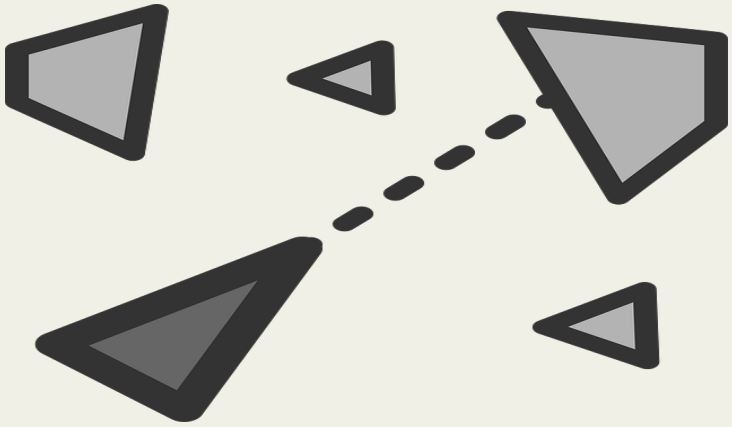
Asteroids

Software Requirements Specification



Team 4 - ISEtroids

Damien Girard
Siddhart Ghandi
Thuy-Anh Le
Genevieve Nantel
Vincent Petrella
Chi-Wing Sit

2/10/2013

TABLE OF CONTENTS

1. INTRODUCTION

1.1 PURPOSE AND SCOPE

Asteroids is a two-dimensional view video arcade game in which the player controls a triangular-shaped ship in an asteroid field, periodically, a flying saucer will appear on the screen and fire at the player. The goal of the game is to destroy as many asteroids and flying saucers as possible while avoiding collision with either. As the player shoots an asteroid, the asteroid breaks into smaller asteroids which are smaller and harder to hit.

1.2 DEFINITIONS

Ship: The ship is represented by a triangle and can rotate left and right, move forward and fire shots
Application Programming Interface (API): A protocol intended to be used as an interface by software components to communicate with each other
Asteroid: In the shape of a simple polygon, asteroids break into smaller asteroids when hit
Flying saucers:

- Small flying saucer aims at the player's direction
- Large flying saucer shoots in random direction

Frame per Second (fps, frame rate): The number of frames executed and rendered by the program in one second. The higher the fps is, the smoother the game will be.
Hyperspace: Make the player reappear in a random location on the screen
Java Runtime Environment (JRE): Provides the libraries, the Java Virtual Machine, and other components to run applications in Java
Level: Determines the difficulty of the game by adjusting the moving speed of the enemies and the number of asteroids
Lives: The number of collisions with asteroids or flying saucers allowed
Open Graphics Library (OpenGL): Cross-language and multi-platform API to render 2D and 3D computer graphics

1.3 REFERENCES

http://web.archive.org/web/20080801211047/http://www.gamearchive.com/General/Articles/ClassicNews/1981/Esquire2-81-pg62.htm

1.4 OVERVIEW

The Overall Description section describes the general factors that affect the game and the requirements. With the help of use case diagrams and the descriptions of the product, the Overall Description provides a background to the reader for the requirements. This section does not state any specific requirements. The third section contains all the software requirements with a level of detail sufficient to enable the developers to design the game by satisfying those requirements.

2. OVERALL DESCRIPTION

2.1 PRODUCT PERSPECTIVE

2.1.1 USER INTERFACES

The game will be displayed to the user via a resizable window.

The user will navigate the menus using the arrow keys, and select a choice with the return key, all the controls are located on the keyboard.
The layouts of the menus are a simple vertical enumeration of choices.

During a game session, the user will only input the control of its space ship, the pause button, or exit buttons, which will all be located on the keyboard.
The layout of the game session will be composed of the actual game action itself (including the playable character, the enemies, and a background), and a GUI indicating the state of the game session (Score, number of lives left).

Each of these components will be updated every frame.

2.1.2 HARDWARE INTERFACES

The program will make use of the OpenGL API up to its 2.0 revision.
And will thus require the hardware to be equipped with an OpenGL 2.0 compatible GPU.
It will also require a sound processor in order play the sounds.
A keyboard will also be required to play the game.
Most of these devices are integrated by default in modern computers.

2.1.3 SOFTWARE INTERFACES

In order to execute the program, the following pieces of software are required:
Drivers for both the OpenGL 2.0 compatible video and sound devices as well as for the keyboard.
The JRE must be installed on the operating system.

2.1.5 MEMORY CONSTRAINTS

There is no real memory constraint for our project, as it is intended to be executed on modern desktop computers. However, a preferable size of around less than 100 Mbytes of memory used during runtime will be aimed.

2.2 PRODUCT FUNCTIONS

2.2.1 USE CASES

a. Main Menu (Initiated by non-logged-in user)

Flow of Events:

1. Three options are displayed to the user, with the behaviour specified as follows:

a. New Profile: Jump to use case *New Profile*
b. Load Profile: Jump to use case *Load Profile*
c. 2 Player Mode: Jump to use case *2 Player Mode*
d. Quit: application terminates

2. The behaviour of each option is specified as above

Entry Condition(s): Asteroids application launches
Exit Condition(s): The user quits application or selects one of the options

b. Create New Profile (Initiated by non-logged-in user)

Flow of Events:

1. User is prompted to enter username with eight characters
2. Two buttons are displayed:

a. Submit: If username does not already exist, new profile is successfully created and user is directed back to the main menu. If it does already exist, an error message is shown and the user is prompted again for a new username
b. Cancel: Returns the user to the main menu (see use case *Main Menu*

Entry Condition(s): User selects the "New Profile" option in the *Main Menu*
Exit Condition(s): User creates or cancels the creation of a new account

c. Load Profile (Initiated by non-logged-in user)

Flow of Events:

1. User is prompted to enter eight-character username
2. Two buttons are displayed:

a. Load: If username exists in the database, the profile is successfully loaded and player is directed to the game menu (see use case *Game Menu)*

b. Cancel: User is directed to the main menu (see use case *Main Menu)*

Entry Condition(s): User selects the "Load Profile" option in the *Main Menu*
Exit Condition(s): User creates or cancels the creation of a new account

d. Game Menu (Initiated by logged-in user)
Flow of Events:

1. Five options are displayed to the user, with the behavior specified as follows:

a. Play Asteroids: see *Play Asteroids* use case
b. Select Level: see *Select Level* use case
c. View Statistics: see *Statistics Page* use case
d. Instructions: displays a game instructions page
e. Main Menu: see *Main Menu* use case

Entry Condition(s): The user successfully loads his or her profile
Exit Condition(s): The user selects one of the home screen options

e. Statistics Page (Initiated by logged-in user)
Flow of Events:

1. The screen displays a list of the five highest scores associated with the current user's profile in descending order
2. Two additional options are displayed:

a. View Global High Scores: User is directed to a page containing a list of the ten global highest scores in descending order
b. Back to Game Menu: User is directed back to the Game Menu (see use case *Game Menu*)

Entry Condition(s): The user selects the "View Statistics" option in Game Menu
Exit Condition(s): The user is taken back to the Game Menu

f. Select Level (Initiated by logged-in user)
Flow of Events:

1. User is shown a pop up menu, giving the user:

a. Three levels to select from: The game begins upon selection of any of these (see use case *Play Asteroids)*
b. Back to Game Menu: The user is returned to the game menu (see use case *Game Menu*)

Entry Condition(s): The user clicks the "Select Level" option on the game menu
Exit Condition(s): The user begins game or is taken back to game menu

g. Play Asteroids (Initiated by logged-in user)
Flow of Events:

1. A GUI is loaded
2. A five second countdown is given while the word "Ready" flashes at the center of the screen
3. Play begins with a few asteroids drifting in random directions on the screen. Asteroids wrap around screen edges, meaning if they move off the top, they reappear at the bottom and keep moving in the same direction
4. Periodically, a flying saucer appears on one side of the screen and moves across to the other side before disappearing again. The saucers come in two types:

a. Large Saucers: these fire in random directions
b. Small Saucers: these aim at the player

5. The system waits for the user to press a control key or the pause button, and does nothing if the user presses any other key
6. The user presses one of five control keys:

. Move Forward: ship moves in the direction it is facing
a. Turn Left: ship angle turns counter clockwise
b. Turn Right: ship angle turns clockwise
c. Fire: ship fires shots
d. Hyperspace: ship instantly disappears and reappears at a random point on the screen at the risk of appearing right next to an asteroid

7. There are several scenarios possible:

    a. If user fires and shoots asteroid, see use case *Shoot Asteroids*
    b. If the user collides with an asteroid, see use case *Collide with an Asteroid*
    c. If the user fires and shoots a flying saucer, see *Shoot Flying Saucer* use case
    d. If the user is hit by an enemy saucer shot, see *Get Hit by Enemy Fire* use case

e. If the user collides with a flying saucer, see use case *Collide with a Flying Saucer*
f. If the user picks-up an item, see use case *Item Pick-Up*

8. Once the screen has been cleared of all asteroids and flying saucers, a new set of large asteroids appears. The number of asteroids increases each round up to a maximum of 12.
9. If the pause button is pressed, refer to the *Pause Game* use case
10. The game responds to user input until the game ends or is paused

Entry Condition(s): The user selects level and starts or simply clicks "Play Asteroids" on the game menu

Exit Condition(s): The player runs out of lives and the game ends or is paused

h. 2 Player Mode (Initiated by non logged-in user)

Flows of Events:

1. User is shown the Select Level pop-up, see *Select Level*
2. Player 1 starts his game, see *Play Asteroids*
3. Once Player 1 runs out of lives, the system will display a message asking if Player 2 is ready, the system will then wait for an answer
4. Player 2 starts his game
5. Once Player 2 runs out of lives, the system will display the following message "Player X Wins" where Player X is the player who achieved the highest score

Entry Condition(s): The user selects 2 Player Mode in the main menu

Exit Condition(s): Player 2 runs out of lives and the game ends or is paused

i. Shoot Asteroids (Initiated by user)

Flow of Events:

1. Asteroid breaks up into asteroids of smaller category which move faster
2. The player scores points on hitting an asteroid (see *Scorekeeping* use case)
3. Whenever an Asteroid is shot, there is chance that it drops an item

Entry Condition(s): User presses the fire button in-game and hits an asteroid

Exit Condition(s): None

j. Collide with an Asteroid (Initiated by user)
Flow of Events:

1. The ship moves to the position of an asteroid or an asteroid moves to the position of the ship
2. The round ends and the number of remaining lives is decremented by one. If the number of lives after decrement reaches zero, see *End Game* use case
3. Proceed to step 2 of use case *Play Asteroids*

Entry Condition(s): An asteroid and the ship end up in the same grid position
Exit Condition(s): None

k. Shoot Flying Saucer (Initiated by user)
Flow of Events:

1. The flying saucer is destroyed
2. Points are awarded (see *Scorekeeping* use case)
3. Whenever a Flying Saucer is shot, there is chance that it drops an item

Entry Condition(s): The user presses the “Fire” button and hits a saucer
Exit Condition(s): None

l. Get Hit by Enemy Fire (Initiated by user)
Flow of Events:

1. The round ends and the number of lives is decremented by one
2. Proceed to Step 2 of *Play Asteroids* use case

Entry Condition(s): Player’s ship makes contact with a shot fired by an enemy
Exit Condition(s): None

m. Collide with a Flying Saucer
Flow of Events:

1. The ship moves to the position of an asteroid or an asteroid moves to the position of the ship
2. The round ends and the number of remaining lives is decremented by one. If the number of lives after decrement reaches zero, see *End Game* use case
3. Proceed to step 2 of use case *Play Asteroids*

Entry Condition(s): A Flying Saucer and the ship end up in the same grid position
Exit Condition(s): None

n. Item Drop
Flow of Events:

1. If the item is a Bonus Life, the number of remaining lives is incremented by one
2. If the item is a Weapon Upgrade, the user will start shooting multiple shots at the same time in the shape of a V
3. If the item is a Bomb, the screen will be cleared and point are awarded

Entry Condition(s): An item and the ship end up in the same grid position
Exit Condition(s): None

o. Pause Game (Initiated by user)
Flow of Events:

1. All game controls and operations are suspended
2. Two options are displayed on screen, with the behavior as follows:

a. Resume Game: Game becomes active again and preserves current progress
b. Quit Game: See *Main Menu* use case

Entry Condition(s): Player presses the pause button while game is active
Exit Condition(s): Game resumes or ends

p. End Game (Initiated by user)
Flow of Events:

1. Flashing text “Game Over” is displayed on screen for three seconds
2. Score is saved if:

a. It falls in the range between the lowest and highest scores associated with the user’s profile
b. It falls in the range between the lowest and highest global maximum scores

3. User is returned to the game menu (see use case *Game Menu*)

Entry Condition(s): Number of lives becomes zero or the user selects “Quit Game” from the pause menu

Exit Condition(s): Game menu is displayed again

q. Scorekeeping (Automated by score controller)

Flow of Events:

1. Game score is initialized to zero (0) at the beginning of a new game
2. Points are awarded for hitting targets as follows:

a. For hitting a large sized asteroid, the player is awarded twenty (20) points
b. For hitting a medium sized asteroid, the player is awarded fifty (50) points
c. For hitting a small sized asteroid, the player is awarded one hundred (100) points
d. For hitting a large flying saucer, the player is awarded two hundred (200) points
e. For hitting a small flying saucer, the player is awarded one thousand (1000) points

Entry Condition(s): The *Play Asteroids* use case begins

Exit Condition(s): The *Play Asteroids* use case ends

2.2.1 USE CASE DIAGRAMS

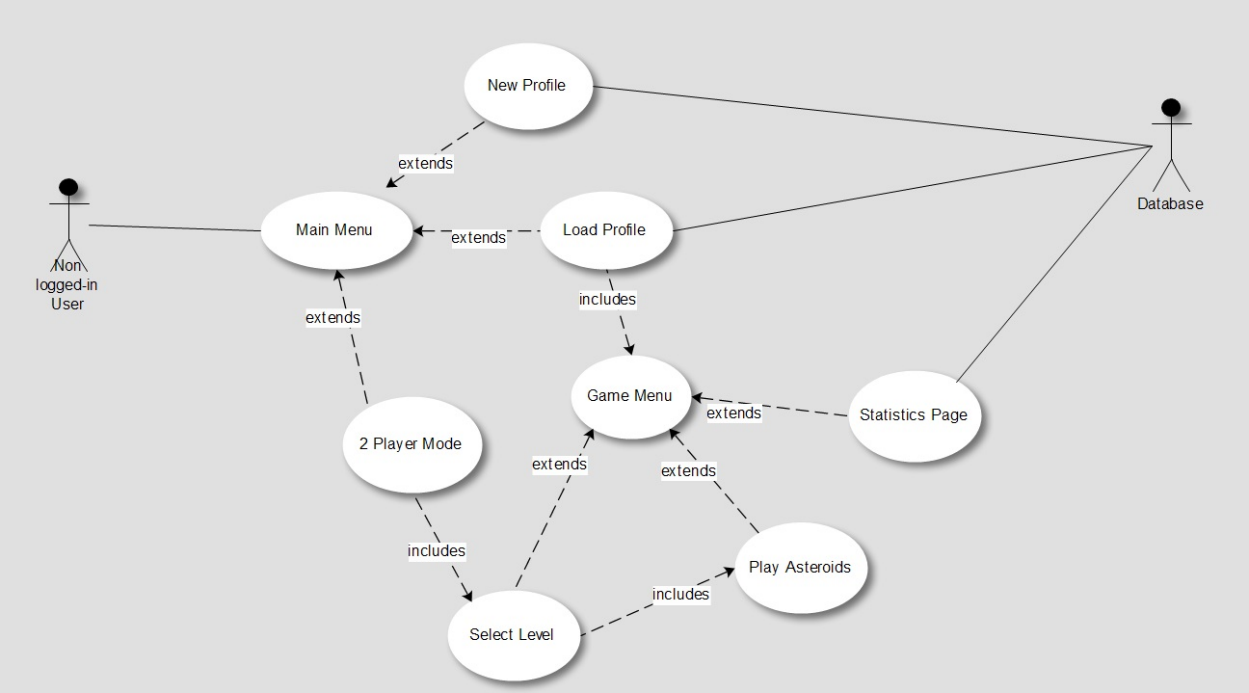


Figure 1 - Main Use Case Diagram

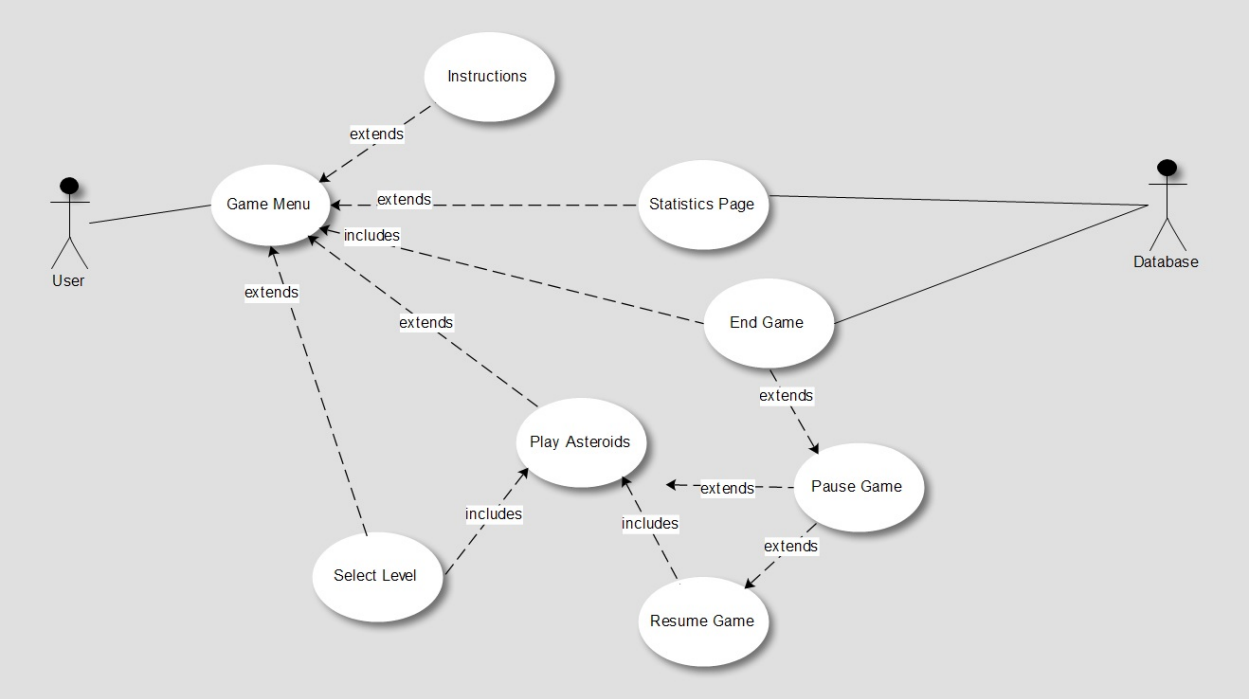


Figure 2 - Game Menu Case Diagram

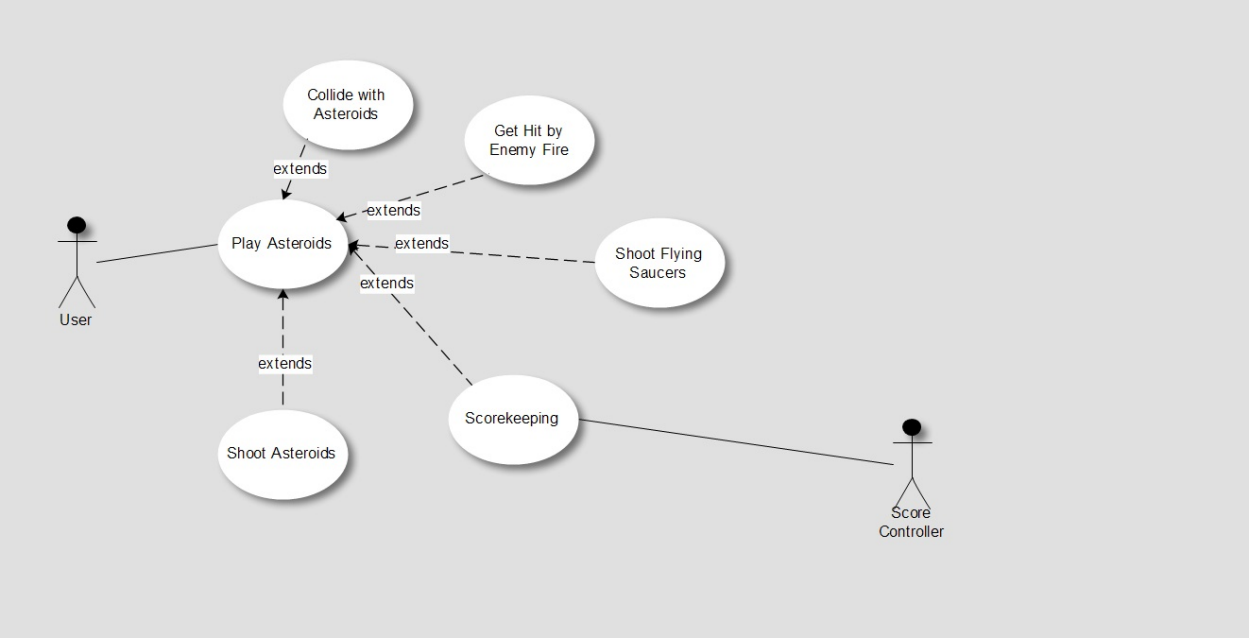


Figure 3 - Play Asteroids Case Diagram

2.3 USER CHARACTERISTICS

All users of the software are expected to have the following common characteristics:

- Ability to read and understand instructions presented in the English language
- Knowledge of the operation of basic operating system GUI components

2.4 CONSTRAINTS

The development methodology of the software is expected to be subject to the following constraints:

- **Hardware Constraints**: Making the software as platform-independent as possible involves the capability to run on a variety of distinct devices and operating systems, from Android devices to Personal Computers.
- **Software Constraints**: The system will be constructed using Java technologies for reasons of portability and flexibility of platform. The client's device can use any platform so long as it supports Java applets.

2.5 ASSUMPTIONS AND DEPENDENCIES

- **Artificial Intelligence:** The client did not specify which specific type of AI to use, stating ''simple AI implementation for the alien ships''. We will developed an AI based on the actual game, the flying saucers will move in a straight line and shoot, at a constant rate, at the player or in random directions depending on the type of the saucer.
- **Bonus life drops**: The client vaguely mentioned that he wanted the ''implementation of bonus life drops''. In most asteroid games, every time the player shoots an asteroid, there is a certain percentage chance that the asteroid will drop a bonus life or a weapon. We assumed that the client wants to follow the trend and go for this option. Another option is where the player gets a bonus life after each 5000 points accumulated.
- **Sound System**: The client requires the implementation of a sound system. *Asteroids* game features various sound effects. We decided to use the original sounds that were implemented by Howard Delman instead of creating our own sounds.

3. SPECIFIC REQUIREMENTS

3.1 FUNCTIONAL REQUIREMENTS

3.1.1 INITIALIZATION

3.1.1.1 GUI

3.1.1.1.1 [Essential/Easy] System available with a GUI, developed using java swing components.

3.1.1.2 DATABASE

3.1.1.2.1 [Essential/Medium] Creating and initializing a database to store information upon first execution

3.1.1.2.2 [Desirable/Medium] Loading and saving your current game

3.1.1.2.3 [Essential/Easy] Only one game per username can be save

3.1.1.2.4 [Essential/Medium] Opening and loading information from database when user wants to view statistics and high scores

3.1.1.2.5 [Essential/Medium] Saving information in database when user ends or saves game

3.1.2 USER INTERFACE

3.1.2.1 IDENTIFICATION WINDOW

3.1.2.1.1 [Essential/Easy] User must be identified by entering user name before he can see game panel

3.1.2.2 HELP WINDOW

3.1.2.2.1 [Essential/Easy] The help window shall provide a way of viewing documentation about the game.

3.1.2.3 MAIN WINDOW

3.1.2.3.1 [Essential/Easy] After user logged in successfully, the game panel will be replaced by the user information: You are logged in as: username.

3.1.2.3.2 [Essential/Easy] User game Interface allows you to manage the game state: seeing your statistics and high scores, new game, game rules, player mode and change settings.

3.1.2.3.3 [Essential/Easy] When user clicks on new game, the game option panel will disappear and be replaced by the game itself.

3.1.2.4 CHANGE SETTINGS WINDOW

3.1.2.4.1 [Essential/Easy] Two player mode, alternating turns

3.1.2.4.2 [Desirable/Medium] Two player team up with simultaneous play

3.1.2.4.3 [Essential/Easy] Different levels of difficulty

3.1.2.5 STATISTICS WINDOW

3.1.2.5.1 [Essential/Easy] User will see he high score list with statistics

3.1.2.6 GAME WINDOW

2.2.6.1 [Essential/Medium] The graphical implementation of the game will include multi-sized asteroids, aliens, bonus drops, weapon drops, spaceship animations and environment wraparounds.

3.1.3 FUNCTIONALITIES

3.1.3.1 ARTIFICIAL INTELLIGENCE

3.1.3.1.1 [Essential/Medium] A HOSTILE AI implementation for alien ships

3.1.3.2 SCORING

3.1.3.2.1 [Essential/Easy] Implementation of point system

3.1.3.2.1.1 - Large Asteroid: 50 points

- Small Asteroid: 100 points

- Large Saucer: 200 points

- Small Saucer: 1000 points

3.1.3.2.2 [Essential/Easy] Implementation of life system

3.1.3.2.2.1 [Essential/Easy] Set starting lives at 3

3.1.3.2.2.2 [Essential/Easy] Implementation of bonus life drops; every time the player shoots an asteroid or a flying saucer, there is a certain percentage change that it will be a bonus life or a weapon

3.1.3.3.2.3 [Essential/Easy] Collision with an asteroid results in loss of life

3.1.3.2.2.4 [Essential/Easy] Game is over when player has lost all his lives

3.1.3.3 SOUND SYSTEM

3.1.3.3.1 [Essential/Medium] Implementation of the sound system for the feature of various sound effects.

3.1.3.4 WEAPONS

3.1.3.4.1 [Essential/Medium] Implementation of shooting, firing rate control and impact detection.

3.1.3.4.2 [Desirable/Easy] Implementation of different types of weapons: regular, spread, screen clear

3.1.3.5 SPACESHIP

3.1.3.5.1 [Essential/Medium] Player controls a triangular-shaped ship that can rotate left and right, fire shots straight forward, and thrust forward.

3.1.3.4 GAME CONTROLS

3.1.3.4.1 Game controls to allow the control of the ship

3.1.3.4.1.1 [Essential]/ [Medium] Implementation of game physics for spaceship

3.1.3.4.1.2 [Essential]/ [Medium] Player will be able to change the direction of the navigated space shuttle.

3.1.3.4.1.3 [Essential]/ [Medium] Player will be able to change the speed of the navigated space shuttle.

3.1.3.4.1.4 [Essential]/ [Medium] Momentum is not conserved, the ship eventually comes to a stop again when not thrusting

3.1.3.4.2 Basic Game Control

3.1.3.4.2.1 [Essential]/ [Easy] Stop Game

3.1.3.4.2.2 [Essential]/ [Medium] Pause Game

3.1.3.5 ASTEROIDS AND SAUCERS

3.1.3.5.1 [Essential/Medium] Implementation of game physics for asteroids and saucers

3.1.3.5.1.1 [Essential/Medium] Each kind of asteroids/saucers should be able to change their speed. They have different minimum and maximum speed.

3.1.3.5.1.2 [Essential/Medium] Saucers will appear periodically on one side of the screen to the other before disappearing.

3.1.3.5.2 Saucers

3.1.3.5.2.1[Essential/Medium] Implementation of saucers: Large saucers fire in random directions, while small saucers aim at the player's ship

3.1.3.5.2.2[Desirable/Medium] Implementation of different types of flying saucers: regular, big boss.

3.1.3.5.3 Asteroids

3.1.3.5.3.1 [Desirable/Medium] Implementation of two different kinds of asteroids: One is small and fast and does not change its direction until it reaches the boundary of the game board. The other is big and slow, and it may change its direction any time.

3.1.4 DIFFICULTY

3.1.4.1[Essential/Medium] Every time the screen has been cleared of all asteroids and flying saucers, a new set of large asteroids appears. The number of asteroids increases each round up to a maximum of 12.

3.2 QUALITY REQUIREMENTS

3.2.1 PERFORMANCE

3.2.1.1 [Essential/Easy] Minimum delay between user input and game reaction, and more generally a minimum delay between two game frames. The desired frame rate is a constant 60 fps on current mid-range hardware, which is equivalent to a 16ms delay between two game refresh.

3.2.1.2 [Desirable/Easy] Query and reporting times (initial loads and subsequent loads) should load in less than 20 seconds on current mid-range hardware.

3.2.1.3 [Essential/Medium] Calculations to keep the score of the player up to date should be executed instantaneously (i.e., during the frame of the change)

3.2.1.4 [Essential/Easy] A high score table will appear in less than 5 seconds after “game over” (processing time) on current mid-range hardware.

3.2.2 USABILITY

3.2.2.1 [Essential/Easy] The language used for the game is English

3.2.2.2 [Desirable/Easy] Shortcuts on the keyboard should be implemented to facilitate the user’s game.

3.2.2.3 [Essential/Easy] Game interface designed in a way to appeal to users of all ages and you can learn how to play while playing the game, i.e. Asteroids must be simple to play: The user interface and the rules of playing Asteroids must be intuitive.

3.2.3 RELIABILITY AND AVAILABILITY

3.2.3.1 [Essential/Medium] In case the system crashes, all high scores will be maintained in the database.

3.2.3.2 [Essential/Difficult] All predictable exceptions should be handle by the program during execution, and should not produce a crash. If an exception occurs forcing the program to terminate, the current state of the session should be saved before termination.

3.2.4 SAFETY

3.2.4.1 [Essential/Easy] If the user clicks outside of the system's window, the game is paused.

3.2.5 MAINTAINABILITY

3.2.6.1 [Essential] The code should be fully documented and modular in order to permit future modifications.

3.2.6 EXTENSIBILITY

3.2.8.1 [Desirable/Medium] The game should be extensible and allow easy changes or improvements in the future.

3.3 DESIGN CONSTRAINTS

3.3.1 CONSTRAINTS ON THE APPLICATION OR ITS DEVELOPMENT

3.3.1.1[Desirable/Easy] Less than 30 Mb on the hard drive

3.3.1.2 [Desirable/Easy] Less than 100mb of ram during runtime

3.3.1.3 [Essential/Medium] Game lag must be avoided at all cost as it takes away from user experience and breaks the game experience.

3.3.1.4 [Essential] The system must be realizable by 7 people-each working under 9 hours a week.

3.3.2 GAME AESTHETIC

3.3.2.1 Game graphics, including sprites, animations, backgrounds, etc. must be visually pleasing, professional and conform to the original game's concept.

3.3.2.2 The final system should not include any graphical glitches.

3.3.3 USER INTERFACES

3.3.3.1 [Essential] The interface of the system should be user-friendly for all ages. Users of all level of computer experience should be able to easily engage in the game.

3.3.3.2 [Desirable] The elements involving game states (save/pause, etc.) not directly involving game play should be intuitive as well as visually appealing to the user while retaining a professional look.

3.3.3.3 [Essential] The interface must accurately inform the user in case of input errors.

3.3.4 ERROR HANDLING

3.3.4.1 [Essential/Hard] All exception raised during runtime must be handled so that the system meets a minimal standard of stability. The game should never crash but a realistic best scenario should make a game crash a rare occurrence.

3.3.4.2 [Essential] In extreme or high stress conditions, the system must retain stability. The conditions should be determined in testing

3.3.4.3 [Essential] Errors raised by the program must be minimized.

3.4 PROCESS CONSTRAINTS

3.4.1 RESOURCES

3.4.1.1 [Essential] Time must be managed effectively. The final deadline for the submission of deliverables is April 16th. Time must be allocated for the completion of the system, the finalization of all the documentation involved in its development as well as the preparation for the final presentation.

3.4.1.2 [Desirable] The schedule of all members must be accommodated for regular group meetings. Such meetings must be put in place to periodically review the state of the project with all group members present.

3.4.1.3 [Essential] The allocation of work must be executed fairly. No group member should be overloaded with work in order to pick up the slack of other members.

3.4.2 DOCUMENTATION

3.4.2.1 [Essential] Documents must all be completed before their respective due dates.

3.4.2.2 [Essential] Documentation must maintain a consistent format and meet client requirements. the tone must remain professional at all times.

3.4.2.3 [Essential] All documentation must be made available to team members and client.

3.4.2.4 [Essential] Documents must be unambiguous, concise and clear in order to be easily understood by both parties.