Software Requirements Specification

for

Submarine Survival

Version 1.0 approved

Prepared by Lynsie LaPointe, Olivia Emery, Angela Deslandes, Amelia Vogel, Jose Bulux, Yusra Suhail, Lakshinee Rungadoo

URI CSC 305-F21-G

27 September 2021

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Jose Bulux	10/18/2021	Adding the Menu and making the exit button work	1.0
Jose Bulux	10/26/2021	Adding comments and the background for the play screen	1.0
Olivia Emery	11/1/2021	Adding the submarine to the play screen	1.0
Olivia Emery	11/3/2021	Adding player class	1.0

Yusra Suhail	11/3/2021	Adding a help class for game instructions	1.0
Yusra Suhail	11/4/2021	Adding buttons on the help screen which can take to the game, main menu or quit the game	2.0
Olivia Emery	11/14/2021	Fixed the enemy class, enemies appear at random Y locations and then re-appear at a new random y starting from the right side again if it passes by the submarine.	2.0
Lynsie LaPointe	11/15/2021	Got submarine to move up and down. Added torpedo class and got them to appear when space is pressed	2.0
Lynsie LaPointe	11/15/2021	Fixed the screen size and made submarine and enemy smaller	2.0
Olivia Emery	11/15/2021	Began the collision class, added a second enemy class for level 3	2.0
Olivia Emery	11/26/2021	Got collision working, deleted unused files, added comments and cleaned up enemy, player, and controller files	3.0
Lynsie LaPointe	12/5/2021	Adding the kraken boss and made it move up and down	3.0
Yusra Suhail	12/5/2021	Got the shark threshold fixed. 15 sharks appear on the screen and keeps on decreasing once player hits it with missile.	3.0
Olivia Emery	12/6/2021	Got shark collision to work	3.0
Amelia Vogel	12/13/2021	Added lives in controller class	3.0

1. Introduction

1.1 Purpose

Our game's main purpose would be to entertain our users by providing them with the latest version of arcade games that include submarines too. It targets mostly gamers of age 15 and up. Another The hidden purpose of our game is to represent our distaste for trash in the ocean. The enemies are the ones who are polluting the ocean with trash, and our goal is to avoid them throwing trash as much as possible by eliminating them.

1.2 Document Conventions

1.3 Project Scope

A main feature of the software will be allowing the player to adjust the speed and direction in horizontal and vertical directions to evade the enemies. Suspenseful music will be played in the background to increase the user's entertainment. In an underwater setting, the user will navigate a submarine and avoid obstacles such as sharks or squids. The game will include a total of seven levels, each increasing in difficulty. As each level increases in difficulty, more enemies will be added that the player must destroy, and the enemies will increase in speed as well. The final level will include a final boss, with the highest level of difficulty. The player will start the game with three lives, and when those lives are used, the game will start over completely. To advance to the next level, the player must destroy all the enemies on screen. The player will have an opportunity to collect extra lives in the final level.

1.4 References

2. Overall Description

Our game is inspired by 1980's arcade games, like Galaga. The inspiration leads us to a game with 2D graphics and shooting objectives. The game will contain levels that increase in difficulty. To use the game, the user will be playing as a submarine that you can move up and down using the arrow keys while trying to avoid and defeat objects surrounding you. There will be enemies that throw trash at the submarine, our goal is to eliminate the enemies and avoid the trash. Once we clear the pathway by eliminating all enemies on the screen, we will move onto the next level, where the difficulty will increase by having more enemies that are faster and more difficult to kill (bigger enemies must be shot more than once to kill). There will be around 4-7 levels, with 3 lives for the whole game. There will be opportunities to gain more levels in case you lose some, but no opportunity to gain more than 3 levels at once.

The overall use of this product is to be played and enjoyed at any time of the day. The targeted audience will be any gamers, average players or any person who is looking to play a game. The quality of the game will be simple and easy to understand. It will provide you with information about how to play and control the game.

2.1 Product Perspective

The game Submarine Survival takes inspiration from old arcade style games that could be played with a button and joystick. It will be a little bit more advanced with the graphics, however the concepts and features will mostly be similar. We will make the requirements for the software and use them to program each feature. We will use the up and down arrows to control the submarine going up and down. We will use the spacebar to shoot missiles out of the submarine. The missiles will travel straight out to the right. Once we are done with the program, we could upload it to a website for users to play, which would be a server other than our own.

2.2 User Classes and Characteristics

Our project is made for novice users who will play this game to relax, and we have made some beginner levels easier for them. It is also made for adept users who like arcade games and want to get challenged, so we have made levels up to seven with increasing difficulty that will keep their attention till the end.

2.3 Operating Environment

This game can run on any computer that supports Java since that is the language our program will be in. Any operating system that has a Java Virtual Machine installed can run this game. Having an IDE installed is also recommended—but not required—to run the program.

2.4 Design and Implementation Constraints

Some things that our team may run into downloading images and making sure that there aren't any copyright issues. Figuring out how to code the graphics and make it work in the way we want so that it fits our game. This could be figuring out how to make the lives decrease when you get hit, have the water moving while the sea creatures and submarines are also moving, program missiles or projectiles moving towards an object. Finding music that we can use and figuring out how to overlay it in the levels and incorporate other sounds into the code/game as needed.

2.5 Assumptions and Dependencies

One major factor that can determine the course of the project is our expertise and knowledge level of Java since that is the language our program will be in. If we have trouble understanding and coding in Java, this might lengthen the coding process, and cause us to run into issues later when we need to debug and test the program. Another factor that is of concern is our knowledge of the JetBrains IDE that we will be using to code the program since we need to get familiar with the software to ensure we can accurately complete the project. We will also be depending on both the IDE and GitHub for this project, so if the software is not working as intended—like if the website goes down—this may hinder our progress and development of the project. Since each member of the team has a unique experience level in Java, we will need to ensure that time is managed well to allow a stress-free process to eliminate bugs and errors. If a group member is unable to complete part of the project due to an emergency, this might also make things more difficult since we might start to fall behind.

3. System Features

3.1 Moving and Shooting (Submarine)

3.1.1 *Description*

As a game player, I need the submarine to move and shoot off a missile when I press certain buttons so that I can eliminate the enemies and trash on the screen and move out of the way of incoming projectiles. (High priority)

3.1.2 Stimulus/Response Sequences

3.1.2.1 Move Up

Stimulus: the player presses the up-arrow key. Response: the submarine moves upward on the y-axis.

3.1.2.2 Move Down

Stimulus: the player presses the down arrow key. Response: the submarine moves downward on the y-axis.

3.1.2.3 Shoot

Stimulus: the player presses the spacebar. Response: the submarine shoots off a missile from the front and moves forward on the x-axis.

3.1.3 Functional Requirements

3.1.3.1 Must start triggering the correct action when pressing the appropriate keys for that action.

3.1.3.2 Must stop triggering those actions when the user stops pressing the keys.

3.1.3.3 Must keep track of the submarine's coordinates to move up and down and release a missile in the same location as the submarine.

3.1.3.4 Must keep track of the missile's location to move it across the screen.

3.2 Collecting Lives

3.2.1 Description

As a game player, I need to keep an eye out on special objects or creatures so that I can collect extra lives to help me at higher levels. (Low priority)

3.2.2 Stimulus/Response Sequences

3.2.2.1 Special Objects

Stimulus: The special object enters the water. Response: The user uses the space bar to shoot missiles at the object to destroy it and gain extra lives.

3.2.2.2 Special Creatures

Stimulus: The special creature enters the water coming towards the user. Response: The user shoots missiles at the creature three times before destroying it to receive an extra life.

3.2.3 Functional Requirements

3.2.3.1 The program will need to recognize that special objects and creatures may have the same behavior as ordinary objects and creatures, but their outcomes must be different.

3.2.3.2 As such, special objects and creatures should be a subclass of ordinary objects and creatures, so that they can still be derived from the objects and creatures' class.

3.3 Sea Creatures/Enemies

3.3.1 Description

As a game player, I need to focus on the playability of the submarine and its attacks in the game environment so that I can destroy all required obstacles to advance levels. (High Priority)

3.3.2 Stimulus/Response Sequences

3.3.2.1 Enemies

Stimulus: the player is traveling by submarine. Response: every 5 seconds, a series of "enemies" will deploy in random order in multiple locations across the screen to attack the submarine to destroy it. Enemies consist of various sea life and bombs coming in from another submarine.

3.3.2.2 Final Boss

Stimulus: the player enters a cave with skeletal remains. Response: a final boss will appear in the form of a large sea animal and attack the submarine at increased speeds and time intervals of 3 seconds. 2-3 other smaller sea animals will appear every 20 seconds and attack the submarine.

3.3.3 Functional Requirements

3.5.3.1 The system must be able to accurately track the collision boxes of the submarine and the enemies.

3.5.3.2 Must be able to "shoot" missiles, aka make the missile appear at the front of the submarine and shoot to the right when the space bar is clicked.

3.5.3.3 Must be able to make the missile and the enemy disappear once the missile hits the enemy.

3.5.3.4 Must be able to see if all enemies are removed from the screen and advance to the next level if they are.

3.4 Instructions Page

3.4.1 Description

As a user, I would like to see an instructions page so that I can get help on how to play the game and win it. From the help screen I want to navigate back to the menu screen; the game screen and I should also have the option to quit the game. (High priority)

3.4.2 Stimulus/Response Sequences

3.4.2.1 Help Button

Stimulus: The player clicks a button that says help. Response: A help page is displayed that shows all the instructions on how to play the game.

3.4.2.2 Menu Button

Stimulus: The player clicks the menu screen button visible to him on the help screen.
Response: The menu screen is displayed to him.

3.4.2.3 Play Button

Stimulus: The player clicks the play button visible to him on the help screen.
Response: The game takes the player to the game page where he can play the game.

3.4.2.4 Quit Button

Stimulus: The player clicks the quit button displayed on the help page. Response: The game quits.

3.4.3 Functional Requirements

3.5.3.1 The software must only display the instructions page if a user clicks the help button.

3.5.3.2 The software must show the three buttons on the help screen on which the user can click.

3.5.3.3 Must be able to detect user clicks on the buttons to show respective pages.

3.5.3.4 Must not change the screen if user clicks on any other area other than that covered by the buttons.

3.5.3.5 Must be able to show the instructions to play the game.

3.5 Levels

3.5.1 Description

As a user I need to have multiple levels with varying degrees of difficulty so that the game is entertaining and not repetitive. (High priority)

3.5.2 Stimulus/Response Sequences

3.5.2.1 Threshold Number

Stimulus: The user will hit creatures until they reach a threshold number. Response: Once the user reaches the threshold number, they advance to the next level. The threshold number determines the difficulty of the level—the higher the level, the larger the threshold number.

3.5.1 Functional Requirements

3.5.3.1 The program must keep track of all objects that the user hits, and as soon as the user hits the threshold number, it should display the next level and the counter must be reset and the threshold number should increase, in accordance with the level number.

3.6 Object Collision

3.6.1 Description

As a game player, I need the sea enemies and the trash objects to disappear once I shoot them with missiles, so that I can advance to the next level of the game.

3.6.2 Stimulus/Response Sequences

3.6.2.1 Missile Hits Enemy

Stimulus: the player shoots a missile that hits an enemy object. Response: The enemy object disappears from the screen. (TBD: hitting enemies could lead to an increase in the user's score.)

3.6.2.2 Missile Hits Trash

Stimulus: the player shoots a missile and hits a trash object. Response: The trash object disappears from the screen, cleaning up the ocean.

3.6.2.3 Level Advance

Stimulus: the player eliminates all enemy and trash objects. Response: The screen is cleared and the player advances to the next level, which will contain more enemy and trash objects that may or may not be harder to eliminate.

3.6.2 Functional Requirements

3.6.3.1 The program must keep track of all objects' locations in order to trigger the appropriate response when the user hits the object with their missile; this includes the location of the user's missile as well as the trajectory of the missile until it hits a target.

3.6.3.2 There also needs to be a counter running for the duration of the level, keeping track of the objects getting hit so that the user can move on to the next level after hitting enough objects.

3.7 Menu

3.7.1 Description

As a game player, I need a menu to navigate the game. I want to be able to have options to choose between play, help, and quit buttons.

3.7.2 Stimulus/Response Sequences

3.7.2.1 Game

Stimulus: the player clicks on the play button. Response: the game begins causing the screen to change and start the game.

3.7.2.2 Help

Stimulus: the player clicks on the help button. Response: the menu screen appears giving useful tips and tricks in how to play the game and control the game

3.7.2.3 Quit

Stimulus: the player clicks on the quit button. Response: the quit button will terminate the game causing the game to quit and end.

3.7.2 *Functional Requirements*

3.7.3.1 The program must respond correctly depending on what the user clicks on. If the user clicks out anywhere else on the page nothing will happen.

3.7.3.2 Only when the user clicks on the correct place that is when it will take the user to where it wants it to go.

4. Data Requirements

4.1 Logical Data Model

Lives Class

Player Class:

Levels Class:

4.2 Data Dictionary

Lives Class: This class deals with the data of the lives and increasing and decreasing the number of lives based on what happens within the game with the player and the enemies. This class didn't really workout correctly, so it was integrated into the controller class where they check the enemies hitting the player.

Controller Class: This class deals with the collisions of the torpedo with squid and shark enemies as well as the kraken and has them disappear from the game once they are hit. It also checks for when the enemies and ink splats hit the player and calculates the number of lives that are lost each time the player is hit. When the number of lives equals zero the game then terminates after setting the number of lives back to five and resetting the game that when it goes back to the main menu they can just replay. This class also checks and sees if conditions are met such as no more enemies or a specific

number of enemies hit to be able to move onto the next level and once all the levels are completed the game finishes and a statement pops up saying that the game has been won.

Menu Class: This class sets up the game and has the introduction to the name of the game and the opportunity to play the game, go to the help page, or quit the game. It relies on the mouseInput/keyInput class so that when the buttons are pressed it takes to the desired screen,

Help Class: This class displays the instructions for the game that the users can read to grasp a better understanding of the game based on the help button being pressed. It also provides buttons to go back to the main screen, play the game, and quit the game to exit and go back to your code.

Enemy Class: This class sets up the enemies so that their images are rendered and sets up their dimensions and movements within the game so that they move at specific. Speeds and goes from right to left on the screen.

Torpedo class: This class creates the torpedoes and determines how they're going to be shot out from the sub when the spacebar is pressed by the user. It controls the amount of torpedoes that are launched at each spacebar press and at what speeds they move at. This class plays off the keyPressedInput class where if a specific button is pressed a specific action is executed and, in this case, it is the launching of the missiles from the sub by the user to eliminate the enemies.

Player class: The player class is what creates the submarine and determines the size of the sub as it appears on the screen and the movement of the submarine. The sub moves from up to down on the screen and is positioned on the left side of the screen and doesn't move other than the up and down actions that are controlled by the user pressing the up and down keys on the computer allowing for the user to control the subs movement.

4.3 Reports

Not Applicable

4.4 Data Acquisition, Integrity, Retention, and Disposal

If we had more time, we could have developed a feedback button where the user can voice their opinion on how to better the game or if there are any glitches we can patch up. No information will be taken from the user, all feedback will be anonymous. The only data that will be saved and stored is the feedback. Data will be saved and depending on the feedback we will improve the game to keep the players happy.

5. External Interface Requirements

Our program uses interaction with keyboard input to help the software. The up and down arrow keys make the submarine move while the space bar makes a missile shoot out from the submarine.

5.1 User Interfaces

This is our home menu, where the user must use the mouse to click any of the three buttons. The play button will change the state of the game to be true which will begin play on the game. The help button will bring you to a page with instructions on how to play the game and re-give the user the 3 options seen above. Finally, the quit button will terminate the game window.

This is level 2 of our game, where the submarine will be moving up and down based on the user's key input, and torpedoes will also shoot based on the user's key input. The user will have to avoid the oncoming enemies as well as try to shoot at them which will eliminate them.

This is our final level, the boss level, where the kraken will move up and down constantly, and it will shoot ink splats towards the player. The user will have to avoid the ink splats and the enemies as well as shoot the kraken and the enemies.

There is a shortcut all throughout the game that any time the user clicks on the "q" key, the game window will terminate, ending the game.

5.2 Software Interfaces

Since this is just the start of a brand-new game, we do not have any connections to other software applications. However, we are working on this game on different operating systems. We are working on Windows as well as macOS. One problem we have come across using two different operating systems is that IDE's function differently on different operating systems. Also, one specific problem we had was that enemies were travelling at a normal speed on one computer while they were travelling like 4 times as fast on another computer that was coding on a different operating system. Although we faced these challenges, we used Git to share code between computers, and were successfully able to pull from each other's branches and pull from main to always work on the most updated code. At the start of coding our game we printed out the frames per second that we were getting, so we knew the response time of our game and seeing if the code we added would optimize the code or not.

5.3 Hardware Interfaces

The hardware we use is a mouse, keyboard, monitor, and computer. With the keyboard, we use a key listeners to see when each key is pressed. With this information, we can move the player object up or down or have it idle otherwise. When the space bar is pressed, we create a new instance of the torpedo class and add it at a specific location on the submarine. With the mouse, we use the left-click button to press each button on the menu screen, and to press the continue button in between levels. The monitor presents the software on a viewable screen, showing the working code with the moving objects. The computer is what stores the memory of the code and compiles and runs the code. It saves the data that we ask it to save and uses it to manipulate the objects when we ask it to. So far, we have only tested running this code on a local computer that has access to the repository on GitHub. However, if we had time to figure out how to make it mobile then that would definitely be something we would want to do.

5.4 Communications Interfaces

At the moment, our program does not use any sort of communication with email, web browsers, network protocols, or electronic forms. However, if we could, we would add our game to a web browser for public users to enjoy playing our game as well. Another way we could improve our code is to have users create an account to keep track of their best score. To do this, we would need to have

the user input their email address and username and password. Then we would need a communication function in order to get their previous score information to show them what they have gotten as a score in the past, so that they have a goal to beat. If we did this, we would have to make sure that we had encryption for their user information such as their password. Although there may not be sensitive information within their account, they could use the same password for other accounts, so we want to keep that secure for the user. Also, we wouldn't want the reputation of having leaked passwords because that makes the user not trust the program which would make them avoid playing our game, which we do not want. We are not sure if we could get user input for taking in emails, but that is something we could investigate with further development of this game.

6. Quality Attributes

6.1 Usability

Our program is almost fully user friendly. The user begins with the three button menu options. The quit button terminates the window, however, the other buttons take the user to a new screen. If they click the help button, they will visit the help page, but still have the option to return to the menu or quit the game right in the help screen. If they click play, they will begin playing the game, and if at any point they want to quit, they can push the "q" key on the keyboard and it will terminate the window. Another option is to just exit the window which will also terminate the game. One usability function that may be only beneficial to right hand users is that the more important functionality buttons: the up and down arrow keys, are used by the right hand while only the space bar is used by the left hand.

6.2 Performance

Submarine Survival is compatible to work on a laptop or computer only. Any IDE that supports Java can help to play the game, only code is required. The computer should also have any modern-non-entry-level graphics card so that they can enjoy the graphics in the game. The computer should also have a space bar and up down arrow buttons to play the game.

6.3 Security

The game does not require the user to enter their personal details anywhere in the game. It also does not require internet connection so the privacy of the user will not be compromised.

6.4 Safety

Not Applicable

7. Internationalization and Localization Requirements

Submarine survival will be access by anybody who has a laptop or computer and Wi-Fi. This game's main language is US English. Those who do not know English will not understand what the buttons say or understand the help page. The game does not show any blood or gory content although, people

who may be triggered or take offense to shooting sea creatures should not play this game. Submarine survival recommends people from the age 6+ to play this game.

Appendix A: Glossary

Collision
When the submarine or its missile interacts with any of the objects on screen; depending on object type, lives can be lost, or the user can get closer to the threshold by hitting or getting hit by the object.

Enemy
Creatures that appear on the screen during all the levels. There are 3 different enemies that will appear in the game and the player must hit a certain number of enemies in order to advance in the game and win.

Ink
An obstacle thrown at the player by the kraken that they must avoid; getting hit by an ink splat costs them a life.

Kraken
Boss enemy that appears during Level 3. To defeat the kraken, the player must shoot it 10 times. The kraken can shoot ink splats at the player that the player needs to steer clear off.

Levels
As of 13 December 2021, there are 3 levels that the player must complete to finish the game. The first level involves getting rid of squids; in the second level, the player can hit both squids and sharks; in the third and final level, the player can hit squids, sharks, and a kraken to win.

Lives
Players start out with 5 lives, and they lose lives by getting hit by enemies such as squids, sharks, or kraken ink splats. To gain lives, players would have had to hit special creatures.

Missile
The weapon that the player uses to shoot the enemies. Missiles are launched by pressing the spacebar key.

Shark
The enemy that first appears in Level 2 of the game and is also seen during Level 3.

Special Creatures
Rare objects that would have appeared randomly at all levels, giving players an opportunity to gain lives that they lost; however special creatures are not currently included in the game due to time constraints.

Squid
First enemy that appears on the screen; present at all levels.

Submarine

The character or object that the player controls on the screen. The submarine can move up and down, and it can shoot missiles.

Threshold
The maximum number of enemies that the player has to hit in order to move on to the next level.