Object Oriented Software Engineering Project

Design Report

CS 319 Project: Saving Humanity

Group 24

Can Bayraktar
Erdem Karaosmanoğlu
Mert Sebahattin Kutluca
Oğuzhan Karakahya

Course Instructor: Uğur DOĞRUSÖZ

Table of Contents

1.Introduction

1.1 Purpose Of The System

Saving Humanity is a game which we design to entertain people and cherish nostalgia to the ones that grew up with legendary atari game "Tank 1990". Our game is a remake of "Tank 1990" with some extra features. With that features our game promises more fun than its ancestor. Additionally with our multiplayer options, players can have fun with their friends in Saving Humanity.

1.2 Design Goals

Usability:

It is crucial for a user to feel comfortable in a game. Menu should help users to travel all features of game easily to attract the players. With "View Help" option you can see the button configurations of system. Help screen is shown while starting the game too. For Multiplayer Game options we will select the buttons so carefully to present comfortable setting for all of the four users in max.

Functionality and Performance:

Our main purpose for performance is a fluent gameplay. It is not hard to achieve because graphical requirements of game is low. Since it is a remake of old atari game, systems of today can easily run "Saving Humanity" at highest performance. Our purpose is to run the game with 120 FPS and smooth graphics. We will enable anti-aliasing for smooth graphics as we mentioned it in Analysis Report.

Good Documentation:

We aim to arrange our program well documented. We will try to provide JavaDoc documentation for all of the exposed classes and their exposed methods.

Extensibility:

Our game will be easy to extend because map files will be read from outside. So we can change map files easily. Even an editor program can be written to create new maps easily.

2. Software Architecture

2.1 Subsystem Decomposition

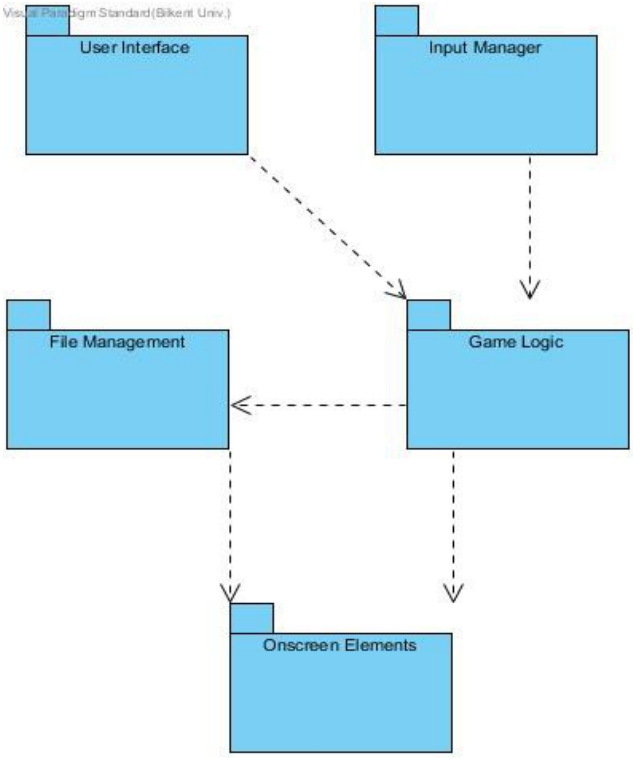

Figure 1

The architectural style of the system is a 3-Layer architectural style.It is also closed architecture(opaque layering).A layer's component can only communicate with the components in the same layer or the layer below.There are three hierarchically ordered layers. First one is the user interaction layer. It includes two components called “User Interface” and “Input Manager”.These components have classes which take inputs from user to initialize the game.
The second layer is the game logic layer. There are two components in this layer. “Game Logic” component handles all the game logic and mechanics with help from file management component. Finally, the third layer is called the “Display Layer” which represents all the entity objects of the game. All these object gets their information from the layer above.

2.2 Hardware/Software Mapping

Our project is going to be implemented in Java and because of that, the PC that is running the game is going to need the latest version of Java Runtime Environment. The game will require a keyboard to control the tanks and to enter a name whenever you beat a highscore in the game. Also to save the highscores we will a use .txt file which is supported in almost every operating system, therefore there isn’t any problem. The game won’t require a mouse and there won’t be any internet connection needed since our multiplayer is hot-seat.

2.3 Persistent Data Management

The disk is only used for storing game maps and keeping highscore file and the reads and writes are controlled by the program, meaning that there will be no concurrent writes. Therefore a simple file management system will satisfy our needs in term of data management instead of a database management system. Game maps will be stored as text files with predefined syntax and highscore file will contain name of players with their highscores. Map files will be read only while highscore file will be read and written consequtively by the program without any concurrency.

2.4 Access Control and Security

This game is played without any internet connection and there is only one type of user that is called as the player. As a result of that, we are not concerned with any security issues and also we do not need to implement and access control system.

2.5 Boundary Conditions

Initialization: Program needs to load proper images, sprites, map files and highscore file in order to run the game smoothly. When the user runs the executable file and start a game, all files will be read and game scene will be initialized.

Termination: Upon closing the game, each component related to the game will be terminated. Since there will be only one thread related to the program, it will be closed and program will exit.

Errors: Our aim is to provide best experince to users while playing the game, thus our main aim is to try recovering from errors as much as possible. However, if an error is crucial, such as failure of loading main game images, then game should be terminated. Thus, unless the error is not affecting the gameplay significantly, program will try to recover from erroneous situation.