Criteria B

# Detailed Design Specifications

Below, Criteria A’s specifications are expanded on to show, in detail, what the expectations are for each.

|  |  |
| --- | --- |
| Criteria A | Criteria B |
| 1. Game allows students to practice spatial, logical or linguistic skills. | 1. A Grid is used |
| 2. Game is mainly visual, with limited written content. | 1. GUI used for all of the game 2. User friendly, easy to play / use / understand 3. Many Pictures 4. Themed |
| 3. Good user-interface design principles are used. | 1. Experimentation / usage of different widgets 2. Visually Appealing 3. Clear instructions 4. Restriction of Input 5. Widget Arrangement 6. Error handling 7. Scaled well |
| 4. Game is fun and engaging for grade 5 learning-disabled students. | 1. Suitable subject matter for grades 5-6 2. Parent-approved game |
| 5. Code is easily adaptable. | 1. Comments are used throughout 2. Many methods that are simple, therefore easily adaptable 3. Array based code |
| 6. Requires 1 or 2 players. | 1. 2 Player game, Computer AI (possibly) |
| 7. A new widget is used. | 1. Usage of many widgets, such as: 2. Pull-Downs, Radio Buttons, Menus |

# Feasible Design Ideas

Theme

As you can see of the left, I started by brainstorming themes for the game. My biggest reasoning for choosing these themes is that they could work well with the game of battleship (spaceships for ships, etc.)

Option 1: Star Wars

High Concept Statement: Galactic battles between spaceships! Ships and titles would be themed accordingly, you must launch missiles in attempt to destroy your enemy’s fleet.

Option 2: Pirates

High Concept Statement: Ahoy! Board your ships as captain and attempt to crush the enemy pirate. Ships themed accordingly, and pirates for players.

Option 3: Army / Battleship

High Concept Statement: Very similar colour scheme to the real game, but the most real because of that. Ships will be army ships, and themed as such. Hits will be splats or dots, and misses white spots.

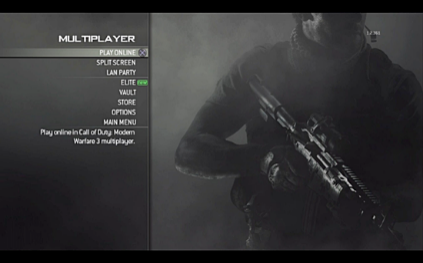
I picked option 3 (Army/ Battleship) as the theme for my game. This is because it was the one that fit best with how the game is played, and really puts the user into the army commander position. Conveniently, it also works really well for having two players (specification 6), as they are facing off against each other (two different sides). Because this theme has been done many times before, there are many potential palettes and pictures to use for it, making it visually appealing (specification 2). Alongside this, the theme will be fun and engaging for grade 5s, while being still parent-approved (hit splats aren’t very violent) (specification 4). In contrast, neither star wars nor pirates have been done before, making colour schemes and ships hard to find and create. This would make them a lot less visually appealing as well as engaging, not fully fulfilling the requirements of specifications 2 and 3. For these reasons, an army or battleship theme would be the best choice for my game!

Colour Scheme Selection

 Option 1: Legend of Zelda

* Background: either light pink or grey
* Foreground: Black writing, light or dark brown subtitles
* Accents: Grass Green, white, light blue, dark green

Option 2: Call of Duty / Army

* Desaturated in general
* Background: grey/black
* Foreground: White writing for most of it
* Accents: a colour dependent on the game – orange for black ops, yellow for modern warfare

 Option 3: Battleship

* Based on what I’m playing, but match well
* Background: grey/dark green/teals
* Foreground: lighter colours, desaturated
* Accents: green/blues, and reds for hits.

I am going to use a mix of the “Battleship” and “Army” themes to create the most effective theme. I will ensure to use colours that complement each other so that there is no conflict.

|  |  |  |
| --- | --- | --- |
| Colour | Use | RGB Code (R, G, B) |
| 9fc27a729df046683d885852edf553bc | Background | (120, 120, 120) |
| color-scheme-234-main.png (576×290) | Grids |  |
|  | Buttons | (51, 98, 106) |
|  | Info Buttons | (106, 139, 146) |
|  | Misses | (233, 238, 234) |
| 2000px-Red_Army_flag | Hits | (222, 0, 0) |

# Accurate and Detailed DEsigns

Screen Flow Diagram

There are 5 screens: start, instructions, selection, and both player screens. The battles will be done on the player screens, and you can get to start from any screen. (Except selection)

**Start**

0

Play

Instructions

**P1 Screen**

3

Fire!

Reset

**P2 Screen**

4

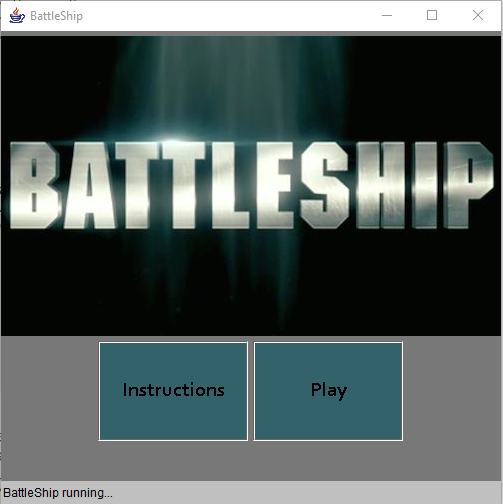
Reset

Fire!

Back

**Instructions**

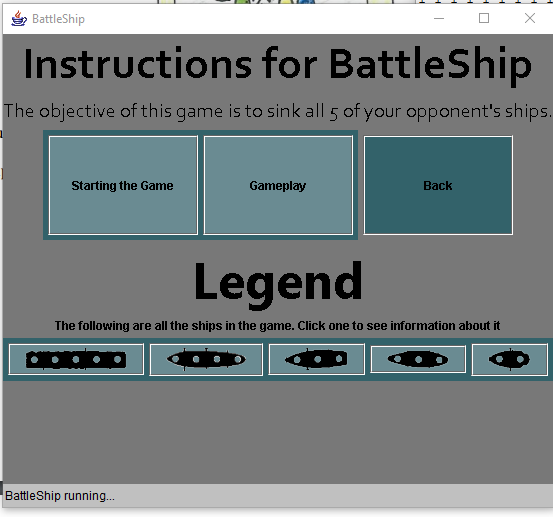
1



Screen Layout

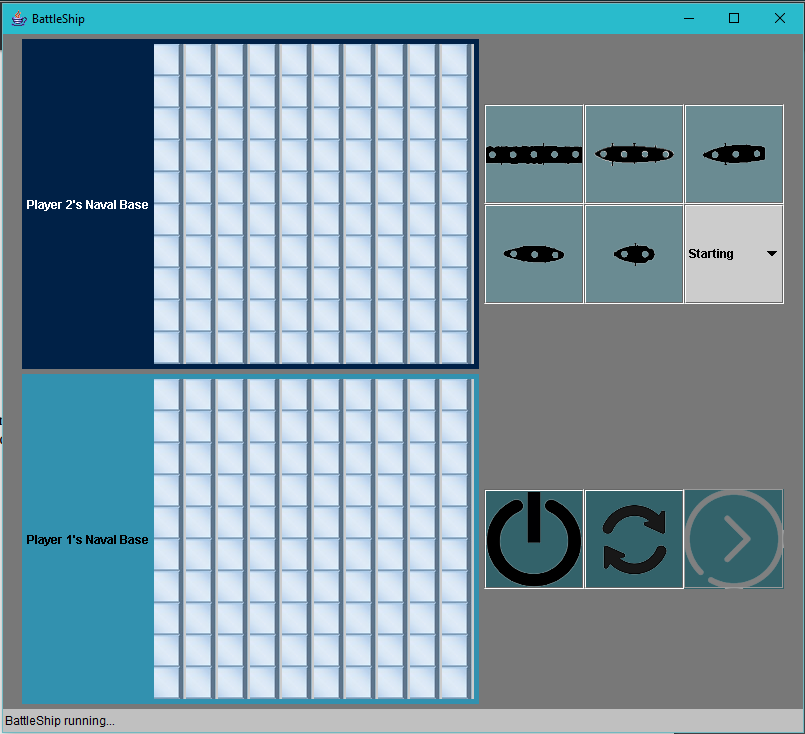
On the main menu screen, there are two buttons at the bottom of the screen, accompanied by a splash picture for my game (specification 2 – pictures!)

The instructions screen has a few pictures, as well as a title and boxes for each set of instructions. The legend is also divided under, so that players can understand how each ship is placed (drawn) (specification 2 – easy to understand).



Both the screens for player 1 and player 2 (specification 5 – 2 players) are created with two grids, and buttons, for both the ships and passing / resetting / rotating the ships. Upon use, (depending on the button), they are disabled (specification 3 – restriction of input). Also, these ships are unable to be placed on top of one another, or in an area which they would be outside the grid (specification 3 – error handing). Finally, both screens are themed, and appropriately so for the game and for their age group (specifications 2 and 4)

All screens are resized according to their content, but all fall below 1280x1024, or the required screen size. Also, there is a new widget used in the form of the JComboBox, as well as the JPanel (using flowlayouts) (specification 7).



Game Instructions

These instructions are taken straight from the official battleship website to ensure they are clear, consice and easy to understand.(specification 2 – user friendly). Also, the game is coded efficiently through many arrays, and else ifs used as often as possible, accompanied by both while and for loops, where needed (specifications 1 and 5).

These instructions are taken form the official Battlehip site:[[1]](#footnote-1)

Start by placing each of your five ships on the lower part (your part) of the battlefield. Use the rotate button to choose between vertical and horizontal, both orientations are fine! Your ships may NOT overlap! Once your ships are placed, you may not reset their position for the remainder of the game.

When it is your turn, you may click a position, then click lock in to fire at that position. Upon firing, you will be notified if you have hit a ship, or missed.If you hit a ship, you will be notified which one. Based on the key, your following shots should keep in mind the length of the ship (If you've hit a carrier, it's a length of five, so ensure to keep firing till it's sunk). Ensure to have fun while playing!

They have been somewhat modified for the purposes of my game (click instead of place, etc.).

To the left is a visual demonstration of the game, or what I’m trying to simluate. The image is made my hasbro, but distributed to many sales locations[[2]](#footnote-2).

Pictures

More than helping specification 2 (visually appealing), these pictures will be used to create my game in all ways (setting up my board, firing, etc.)

Board

|  |  |  |  |
| --- | --- | --- | --- |
| 1.jpg | 501.jpg | 1001.jpg | 10001.jpg |
|  |  |  |  |

Progression Buttons

|  |  |  |
| --- | --- | --- |
| Crosshair.png | Reset.png | Rotate.png |
|  |  | D:\School\Comp Sci\Unit 6 - Battleship\rotate.png |

Boats

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Carrier.png | Battleship.png | Crusier.png | Submarine.png | Destroyer.png |
|  |  |  |  |  |
| Carrier2.png | Battleship2.png | Crusier2.png | Submarine2.png | Destroyer2.png |
|  |  | C:\Users\Maanav\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Cruiser2.png |  |  |

Grid Pieces

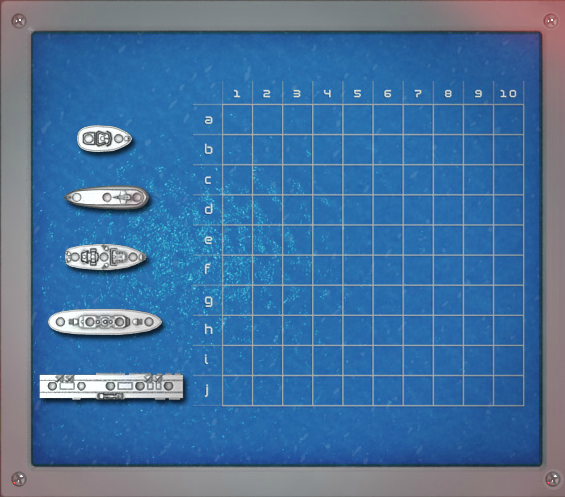
Because Nobody wants to see all 17 pieces of all the ships divided into 4 different types for a total of 68 images, I’ll show a demonstration using the three-hole crusier of all the types of image.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name / Picture | Name / Picture | Name / Picture | Part of the ship | Orientation |
| 31.jpg | 32.jpg | 33.jpg | Ship Parts 1-3 | Vertical |
| 231.jpg | 232.jpg | 233.jpg | Ship Parts 1-3 | Horizontal |
| 1031.jpg | 1032.jpg | 1033.jpg | Hit parts 1-3 | Vertical |
| 1231.jpg | 1232.jpg | 1233.jpg | Hit Parts 1-3 | Horizontal |

Main Menu



Variables and Array Structures

Actual Game Board(s)

[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

(there’s two like this for each player)

[0] [1] [2] [3] [4] [5] [6] [7] [8] [9]

They will all hold data as follows:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | [0] | [1] | [2] | [3] | [3] | [4] | [5] | [6] | [7] | [8] | [9] |
| [0] | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| [1] | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| [2] | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| [3] | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| [4] | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| [5] | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| [6] | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| [7] | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| [8] | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| [9] | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

500 will be added for misses, 1000 for hits (for both friendly and enemy grids).

**The arrays will be declared like this:** int p1array[] [] = {{1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, {1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, {1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, {1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, {1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, {1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, {1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, {1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, {1, 1, 1, 1, 1, 1, 1, 1, 1, 1}, {1, 1, 1, 1, 1, 1, 1, 1, 1, 1}};

It’s the same for all four arrays (p1array, p2array, p1enemy, p2enemy).

**The grids will be displayed using a one-dimensional button array:**

JButton p1b[] = new JButton [100];

JButton p1eb[] = new JButton [100];

JButton p2b[] = new JButton [100];

JButton p2eb[] = new JButton [100];

**These together will allow me to draw all of my grids very easily throughfour loops:**

for (int i = 0 ; i < p1b.length ; i++)

{

p1b [i] = new JButton (" ");

p1b [i].addActionListener (this);

p1b [i].setActionCommand ("a" + i);

p1b [i].setBorder (null);

p1b [i].setBorderPainted (false);

p1b [i].setPreferredSize (new Dimension (32, 32));

p.add (p1b [i]);

}

**There are variables to track multiple things, including the turn, which ship is selected, and even more:**

int home = 1; - to reset and go to home after reset

char selectedship = 'n'; - shows which ship is selected

int stop = 0; - to stop continued firing

int rotatecounter = 1; - to check which way the player wants to place their ships

int turn = 0; to swap turns, therefore screens and boards.

… you get the idea, there’s many variables for many things.

**Finally, changing of ships is done through the fire method:**

(just a small section of it, after all the checks of it being a valid move)

if (p2array [x] [y] > 1 && p2array [x] [y] < 5000)

{

p1enemy [x] [y] += 1000;

p2array [x] [y] += 1000;

}

else

{

p1enemy [x] [y] += 500;

p2array [x] [y] += 500;

}

1. http://www.hasbro.com/common/instruct/Battleship.PDF [↑](#footnote-ref-1)
2. http://www.toysrus.ca/product/index.jsp?productId=23386586 [↑](#footnote-ref-2)