Software requirments specification

Team Name: 6 Guys, 1 Asteroid

Jeffrey Leung

[ID 260402139]

Andrew Dennis

[ID 260475864]

Corey Perlman

[ID 260475188]

Neil Sabharwal

[ID 260474982]

HoaiPhuoc Truong

[ID 260526454]

Richard Pringle

[ID 260302515]

Contents

1. [Introduction 2](#_Toc348294987)

[1.1 Purpose 2](#_Toc348294988)

[1.2 Scope 2](#_Toc348294989)

[1.3 Definitions 2](#_Toc348294990)

[1.4 References 3](#_Toc348294991)

[1.5 Overview 3](#_Toc348294992)

2. [Overall description 3](#_Toc348294993)

[2.1 Production perspective 3](#_Toc348294994)

[2.2 Product functions and use cases 3](#_Toc348294995)

[2.2.1) Functions: 3](#_Toc348294996)

[2.2.2) Use cases 5](#_Toc348294997)

[2.3 User characteristics: 8](#_Toc348294998)

[2.4 Constraints 8](#_Toc348294999)

[2.5 Assumptions and dependencies 9](#_Toc348295000)

3. [Specific Requirements 10](#_Toc348295001)

[3.1 Functional Requirement 10](#_Toc348295002)

[3.1.1 Interface 10](#_Toc348295003)

[3.1.2 Game Environment 10](#_Toc348295004)

[3.1.3 Multiplayer 11](#_Toc348295005)

[3.1.4 Levels of difficulty 11](#_Toc348295006)

[3.1.5 In-Game Health 11](#_Toc348295007)

[3.1.6 In-Game Asteroids 12](#_Toc348295008)

[3.1.7 In-Game Bonuses 12](#_Toc348295009)

[3.1.8 In-Game Weapons 13](#_Toc348295010)

[3.1.9 Background Calculations 13](#_Toc348295011)

[3.2 Quality requirements 13](#_Toc348295012)

[3.2.1 Intuitive user interface 13](#_Toc348295013)

[3.2.3 Challenging 13](#_Toc348295014)

[3.2.4 Unpredictability 13](#_Toc348295015)

[3.2.5 Audio Capabilities 14](#_Toc348295016)

[3.2.6 Network connectivity 14](#_Toc348295017)

[3.3 Design Constraints 14](#_Toc348295018)

[3.3.1 No code from outside sources 14](#_Toc348295019)

[3.3.3 Easy to use 14](#_Toc348295020)

[3.3.7 Similar to Traditional Asteroids 14](#_Toc348295021)

# Introduction

## 1.1 Purpose

This SRS document will focus on the functions of the game in terms of high level features, low level controls. It will analyze the priority of each function and outline its complete implementation. The document will also explain how the primary focus of the project (maintenance, end user, client) has an effect on the various functional and non-function requirements of the game. Therefore, the primary audience of this document will be those who want to understand the ideas that went into the creation of the program and how each individual feature was developed. Furthermore, the document is important for those who would want to continue to develop the program in the future.

## 1.2 Scope

We intend to revolutionize the classic arcade game Asteroids, which was released in 1979 [3]. In addition to implementing basic elements such as shooting, movement, the classic sound track, etc. We will also be adding a multi-player mode, save points, assistance AI (more than one ship), etc (for more information on additional features and priorities please refer to Section 3).

We aim to build the game from scratch that will provide game physics, a friendly user interface and that can be played smoothly over the internet. We will be developing the game using the Java programming language with the help of Java Swing components. The final result (the game) will be presented in a GUI interface.

## 1.3 Definitions

**Asteroids**: This is video arcade game that was created by Atari Inc. in 1979 and is one of the most popular and influential games in the history of its genre. [3]

**AlienShips**: fixed shapes controlled by the AI system which can move, shoot at players’ spaceship, and explode when collide with asteroids (see below).

**Projectiles**: polygons with fix shapes produced by the players’ spaceships when they shoot.

**Weapon**: attributes associated with player’s spaceship which determines projectiles strength, spread and range.

**Highscore**: statistics collected during the game showing the highest achievement of players (calculated through deaths/ enemy destroyed/ asteroids destroyed)

**Lives**: number indicating how many more turn the player can play before the game is over.

**Bonuses**: different types of rewards appearing randomly in game to support players’ with health, lives and weapons.

**Asteroids (in-game object**: objects with random sizes, moving at random speeds, which is broken into smaller pieces (or vanish if is at smallest size) when collide with alien ships or spaceship.

**Multiplayer**: a game mode which allows two players to play alternatively.

**GamePhysics**: specific law of physics applied in collision detection and explosion of asteroids, spaceships and alien space ships in the game.

**Animations**: visualizations of game actions and activities (explosion, collision…)

## 1.4 References

[1] <http://www.math.uaa.alaska.edu/~afkjm/cs401/IEEE830.pdf>

[2] <http://standards.ieee.org/findstds/standard/830-1998.html>

[3] <http://en.wikipedia.org/wiki/Asteroids_%28video_game%29>

## 1.5 Overview

This document contains three main sections: The introduction, overall description and specific requirements. Within the overall description, the document discusses our expectation of the final product, major functions of the program and additional function descriptions (high level). Through use cases, the actions of the program are outlined in detail along with their specific functions. Moreover, project constraints and general assumptions will be discussed in this section.

The last part of the document, specific requirements, will explain the functional/quality requirements for the project as well as the design constraints that are present throughout the project.

# Overall description

## 2.1 Production perspective

This product is an extension of the classic asteroid game, which performs all the essential components of the traditional game along with several extra features: a multiplayer mode, a bonus system and game statistics. The product is independent and totally self-contained, and can function normally on its own.

## 2.2 Product functions and use cases

### 2.2.1) Functions:

2.2.1.1) Random appearance and movement for asteroids (with varied sizes)

2.2.1.2) Simple AI design for alien (movement and shooting)

2.2.1.3) Players’ spaceships can be controlled using and keyboard

2.2.1.3.1) Rotation using designated keys on keyboard

2.2.1.3.2) Acceleration and deceleration using designated keys on keyboard

2.2.1.3.3) Spaceship moves to the other end of the screen if reached one end

2.2.1.4) Limited amount of lives for spaceship (3 lives)

2.2.1.5) Collision detection and explosion

2.2.1.5.1) Spaceship/asteroid/alien will explode if collide with another object

2.2.1.5.2) Asteroid will explode into smaller pieces unless reached the minimum size

2.2.1.5.3) Projectile will disappear after travelling certain distance/ encounter collision

2.2.1.6) Multiplayer (through networking/ on the same PC)

2.2.1.7) Save and load game

2.2.1.8) Game statistics

2.2.1.8.1) Average game time

2.2.1.8.2) Highest score

2.2.1.8.3) Accuracy

### 2.2.2) Use cases

The program expects following use cases under normal operation:

|  |  |
| --- | --- |
| Use case name | Spaceship movement |
| Participating actor | User, spaceship |
| Entry conditions | Movement keys (rotate/ accelerate/decelerate) pressed |
| Flow of events | 1. If rotate key pressed    1. Increase/decrease moving angle by certain degree (10o) 2. If accelerate key pressed    1. Increase/ decrease velocity in the moving direction 3. Update spaceship position on game screen according to the changes    1. If spaceship reaches one end of the screen, appears on the other end 4. Detect collision    1. Trigger collision if detected |
| Exit conditions | Finished collision detection process |
| Quality requirements | The change should be visible (if applicable) on game screen no later than 0.5 second after the key was pressed |

|  |  |
| --- | --- |
| Use case name | Shooting |
| Participating actor | User, spaceship |
| Entry conditions | “Shoot” key was pressed |
| Flow of events | 1. Introduce new projectile at the nose of the spaceship 2. Initialize the moving direction and velocity of the projectile according to the moving direction of the spaceship |
| Exit conditions | Position and movement initializations completed |
| Quality requirements | The change should be visible (if applicable) on game screen no later than 0.5 second after entering the case |

|  |  |
| --- | --- |
| Use case name | Collision |
| Participating actor | Spaceship /Asteroid/ Alien/ Projectile |
| Entry conditions | Two shapes representing the two objects intersect/ overlaps |
| Flow of events | 1. Trigger explosion/ vanish for the object 2. Display changes (if applicable) |
| Exit conditions | Display updated |
| Quality requirements | The change should be visible (if applicable) on game screen no later than 0.5 second after entering the case |

|  |  |
| --- | --- |
| Use case name | Explode/ Vanish |
| Participating actor | Spaceship |
| Entry conditions | Triggered by collision/ Collision detected |
| Flow of events | 1. Set velocity and moving angle to zero 2. Become invisible on screen 3. Show explosion pieces on screen 4. Delete the object 5. Place a new spaceship at the center of the screen if there is life left, else show “Game Over” message |
| Exit conditions | Displayed all the changes |
| Quality requirements | The change should be visible (if applicable) on game screen no later than 0.5 second after entering the case |

|  |  |
| --- | --- |
| Use case name | Select weapon |
| Participating actor | User, spaceship |
| Entry conditions | User presses the “select weapon” button/key |
| Flow of events | 1. If there is only 1 weapon in the weapon list, do nothing 2. Change the equipped weapon to the next weapon on the list |
| Exit conditions | Equipped weapon successfully updated |
| Quality requirements | The process should not take more than 0.25 second to complete |

|  |  |
| --- | --- |
| Use case name | Close game |
| Participating actor | User |
| Entry conditions | User presses exit button or close the window using menu bar or hotkey |
| Flow of events | 1. Pause the game 2. Prompt message asking to save game 3. If user presses “Yes”, enter “save” use case, and quit the program 4. If user presses “No”, quit the program without saving 5. If user presses “Cancel”, return to main game screen and resume game. |
| Exit conditions | Either quit the game successfully, or resumed game |
| Quality requirements | The save files procedure should not take more than 1 second. Detect window closing even when program is terminated. |

|  |  |
| --- | --- |
| Use case name | Save and load |
| Participating actor | User |
| Entry conditions | User presses save/load button |
| Flow of events | 1. Pause the game (if there is a game playing) 2. If save button was pressed    1. Prompt save screen asking for save location (default is the save file of the game)    2. If user presses save button then save the current state 3. If load button was pressed    1. If there is no previously saved game, show message “No previously saved game”.    2. Prompt load screen asking for loading previously saved games.    3. If user presses load button then load the chosen game. 4. Return to main game screen and resume game.  * Exception   + Previously save file corrupted or invalid format     - Show Error: “Cannot Load Game” message     - When user presses OK, close the message, return to main game and resume game.   + Cannot write to file to save the game     - Show Error: “Cannot Save Game” message     - When user presses OK, close the message, return to main game and resume game. |
| Exit conditions | Game successfully loaded or saved, and returned to main game screen. |
| Quality requirements | The save files and load files procedure should not take more than 1 second. |

|  |  |
| --- | --- |
| Use case name | Show statistics |
| Participating actor | User |
| Entry conditions | User presses statistics button |
| Flow of events | 1. Pause the game 2. Load statistics from files 3. Show table containing game statistics 4. When the user presses OK button, close the statistics table 5. Return to main game screen and resume game  * Exception   + Statistics file corrupted.     - Show Error: “Cannot Load Statistics” message     - When user presses OK, close the message, return to main game and resume game. |
| Exit conditions | Statistics table closed and returned to main game screen. |
| Quality requirements | The load files procedure should not take more than 1 second. |

2.3 User characteristics:

|  |  |
| --- | --- |
| Characteristics | Description |
| Demographic |  |
| Gender | Both male and female |
| Age | 6 years-old kids to 70 years-old adults |
| Physical characteristics | Irrespective of the user handedness (left-handed, right-handed or ambidextrous), the user should be able to use the program easily. |
| Disabilities | Normal people without any perceptual or physical disability. |
| Experience characteristics |  |
| Prior experience | Little or no prior knowledge about the game |
| Computer literacy | Basic computer knowledge (use mouse and keyboard). |
| Other |  |
| Linguistic ability | Basic English at conversational level. |

## 

## 2.4 Constraints

2.3.1 Hardware Limitations:

Since the game is a relatively light program in comparison to others, it will only require a normal functional computer that can run java.

2.3.3 Interface to other applications:

There is not going to be any interaction with other programs within this game.

2.3.3 Parallel operations

There will be parallel operation within this game since positions of asteroids, players and other elements will be calculated and moving at the same time. Hence if the computational power of the PC is weak, it might lead to a slower or bad gaming experience.

2.3.4 Control functions

The game will only support keyboard control. Mouse or External controller will not work unless mapped to keyboard keys.

## 2.4 Assumptions and dependencies

The game base will be designed on the following assumptions

* The computer running the game has java installed already
* User/client has a functional keyboard
* User/client has a function speakers
* User/client has a standard computer which can do low level parallel computing and graphic processing power

All the above is based on the core function and control of the program.

# Specific Requirements

## 3.1 Functional Requirement

### 3.1.1 Interface

**3.1.1.1 User Interface**  
The user interface of the asteroid game will allow the player to play the game, view their 5 highest scores and view their statistics (number of asteroids destroyed and number of alien ships destroyed).   
Priority: Essential  
Implementation Difficulty: Difficult

**3.1.1.2 High Score List**  
There must be a list of previously attained high scores that have been reached by the player. These high scores will be logged along with the player name and statistics. There will only be 5 attainable high score fields.  
Priority: Desirable  
Implementation Difficulty: Medium

**3.1.1.3 Game Control**   
The ship will be controlled through the use of the arrow keys along with the spacebar button. The arrow keys will control the direction that the ship will travel in, and the spacebar will control the firing of the spaceship.  
Priority: Essential  
Implementation Difficulty: Medium

**3.1.1.4 Loading and saving current game state**  
The player can choose to save the current game’s state to a file which includes the asteroids and ship positions, the score, lives and all other game elements. The player can load a saved file to resume a previous game state.  
Priority: Optional  
Implementation Difficulty: Difficult

### 3.1.2 Game Environment

**3.1.2.1 Animations**  
There must be animations for the user-controlled ship, such as firing animations, impact animations and movement animations.  
Priority: Desirable  
Implementation Difficulty: Medium  
  
**3.1.2.2 Sounds**  
A sound system for various game features will be implemented. There will be sound effects for the ship/alien shooting, ship collisions, ship movement, asteroid explosions, and high scores.  
Priority: Optional  
Implementation Difficulty: Medium

**3.1.2.3 Impact Detection**  
When the player collides with an asteroid or alien ship, the game should detect the impact at decrement the lives (see 3.1.5.2 for more detailed specifications) of the user.  
Priority: Essential  
Implementation Difficulty: Difficult

**3.1.2.4 Game Physics:**  
The ship will continue on its current course until the movement key is released. Once the key is released, the ship will decelerate at a constant rate until it either stops moving or another arrow key is pressed.   
  
Asteroids will break into smaller pieces (see 3.1.6.2 for more detailed specifications) and they will keep travelling after they have been shot by the space ship. They will travel in the direction of the fired shot with a speed dependant on the size of the asteroid. Smaller asteroids will travel faster after a hit.  
  
Priority: Desirable   
Implementation Difficulty: Difficult

### 3.1.3 Multiplayer

**3.1.3.1 Multiplayer mode with alternating turns**  
One player will play through the first level of difficulty in the asteroid game, and then the second player will do the same. This will continue until one player fails to reach the level of difficulty of the other player.  
  
Priority: Essential  
Implementation Difficulty: Medium

**3.1.3.2 2-player Team Up With Simultaneous Play**   
2 players can connect to the same game over a network and play the game as a team. There will be two independent ships displayed on the screen.  
Priority: Optional  
Implementation Difficulty: Difficult

### 3.1.4 Levels of difficulty

**3.1.4.1 Levels of difficulty**  
The level of difficulty of the game (5 levels) will increase after a certain amount of asteroids (30 – 40) are completely destroyed. After each level, the amount of asteroids that need to be broken will increase, along with their speed and the addition of alien ships.  
  
Priority: Essential  
Implementation Difficulty: Medium  
  
**3.1.4.2 Levels of difficulty for alien ships**  
5 different levels of alien ships are available to appear with increasing level of difficulty of the game. These levels of aliens will have an increasingly difficult level of artificial intelligence and increased levels of weaponry.  
Priority: Optional  
Implementation Difficulty: Difficult

### 3.1.5 In-Game Health

**3.1.5.1 Lives**  
The player is given a certain amount of lives to finish the game. The player can collect additional lives through drops or score bonuses. If the player reaches a lives value of 0, the game is lost and must be restarted from the starting level.  
Priority: Essential  
Implementation Difficulty: Easy  
  
**3.1.5.2 Ship Health**  
The ship has a health level that ranges from 0 to 100. The ship can take damage caused by asteroids (5 – 10 damage depends on the difficulty level) or aliens (10 – 30 damage depends on the difficulty level) till its health reaches 0. When it has no more health, a live is lost.  
Priority: Optional  
Implementation Difficulty: Medium

### 3.1.6 In-Game Asteroids

**3.1.6.1 Multi-Sized Asteroids**  
The system can generate multiple sizes of asteroids based on the current level of difficulty. These sized asteroids should be presented to the user in the game’s interface.   
Priority: Essential  
Implementation Difficulty: Medium  
  
**3.1.6.2 Breakable Asteroids**  
Larger asteroids break up into smaller asteroids (maximum of 5 times) when hit until a sufficiently small asteroid (asteroid which is split 5 times) can be destroyed.   
Priority: Essential  
Implementation Difficulty: Difficult  
  
**3.1.6.3 Alien Ships**  
Other ships appear in the game’s interface space area. These ships are alien characters with artificial intelligence. They attack the player’s ship. The player must defend themselves by attacking the alien ships and eliminating them.  
Priority: Essential  
Implementation Difficulty: Difficult

### 3.1.7 In-Game Bonuses

**3.1.7.1 Bonus Drops**  
Items will appear in the game’s interface space area that the player can collect by colliding with them or attacking them. These items include extra lives, health, or new weapons and ships. Better bonuses are dropped as the difficulty of the levels increase.  
Priority: Optional  
Implementation Difficulty: Medium

**3.1.7.2 Score Bonuses**  
If a user achieves a high enough score, or collects all the required items during one round of the game, or achieves a high enough shooting accuracy, bonuses to the score can be obtained. These bonuses include extra points, lives, health, or new weapons and ships.  
Priority: Optional  
Implementation Difficulty: Easy

### 3.1.8 In-Game Weapons

**3.1.8.1 Weapons Upgrades**  
The player can attain weapons upgrades through item drops or bonuses. The player can choose which weapon to equip their ship with. Different weapons will have different firing rates and additional firing paths  
Priority: Optional  
Implementation Difficulty: Difficult  
  
**3.1.8.2 Fire Rate Control**  
The game controls the amount of fires that the user can do per second: determined by the weapon with which the ship is currently equipped.  
Priority: Essential  
Implementation Difficulty: Medium

### 3.1.9 Background Calculations

**3.1.9.1 Round and Total Game Points**  
The game keeps track of all the asteroids and aliens destroyed during a round of the game and assigns point values to this. More points are gained with a higher number of items hit.  
Priority: Desirable  
Implementation Difficulty: Easy  
  
**3.1.9.2 Shooting Accuracy Measurement**  
The game measured the accuracy of shots fired at asteroids and aliens and displays it at the end of the level. Promoting accuracy ensures the player will not constantly shoot into space.  
Priority: Optional  
Implementation Difficulty: Medium

## 3.2 Quality requirements

3.2.1 Intuitive user interface  
The interface must be aesthetically pleasing, and easy to use. This means we will have separate properly sized buttons for each game option with clear and concise descriptions of the button so as to avoid user confusion.  
  
3.2.2 Smooth running game

This means that there is a quick response from the user key input to the graphic output. To accomplish this we must have a proper refresh rate of the display. We also need a glitch free game where the animations are smooth and do not encounter errors. This means that animations are well done, happen quickly and are tested so that no errors happen such as the asteroid not blowing up after being shot or the ship going through an asteroid without losing a life.

3.2.3 Challenging

The game must be reasonable difficult with options available for different experience levels. This means that all players can enjoy the game and have a reason to continue playing it and improve.

3.2.4 Unpredictability

To be entertaining the game must also be a new experience for the user every time so that they do not get bored of it. Example: the location and spawn time of the asteroids must be random or inconsistent so that it is a new and enticing game experience every time.

### 3.2.5 Audio Capabilities

The game must be able to play appropriate sounds at the right time and at a proper volume so as to enhance the game experience.

### 3.2.6 Network connectivity

A proper internet connection protocol that does not drop and communicates scores and player names to record a high score board without loss of information (Optional).

## 3.3 Design Constraints

3.3.1 No code from outside sources  
Unless given explicit permission from the course instructor, no code may be used that was procured from sources outside of group members. Any classes of the Java SDK may be used. The nominal interface must be developed using Java Swing components.  
  
3.3.2 Design deadline  
The code must be written over the course of one semester and the final code is due on April 16th 2013. A total of 6 team members are available to work on the project.

3.3.3 Easy to use  
The game must be operable by a user of almost any background, regardless of their computer experience or age.  
  
3.3.4 Single programming language  
Only one programming language should be used to code this game. It has been decided that the Java programming language will be used.  
  
3.3.5 Limited access to design applications  
We will be constrained by the visual effects possible using JAVA SWING and JAVA AWT.  
  
3.3.6 X-Y Plane  
We do not currently possess the skills to create a three dimensional game because the majority of our team has no experience in the implementation of 3D models

### 3.3.7 Similar to Traditional Asteroids

Calculating high scores, recording of scores and ability to see statistics, multi-sized asteroids, aliens, bonus drops, weapon drops, bonus life drops, spaceship animations and environment wraparounds, keyboard control of the spaceship with gun, controlled firing rate and impact detection, proper game physics for asteroid and spaceship, sounds, a simple AI for the aliens, possibility of different difficulty levels, and 2 player mode (alternating turns).