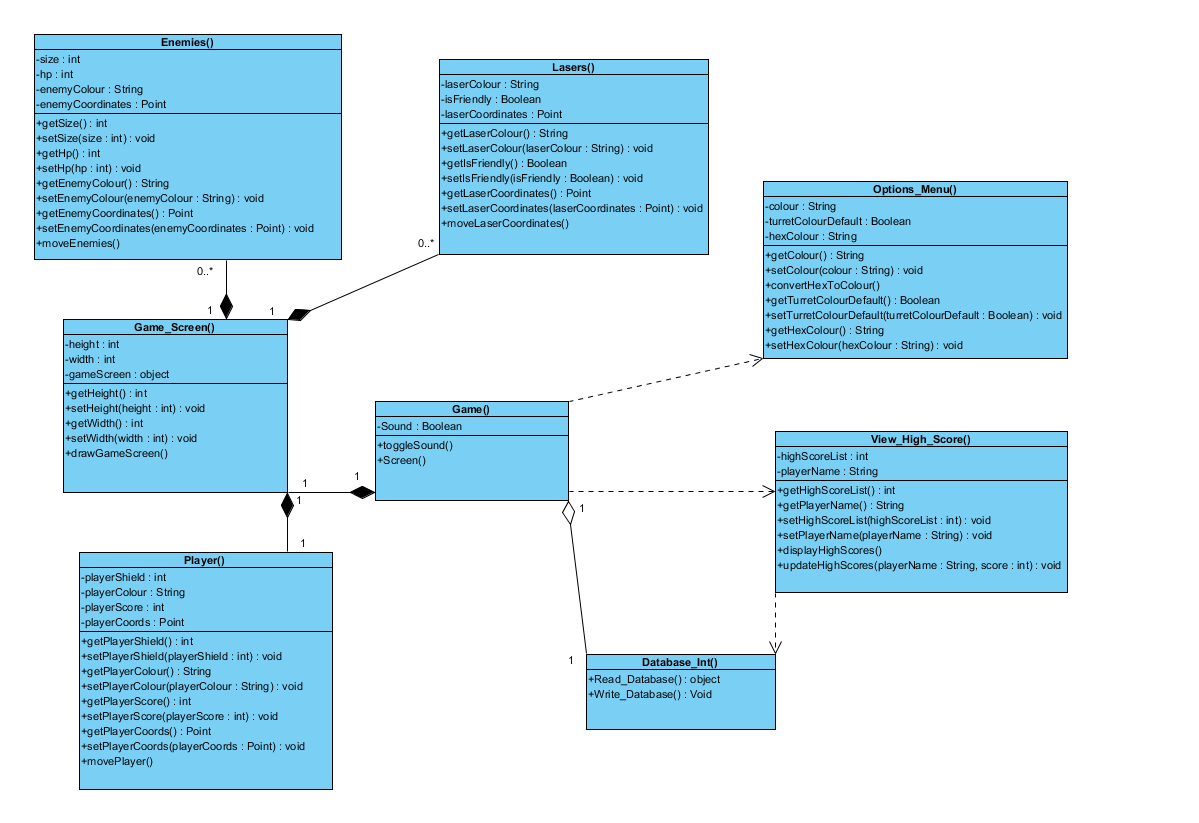
|  |  |
| --- | --- |
| Class Name: View\_High\_Score() | |
| Attributes | Private String highScoreList  Private String playerName |
| Methods | getHighScoreList()  setHighScoreList(String highScoreList)  displayHighscores()  updateHighscores(String playerName ,int score)  getPlayerName()  setPlayerName(String playerName) |

|  |  |
| --- | --- |
| Class Name: Lasers() | |
| Attributes | Private string laserColour  Private point laserCoordinates  Private Boolean isFriendly |
| Methods | getColour()  getLaserCoordinates()  getIsFriendly()  setColour(string colour)  setLaserCoordinates(point laserCoordinates)  setIsFriendly(Boolean isFriendly)  moveLaserCoordinates() |

|  |  |
| --- | --- |
| Class Name: Database\_Int() | |
| Attributes | - |
| Methods | Read\_Database(object)  Write\_Database() |

## Class Diagram

Class diagrams are yet another useful tool unique to UML, acting as a method for developers to highlight the relationships between classes.



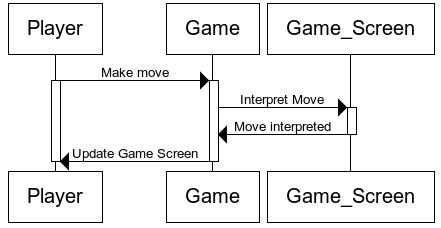
|  |  |
| --- | --- |
| Connection Type | Example |
| Aggregation  This connection type describes a relationship where the object can exist outside its container object |  |
| Composition  This connection type describes a relationship where the object cannot exist outside its container object |  |
| Dependency  This connection type describes an object that is used by the object it is connected to |  |

## Sequence Diagrams

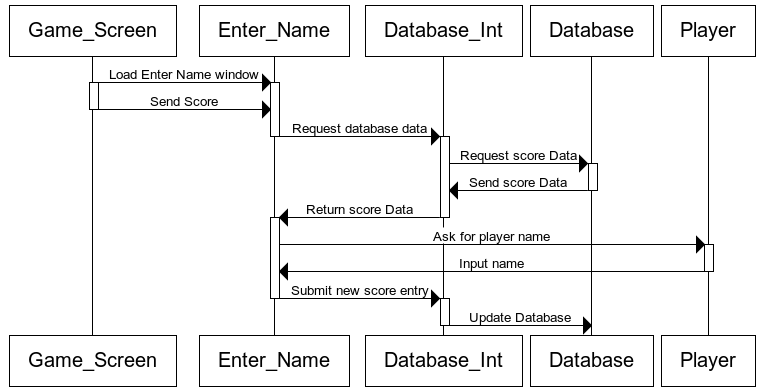
Another useful UML tool, sequence diagrams act as a medium for developers to plan and order all interactions between the objects contained in the system.

### I:\Graded Unit\Sequence Diagrams\New game.PNGStart Game

### Move



### End Game



## Data Dictionary

A data dictionary is a detailed list of all the attributes used in the project, for the purposes of

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute Name | Data Type | Accessibility | Location | Description |
| Width | Int | Private | Game\_screen() | Stores the width of game screen |
| Height | Int | Private | Game\_screen() | Stores the height of game screen |
| Sound | Bool | Private | Game() | Indicates if sound is ON or OFF |
| Screen | Object | Private | Game() | Instance of game\_Screen class |
| playerName | String | Private | View\_High\_Score() | Stores the player’s name |
| gameScreen |  | Private | Game\_Screen() | 2D array of game\_Screen class |
| playerScore | Int | Private | Player() | Stores the players current score |
| highScoreList | String | Private | View\_High\_Score() | Displays the list of high scores |
| player\_Shield | Int | Private | Player() | Stores the players current shield level |
| enemySize | Int | Private | Enemies() | Stores the current size of the enemy |
| enemyHP | Int | Private | Enemies() | Stores the current health of the enemy |
| enemyCoord | Object | Private | Enemies() | Stores enemy position |
| enemyColour | String | Private | Enemies() | Stores enemy colour |
| playerColour | String | Private | Player() | Stores player colour |
| playerCoords | Object | Private | Player() | Stores player position |
| laserColour | String | Private | Lasers() | Stores laser colour |
| laserCoordinates | Object | Private | Lasers() | Stores laser position |
| isFriendly | boolean | Private | Lasers() | Indicates laser team |
| playerColour | string | Private | OptionsScreen() | Stores player colour choice |
| hexColour | String | Private | OptionsScreen() | Stores player hex entry |

## Methods / Functions

The following is a detailed list containing all the functions/methods to be used in the game:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Method Name | Return Type | Parameters | Location | Description |
| Read\_Database | String | - | Database\_Int() | Gets data from database |
| Write\_Database | Void | String | Database\_Int() | Updates the database |
| getColour | String | - | Lasers() | Getter for laser colour |
| getLaserCoordinates | Point | - | Lasers() | Getter for laser coordinates |
| getIsFriendly | Boolean | - | Lasers() | Getter for laser team |
| setColour | Void | String laserColour | Lasers() | Setter for laser colour |
| setLaserCoordinates | Void | Point value | Lasers() | Setter for laser coordinates |
| setIsFriendly | Void | Boolean isFriendly | Lasers() | Setter for laser team |
| moveLaserCoordinates | Void | Point laserCoordinates 1, Point laserCoordinates 2 | Lasers() | Moves the laser according to coordinates |
| readFrom\_Database | String | - | View\_High\_Score() | Getter for player name |
| displayResults | Void | String Value | View\_High\_Score() | Sets player name |
| updateHighScores | Void | String value | View\_High\_Score() | Updates the high score list |
| toggleSound | Void | - | Game() | Toggles sound ON or OFF |
| getColour | String | - | Options\_Menu() | Getter for player colour |
| setColour | Void | String playerColour | Options\_Menu() | Setter for player colour |
| convertHexToColour | Void | String hexColour | Options\_Menu() | Converts hex codes to colour |
| drawGameScreen | Void | Int width, int height | Game\_Screen() | Creates game screen |
| getSize | Int | - | Enemies() | Getter for enemy size |
| getHP | Int | - | Enemies() | Getter for enemy Hitpoints |
| getColour | String | - | Enemies() | Getter for enemy colour |
| getEnemyCoord | Point | - | Enemies() | Getter for enemy coordinates |
| setEnemyCoord | Void | Point value | Enemies() | Setter for enemy coordinates |
| setHP | Void | Int enemyHP | Enemies() | Setter for enemy hitpoints |
| setColour | Void | String enemyColour | Enemies() | Setter for enemy colour |
| moveEnemies | Void | Point enemyCoordinates 1, Point enemyCoordinates 2 | Enemies() | Moves enemies according to coordinates |
| getPlayer\_Shield | Int | - | Player() | Getter for player shields |
| getColour | String | - | Player() | Getter for player colour |
| getPlayerCoords | Point | - | Player() | Getter for player coordinates |
| setPlayer\_Shield | Void | Int playerShield | Player() | Setter for player shield |
| setColour | Void | String playerColour | Player() | Setter for player colour |
| setPlayerCoords | Void | Point playerCoords | Player() | Setter for player coordinates |
| movePlayerCoords | Void | Point playerCoords 1, Point playerCoords 2 | Player() | Moves player according to coordinates |

## High Score Database Design

The database will only serve to hold the name and high scores of the players, to a total of 10 entries, and as a result, is very simple.

As MS Visual Studio is being used for the developer environment, MS Access will be used to create the database. MS Visual Studio has synergy with MC Access, and so it is the most reasonable database platform choice.

## ERD

**Score**

#Rank

\*player\_ID

\*Score

\*Mode

**Player**

#player\_ID

\*Name

### Score

Primary Key: Rank

Foreign Key: player\_ID

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute Name | Database name | Optionality | Data type | Description |
|  | Rank | Not Null | Int(2) |  |
|  | Player\_ID | Not Null | Int(4) |  |
|  | Score | Not Null | Int(16) |  |
|  | Mode | Not Null | String |  |

### Player

Primary Key: player\_ID

Foreign Key: N/A

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute Name | Database name | Optionality | Data type | Description |
| player\_Name | Player\_ID | Not Null | Int(4) |  |
|  | Name | Not Null | String |  |

## Class Responsibility Collaboration Cards (CRC Cards)

CRC Cards are an object oriented technique useful to developers as a tool that assists with identification of classes and how they interact with each other. The cards display the responsibilities (what the class should do) and the collaborators (the classes it interacts with) of each class.

|  |  |
| --- | --- |
| Class Name: Options\_Menu | |
| Responsibilities | **Collaborators** |
| Toggles sound ON/OFF | N/A |
| Allows player to pick turret colour |  |

|  |  |
| --- | --- |
| Class Name: Game | |
| Responsibilities | **Collaborators** |
| Allows player to choose game mode | Menu |
| Initialises GUI |  |
| Initialises game\_Screen |  |
| Loads High Score Window |  |

|  |  |
| --- | --- |
| Class Name: Game\_Screen | |
| Responsibilities | **Collaborators** |
| Deals with user input | Game |
| Controls Game flow |  |
| Loads Player\_Name\_Window |  |

|  |  |
| --- | --- |
| Class Name: Player | |
| Responsibilities | **Collaborators** |
| Stores player coordinates | game\_Screen |
| Stores player lives/shield |  |

|  |  |
| --- | --- |
| Class Name: Enemies | |
| Responsibilities | **Collaborators** |
| Stores what enemy is being represented | game\_Screen |
| Stores individual enemy coordinates |  |

|  |  |
| --- | --- |
| Class Name: View\_High\_Score | |
| Responsibilities | **Collaborators** |
| Displays high\_Score\_List | Game |
| Converts high score database to String |  |
| Reads database using Database\_Int |  |

|  |  |
| --- | --- |
| Class Name: Player\_Name\_Window | |
| Responsibilities | **Collaborators** |
| Asks user to input name | Game\_Screen |
| Checks if inputted name is valid |  |
| Reads database using Database\_Int |  |
| Updates database using Database\_Int |  |
| Class Name: Lasers | |
| Responsibilities | **Collaborators** |
| Stores laser coordinates | Game |
| Creates lasers | Enemies |
| Stores laser team | Player |

|  |  |
| --- | --- |
| Class Name: Database\_Int | |
| Responsibilities | **Collaborators** |
| Reads Database | Player\_Name\_Window |
| Updates Database | View\_High\_Score |

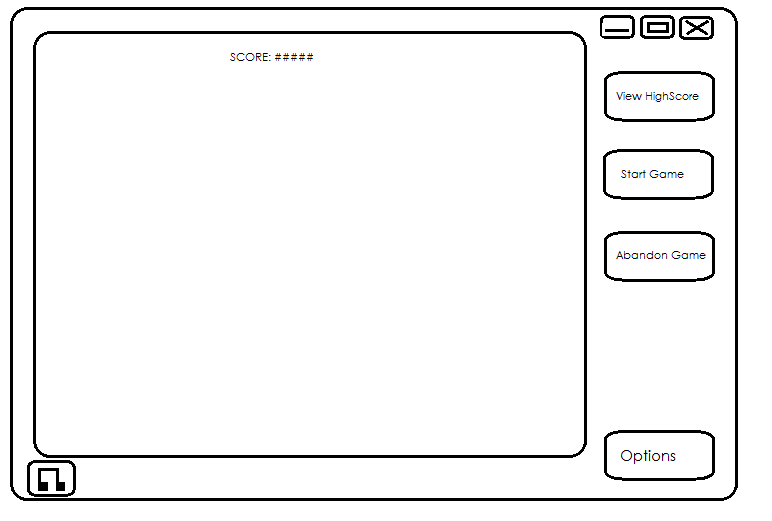
## Screen Designs

Below are the prototype designs for the proposed screens and game elements. All screens and elements seen below are subject to change during actual development.

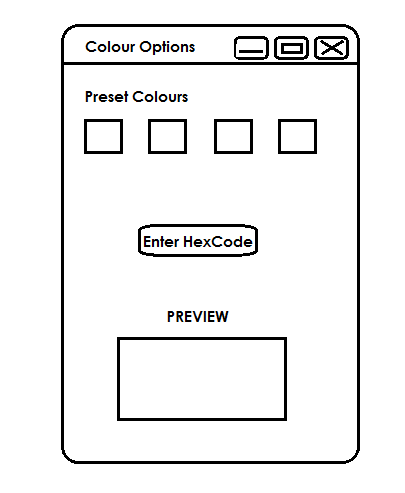
### Game Elements

|  |  |
| --- | --- |
| Element Type Design | |
| Enemy 1 |  |
| Enemy 2 |  |
| Enemy 3 |  |
| Enemy 4 |  |
| Bonus Enemy | **SUPER SECRET.** |
| Boss Enemy | C:\Users\Siobhan\Downloads\angry batman.png |
| Boss Laser | I:\Graded Unit\Game Resources\boss laser.png |
| Boss Health Bar (Full)  Boss HP is split into quarters, as shown. | I:\Graded Unit\Game Resources\boss full hp.png |
| Boss Health Bar (Damaged, 75%)  Red fills to show boss remaining Health | I:\Graded Unit\Game Resources\boss 75%.png |
| Player Shield (Fully Charged) | I:\Graded Unit\Game Resources\plasyer Shield.png |
| Player Shield (Nearly Depleted) | I:\Graded Unit\Game Resources\player shield depleted.png |
| Player Ship |  |
| Laser (Player) | I:\Graded Unit\Game Resources\LASER.png |
| Laser (Enemy) | I:\Graded Unit\Game Resources\Alien Laser.png |

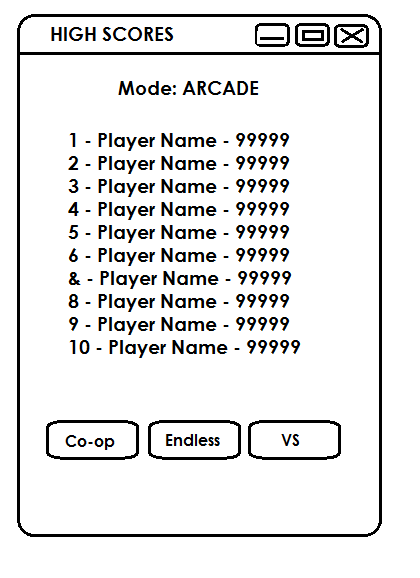
### Game Screen



### Options Screen



### High Score Screen



----------------------------------------------------------------------------------------------------

# Development

## Test Plan

## Full Pseudo Code

## Final Game Design

# Evaluation

## Testing

## Self-Assessment

# Appendix

# References

<https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/382206/Copyright_Designs_and_Patents_Act_1988.pdf>

<http://www.legislation.gov.uk/uksi/1992/3233/made>