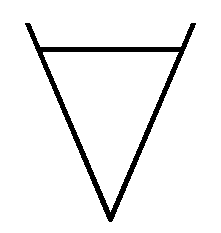
ASTEROIDS ++ Software Requirements Specification



**VERSION 1.0**

**ECSE-321-001: Introduction to Software Engineering**

**Instructor: Professor Haibo Zeng**

**2/10/2013**

Prepared by

|  |  |  |
| --- | --- | --- |
| 260477045 |  | Juan Guzman |
| 260395716 |  | Graham Ludwinski |
| 260376364 |  | Khoi Tran-Quang |
| 260445956 |  | Nicole Witter |
| 260549690 |  | Zhipeng Zheng |

# Contents

**1 INTRODUCTION3**

1.1 Purpose3

1.2 Scope3

1.3 Definitions, acronyms, and abbreviations5

1.4 References6

1.5 Overview7

**2 OVERALL DESCRIPTION7**

2.1 Product Perspective7

2.1.1 System Interfaces7

2.1.2 User Interfaces8

2.1.3 Hardware Interfaces8

2.1.4 Software Interfaces8

2.1.5 Communication Interfaces9

2.1.6 Memory Constraints9

2.1.7 Operations9

2.1.8 Site Adaptation Requirements10

2.2 Product Functions and Use Cases10

2.3 User Characteristics17

2.4 Constraints18

2.5 Assumptions and Dependence18

2.6 Apportioning of Requirements19

**3 SPECIFIC REQUIREMENTS19**

3.1 Functional Requirements19

3.2 Quality Requirements20

3.2.1 Availability20

3.2.2 Reliability21

3.2.3 Compatibility21

3.2.4 Maintainability21

3.2.5 Usability22

3.2.6 Performance22

3.2.7 Stability22

3.2.8 Open Source22

3.3 Design Constraints22

3.4 Other Requirements22

# Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Author | Date | Description | Version |
| Peter Zheng | 2/10/2013 | First version | <1.0> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

* 1. **Purpose**

This document serves to outline and describe the software specifications of the shooting game *Asteroids++*, which shall be based off Atari’s successful and popular retro game *Asteroids*, released in November of 1979.

The intended audience of this document includes the software developers, the supervisors of this project, and the client base interested in the development of the game.

* 1. **Scope**

The original *Asteroids* is a 2D shooting game in which the player takes control of a spaceship in the middle of an asteroid field. The world wraps around in both axes of the screen. Using a ship that can rotate left or right, accelerate forward and can shoot forward, the player must destroy as many asteroids and flying saucers as possible while avoiding collisions of any type; a collision shall destroy the ship.

*Asteroids++,* as the name implies, will be an improved version of the original *Asteroids*. The system will keep many aspects of the original game, but will have new features which shall greatly enhance the gameplay. A large variety of weapons, the possibility of having an alternate and more efficient method of controlling the ship, and an enhanced yet user-friendly graphical user interface will all be implemented. There will be different levels of difficulty, as well as appropriate sounds or music to enhance the gaming experience. Two players will be allowed to play simultaneously. There will also be different saucers, the potential to save a game, and a possibility of online play. Also implemented will be a high score system that displays achievements and statistics. The goal of *Asteroids++* is to enable users to relive the glory of Atari’s *Asteroids* in a different, but significantly improved light.

* 1. **Definitions, acronyms, and abbreviations**

|  |  |
| --- | --- |
| Term | Definitions |
| Software requirements specification (SRS) | A complete description of the behavior of a system to be developed and may include a set of use cases that describe interactions the users will have with the software. (1) |
| Javadoc | A documentation generator from Oracle Corporation for generating API documentation in HTML format. (2) |
| Application programming interface (API) | Protocol intended to be used as an interface by software components to communicate with each other. (3) |
| HyperText Markup Language (HTML) | Main markup language for creating web pages and other information that can be displayed in a web browser. (4) |
| Graphic user interface (GUI) | Type of user interface that allows users to interact with electronic devices using images rather than text commands. (5) |
| Local area network (LAN) | Computer network that interconnects computer in a limited area. (6) |
| Artificial intelligence (AI) | Used to produce the illusion of intelligence in the behavior of non-player characters. (7) |
| TCP/IP protocol | Is the set of [communications protocols](http://en.wikipedia.org/wiki/Communications_protocol) used for the [Internet](http://en.wikipedia.org/wiki/Internet) and similar networks, and generally the most popular [protocol stack](http://en.wikipedia.org/wiki/Protocol_stack) for [wide area networks](http://en.wikipedia.org/wiki/Wide_area_network). It is commonly known as TCP/IP, because of its most important protocols: [Transmission Control Protocol](http://en.wikipedia.org/wiki/Transmission_Control_Protocol) (TCP) and [Internet Protocol](http://en.wikipedia.org/wiki/Internet_Protocol) (IP), which were the first networking protocols defined in this standard. (8) |
| WebRTC | Is an [API](http://en.wikipedia.org/wiki/Application_programming_interface) definition being drafted by the [World Wide Web Consortium](http://en.wikipedia.org/wiki/World_Wide_Web_Consortium) (W3C) to enable browser to browser applications for [voice calling](http://en.wikipedia.org/wiki/Voice_calling), [video chat](http://en.wikipedia.org/wiki/Video_chat) and [P2P file sharing](http://en.wikipedia.org/wiki/P2P_file_sharing) without [plugins](http://en.wikipedia.org/wiki/Plugins). (9) |
| Application Programming Interface (API) | A protocol intended to be used as an [interface](http://en.wikipedia.org/wiki/Interface_%28computing%29) by [software components](http://en.wikipedia.org/wiki/Software_component) to communicate with each other. (10) |
| SAD | Software architecture document |
| User (Player) | Users use the application without complete technical expertise required to understand the system fully. (11) |
| Application | Computer software designed to help the user perform specific tasks. (12) |

* 1. **References**

1. Wikipedia contributors, “Software requirements specification.” *Wikipedia, The Free Encyclopedia,* http://en.wikipedia.org/wiki/Software\_requirements\_specification (Accessed February 7, 2013).
2. Wikipedia contributors, “Javadoc.” *Wikipedia, The Free Encyclopedia,* <http://en.wikipedia.org/wiki/Javadoc> (Accessed February 7, 2013).
3. Wikipedia contributors, “Application programming interface.” *Wikipedia, The Free Encyclopedia,* <http://en.wikipedia.org/wiki/Application_programming_interface> (Accessed February 7, 2013).
4. Wikipedia contributors, “HTML.” *Wikipedia, The Free Encyclopedia,* <http://en.wikipedia.org/wiki/HTML> (Accessed February 7, 2013).
5. Wikipedia contributors, “Graphical user interface.” *Wikipedia, The Free Encyclopedia,* <http://en.wikipedia.org/wiki/Graphical_user_interface> (Accessed February 7, 2013).
6. Wikipedia contributors, “Local area network.” *Wikipedia, The Free Encyclopedia,* <http://en.wikipedia.org/wiki/Local_area_network> (Accessed February 7, 2013).
7. Wikipedia contributors, “Artificial intelligence.” *Wikipedia, The Free Encyclopedia,* <http://en.wikipedia.org/wiki/Artificial_intelligence_%28video_games%29> (Accessed February 7, 2013).
8. Wikipedia contributors, “Asteroids (video game).” *Wikipedia, The Free Encyclopedia,* <http://en.wikipedia.org/wiki/Asteroids_(video_game)> (Accessed February 7, 2013).
9. Wikipedia contributors, “Internet protocol suite.” *Wikipedia, The Free Encyclopedia,* <http://en.wikipedia.org/wiki/Internet_protocol_suite> (Accessed February 7, 2013).
10. Wikipedia contributors, “WebRTC.” *Wikipedia, The Free Encyclopedia,* <http://en.wikipedia.org/wiki/WebRTC> (Accessed February 7, 2013).
11. Wikipedia contributors, “Application programming interface.” *Wikipedia, The Free Encyclopedia,* <http://en.wikipedia.org/wiki/Application_programming_interface> (Accessed February 7, 2013).
12. Wikipedia contributors, “User (computing).” *Wikipedia, The Free Encyclopedia,* <http://en.wikipedia.org/wiki/User_%28computing%29> (Accessed February 7, 2013).
13. Wikipedia contributors, “Application software.” *Wikipedia, The Free Encyclopedia,* <http://en.wikipedia.org/wiki/Application_software> (Accessed February 7, 2013).
14. IEEE. *IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.* IEEE Computer Society, 1998.
    1. **Overview**

The rest of this document contains an overall description of the *Asteroids++* video game (section 2 of the SRS), and the specific requirements of the game (section 3 of the

SRS).

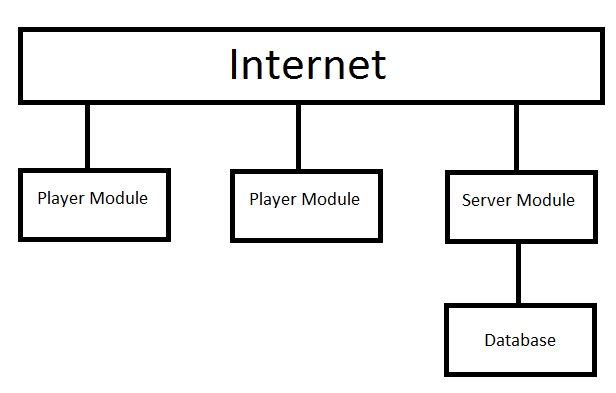
# Overall Description

* 1. **Product Perspective**

This section aims to describe how the relevant interfaces will work within the system

* + 1. **System Interfaces**

The game is a completely independent application that allows a user to play in two different modes: single player or multiplayer. The system contains three major components: the *player module,* the *server module* and the *database*.



The player module contains the main components of the video game. This module shall ensure the stability and functionality of the game.

The server module shall handle connections between different players to create a multiplayer environment for the game.

The database shall contain all the information about the users.

* + 1. **User Interfaces**

The player module shall be the only module that shall have a graphic user interface (GUI). The server module shall be initialized automatically when the users launch the application and it won’t provide any user interface.

* + 1. **Hardware Interfaces**

*Asteroids++* must be able to run on any conventional computer.

* + 1. **Software Interfaces**

The player module shall be developed using Java. After this module is complete, it shall be implemented as a Java applet to allow the game to run in a browser. Furthermore, JavaScript shall be used to create a communication link between online players.

* + 1. **Communications Interfaces**

The player module shall communicate with the server module using a local area network (LAN). The server module requires access to the database.  
 The player module shall be available for download through any modern browser on a computer that has Java installed.  This communication shall happen via HTTP through the TCP/IP protocol over an internet connection.  Any multiplayer communication shall happen either via HTTP through TCP/IP or via web sockets over a single TCP/IP connection per user.  Additionally, the newly released WebRTC specification could be used to allow for direct serverless browser-browser communication, effectively reducing the latency for online players.

* + 1. **Memory Constraints**

The game should be able to run on everyday consumer PC’s which means that the game should not use more than 500MB of memory.  In practice, a game of this size and scope shall require much less than that.  As such, there are not any practical limits on memory. However, *Asteroids++* shall be optimized as much as possible.

* + 1. **Operations**

Most of the player module operations shall be invisible to the users. The users shall only be capable of initializing the game, controlling the spaceship and accessing to the options menu. Once the game is initialized, the player module shall be completely responsible for the game. This module must take control of any operation involving the artificial intelligence and the physics of the game. The options menu shall allow the users to customize the controls or gameplay style of the spaceship, and to turn off/on the sounds of the game.

The server module shall be responsible for handling connections between different players, and for maintaining or updating the database. In the database, the server module shall store the highest scores accomplished by the players. There shall not be a backup tool developed for the database.

* + 1. **Site Adaptation Requirements**

There are no special site adaptations needed to play the video game.

* 1. **Product Functions and Use Cases**

The main function of *Asteroids++* is emulate *Asteroids* with the intention of making it a vastly improved version over the original. In this game, the users shall be able to choose between two different modes of gameplay: single player or multiplayer.

Four controlling functions shall ensure the functionality of the game:

* The first function shall handle the behavior of the asteroids. This function must be capable of controlling the quantity asteroids and their trajectory. Also, this function must balance the game to maintain a reasonable degree of difficulty for the game.
* The second function shall control the artificial intelligence of the game. This function shall add new levels of difficulty to the game. The artificial intelligence shall operate the enemies’ spaceships. Therefore, this function must provide the saucers with the capability of avoiding asteroids while attacking the player at the same time.
* The third function shall operate the “player spaceship”. This function must be capable of rotating the ship and must allow the ship to thrust. The response time of this function must be as fast as possible to ensure that the game runs with the lowest latency as possible. Also, this function must interact with the game environment such as when the player gains a new power up and destroy the ship when the player collides into an object. Within this function, there shall be a method for controlling the behavior of the shots fired from the ship.
* The last function shall handle and update the database. This database shall store the names of the players and their highest scores.

**Use cases:**

|  |  |
| --- | --- |
|  |  |
| Use Case Name | User Account |
| Participating Actors | Initiated by the user. |
| Entry Conditions | The user selects “Sign up” from the main screen. |
| Flow of Events | 1. The user initializes the game. 2. The user press “Sign up” button. 3. The application opens the Sign up menu window. 4. User fills all required information to make a new user account. 5. The user presses “Create account” button. 6. The application creates the new account and stores the information in the database. 7. The application closes the Sign up window and returns to the main screen. 8. User signs in. |
| Exit Conditions | * User presses “Sign up” button. * User closes the Sign up window. |
| Quality Requirements | * User must be able to operate a keyboard. * The application must be able to recognize input from the user * Application must be able to modify and update the database. |
| Exceptions | * The user tries to create an account using not valid characters. |
| Error Flow | 1. The user fills the required information in the Sign up window. 2. The user presses “Create account” button. 3. The application detects that the user used invalid characters. 4. The application prompts a message telling that the user has to use valid characters and asks the user to correct their information. |

|  |  |
| --- | --- |
|  |  |
| Use Case Name | Single Player |
| Participating Actors | Initiated by the user. |
| Entry Conditions | The user selects “Single Player” option from the main screen. |
| Flow of Events | 1. The user initializes the game. 2. The application initializes the player module. 3. The user logs in. 4. The user press “Single Player” button. 5. The application initializes all the controlling functions (these function are described in the Product function, section 2.2 of the SRS) 6. User operates the spaceship by pressing the arrow keys (or the keys that the users decided to use to operate the ship). 7. If the user closes the application, the system asks if the game progress should be saved. 8. If the user saves the game, the Player Module stores the saved game in the database. 9. The application ends the game when the player loses. 10. The system stores the highest record of the user in the database 11. Application returns to the main screen. |
| Exit Conditions | * All the user’s ships have been destroyed. * User closes the main window. |
| Quality Requirements | * User must be able to press different keys from the keyboard. * The application must be able to respond to the user input. * The application must be able to access and modify the database |
| Exceptions | * The user tries to log in with an invalid user account/password * The application can’t find the user account in the database. |
| Error Flow | 1. The user tries to log in. 2. The application tries to validate the account information. 3. The application fails to validate the account information. 4. The application prompts the following message “Wrong username or password”. 5. The user verifies the account information and tries to log in again. |

|  |  |
| --- | --- |
|  |  |
| Use Case Name | Options |
| Participating Actors | Initiated by the user. |
| Entry Conditions | The user selects “Options” from the main screen. |
| Flow of Events | 1. The user initializes the game. 2. The application initializes the Player Module. 3. The user logs in. 4. The user presses the “Options” button. 5. The application opens the options menu window. 6. The user changes the desired parameters of the game and presses save. 7. The application saves the changes by the user. 8. The application closes the options menu window and returns to the main screen. |
| Exit Conditions | * User presses save button. * User closes the options menu window. |
| Quality Requirements | * User must be able to operate a keyboard. * The application must be able to save the changes made by the user. |
| Exceptions | * The user tries to change the controlling keys with invalid characters. |
| Error Flow | 1. The user tries to modify the controlling keys. 2. The application tries to save the changes. 3. The application fails to change the controlling keys. 4. The application prompts the following message “Invalid combination of keys”. 5. The user tries to change the keys again. |

|  |  |
| --- | --- |
|  |  |
| Use Case Name | High Scores |
| Participating Actors | Initiated by the user. |
| Entry Conditions | The user selects “High Scores” from the main screen. |
| Flow of Events | 1. The user initializes the game. 2. The application initializes the player module. 3. The user logs in. 4. The user presses the “High Scores” button. 5. The application accesses the database. 6. The application loads the highest scores from the database into the high scores window. 7. The user sees the highest scores and press back 8. The system closes the highest scores window and returns to the main screen. |
| Exit Conditions | * User presses back. * User closes the highest scores window. |
| Quality Requirements | * User must be able to operate a computer mouse. * Application must be able to open and show the data stored in the database. |
| Exceptions | * The applications can’t access the database * The database file is corrupted. |
| Error Flow | 1. The user presses the “High score” button. 2. The application tries to access the database. 3. The application fails to load the highest scores. 4. The application prompts the following message “Impossible to access the database, please try again later”. 5. The application returns to the main screen. |

|  |  |
| --- | --- |
|  |  |
| Use Case Name | Multiplayer (2 players) |
| Participating Actors | Initiated by the user. |
| Entry Conditions | The user selects “Multiplayer” from the main screen. Then selects 2 players. |
| Flow of Events | 1. The user initializes the game. 2. The application initializes the player module. 3. The user chooses “Multiplayer”. 4. The application starts the game. 5. Players take turn playing the game. The game switches players when a player either beats a level, or loses a life. 6. If both players lose all their lives, the game is over. 7. The system stores the highest score of both users in the database when one of the two players. 8. Application returns to the main screen. |
| Exit Conditions | * User closes the main window. * All users lose all their lives. |
| Quality Requirements | * User must be able to operate a keyboard * Application must be able to respond to the user input. * Application must be able to create a connection between two players |
| Exceptions | * The user tries to log in with an invalid user account/password (online play). * The application can’t find the user account in the database. |

|  |  |
| --- | --- |
|  |  |
| Use Case Name | Multiplayer (Cooperative play) |
| Participating Actors | Initiated by the user. |
| Entry Conditions | The user selects “Multiplayer” from the main screen. Then selects Online play and Cooperative play. |
| Flow of Events | 1. The user initializes the game. 2. The application initializes the player module. 3. The user logs in. 4. The user chooses “Multiplayer”. 5. The application initializes all the controlling functions (these function are described in the **Product Functions**, section **2.2** of this document) 6. User logs on. 7. The server module creates a connection between users if it is for online play. 8. The application starts the game. 9. “User 1” operates “ spaceship 1” and “User 2” operates “ spaceship 2” 10. If both players lose all their lives, the game is over. 11. The system stores the highest score of both users in the database when one of the two players. 12. Application returns to the main screen. |
| Exit Conditions | * User closes the main window. * All users lose all their lives. |
| Quality Requirements | * User must be able to operate a keyboard * Application must be able to respond to the user input. * Application must be able to create a connection between two players |
| Exceptions | * The user tries to log in with an invalid user account/password (online play). * The application can’t find the user account in the database. |

|  |  |
| --- | --- |
|  |  |
| Use Case Name | Multiplayer (Deathmatch) |
| Participating Actors | Initiated by the user. |
| Entry Conditions | The user selects “Multiplayer” from the main screen. Then selects Online play and Deathmatch. |
| Flow of Events | 1. The user initializes the game. 2. The application initializes the player module. 3. The user logs in. 4. The user chooses “Multiplayer”. 5. The application initializes all the controlling functions (these function are described in the **Product Functions**, section **2.2** of this document) 6. User logs on. 7. The server module creates a connection between users if it is for online play. 8. The application starts the game. 9. User attempts to destroy other ships 10. If time or frag limit is reach, the game is over 11. The system stores the user’s gameplay statistics. 12. Application returns to the main screen. |
| Exit Conditions | * User closes the main window. * All users lose all their lives. |
| Quality Requirements | * User must be able to operate a keyboard * Application must be able to respond to the user input. * Application must be able to create a connection between two players |
| Exceptions | * The user tries to log in with an invalid user account/password * The application can’t find the user account in the database. |

**2.3 User Characteristics**

The player is expected to have basic cognitive ability, but does not need any prior computer experience or experience with the system. The player can be of any age or background and be able to utilize the system. The system should also be compatible for players with a large amount of technical experience with the system and computers in general.

**2.4 Constraints**

The system shall have appeal for users of all ages. The game shall be easily used by anyone without prior knowledge of the system, even if the user has little knowledge of computers in general.

The main system shall be created in Java, using any class in Java JDK, with possible integration into a web page. This shall allow for internet play between users. Java Swing shall be used to develop a GUI that shall be user-friendly.

For the project schedule, the system must be fully functional by the week of April 10th, 2013. This SRS must be completed by February 10th, 2013, followed by the SAD due March 1st, 2013. A prototype of the system should be prepared for the week of March 27th, 2013.These time constraints determine the timeframe of the project.

The system should be able to handle more than one player, as well as running simultaneously via the internet.

The system shall also allow for the creation of a username and password, and the system must keep this information private to allow for a valid high-score records and effective player interaction.

**2.5 Assumptions and Dependencies**

It is assumed that the user’s computer will run Java applets. It is also assumed that the user will have a keyboard that supports the movement functions. For the web page integration, it is assumed that the user’s computer’s web browser will support the system.

**2.6 Apportioning of requirements**

Online play may be implemented in future versions of the system. It shall have cooperative play between two players and would allow users to duke it out in a deathmatch between up to 7 players connected through the internet. The multiplayer option of the game may allow users to battle against other ships controlled by computers. New types of gameplay ideas may be considered and implemented in the future.

**3.1 Functional Requirements**

|  |  |
| --- | --- |
| Requirement | Required? |
| The user shall be able to choose different methods of gameplay such as single player, cooperative multiplayer, competitive multiplayer, or online play. | desirable |
| The user shall be able to change the controls of the game.  This could consist of only changing keyboard keys or enabling mouse aim | desirable |
| The user shall be able to view/reset high scores | required |
| The user must be able to control the ship’s navigation and firing systems. | required |
| Asteroids shall be of different sizes and asteroids that are greater than the minimum size shall break apart when shot.  Asteroids of the smallest size shall disintegrate when shot | required |
| Ships must be destroyed or take damage when they intercept an asteroid or an enemy. | required |
| There must be a user-friendly GUI for the user to interact with. | required |
| The user shall be able to choose a username. | required |
| The user shall be able to choose a password. | required |
| The user shall be able to change their username/password | optional |
| The game shall have sound/music | required |
| The user shall be able to modify the difficulty of the game. | required |
| The user shall be able to use different weapons | required |
| The user shall be able to upgrade their ship, shields, and weapons | optional |
| The user shall be able to teleport within the game | required |
| The user shall be able to access and control settings for the game including potentially sound/music volume, game controls, resolution, and frame rate. | desirable |
| The user shall be able to access instructions or help | required |
| The user shall be able to pause, restart, and quit the game | required |
| The user shall be able to save their game. | required |
| The user shall be able to load their game from a list of previously saved games | required |
| The user shall be able to enable or disable the ability to hurt cooperative teammates with bullets.  This would be available only for cooperative multiplayer. | desirable |
| During competitive multiplayer, users shall be able to shoot each other. | desirable |

* 1. **Quality requirements**
     1. **Availability:**
* Hours of operation: Asteroid s++ must be available at any time.
* Location of operation: The game must be capable of operating on any conventional machine capable of running Java. Also, if the user desires to play the multiplayer mode of the game, the computer must be able of creating a LAN connection with the server module (The function of this module was described in section 2.1 of the SRS).
  + 1. **Reliability:**

This application must be able to run for at least 1 hour without crashing. This period of time is necessary to allow the users to enjoy the game. If the application crashes, the user must be able to re-launch it by simple opening the application again.

* + 1. **Compatibility:**
* Java must be installed in order for the user to play.
* The game may run on Windows, Linux or Mac.
* The computer must have a basic graphic card to run the game.
* The computer must be equipped with a keyboard.
* The computer must have at least 100Mb of free space, a 1 GHz processor, and 512Mb of RAM.
  + 1. **Maintainability:**

This game shall not require any maintainability after its release. There won’t be any future development for this game.

* + 1. **Usability:**

The game shall feature an easy to use graphical user interface where the majority of users can comfortably navigate.  
 The system only requires the user to have very basic computer knowledge.  
 The system shall include a section where users can learn all the gameplay mechanics in under a minute.

* + 1. **Performance:**

The ship shall respond to the controls after it is immediately pressed. This shall be at least under half a second.

* + 1. **Stability:**

The game shall be able to run with 2 users playing at the same time without any issues.

* + 1. **Open source:**

The system may be continued after the semester has ended as an open source project.

**3.3 Design Constraints**

The game shall be developed in a language where the majority of the developers are comfortable with. Another programming language may be used provided that all the members of the developing team agree. The game must be written from scratch. The game interface must be user-friendly such that users with limited computer experience and knowledge may easily be able to enjoy the game. Aspects of the game must be consistent.

**3.4 Other Requirements**

There are no other requirements.