Bilkent University

Department of Computer Engineering

CS 319 – Object Oriented Software Engineering

Traffic Master: Frogger Game

Final Iteration Final Report

Aurel Hoxha, Furkan Bacak, Nursenal Kal

Supervisor: Prof. Bora Güngören

Progress Report
Nov 04, 2017

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Object-Oriented Software Engineering Project, course CS319.

Table of Contents

1. Introduction

This report will talk about the implementation process of our Traffic Master game and will also include a how to play manual. Since our project is a game we will first give a brief description of our game in Section 2. Then, we will teach about how a player can play this game, talking about controls in Section 3. In Section 4 we will start talking about the implementation process; challenges that we have faced and how we have dealt with them, what changes have been made different from the design of the game and the current status of the implementation.

2. Game Description

Traffic Master is a Frogger game in which the player control a frog that moves upwards and downwards in traffic through different levels. In the roads and in the river, where the player can control the frog will be different objects that can hit the player and other object that the player must use to pass the frog on the other side of the river. The goal of the game is passing the frog to the top by avoiding being hit by cars and keeping the frog away from falling into the river. The game will consist of levels where the difficulty increases gradually. The score of the player will also be saved locally. There is also power-ups and power-downs that a user can encounter throughout the game which were explained in detail in our design report.

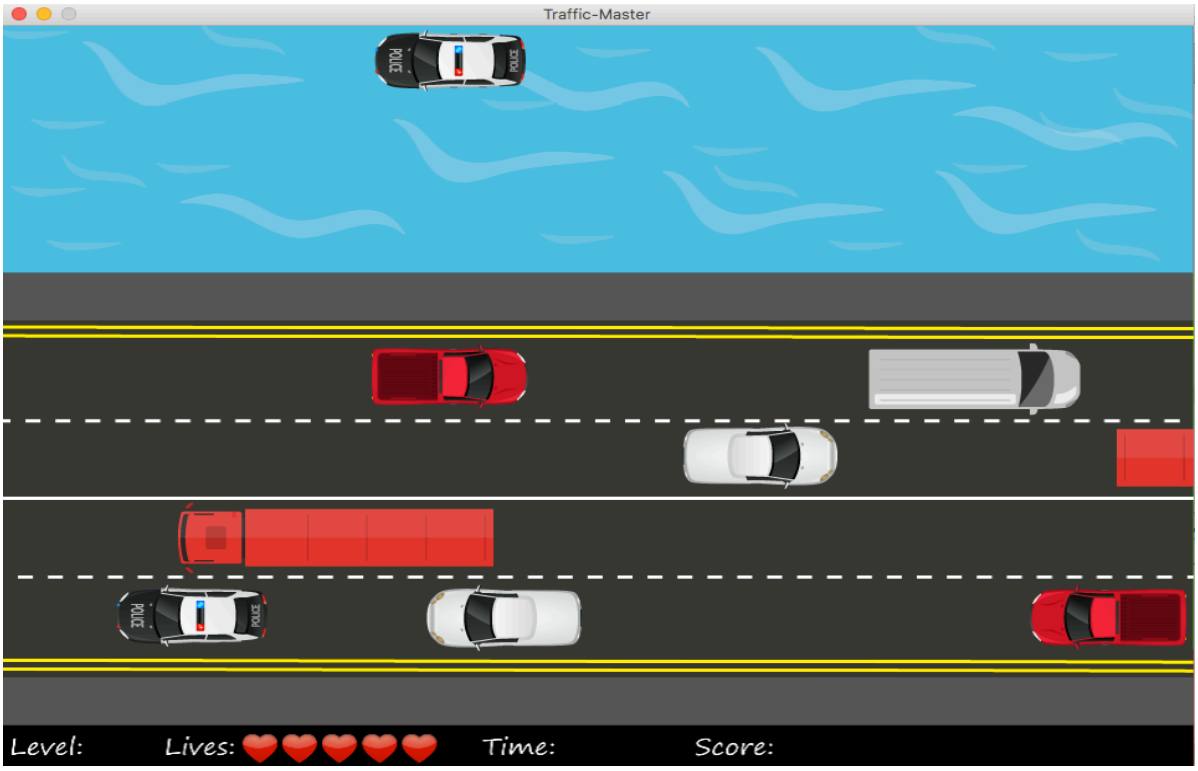
3. User Manual

3.1 Menu

This is the first screen the user will see upon launching the game.

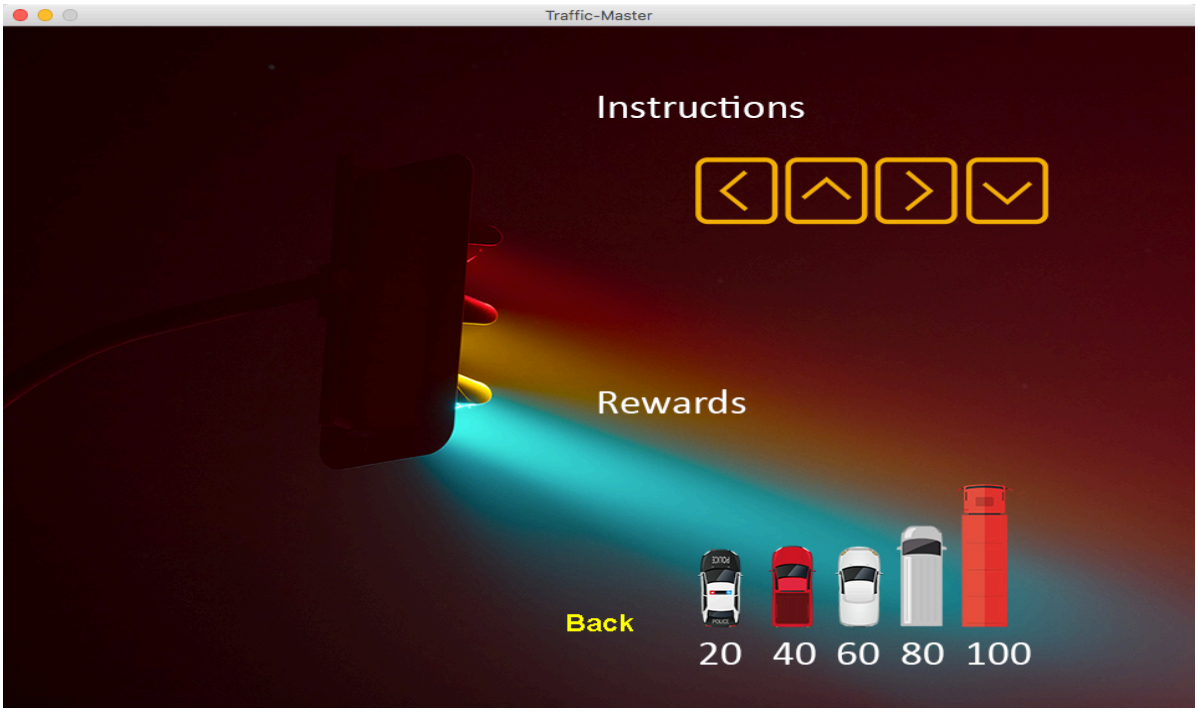
The user will simply be able to have 6 options. Clicking exit terminates the game window.

3.2 Play



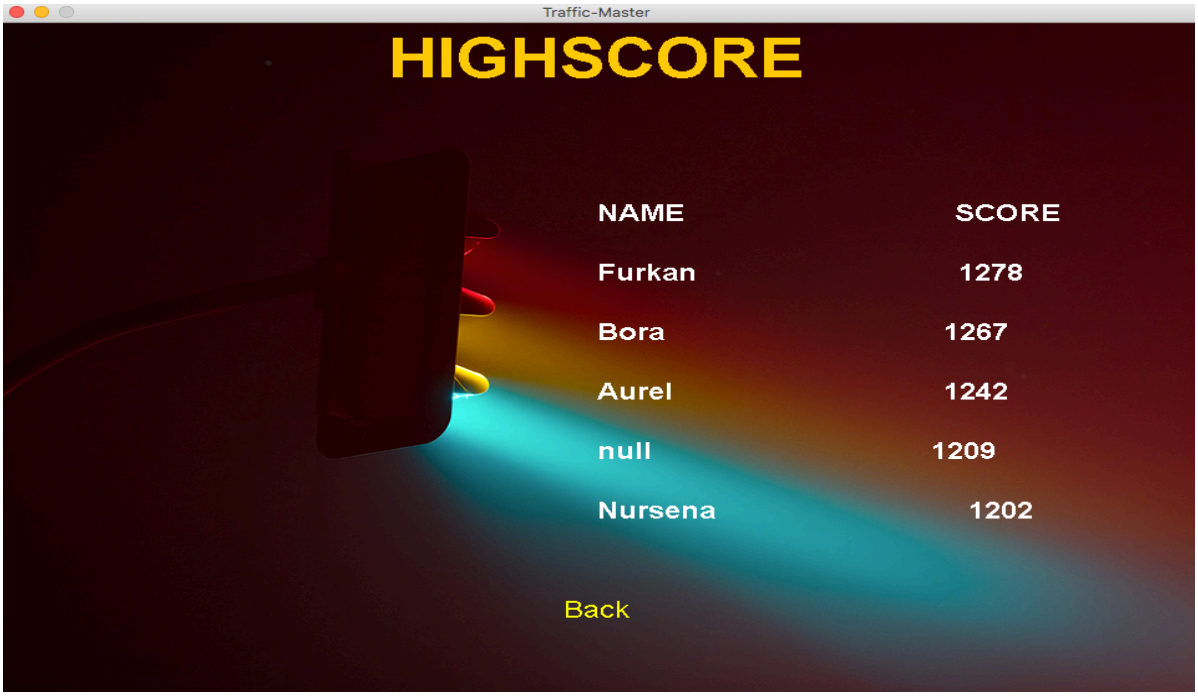
After clicking on Play button in Main Menu the user is led to the game screen under construction which is missing proper looks for the game entities. The user will be able to control the frog using the classic ⇦ ⇧ ⇩ ⇨ arrow button combination. The user must avoid the moving entities on the game board and reach to other side of the river as a main goal. In order to earn more points, the user should also try to finish the game as soon as possible with little or no help from the power-ups.

3.3 Help



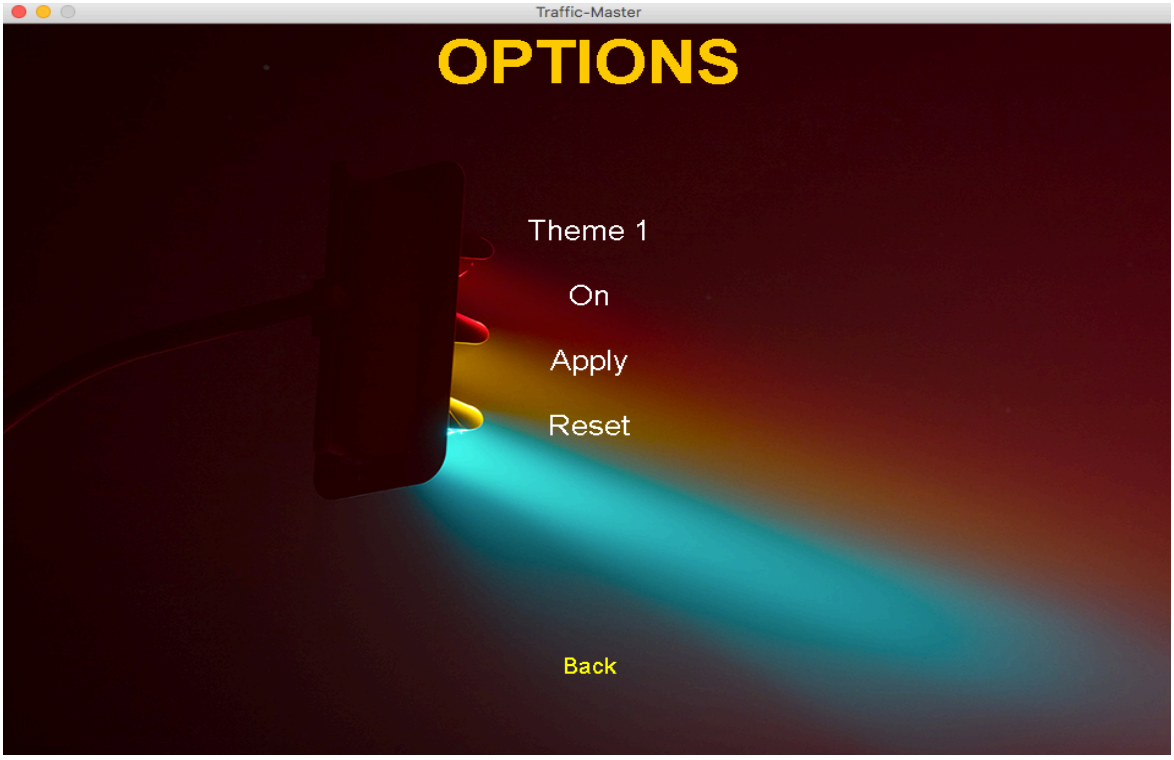
Upon clicking on Help button in Main Menu the user is led to this screen where he can see illustrations on game controls.

3.4 Highscore



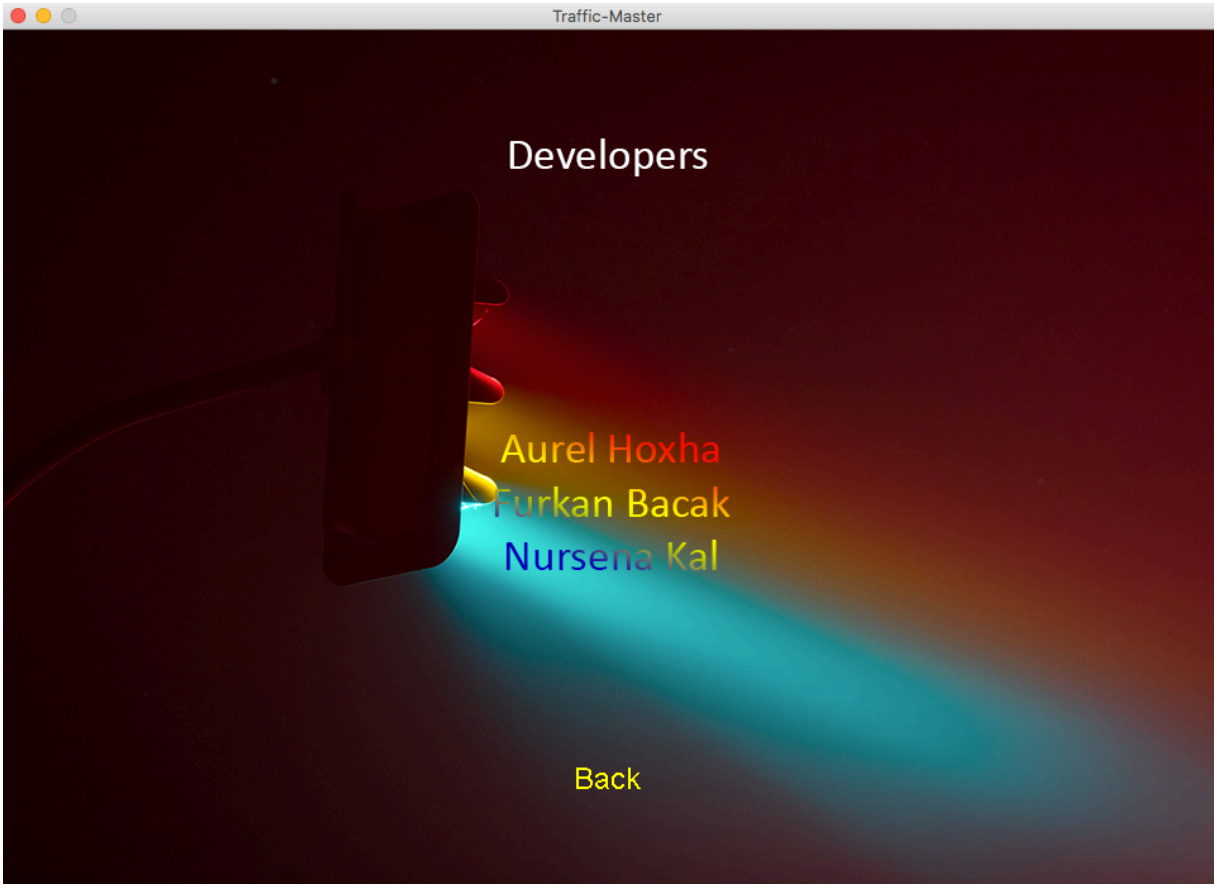
Upon clicking on High Scores button in Main Menu the user will be able to see previous high scores that any player has achieved playing on this particular device.

3.5 Options



Upon clicking on Options button in Main Menu the user can control the sound level of the game and also choose a different look for the frog.

3.6 Credits



This screen is dedicated for the developers of the game.

4. Implementation Process

4.1 Challenged Encountered During Implementation Process

The first and the most non-trivial part was understanding how the game is being developed and how the render works. The main class call the *"lunch()"*, *"init()"*, *"start()"* and *"stop()"* methods. The other part that was difficult was to understand how the sprite works and how all entities are being generated using the sprites and their particular properties. In order to make our game dynamic, which means that the states change over time we encountered a lot of problems. After seeing some online sources, we realized that we had to implement an infinite loop that updates the game objects and renders the scenes ideally at a rate 60 frames per seconds (fps). The problem that we also encountered was managing the inputs of the user and after several hours of trying to find we concluded that keeping the pressed button or part of the mouse into an array and checking whether this location is pressed or not would allow us for better performance.

4.2 Changes from Design to Implementation

In our first iteration design report we were missing class diagrams since we were not aware that they were also needed. Therefore, in terms of a close-up look to implementation design we did not have any change on classes. However, in terms of game functionalities and implementation approaches we had small differences between the design process and implementation process. Since we all took CS102 class and learned GUI components using Swing – AWT libraries, using them was our first idea. After doing some research on game development we also realized that JavaFX is Swing's successor and more user-friendly for game development. After careful consideration of the trade-offs between the libraries and asking about JavaFX in this course to senior students, we still decided to go for what is familiar for us which was Swing – AWT. In terms of game functionality, we were planning to increase the difficulty each time the user manages to complete a level. However, after realizing that increasing or decreasing some variables gradually did not add much of a complexity to the game, we decided to let power-ups and power-downs to take this role. None of the small differences mentioned above did make any difference between design and implementation processes on system decomposition. First of all we were not sure what libraries to use for game development.

4.3 Functionalities Not Implemented Yet

The implementation of the game is not yet complete, there are still power-ups and power-downs to be added. Although the game is functional at the moment in its current condition, it is missing proper object looks which we will be using Adobe Photoshop to create our own looks for our own objects. We are also missing the score system since this functionality is left to be done after completing all other functionalities and object looks of the game. We have also did not yet implement game music or sound effects to the game.

References:

[1]"Frogger Game" http://www.frogger.net/ [Accessed, 23 Sep 2017]

[2] Object - Oriented Software Engineering, Using UML, Patterns, and Java, 3rd Edition, by Bernd Bruegge and Allen H. Dutoit, Prentice - Hall, 2010, ISBN - 10: 0136066836. [Accessed, 24 Sep 2017]

[3] "JavaFX Game Development": https://gamedevelopment.tutsplus.com/tutorials/introduction-to-javafx-for-game-development--cms-23835 [Accessed 15 Dec 2017]