Design Document
for
Submarine Survival

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

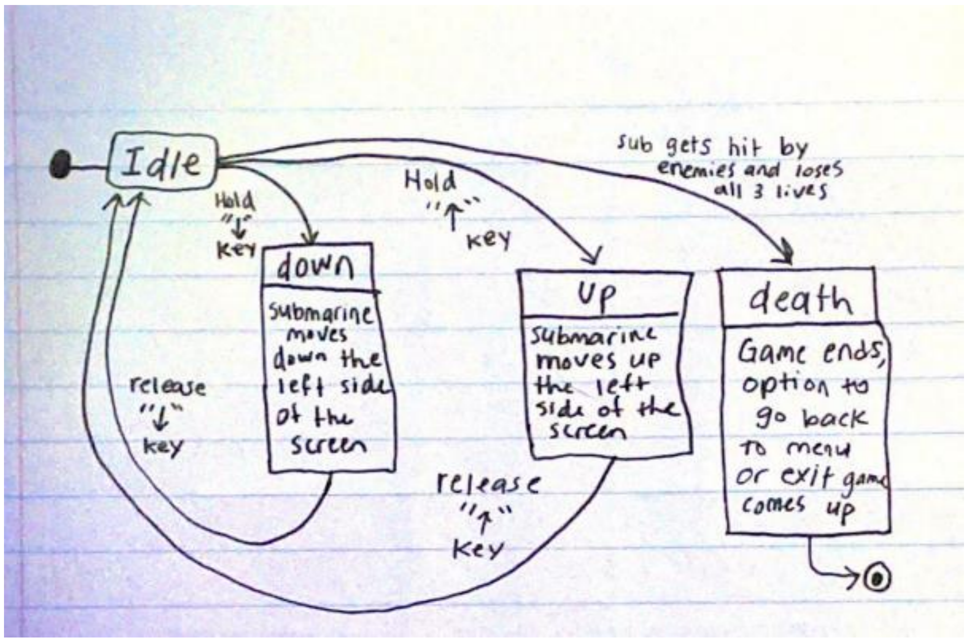
URI CSC 305-F21-G

22 November 2021

Table of Contents

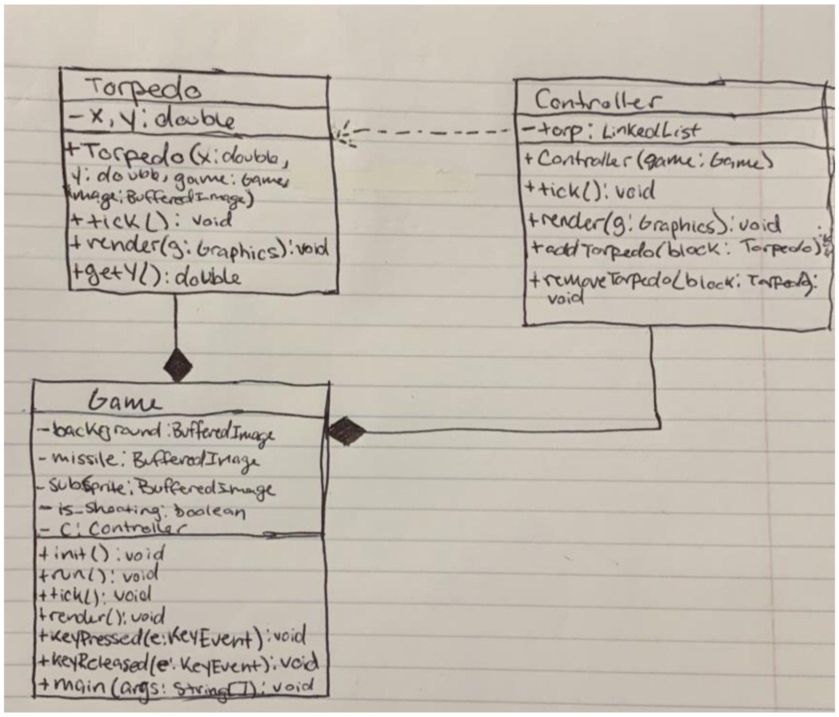
3.1 Submarine

This state diagram shows our section 3.1.2.1 and 3.1.2.2. The functionality includes the movement of the submarine to go up and down the left side of the screen while the up and down arrow keys are held down. Otherwise, the submarine will stay idle at its ‘y’ position. If the submarine gets hit by enemies enough times that the player loses all 3 lives and has no opportunity to gain any lives back, the submarine ‘dies’ and the game ends. Then, the player will have the option to exit the game or return to the menu screen where they will have the option to restart the game if they wanted to.


Idle
Hold "↓" key
down
Submarine moves down the left side of the screen
release "↓" key
Hold "↑" key
Up
Submarine moves up the left side of the screen
release "↑" key
sub gets hit by enemies and loses all 3 lives
death
Game ends, option to go back to menu or exit game comes up

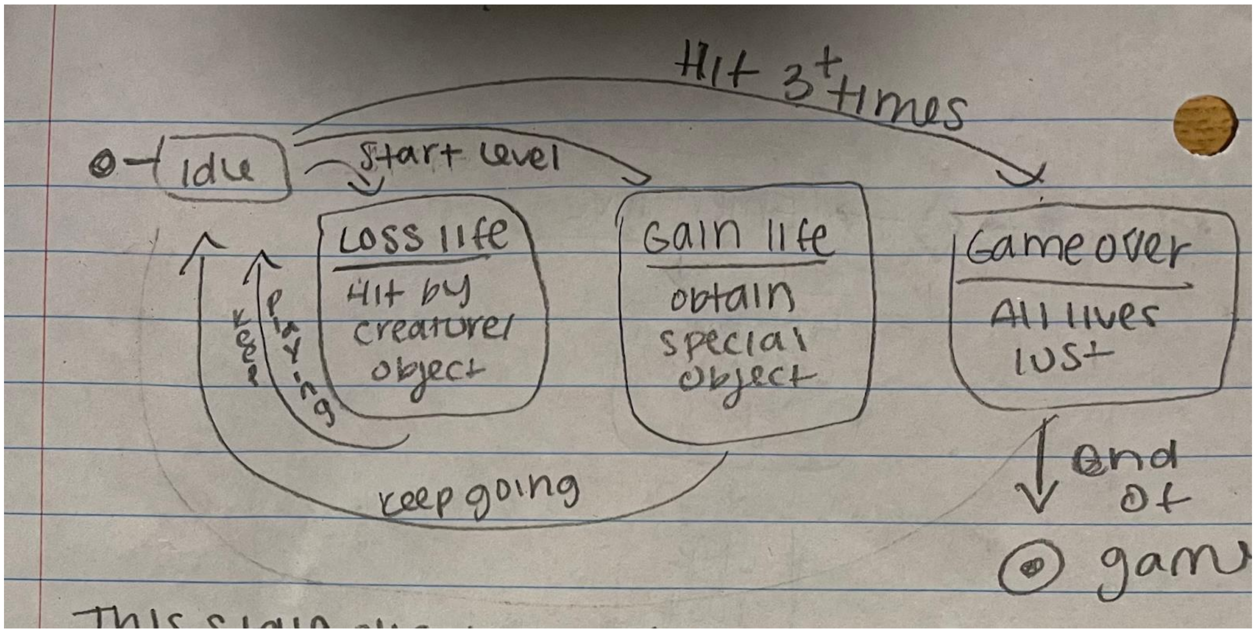
3.1 Torpedo

In order to represent our program’s Moving and Shooting (SRS 3.1), but more specifically the shooting part (SRS 3.1.2.3), I chose to use an UML class diagram. The class diagram is modelled off the class diagram for a Command design pattern. The Game class is our main file where the game runs and calls all the other classes. Therefore, without the Game class, the torpedo and controller classes would be useless. Inside the Game class there is the method keyPressed which calls the controller class to add a torpedo when the spacebar is held. It also has the method keyReleased which will stop the class from producing more torpedoes when the spacebar is no longer held. Inside the controller class is where the torpedo class is called to draw the torpedo image at the x and y coordinates of the submarine. It also determines the speed at which the torpedo will go. Since the torpedo class is called through the controller class, the torpedo class is dependent on the controller class every time it is called.


Torpedo
-x, y: double
+Torpedo(x: double, y: double, game: Game, image: BufferedImage)
++tick(): void
+render(g: Graphics): void
+getY(): double
Controller
-torp: LinkedList
+Controller(game: Game)
++tick(): void
+render(g: Graphics): void
+addTorpedo(block: Torpedo)
+removeTorpedo(block: Torpedo): void
Game
-background: BufferedImage
-missile: BufferedImage
-subSprite: BufferedImage
-is_Shooting: boolean
-C: Controller
+init(): void
+run(): void
++tick(): void
+render(): void
+keyPressed(e: KeyEvent): void
+keyReleased(e: KeyEvent): void
+main(args: String[]): void

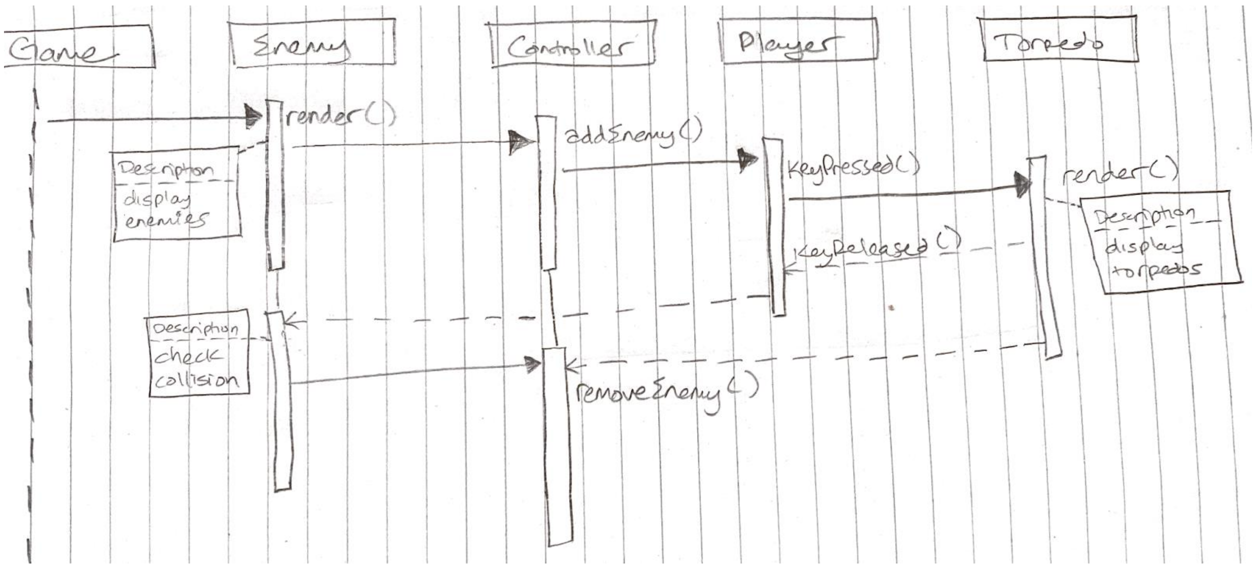
3.2 Lives

This state diagram shows our section 3.2.2.1, the lives class. The functionality includes losing a life, gaining a life, and game over once all lives are lost. When starting the level if you get hit by a creature or object that is thrown at you then a life is lost each time that you are hit but you resume playing the game. If you hit a special object or special creature during the level then you get an extra life that is added to your life count and keep playing until all lives are lost. When you get hit 3 or more times depending on if you acquired any special creatures, then the game is over and you lose the level and have to restart.



3.3 Enemies

The diagram is a model of our SRS document section 3.3 Sea Creatures/Enemies and all of the functional requirements for the Submarine Survival game found in 3.3.3. The following sequence diagram first shows the display of enemies when the game screen is active by using the render function to display the image of the enemy. The enemies will be coming on screen from right to left while the player stays on the left side of the game environment, the game only allowing them to move up and down with the arrow keys. The Controller class handles enemy and torpedo movements. When the game starts, a certain number of enemies will be displayed. When the player presses then releases the space bar key, a torpedo will be displayed, going from left to right, in the direction of the enemies. The game environment will check the collison, and if it returns true, the Controller class will then implement the removeEnemy function. Once all enemies have been destroyed, the player will move on to the next level until the final level is reached (5).

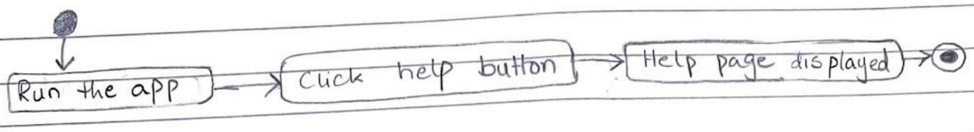


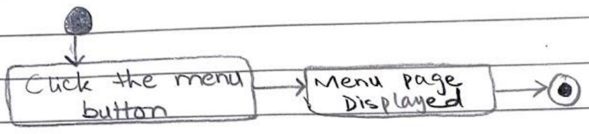
3.4 Instructions

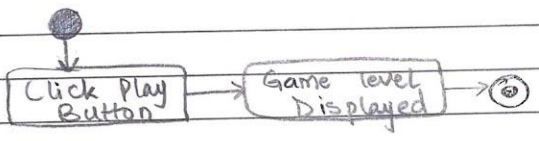
Following is the activity diagram for the feature 3.4 instruction page of our game. The instruction page of our game has buttons on it that user can click and it will take them to different screens. The player can access the instruction page by clicking on the help button 3.4.2.1 on the menu page feature 3.7.The buttons on the help screen include the menu button 3.4.2.2 that can take the player to the menu for the game, then there is a play button 3.4.2.3 that takes them to the game screen where the player can play. Third button is the quit button 3.4.2.4 that exits the game. The instruction page has all the instructions that the player needs to play the game. These instructions tell the user which keys to press to help them do the movement. It tells them how to win from enemies which he can do by shooting missiles. It also tells them about their lives- how many they are, how he will lose them, and how he can boost them up. The buttons on the help screen are configured according to the dimensions of the button so that the buttons perform their functionality only when the player clicks on them. If they click somewhere else the current screen should stay intact. The buttons are also configured to do their own required functionality and not any other screen.

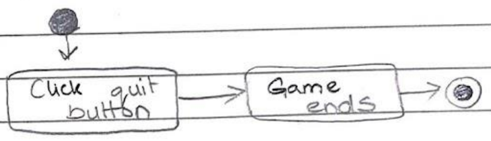
Four activity diagrams are made which explains the activity of these three buttons on the help page and fourth explains the activity which takes the player to the instruction page.

Activity Diagrams for the Instruction Page



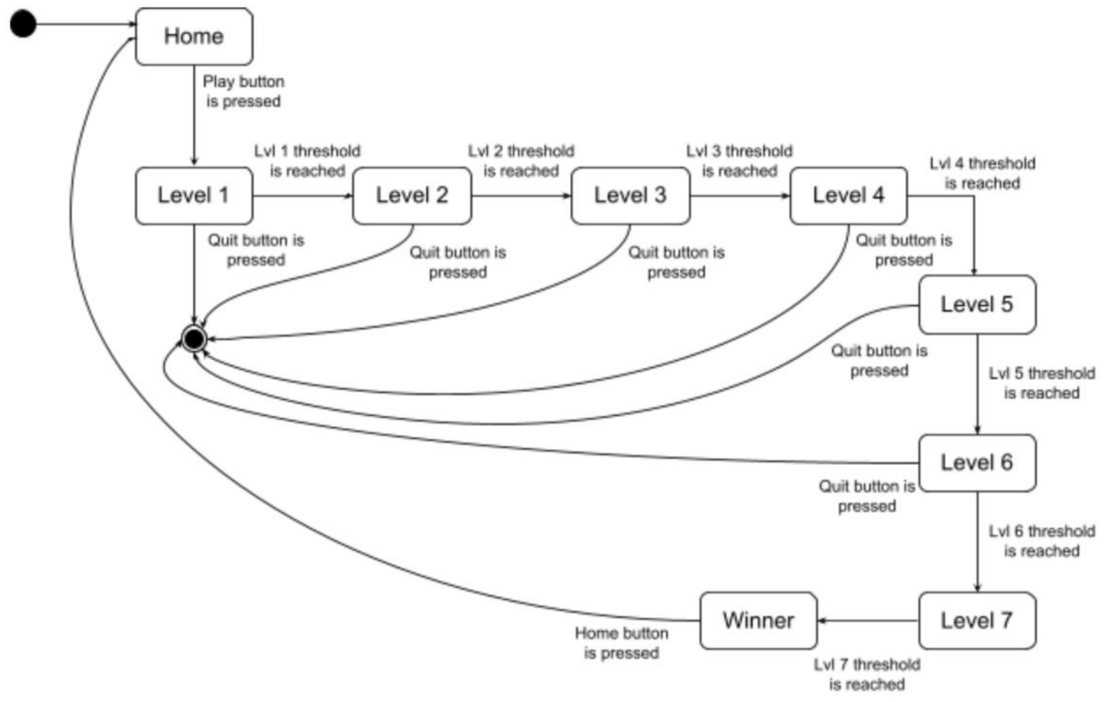






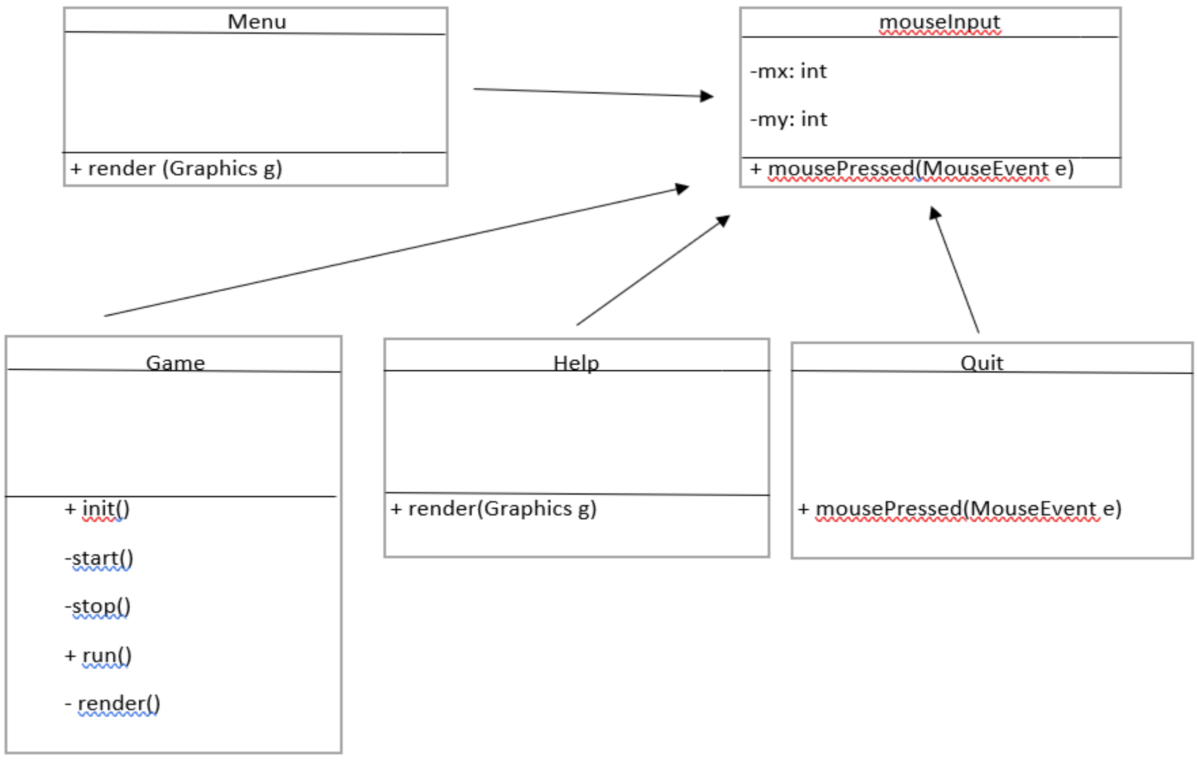
3.5 Levels

To illustrate the workings of our Levels class as described in the SRS under section 3.5, I have created the following state diagram that shows the transition between each level. To begin with, the user will find themselves on the Homepage screen when they run the game. From there, they press the Play button to get to the next state, which is Level 1. By killing enemies and avoiding obstacles or hitting special objects, they get closer to the threshold for the next level. Once they reach the threshold number of points set for the particular level they are on, they transition to the next level, or the next state. This process is repeated until they reach Level 7, which is the final level of the game. For this level, when the user reaches the threshold, the next state they will see is the Winner page, which will congratulate them on completing the game. During any level, if the Quit button is pressed, the game ends and the program terminates.



3.7 Menu

The diagram is a model of our SRS document section 3.7 Menu system and all of the functional requirements for the Menu system are found in 3.7.1. For my UML diagram contribution, I created a sequence diagram for the menu system The menu system is what the user will interact with before playing the game. The menu will display three buttons but the file that's doing all the work is mouseInput. MouseInput jobs is to create an interactive button, you will give it an x and y coordinate to tell the computer where the button to click will be, if it's not clicked within the area nothing will happen. The menu file is what's being displayed and what the user sees, mouseInput is for the program to know what the user is doing. In the SRS 3.7.2.1, the first button is the play button, when the play button is pressed the game will start and the user can begin playing. The second button in the SRS 3.7.2.2 is the help button, this button will provide useful tips and tricks in how to play the game and control the game. This button is meant to be informative so the user can get a better understanding of the game. The third and last button in the SRS 3.7.2.3 is the exit button, if the user wants to exit the game or play later the user can quit the game and the game closes.


Menu
+ render (Graphics g)
mouseInput
-mx: int
-my: int
+ mousePressed(MouseEvent e)
Game
+ init()
-start()
-stop()
+ run()
- render()
Help
+ render(Graphics g)
Quit
+ mousePressed(MouseEvent e)