Software Requirements Specification

for

Battleship

Version 1.2 approved

Prepared by Logan Price

LP Games

August 27th, 2020

Copyright © 1999 by Karl E. Wiegers. Permission is granted to use, modify, and distribute this document.

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Logan Price	8/28/20	Original Document	1.0
Logan Price	9/7/20	Updated System Features	1.1
Logan Price	9/24/20	Added Rules to System Features	1.2

1. Introduction

1.1 Purpose

The purpose of this program is to provide a two-player Battleship game playable on a Windows 10 computer using Python 3. This game will allow two player on one computer to enjoy the traditional board game of Battleship without the need for a physical copy of the game.

1.2 Document Conventions

This document uses no specific formatting conventions.

1.3 Intended Audience and Reading Suggestions

This document is intended for those writing the software to provide guidelines for what the software must fulfill. This document is also for those who wish to learn more about the process of this code's development.

1.4 Product Scope

Sometimes it is more convenient to have a computer with software on it to play games with friends because carrying around all the parts to play some board games, Battleship included, can be too much. This program aims to ease the load of carrying around a large box full of pieces and replacing that with a thin laptop that someone is likely to already have with them. This program also does not have the possibility of losing small pieces of having any parts break like a physical game.

1.5 References

Not applicable to this project.

2. Overall Description

2.1 Product Perspective

This program is not associated with another system in particular, but it is meant to be a replacement to the popular board game of the same name.

2.2 Product Functions

This product must allow two players to play the Battleship game on one computer. In this game of Battleship, players will place 5 ships on their ship grid and their opponent will have to guess where the ships are on that grid by firing missiles at it. Each ship has a set number of sections that it takes up on the grid. A player can fire one missile on their turn by selecting a tile on their missile *grid. Once all of the sections of a ship has been struck with a missile, that ship sinks. Once all ships of a player have been sunk, that player loses.*

2.3 User Classes and Characteristics

Game: Holds information about the current game and objects of the current instance.

Player: Holds information about the player including their grids, ships, and names.

Grid: Parent class for each the ship grid and the attack grid (10x10 grid)
ShipGrid: grid which a player places their ships onto
AttackGrid: grid which a player fires their missiles on.

Ship: Parent class for each of the ship pieces
Carrier: ship piece of length 5
Battleship: ship piece of length 4
Destroyer: ship piece of length 3
Submarine: ship piece of length 3
Patrol Boat: ship piece of length 2

Missile: piece that players fire to try and sink enemy ships.

App: Tkinter GUI manager class

(tk.Frame Classes)
I'm not sure how many screens I will need yet but each screen needs its own class in tkinter

2.4 Operating Environment

This program will operate on computers that are running Windows 10.

2.5 Design and Implementation Constraints

Must be completed before the end of Gardner-Webb's fall 2020 semester.

2.6 User Documentation

There will be a brief set of rules explaining how the game works to those who have never played Battleship.

2.7 Assumptions and Dependencies

Not applicable to this project.

3. External Interface Requirements

3.1 User Interfaces

This program will operate using a GUI. In this GUI the players will be able to place their pieces, shoot their missiles, and play the game as they normally would physically.

3.2 Hardware Interfaces

Not applicable to this project.

3.3 Software Interfaces

Not applicable to this project.

3.4 Communications Interfaces

Not applicable to this project.

4. System Features

4.1 Two Players

This program should allow two players to play the battleship game.

4.2 Player Names

Players should be able to enter their names when starting the game. They will be referred to as this name within the session, to tell them that its their turn, for example

4.3 Ship Placement

Players should be able to place their ships on their ship grid so long as they are completely on the grid and overlap no other ships.

4.4 Horizontal and Vertical Placement

Players should be able to place their ships in either a horizontal or vertical position.

4.5 Setup Change

Players should be able to change the position of any of their boats before they confirm their selection

4.6 Setup Confirmation

Players should be able to confirm the setup of their boat locations, locking them in place for the duration of the game.

4.7 Game Start

The game should begin once both players have entered their name and finished (confirmed) placing all of their ship pieces on their ship grid.

4.8 Ship Grid Display

Players should be able to see their ship grid, which will show the ships that they placed along with enemy missile strikes when it is their turn.

4.9 Attack Grid Display

When a player starts their turn, they should be able to see their attack grid, which will show the missile strikes that they have executed this game.

4.10 Enemy Activity Message

At the start of a player’s turn, a message should be displayed saying what happened on the last turn. This message will say what location the player shot at and what it if, if anything.

4.11 Missile Attack Location Selection

During a player’s turn, that player should be able to select a location on their attack grid to send their missile for that turn on.

4.12 Missile Location Confirmation

After selecting a space on the attack grid, players should be able to confirm their attack, sending a missile to their opponents ship grid.

4.13 Missile Strike Message Feedback

After confirming a player’s missile strike, the program should deliver a feedback message saying if that player hit or not.

4.14 Missile Feedback Attack Grid

After a player confirms their missile strike, that missile should appear on their attack grid. It will be red if it hit or white if it missed.

4.15 Turn Change

Players should see an intermediate screen between turns so they can swap who is looking at the screen without either one seeing their opponent's ships.

4.16 Surrender

During a player's turn, that player should be able to surrender, causing them to automatically lose the game.

4.17 End Turn

Player's should be able to end their turn, either ending the game or going to the next player's turn.

4.18 Game Over

After a player surrenders or all of a player's ships are sunk, the game should end, displaying who won the game.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

This program should be able to run 90% of the times it is opened.

5.2 Safety Requirements

Not applicable to this project.

5.3 Security Requirements

Not applicable to this project.

5.4 Software Quality Attributes

The main attribute that needs to be present in this project is that is fulfills the requirements of this document and the software does what its advertised to do.

5.5 Business Rules

Not applicable to this project.

6. Other Requirements

Not applicable to this project.

Appendix A: Glossary

Ship – general term for the pieces of the game that each player has. These are the pieces each player's enemy wishes to destroy.

Carrier – length 5 ship

Battleship – length 4 ship. Also the name of the game.

Destroyer – length 3 ship.

Submarine – length 3 ship

Patrol Boat – length 2 ship

Missile – Object used to sink ships in the game.

Grid – a 10 by 10 plane that ship and missiles can be on.

Ship grid – a grid that a player places their ships onto. Enemy players shoot missiles onto this grid using their attack grid

Attack grid – a grid used for a player to shoot their missiles from.

Appendix B: Analysis Models

Not applicable in this project.

Appendix C: To Be Determined List

Not applicable in this project.