Group 1 **2017-03-28**

Brooklyn College Spring 2017

Software Design & Implementation 2

Group 1 Class Project

“College Invaders” Specification

n*th *Edition

Group Roster

Specifications	Q.A.	Graphics	Backbone
Ira Bletz-Fuller	Christian Butron	Aniss Fadel	Maxavier Jeanphilippe
Ricles Vigni	Zainab Dandia	Marc Pfeiffer	Noam Swisa
Nishat Anjum	Edward Heifetz	Jeremy Elmani	Wen Huang
Andrew Castillo	Oscar Su	Maogeng Lin	Janhua Lui
Al-John Sakasamo		Randy Cisneros	Shayan Jafri
Nathan Antebi		Denis Barabanov	Maggie Cao

CISC 3140 Spring 2017 College Invaders Project SRS

Group 1 **2017-03-28**

Table of Contents

Group 1 **2017-03-28**

1. Introduction

1.1 Purpose

- The purpose of this SRS is to provide a comprehensive and exhaustive description of the software being developed by Group 1 of CISC 3140-ET6, laying out both the functional and the non-functional (dependent) requirements.
- This SRS is intended to be a guideline for the entire group while developing the project for what to expect of the end product. This SRS will be use by each team in the following ways:
  - *Specs* will be developing this document and making sure that the entire scope of the project is laid out, and a framework is put in place for all other teams to follow.
  - *Backbone* will use this as a guide for coding the game, developing classes based on the requirements laid out here.
  - *Graphics* will use this as a checklist for creating the visual and audio assets necessary for the game.
  - *QA* will test the software against this SRS, making sure that development of the game is going according to plan. They will also use this document to identify conflicts in design vs. implementation of features, so that they can be resolved.

1.2 Scope

- The software being produced is “College Invaders”, A Space Invaders style game with a Brooklyn College “students vs. administration” theme.
- The software will be runnable on any reasonably recent computer with Java 7 (1.7.0_80)
- The software will allow the user to play our game, attempting to attain high scores. Scores will be saved to a list with initials, arcade-style, and will be viewable via an in game scoreboard.
- The software will not require any network connectivity, running locally on the user’s machine.

1.3 References

- Documents and other material referenced in the creation of this SRS:
  - IEEE Std 830-1998 - IEEE Recommended Practice for Software Requirements Specifications. Approved 25 June 1998 IEEE-SA Standards Board. Available at https://standards.ieee.org/findstds/standard/830-1998.html
  - Example of classic space invaders gameplay in flash, published by phatcatmedia.net. Available at http://www.pacxon4u.com/space-invaders/
  - Wikipedia article on Space Invaders, originally published June, 2004 by Wikipedia. Available at https://en.wikipedia.org/wiki/Space_Invaders

- Java Language Specification, Java SE 7 Edition, final release published by Oracle Corporation July 2011. Available at http://docs.oracle.com/javase/specs/index.html

1.4 Overview

The remaining two sections of this SRS Cover the overall description and the specific technical requirements of the software.

- *Section 2* broadly encompasses all the details of this software's interface with the user, as well as other software and hardware systems. A summary of features and functionality is included, as well as a profile of the typical user. Finally, project constraints and hardware/software dependencies are addressed.
- *Section 3* explains the technical specification of the software. This section lists all requirements for the software's inputs and outputs to a level of detail sufficient for developers to write code needed to execute these specifications, and for testers to verify their correct functionality. This section is exhaustive and explicit so that developers, designers, and users may be able to perceive the requirements unambiguously

2. Overall description

2.1 Product perspective

This project is a new, self-contained application written in Java for use by individual users. As a simple arcade-style game, it is small in scope and requirements. The ways in which the software will interface with the system and the user are outlined below and elaborated on in section 3.1

- User Interfaces
  - The user will interface with the game through the normal input devices (keyboard, mouse), and will receive visual and audio feedback from the monitor and speakers
- Hardware interfaces
  - As a stand alone, self contained game, this software has minimal hardware interface requirements.
- Software Interfaces
  - The software will run on any computer able to run the JVM with JRE SE 7
- Communications Interfaces

  - This software is intended to function entirely on a local machine and therefor does not require any network connectivity.
- Memory
  - Considering the scope of this software project, and the only saved data being a text list of high scores, the memory requirements for this software can be considered negligible.

2.2 Product Functions

- To re-create the classic game of Space Invaders, with as “college student vs. administration theme”
- Show the user (via in game menu options) the object of the game, and how to play
- Allow the user to adjust the volume via in game menu options
- To allow the user to start new games, pause and unpause during gameplay, set high scores along with their name, saved to a file that can be viewed on an in-game scoreboard.
- To allow the user to reset the score list via in game options

2.3 User Characteristics

- This game is intended for users who possess rudimentary computer and/or video game knowledge and and ability to make use of computer input and output devices, and for professors who have assigned it as a group project.

2.4 Constraints

- All code for the game must be written in Java
- Should run “smoothly” on any machine capable of installing and running Java 7 JVM.
- Should not consume unnecessary memory (hard drive, RAM).
- Must be able to read and write to a file for recording and retrieval of high scores

2.5 Assumptions And Dependencies

- It is assumed that the end user will have access to a computer running at least Java 7 JVM, with a mouse and keyboard for input and a monitor and speaker(s) for outupt. Additionally the user should have permissions sufficient to save a text file of high scores so they can be maintained across different game sessions.

2.6 Apportioning of requirements

- None, this software will be delivered as complete as possible upon delivery, and will be considered final.

3. Specific requirements

Backbone team members feature responsibilities:

- Menus, including start menu, pause menu, and game over/replay screen: **Max Jeanphilippe**
- Collision detection (including shooting hit/miss logistics) and the creation of a "shield" for the player to block with (not present in current skeleton): **Noam Swisa**
- Enemy movement and automated shooting: **Shayan Jafri**
- Player movement and shooting (including controls for sprite): **Maggie Cao**
- Graphics integration (more details to come on that, graphics team feel free to chime in here): **Wen Huang**
- Scorekeeping, damage, lives and game over logistics: **Jay Lui**
- Level progression and win logistics: **William Ventura**

3.2 External Interface Requirements

3.2.1 User interfaces

- Main Menu
  - start game
  - view high scores
  - options
    - instructions
    - controls
    - sound
    - screen size/resolution
    - high score reset
  - quit

- Pause Menu
  - continue game
  - options(same as above)
  - quit to main menu

- Scoreboard
  - number of scores to track
  - length of strings for high score names

- Gameplay Screen
  - element layout
    - lives
    - score

    - enemies
    - special(UFO, boss) enemies
    - barriers
    - player

- Game over screen
  - name entry for high score(if score is high enough)
  - new game
  - view high scores
  - quit to main menu

- General UI
  - Font(s)
    - TBD

  - Menus
    - buttons
    - sliders

3.2.2 Hardware interfaces

- Keyboard input
  - Input should be based on US QWERTY standard keyboard.
- Mouse input
  - Menus will be navigable with mouse input for ease of use.

3.2.3 Software interfaces

- Java Standard Edition Runtime Environment 7
  - (JRE) 7 (1.7.0_80)
  - JSR-000901 Java Language Specification
  - Version 7
  - **https://docs.oracle.com/javase/specs/**
- Code should be completed using Standard Java Libraries for which there exists an API. Currently there are no constraints on software components, or data sharing. Simply follow standard Java programming practices.

3.2.4 Communications interfaces

- None, local scope only

3.2.5 Memory

- Total game files, assets, and save data will take up a negligible amount of system memory (< 20 MB)

3.3 System Features

This section outlines each major “feature” of the game, broken down into specifics for each feature, describing exactly how each aspect of the feature should appear and behave.

3.3.1 Gameplay Mechanics

- **TBD**
- This mode is simply the actual game.
- Bots/A.I the bots that players will shoot at using the keyboard.

- Controls
  - In menu
    - keyboard selection (up, down, left, right, confirm)
    - mouse selection
  - In game
    - move left
    - move right
    - shoot
    - pause
- Player
  - player movement
  - player projectile type/movement
  - player health/lives
  - player score
- Barriers
  - barrier number
  - barrier health

- Enemies
  - enemy types
    - There are three enemy types arranged in five rows of eleven enemy sprites per row. Rows 1(bottom) & 2 are filled entirely by enemy type 1, rows 3 & 4 are likewise filled with enemy type 2, and row 5 (top) is filled with enemy type 3.
  - enemy movement style
    - The enemies move in unison left and right across the screen until the end of the longest row touches the margin of the gameplay area (TBD). Once this happens all the enemies advance down the screen a set amount(TBD). This continues until the enemies reach the bottom of the screen and cause Game Over, or until all enemies are destroyed.
  - enemy movement speed changes

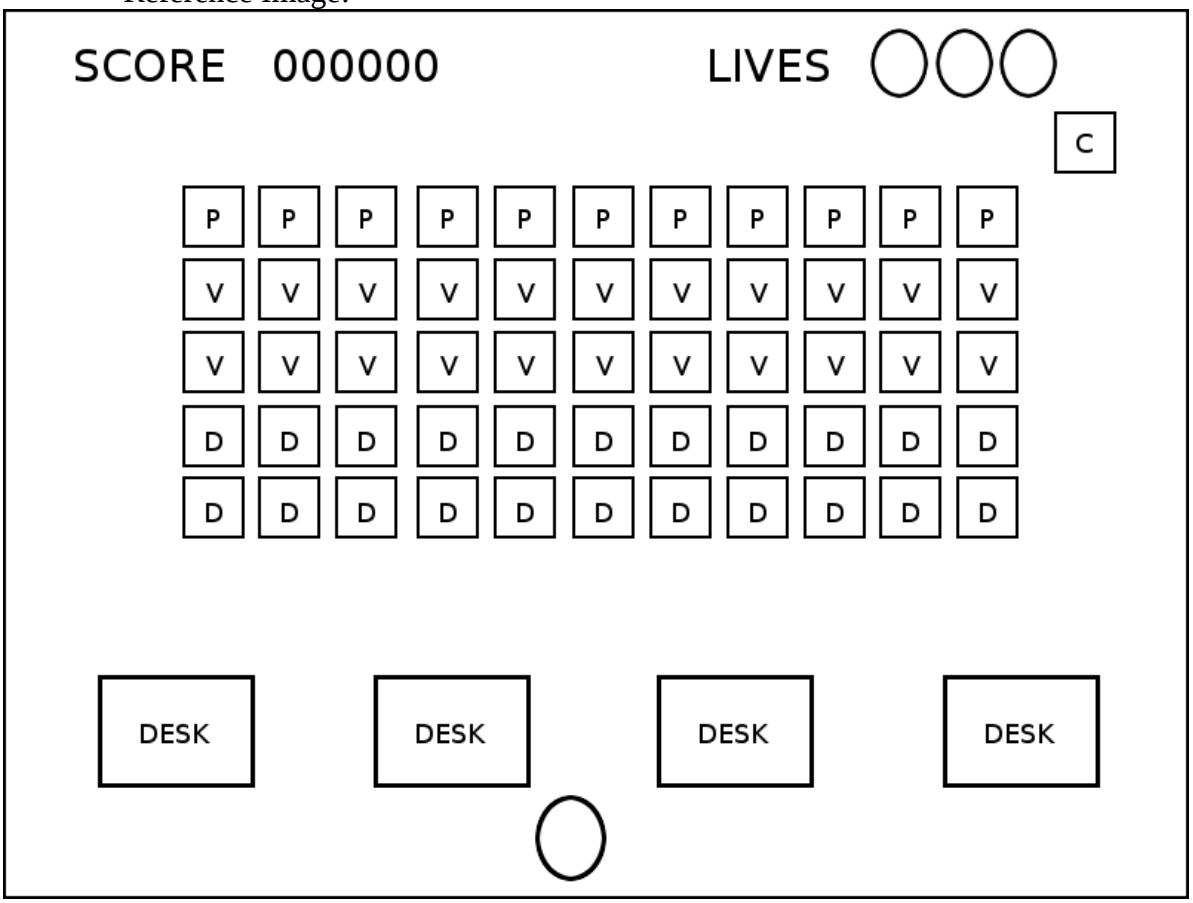
- As enemies are destroyed, the movement speed of all remaining enemies increases up to 1 remaining enemy moving very fast(TBD)

- enemy projectile type/movement/speed
- enemy health
- enemy worth(points)
- special enemy(UFO)

- Score
  - point notification upon kill
  - continually updated current score
  - high score notification?
  - Extra life per certain number of points?

3.3.2 Graphics

Reference Image:



D – dean **V** – vice-chancellor **P** – provost **C** – chancellor

- All graphics assets will be .png format image files, with transparent backgrounds. All dimensions are listed in *WIDTH**x**HEIGHT pixels*
- Enemies
  - Appearance
    - Two sprites for each enemy type, to alternate across and down screen showing movement

Dean pose 1	*dean1.png*	*65x65 px*
Dean pose 2	*dean2.png*	*65x65 px*
Vice-Chancellor 1	*vice1.png*	*65x65 px*
Vice-Chancellor 2	*vice2.png*	*65x65 px*
Provost pose 1	*prov1. png*	*50x65 px*
Provost pose 2	*prov2. png*	*50x65 px*
Chancellor pose 1	*chanc1.png*	*80x65 px*
Chancellor pose 2	*chanc2.png*	*80x65 px*

  - Projectiles
    - Each normal enemy type will have a unique projectile. The Chancellor does not shoot projectiles

Dean projectile	*dean_shot.png*	*20x35 px*
Vice-Chancellor projectile	*vice_shot.png*	*20x35 px*
Provost projectile	*prov_shot.png*	*20x35 px*

  - Death
    - Enemies will have a two-frame death animation, depicting their bloody demise[1]

Dean death frame 1	*dean_death1.png*	*65x65 px*
Dean death frame 2	*dean_death2.png*	*65x65 px*
Vice-Chancellor death frame 1	*vice_death1.png*	*65x65 px*
Vice-Chancellor death frame 2	*vice_death2.png*	*65x65 px*
Provost death frame 1	*prov_death1.png*	*50x65 px*
Provost death frame 2	*prov_death2.png*	*50x65 px*
Chancellor death frame 1	*chanc_death1.png*	*80x65 px*
Chancellor death frame 2	*chanc_death2.png*	*80x65 px*

- Player
  - Appearance
    - The player appears as a single sprite that moves left and right across the screen

Player Sprite	*student.png*	*80x65 px*

  - Projectile
    - The player's projectiles are a pencil and a piece of chalk, fired alternately (or at random TBD)

Player projectile chalk	*student_shot1.png*	*20x35 px*
Player projectile pencil	*student_shot2.png*	*20x35 px*

  - Death
    - Upon being struck by an enemy projectile the player has a two-frame death animation

Player death frame 1	*student_death1.png*	*80x65 px*
Player death frame 2	*student_death2.png*	*80x65 px*

- Barrier
  - The protective barriers the player can hide behind will have a default appearance and 5 frames of increased damage. Upon the 6th hit, the barrier will be destroyed in a 1 frame destruction animation. The barriers appear as school desks.

Barrier default appearance	*desk_new.png*	*130x80 px*
Barrier damage frame 1	*desk_dam1.png*	*130x80 px*
Barrier damage frame 2	*desk_dam2.png*	*130x80 px*
Barrier damage frame 3	*desk_dam3.png*	*130x80 px*
Barrier damage frame 4	*desk_dam4.png*	*130x80 px*
Barrier damage frame 5	*desk_dam5.png*	*130x80 px*
Barrier destruction frame	*desk_broken.png*	*130x80 px*

- Backgrounds
  - There is a different static background image for each screen and menu in the game

Main menu background	*main_bkgrnd.png*	*1024x768 px*
Pause menu background	*pause_bkgrnd.png*	*1024x768 px*
Scoreboard background	*scores_bkgrnd.png*	*1024x768 px*

Gameplay screen background	*game_bkgrnd.png*	*1024x768 px*
Game over screen background	*gover_bkgrnd.png*	*1024x768 px*

- Score
  - Upon destruction of an enemy, the score counter will increment by the point worth of said enemy's type. When the chancellor is destroyed, its point value will be briefly displayed so the player knows how many points were earned.

3.3.3 Sound

Sound assets will be .wav files. Almost all sounds will be loopable.

- Background music
  - Annoying background music will loop while on the main menu, options, and scoreboard screens.

Menu Background Music	*menu_bkrnd.wav*	*10 sec.*

- Gameplay
  - During gameplay, no music will play but sound effects for the movement of the enemies, projectile firing and enemy/barrier/player destruction will be triggered at appropriate times.

  - Movement
    - Normal enemy movement effect is played once each time the enemies move over or down
    - Normal enemy movement effect speeds up as enemies speed up during a wave.
    - Chancellor effect is played, looping continuously, as the Chancellor traverses the screen

Normal enemy movement effect	*enemy_move.wav*	*0.5 sec.*
Special (Chancellor) movement effect	*chanc_mov.wav*	*1 sec.*

  - Projectiles
    - Each of the four projectile firing entities (player, enemies 1,2,3) have a unique projectile firing effect

Player projectile firing effect	*student_shoot.wav*	*0.5 sec.*
Dean projectile firing effect	*dean_shoot.wav*	*0.5 sec.*
Vice-Chancellor projectile firing effect	*vice_shoot.wav*	*0.5 sec.*
Provost projectile firing effect	*prov_shoot.wav*	*0.5 sec.*

- Hits/destructions
  - There are a total of four destruction effects as well one damage effect for enemy hits on the desk barriers. The three normal enemies will share a destruction sound effect, and the Chancellor, player, and desk barrier have their own unique effects.

Player destruction effect	*student_death.wav*	*1 sec.*
Normal enemy destruction effect	*enemy_death.wav*	*0.5 sec.*
Chancellor destruction effect	*chanc_death.wav*	*1 sec.*
Desk barrier hit/damage effect	*desk_hit.wav*	*0.5 sec.*
Desk barrier destruction effect	*desk_death.wav*	*1 sec.*

- Game over/high score
  - Once the player has run out of lives, a **Game Over effect** will be played when the game over message/screen appears. If the player has achieved a new high score during gameplay, the **New high score effect** will be played instead

Game Over effect	*game_over.wav*	*2 sec.*
New high score effect	*new_highscore.wav*	*2 sec.*

- Menus
  - Entering/exiting menus and pausing/unpausing will share a transition sound effect, and all mouse-selectable UI items (buttons sliders) will share a Button click/select effect

Menu open close & pause/unpause effect	*pause.wav*	*0.5 sec.*
Button click/select effect	*click.wav*	*0.5 sec.*

3.4 Performance requirements

- Should run “smoothly” on any machine capable of installing and running Java 7 JVM.
- Should not consume unnecessary memory(hard drive, RAM).

3.5 Design constraints

- Game should be designed using standard programming practices. Java Standards may be read here: https://google.github.io/styleguide/javaguide.html.
- Hardware is limited to computers that are able to run Java 7's JVM.

3.6 Other requirements

- TBD