CS 319

Object-Oriented Software Engineering

Spring 2019

Final Report

Road Blocker: Alpha

Section 1 / Group J

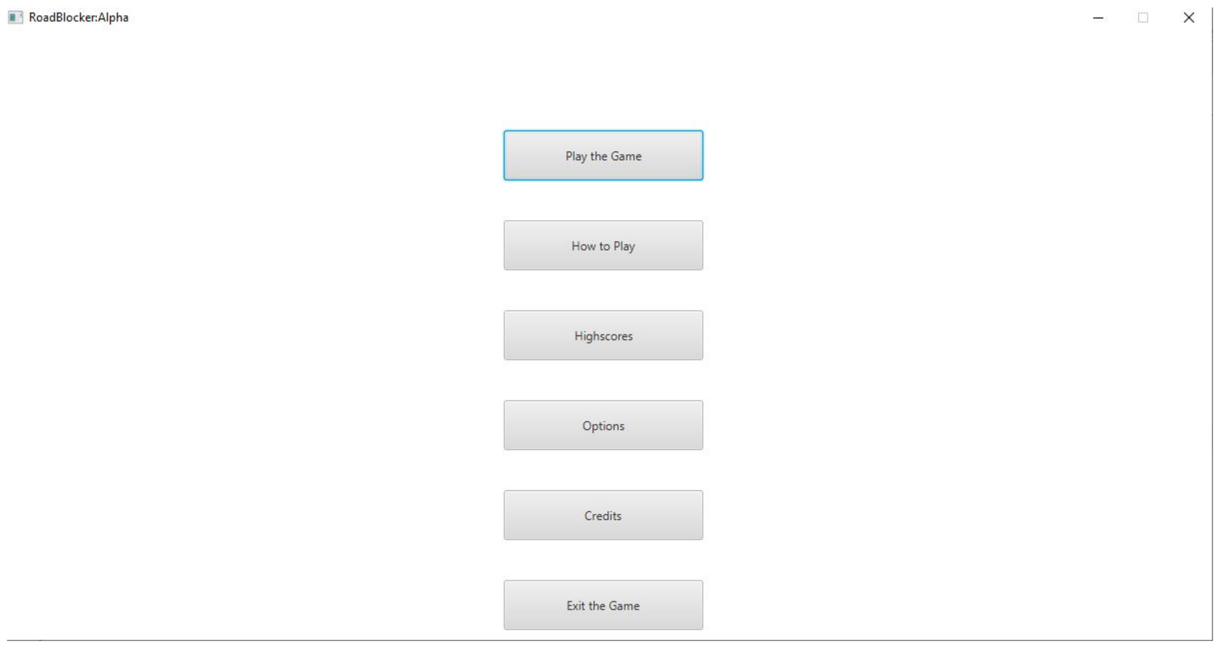
Mert Özerdem
Onur Mermer
Cemil Şişman
Burak Alaydın
Doruk Altan

Table of contents

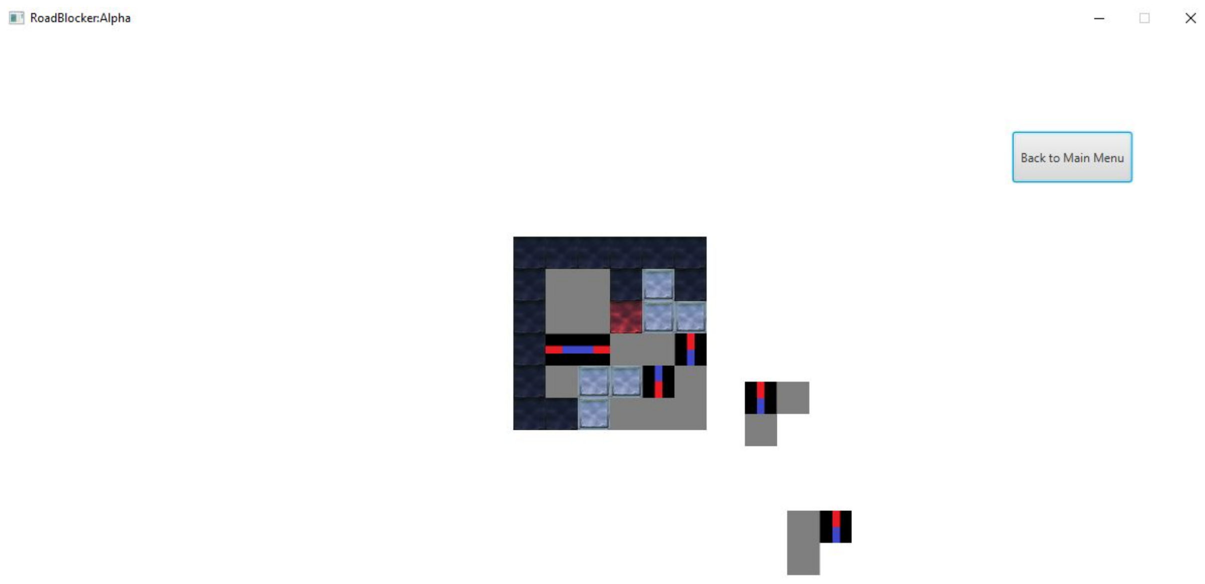
1. Introduction

We have implemented most of the object classes such as Blocks, Table, PoliceCars, Buildings, RedCar etc. Also we have implemented Game Engine class (excluding timer functions). GUI for our GamePlay menu. User can rotate,place and remove blocks. Settings, HighScores, Credits, How to Play etc. are not implemented yet. We have only implemented a sample level, that’s why Level class, File Manager class aren’t implemented yet. We also have not implemented sound and music classes yet. ScoreKeeper class is still not implemented.

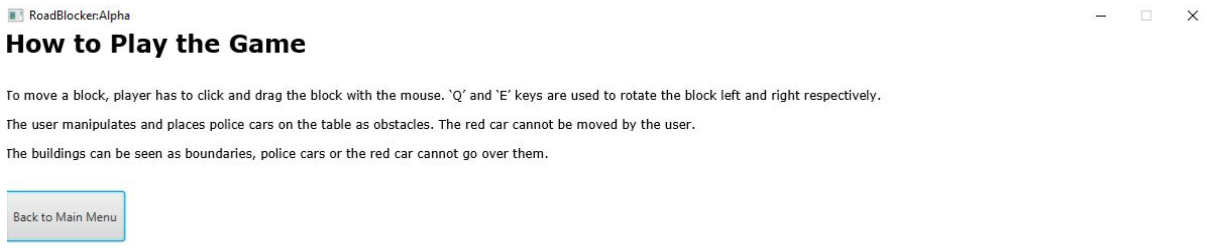
Main Menu:



The picture below shows the table with some police cars already placed and some to be used yet



The following pictures shows the remaining options of the main menu such as How to Play and Credits

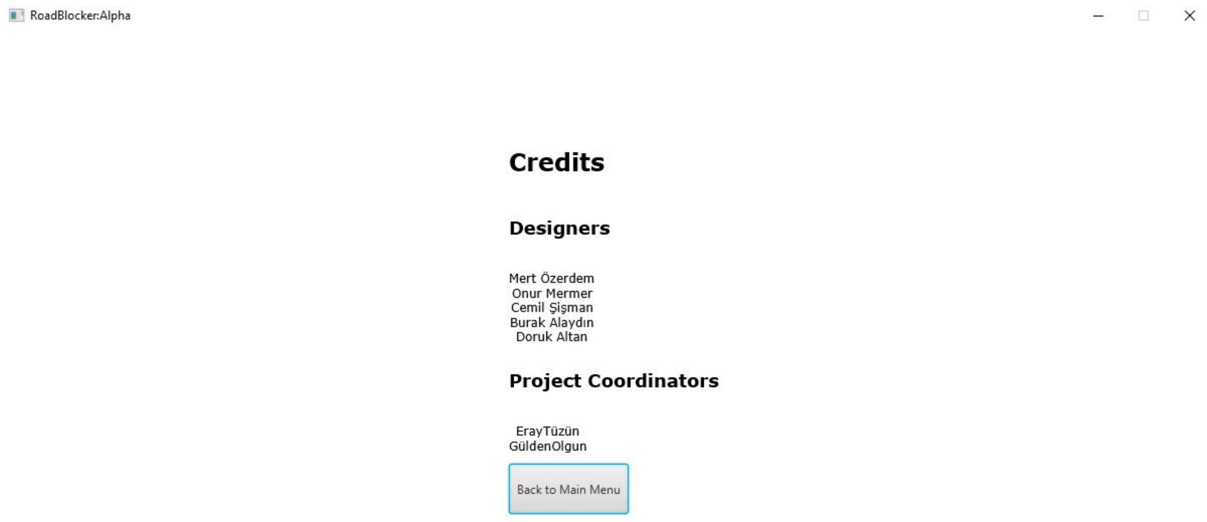

RoadBlocker:Alpha

How to Play the Game

To move a block, player has to click and drag the block with the mouse. 'Q' and 'E' keys are used to rotate the block left and right respectively.

The user manipulates and places police cars on the table as obstacles. The red car cannot be moved by the user.

The buildings can be seen as boundaries, police cars or the red car cannot go over them.


RoadBlocker:Alpha

Credits

Designers

Mert Özerdem
Onur Mermer
Cemil Şişman
Burak Alaydın
Doruk Altan

Project Coordinators

ErayTüzün
GüldenOlgun

2. Design Changes

We were thinking about adding a Show the Template button to our GamePlay screen (to show user the places of buildings and red car first so they can place them) then we decided to have them placed already at the beginning of each level. We also changed the

FileManager class so that the GameEngine class can extract the information of each level from a txt file.

3. Lessons Learnt

Weeks coming up to the deadline were extremely full for all members of the group so meeting in person regularly was a problem. Although, this situation led us to a point where we could not manage our time well and our work being disconnected, we still managed to communicate as a group through various online tools which made our heavy work in a relatively short time much easier. Our choice of Java as our language, though useful, also presented a few setbacks as we needed time to refamiliarize ourselves with the language.

4. User's Guide

4.1 System Requirements & Installation

Road Blocker is implemented in Java, therefore Java Runtime Environment (JRE) must be installed to play the game. If JRE is not installed, you can download it from https://www.oracle.com/technetwork/java/javase/downloads/index.html . Once JRE is installed, Road Blocker can be opened by running its jar file or by compiling and running the code in a Java IDE.

4.2 How to Use

When a player runs the game, he will be directed to the main menu screen. The "New Game" option starts the game. According to the level, a table will be displayed on screen with a red car and building. Beside the table will be blocks with police cars which the user can rotate and place on the grid in such a way that the red car cannot escape. As player completes available levels, more difficult ones will be unlocked.

4.3 Controls & Objects

To move a block, player has to click and drag the block with the mouse. ‘Q’ and ‘E’ keys are used to rotate the block left and right respectively. The user manipulates and places police cars on the table as obstacles. The red car cannot be moved by the user. The buildings can be seen as boundaries/obstacles, police cars or the red car cannot go over them.

5. Step-by-Step Build Instructions

We have four main parts in our implementation:

- GameEngine class which will be the executing the actions of user that is passed by GUI and perform the game logic.
- Object Package which includes all of the objects and tools that we will use in our game.
- Sound Package which includes our sound effects that will be played after user’s actions and the background music that will be played in our game.
- GUI which will display all the visuals and interfaces of our game.

5.1 Game Engine

GameEngine package has two classes: MainEngine and LevelManager. LevelManager class is used to create corresponding levels by using an external file called Level. This file has all the information about the levels such as the table size, block types, car types, building types etc. MainEngine class is where the logical side of the game is executed. All of the user inputs are passed to this class and executed. MainEngine class makes the necessary changes and does all of the checkings. Then it passes the information to the GUI to display them visually.

5.2 Object Package

This package includes all the building, car, table and level informations. The block types that will have the cars and buildings will be on are created in BlockType class. There are four building types and four corresponding building classes that specifies the types such as: BuildingA, BuildingB, BuildingC and BuildingD these classes extends the Buildings class. Buildings class extends the Blocks class as well because the Buildings objects will be our blocks. Then we have four police car classes such as CarA, CarB, CarC and CarD which extends PoliceCar class that also extends Blocks to be able to create distinct blocks with specified cars on them. We also have a RedCar class for the thief car which is trying to escape. This class directly extends the Blocks class because it is a single block with a red car on it.

5.3 Sound Package

This package includes our Sounds class and SoundManager class. We store the urls of our background music and the sound effects such as mouse click sound and collision in the Sounds class. Then in the SoundManager class, we have a method to play the corresponding needed sound.

5.4 GUI

We have MainApp class to display all the visuals and interact with the user. This class will create the visual components of our table, blocks, cars and buildings by taking the necessary data from the GameEngine class. It will also receive the actions of user and pass it to the GameEngine class to process our gameplay instructions.

6. Work Allocation

Burak Alaydın:

- Analysis Report: Non-Functional Requirements, sequence diagram.
- Design Report: Subsystem Services.
- Presentation.
- Code: ScoreKeeper/ScoreBoard, also helped GameEngine.

Mert Özerdem:

- Analysis Report: Introduction, Use Case models, Object Class Models, Improvement Summary, Revision
- Design Report: Subsystem Services, Final Object Design, Revision
- Code: Game logic, Engine Subsystem, Level Subsystem

Cemil Şişman:

- Analysis Report: Drawing Mock-ups and Navigational Path, Revision of Sequence diagram in 2nd iteration.
- Design Report: High Level Software Architecture, Hardware/Software Mapping, Persistent Data Management, Access Control and Security, Boundary Conditions.
- Presentation
- Final Report: Step-by-Step Build instructions, Introduction and Design Changes
- Code: Sound Package and Level class.
- Video Trailer

Onur Mermer:

- Analysis Report: Functional Requirements
- Design Report: Introduction, Design Goals
- Presentation
- Code: GUI package, GameEngine/LevelManager
- Video Trailer recorded

Doruk Altan:

- Analysis Report: Dynamic Models(Activity, Sequence)
- Design Report: Subsystem Decomposition
- Presentation
- Level Design