# Battleships Video Game Specification

by Louis Bennette - 200801270

Battleships is a two-player table-top board game. Both players have a hidden hand comprising of two grids, the first being the placement of their battleships and the other being a grid of the attempts you've made to attack your opponent. The game requires the players to take turns at making guesses as to where the enemy's ships are. Whether the guess is a hit, miss or a ship sinking, the opponent must announce it. More specifically for this project I will be using the official Hasbro rule set[1]. I will be transferring all the concepts and rules over into a playable interactive video game.

The game is intended as a casual strategy game. Being heavily dependent on chance, it is not too competitive and so the target audience are relaxed and patient individuals. The game will be suitable for people of all ages but will appeal more to an audience that: has played the board game and liked it, plays similar quick and simple games or plays strategy games.

The project will be to create the battleships board game as a video game which can be played both against an AI in single player, or against an opponent online in multiplayer.

The project will therefore need to comprise of a game client and a game server, both of which will be written in java.

My aims are to complete the features listed below and to have, firstly, a robust and reliable connection handling and matchmaking server. With instanced orchestration of a game for two connected clients. Secondly, a client, displaying a 2D graphical interface for the user with multiple game modes to choose from and play.

### Features:

The following features are listed in order of importance:

- Windows Vista, 7 and 8 compatibility.
- A navigable menu for players to select what they want to do: Singleplayer, multiplayer, edit options or quit.
- A grid view to drag, drop and rotate ships at the start of a game.
- A grid view that changes based on whose turn it is. When it's the players turn they will see a grid representing their attempts at attacking the enemy. When it's the enemy's turn the player will see their own fleet and the enemy's attempts at attacking them.
- For the player's turn they will select where they want to fire and hit the "Fire Button" to confirm. The player will be told if the missile hit or not and hear an alert sound.
- Singleplayer will pair the player against an intelligent AI.
- Clear indication of whose turn it is. With the use of sounds and banners along the bottom or top of the screen.
- When a singleplayer game is over the player will be asked if they want to play again or exit to main menu.
- Server handling and orchestration of current running multiplayer games.
- For multiplayer, the player will be asked to enter a username before they are queued for matchmaking.

- When a multiplayer game is over the player will be asked if they want to requeue for a new game or exit to main menu.
- When in the menu or when in game the player will be able to select the options menu.
- The player will be able to mute the game.

Optional features:

- When a player fires their missile, a missile will launch from one of their active ships.
- When an enemy fires their missile the player will see it come into view onto the battlefield and fall to that grid cell with either a splash or an explosion.
- The server will allow players to rematch each other.
- Resolution Options for windowed or fullscreen.

## Statement of Deliverables:

Research phase:

In anticipation of the design, I will be making a few test projects to properly learn and understand the java server socket style of programming[5] that needs to be implemented for this game. I will also research whether the use of OO (Object-Oriented) design patterns[4] could help me with the implementation and overall structure of the project.

I will also need to conduct a few test projects to become familiar with the libraries I wish to use. Specifically LWJGL[2] (**L**ight**W**eight **J**ava **G**ame **L**ibrary) to properly understand how java deals with OpenGL (Open **G**raphics **L**ibrary) commands so that I can draw simple, shapes like rectangles. I will be learning to use unit tests properly and efficiently to test all of my code base as I implement the game.

Design documentation:

I will deliver a detailed document describing the classes and methods that will be implemented. This design document will be split between the design of the server and the design of the client. The software will be designed in a client-server approach.

The design of the server will be multithreaded. I will use threads as listeners for new connections, listeners for client to server messages, game instances that orchestrate a game for two players and event based handling threads. This complex structure will require a good design to work smoothly, in order to avoid common issues like deadlock. I will create a number of class diagrams to present the classes and more importantly to show how the threads will interact with each other.

The design of the client will revolve around a classic game main loop. The program will have multiple states that correspond to different views which will be drawn to screen. Assets such as sound and textures will be loaded and managed by a global class. There will be class diagrams for the client and for the server. I will also present statechart diagrams describing how all the program's states interact with each other. I will provide a use case diagram to show how the user navigates the program.

Implementation and Unit Testing:

At the end of the project I hope to deliver a working game with a main menu, single player mode, online multiplayer mode and options to modify sound and other settings.

I expect to have simple graphics consisting mostly of .png files that will be loaded as textures. Sound will also be available. Both the server and the client will be coded in java, I will be working in windows 7 using netBeans when working on my code.
The code will run with a game loop which will loop until the game is asked to shutdown. The process of one loop will update time and all of the entities on screen. Next it will draw all of the objects to an internal buffer representing the screen. Then, the input is taken in and the screen buffer is drawn to screen.

I will be using A GameLoop class, IGame interface, AbstractGame, SingleplayerGame, MultiplayerGame for game classes. These classes will initialise players, ui components and will be able to draw them all to the buffer. The GameLoop will be telling the Game classes when to update and draw everything to screen.

To complete the project, I will be using the following libraries:

- LWJGL[2] : this easily enables openGL programing in java.
- Slick2D[3] (Java Slick-Util) : this deals with textures and sounds by adding onto LWJGL.

Alongside coding my project I will be procedurally testing parts of my code with unit tests. Testing will be heavily done for the heuristic attached to my AI.

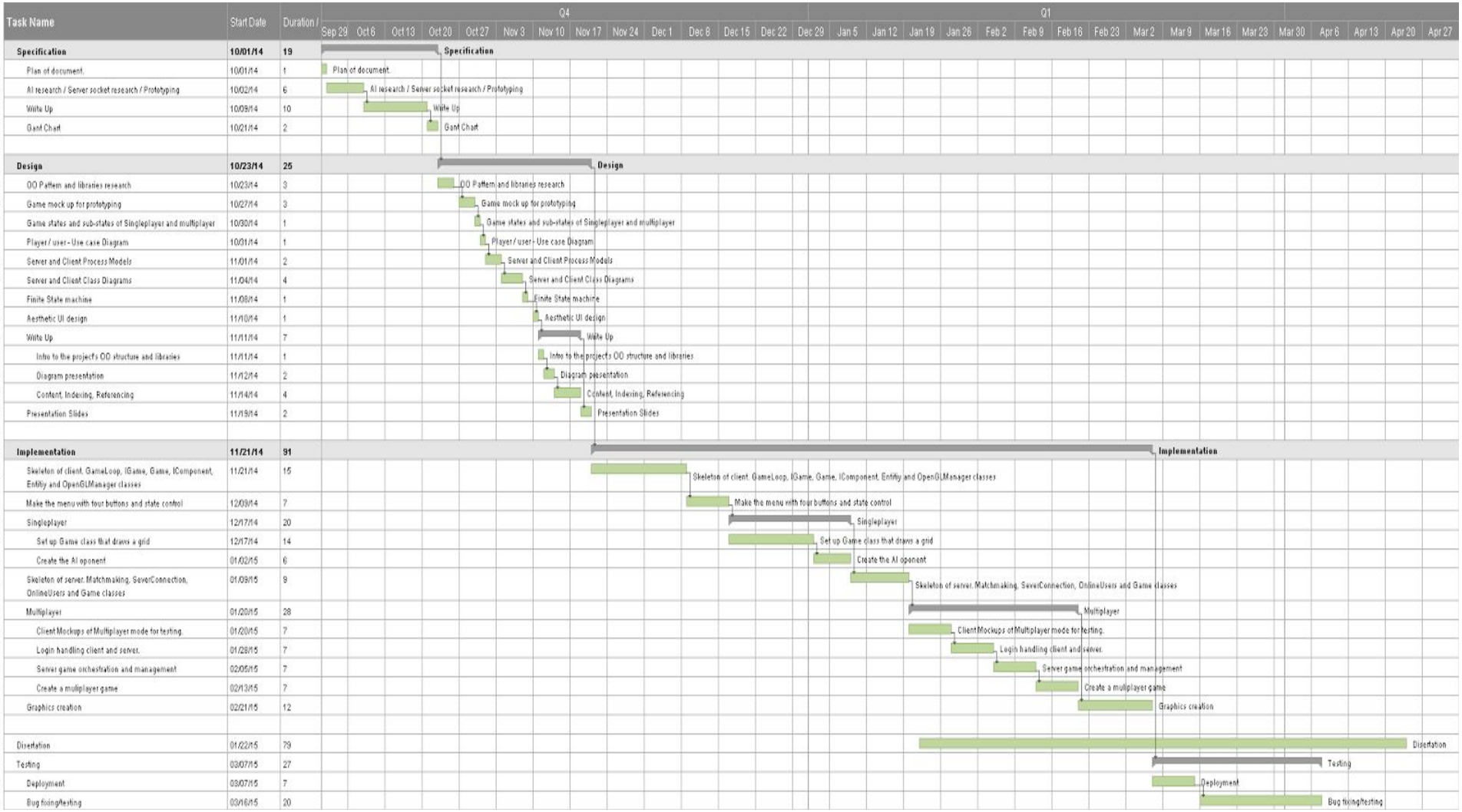
Testing:

At this stage I will freeze all of the features and begin to conduct tests. The implementation will be over. However, the code will be revisited to fix any bugs and “polish” the code. In this stage I will deploy the game to a number of people and ask them to play it, reporting any bugs along the way. Some of the testers, including myself, will be asked to try and break the game by performing odd and unusual actions. I will test the AI by pitting it against itself many times over to see if anything goes wrong.

## Resources:

1 Hasbro Rule Set - Milton Bradley Company:
http://www.hasbro.com/common/instruct/battleship.pdf
2 Official LWJGL website - Lightweight Java Game Library Project:
http://lwjgl.org/wiki/index.php?title=Main_Page
3 Slick2D - NinjaCave:
http://slick.ninjacave.com/javadoc-util/
4 Software Architecture Design Patterns in Java - by Linda Rising and Partha Kuchana
http://eds.b.ebscohost.com.ezproxy.liv.ac.uk/eds/detail/detail?sid=0939d2e5-7b8f-4824-a0db-137336d924dd%40sessionmgr115&vid=0&hid=104&bdata=JnNpdGU9ZWRzLWxpdmUmc2NvcGU9c2l0ZQ%3d%3d#db=cat00003a&AN=lvp.b2644314
5 BSD Sockets from a Multi-Language Perspective - by Jones, M. Tim
http://eds.b.ebscohost.com.ezproxy.liv.ac.uk/eds/detail/detail?vid=12&sid=3cc9cb5c-8d41-4272-8e6a-7d2a6963f58b%40sessionmgr113&hid=104&bdata=JnNpdGU9ZWRzLWxpdmUmc2NvcGU9c2l0ZQ%3d%3d#db=cat00003a&AN=lvp.b2020372

Full sized gant chart of the project available at:
http://louis.bennette.co.uk/downloads/battleships-gant-chart/battleshipsgant.png


Task Name
Start Date
Duration /
Q4
Q1
Sep 29
Oct 6
Oct 13
Oct 20
Oct 27
Nov 3
Nov 10
Nov 17
Nov 24
Dec 1
Dec 8
Dec 15
Dec 22
Dec 29
Jan 5
Jan 12
Jan 19
Jan 26
Feb 2
Feb 9
Feb 16
Feb 23
Mar 2
Mar 9
Mar 16
Mar 23
Mar 30
Apr 6
Apr 13
Apr 20
Apr 27
1 Specification 10/01/14 19
2 Plan of document. 10/01/14 1
3 AI research / Server socket research / Prototyping 10/02/14 6
4 Write Up 10/09/14 10
5 Gant Chart 10/21/14 2
6
7 Design 10/23/14 25
8 OO Pattern and libraries research 10/23/14 3
9 Game mock up for prototyping 10/27/14 3
10 Game states and sub-states of Singleplayer and multiplayer 10/30/14 1
11 Player / user - Use case Diagram 10/31/14 1
12 Server and Client Process Models 11/01/14 2
13 Server and Client Class Diagrams 11/04/14 4
14 Finite State machine 11/08/14 1
15 Aesthetic UI design 11/10/14 1
16 Write Up 11/11/14 7
17 Intro to the project's OO structure and libraries 11/11/14 1
18 Diagram presentation 11/12/14 2
19 Content, Indexing, Referencing 11/14/14 4
20 Presentation Slides 11/19/14 2
21
22 Implementation 11/21/14 91
23 Skeleton of client. GameLoop, IGame, Game, IComponent, Entitiy and OpenGLManager classes 11/21/14 15
24 Make the menu with four buttons and state control 12/09/14 7
25 Singleplayer 12/17/14 20
26 Set up Game class that draws a grid 12/17/14 14
27 Create the AI oponent 01/02/15 6
28 Skeleton of server. Matchmaking, SeverConnection, OnlineUsers and Game classes 01/09/15 9
29 Multiplayer 01/20/15 28
30 Client Mockups of Multiplayer mode for testing. 01/20/15 7
31 Login handling client and server. 01/28/15 7
32 Server game orchestration and management 02/05/15 7
33 Create a muliplayer game 02/13/15 7
34 Graphics creation 02/21/15 12
35
36 Disertation 01/22/15 79
37 Testing 03/07/15 27
38 Deployment 03/07/15 7
39 Bug fixing/testing 03/16/15 20
Specification
Plan of document.
AI research / Server socket research / Prototyping
Write Up
Gant Chart
Design
OO Pattern and libraries research
Game mock up for prototyping
Game states and sub-states of Singleplayer and multiplayer
Player / user - Use case Diagram
Server and Client Process Models
Server and Client Class Diagrams
Finite State machine
Aesthetic UI design
Write Up
Intro to the project's OO structure and libraries
Diagram presentation
Content, Indexing, Referencing
Presentation Slides
Implementation
Skeleton of client. GameLoop, IGame, Game, IComponent, Entitiy and OpenGLManager classes
Make the menu with four buttons and state control
Singleplayer
Set up Game class that draws a grid
Create the AI oponent
Skeleton of server. Matchmaking, SeverConnection, OnlineUsers and Game classes
Multiplayer
Client Mockups of Multiplayer mode for testing.
Login handling client and server.
Server game orchestration and management
Create a muliplayer game
Graphics creation
Disertation
Testing
Deployment
Bug fixing/testing