CS 319 - Object-Oriented Software Engineering

System Design Report

Sea Adventures

<u>Group 2-G</u>

Alper Kağan Kayalı

Büşra Oğuzoğlu

Kasymbek Tashbaev

Salih Zeki Okur

Analysis Summary

1. Description

Sea Adventures is a horizontal scrolling shooter game based on the game called space shooter. The goal of the project is to implement the game using OOP structure. The game will be implemented using JavaFX.

2. Overview

Game's objective is to pass several levels and in the last one to fight the final boss. In addition, in each level on the path of the hero, there will be various enemies who will try to stop him.

The map is basically bottom of the ocean, while there are islands which need to be avoided. In addition, the game will be horizontal, unlike Space Shooters.

Controllers are WASD for maneuvering, spacebar for shooting and "z,x,c" keys for skills.

There will be three levels. Each level has small bosses while the third level will have the final boss. The submarine will gain experience while killing the enemies in order to level up and get better weapons.

Similar Games:

Space Shooters: http://playzool.com/games/spaceShooter/?o=portrait

2.1. Gameplay

The main hero which is a submarine has health and energy. The different enemies try to kill submarine. If health goes to zero, the submarine is destroyed and the game ends. In addition, the submarine can attack to the enemies with special skills.

2.2. Leveling

The submarine's level and the map's level are different. Submarine can level up by killing the enemies. Eventually, when it kills all the enemies in the map, the map's level will increase. In addition, at the third level, the final boss appears.

2.3. Weapons

The submarine will be able to level up 5 times. While the submarine levels up, the damage of its weapons will increase as well. In addition, since the damage skills of the submarine is directly related to the damage of the submarine, the damage that those skills create will increase as well.

2.4. Skills

The skills of the submarine will be affected as same as weapons. At each level a new skill will be unlocked which will affect submarine differently. In addition, the skills won't be used infinitely many times since there is a cooldown time for each skill and energy for submarine. However, the cooldown time will decrease at each leveling up, therefore, using the skills will be easier.

2.5. Enemies

Enemies will try to prevent you from reaching the end of the level in different ways. There are three type of enemies at each level: 1)small enemies, which can only hurt submarine by colliding with it. 2) big enemies, which can shoot bullets to the submarine. 3) boss, which has skills to kill the submarine. At the final level, the final boss appears which is much more stronger than regular bosses.

3. Requirement Specificaiton

3.1. Functional Requirements

3.1.1. Play Game

While playing the game, the user can control only the submarine. The user will have control over its movement, shooting, and skills. Controllers are:

-WASD for maneuvering.

-spacebar for shooting.

-“z,x,c” keys for skills.

By using these three features, the user will try to avoid and kill little enemies, bullets of the big enemies and finally a boss at the end of every level. Enemies will try to shoot or hit the submarine and once they do user will lose some amount of its health according to the strength of the enemy. When submarine shoots an enemy, it will lose health according to the submarine’s strength.

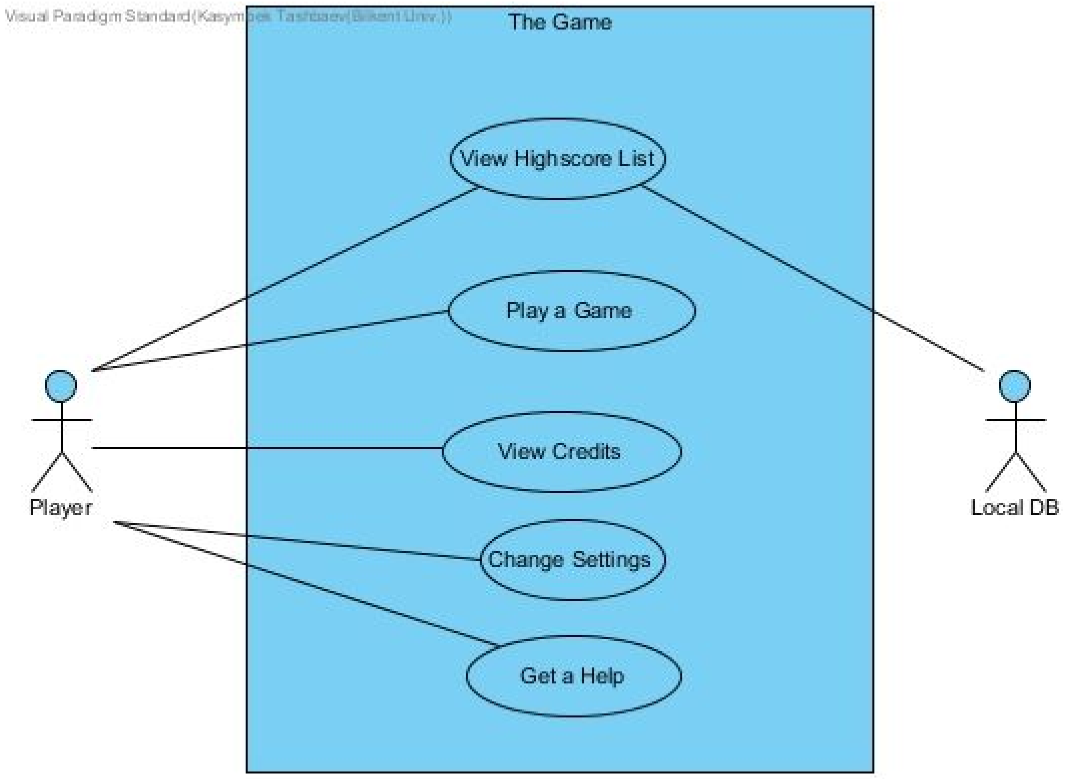
The user will be able to start higher levels once he/she killed the boss in a level. Enemies and bosses will get stronger at every level.

3.1.2. High Scores

In the main menu of the game, there will be a high scores button to reach high scores page. From high scores page, the player can reach highest 10 scores that have been made in the game. Player will get points by killing enemies in the game. There will be several different kinds of enemies that will give the player different amount of points based on their difficulty level.

3.1.3. Pause Game

The game can be paused at any time user desires. In addition, the user can continue to the game at any time (s)he wants at the exact location.



The Use Case Diagram for the game.

3.1.4. View Help

Help page can be viewed by the player from the main menu. From help page, the player can learn about the controls and gameplay of the game.

3.1.5. View Credits

The user will be able to see the developers of the game directly from the main menu, otherwise, credits scene will not bother the user during gameplay.

3.1.6. Change Settings

The user will be able to change music and game buttons from change settings menu which could be reached from the main menu in the beginning of game.

3.2. Non-Functional Requirements

3.2.1. Game Performance

Threads will be used to have better game performance. Basically, the game structure will be run at different thread than the GUI part of the game in order to speed up the game.

3.2.2. Graphical Smoothness

BufferStrategy object will be used to construct interface more smoothly.
Different submarine pictures will be used to make animation.

3.2.3. User-Friendly Interface

The game will be easy to use. First of all, in the main menu, each section will be clickable. Therefore, the main menu will be controlled by just using the mouse. In addition, the game will be controlled by the keys which are used in almost any games out in the market, which means that the game will be easier to be played by the user since (s)he will be familiar with the controller if (s)he played any game.

Design Summary

1. Design Goals

According to non-functional and functional requirements, we decided the following design goals: easy to use, entertaining, extendability, portability, reliability, modifiability, efficiency. In order to achieve our goals, we have sacrificed the following design goals:

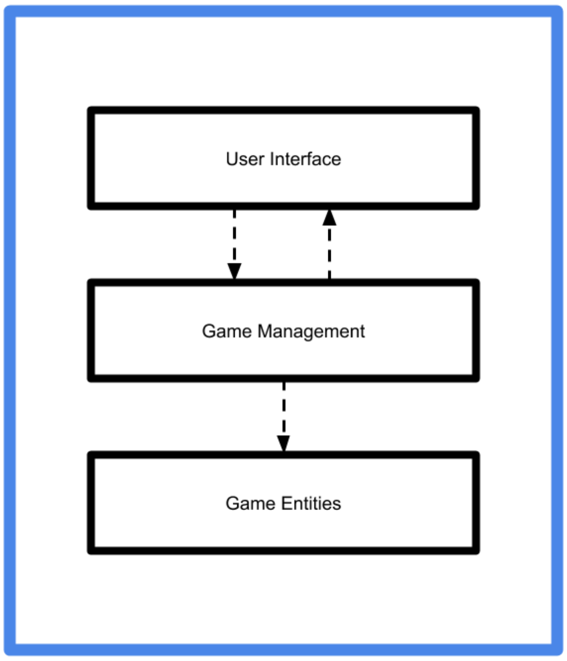
functionality to make the system more simple and more understandable, and memory to increase the performance.

2. Software Architecture

We decomposed our system in order to apply MVC (Model-View-Controller) architectural design into our system. So we divided our system into three subsystems: User Interface, Game Management and Game Entities subsystems. It reduces the coupling between different subsystems and increases the cohesion of the subsystem components.

In our design, Game Entities are mostly the objects of the game, while User Interface includes ScreenManager, InputManager and Menu classes. Other classes such as game engine, sound engine and health manager, are used for Game Management and responsible for controlling general game mechanics therefore, they compose the controller part of the system.

Our game system consists of three different layers, where subsystem is a layer. User Interface interacts with the Game Management and Game Management interacts with the Map class that is responsible for general management of Game Entities layer.



3. Hardware / Software Mapping

Sea Adventures is implemented using JavaFX, so it can run on any computer that has Java Runtime Environment in any operating system. As a hardware aspect for interaction with the game, the player needs a basic keyboard and mouse. Since we will not have a lot of data to save, we will not use a complex database system. Game maps, submarine and other objects' images, sounds and a list of high scores will be stored in the appropriate formats on the hard disk. Therefore, the necessary files will be loaded before the game starts. This gives us the ability to quickly update visual effects and sound effects, but if the files are damaged, the game's user interface will not work properly.

4. Access Control and Security

Sea Adventures has the executable .jar file so it does not require installing before entering the game. The game does not have a player login system, therefore, there will be no database for storing details of users. For this reason, there will not be any controls or restrictions about accessing the game. Everyone who has the game file can access and play the game. Their data will not be saved as a player so if other player wants to access the game from the same system, the second player should start over or continue where the first player left. On the other hand, sea adventures game will not require any network connection, therefore, there will be no security issues.

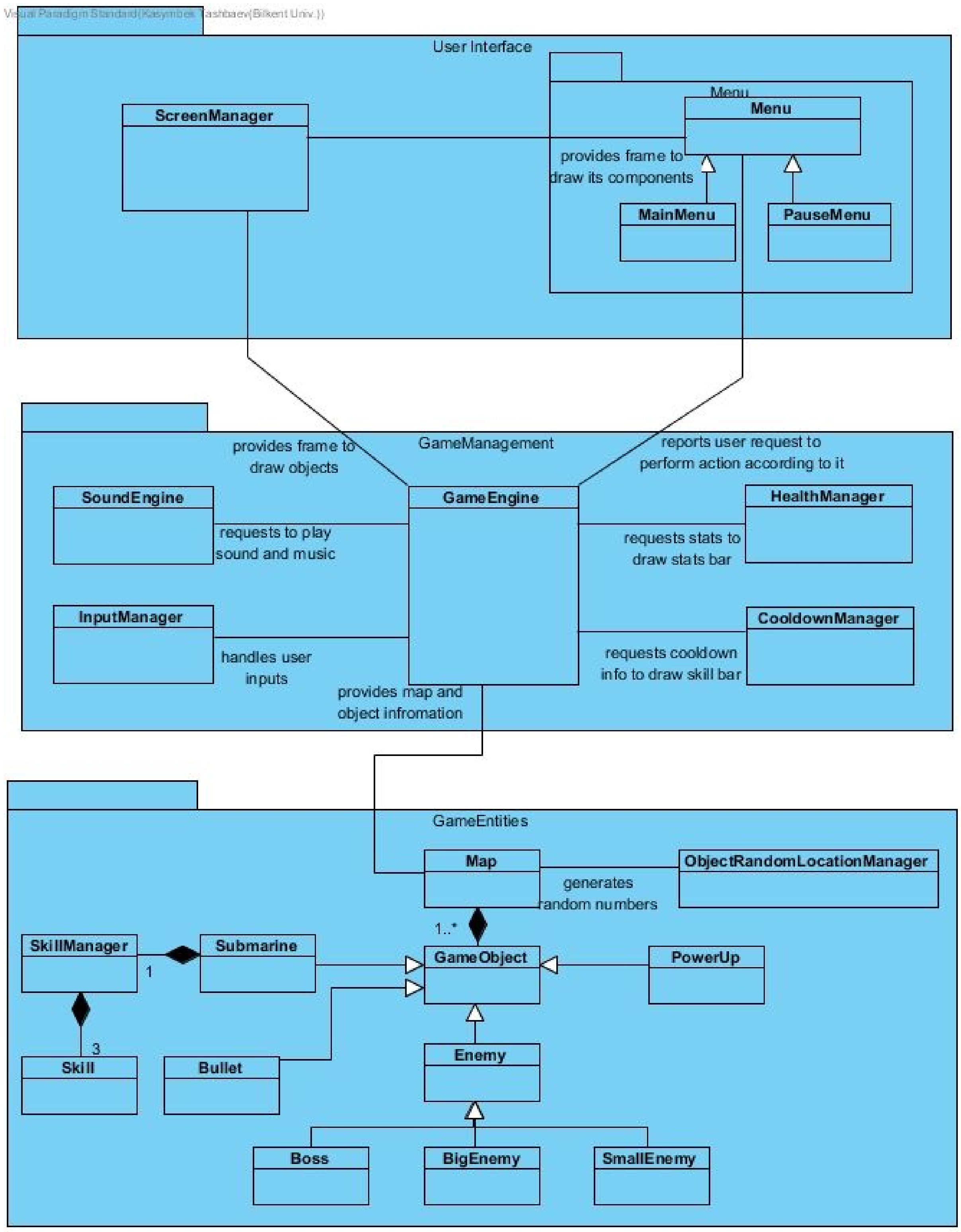
5. Design Patterns

Since our system is divided into subsystems and they communicate with each other, we developed a façade class. This design pattern provides a subsystem to communicate with other subsystems while we still can manage its control. In addition it helps us with extensibility, maintainability and reusability because we can change the specific subsystem through the façade class.

In our design, we created façade pattern for all three subsystems. In User Interface subsystem we use ScreenManager class as our façade class which communicates with the other components in User Interface subsystem based on the request of Game Management subsystem. In Game Management subsystem we use GameEngine as our façade class which communicates with the other components in Game Management subsystem based on the request of both User Interface and Game Objects subsystems. And in Game Entities subsystem Map is façade class and it interacts with Game Management.

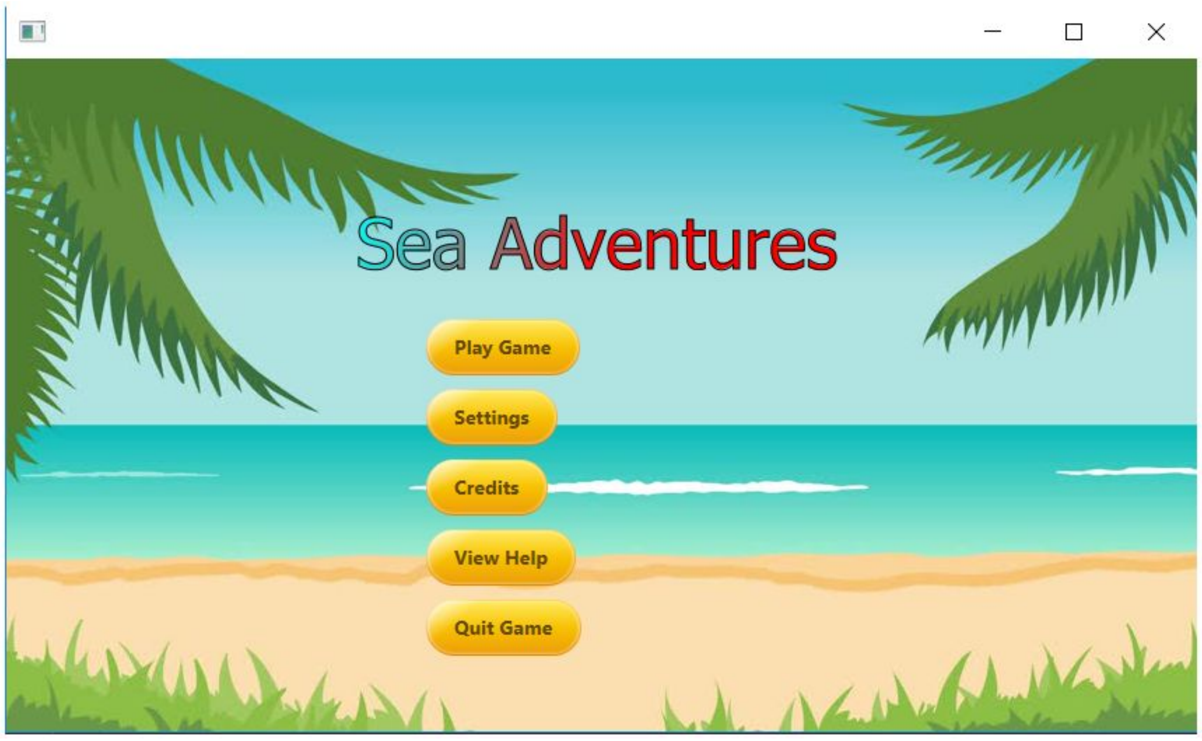
6. Detailed System Design

The high level design of the game:

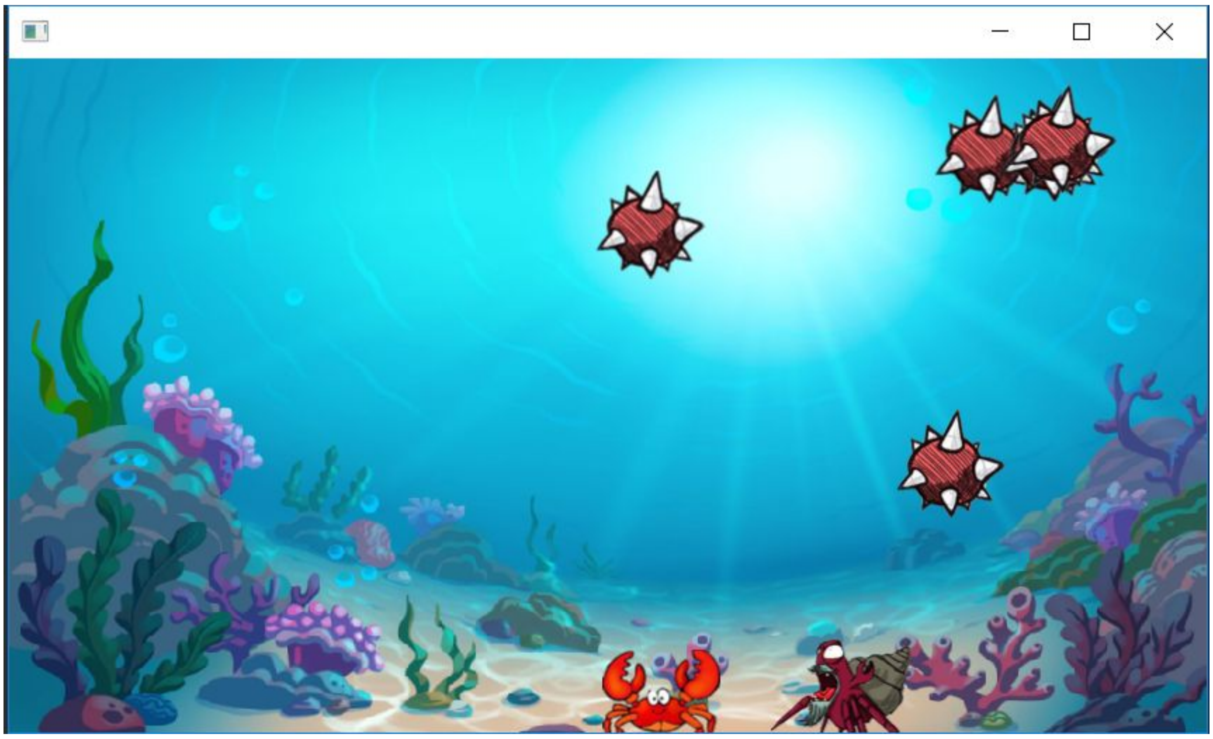


Visual Paradigm Standard(Kasymbek Tashbaev(Bilkent Univ.))
User Interface
Menu
ScreenManager
Menu
provides frame to draw its components
MainMenu
PauseMenu
GameManagement
provides frame to draw objects
reports user request to perform action according to it
SoundEngine
GameEngine
HealthManager
requests to play sound and music
requests stats to draw stats bar
InputManager
CooldownManager
handles user inputs
requests cooldown info to draw skill bar
provides map and object infromation
GameEntities
Map
ObjectRandomLocationManager
generates random numbers
1..*
SkillManager
Submarine
GameObject
PowerUp
1
3
Skill
Bullet
Enemy
Boss
BigEnemy
SmallEnemy

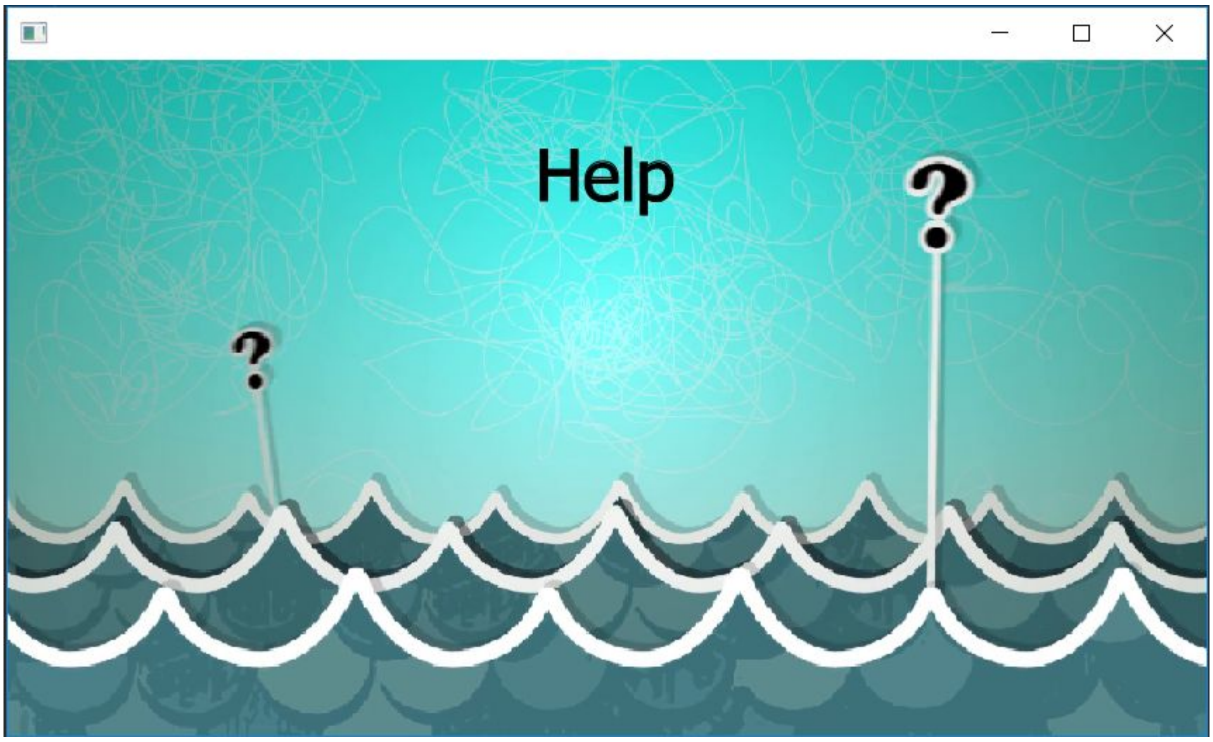
What Has Been Done



Beginning of the Game



The Gameplay and the enemies(not completed)



The help scene(not completed)

We have managed to complete the game logic and the game management. We have also implemented the game object class which helps us into creating other objects in the game such as enemies, power-ups etc.

What Needs To Be Done

We did not implement the submarine, which results in that we do not have collision manager whatsoever. In addition, we did not implement health, energy and the skills of the submarine. We also did not implement different maps, we just implemented first one. Until the end of the second iteration, these features need to be added.