CS 319 - Object Oriented Software Engineering

System Final Report

Iteration 2

Road Block

Group 1A

Denizhan Soydaş
Deniz Ufuk Düzgün
Kaan Atakan Öztürk
Cavid Gayıblı
Selimcan Gülsever
Serkan Delil

Contents Table

1. Introduction

Road Block is a game that we are planning to develop. The main objectives and game play of it are as follows: There is a thief in the game that is trying to escape from the police. The user's main objective is to block the paths which the thief may use in order to get away. The game is played on a map that offers 36(6x6) squares. This map offers various spaces in which the user can place differently shaped blocks and the puzzle in it is to place all the blocks where they have to be. These blocks are offered to the user at a seperate place from the game board and they can be chosen and placed in a random order as long as they are in the places where they have to be in the end. User is also able to, and usually should, rotate these blocks in order to find the solution for that level. The game consists of different levels which increase in difficulty after each one is completed.

Additionally, we are planning to develop the traditional game a few steps further and make some adjustments to it. For instance, since it is a single player game, we wanted to add some self competition and we will be using time limitation for the levels. Again for enhancing the same feeling , we will get involved with score tracking. The player will obtain scores based on the parameter time,Also we will add one element called "park" in order to make some levels a bit harder.In addition, we have elements called “Thief’s House” and “Tunnel”. They both work for the same purpose: increasing difficulty hence entertainment. Detailed information about all of the features can be found in Overview section.

Our aim is to implement this game as a desktop application in Java programming language while abiding by the principles of Object Oriented

Programming. This report offers game overview, our in-game objects and structural information of the application. In this report it is also possible to find functional, non-functional requirements along with the class, use case, state and activity diagrams.

2. Overview

2.1. Gameplay

The game is played using only the mouse. Using the mouse, the user is both able to rotate and move the blocks to the places on the map. Simple left click operations will be sufficient to choose the desired rotation of blocks and place them to the desired spots.

2.2. Map

The game is played on a 2D 6x6 map on which involves the components of the game. The puzzle of our game is the different combinational layouts of the game components on the map and it is the main environment on which the gameplay is conducted.

2.3. The Thief

The thief is the villain of the game and user's main objective is to prevent the thief from escaping the map by manipulating and using different Police Car Blocks.

2.4. Police Car Blocks

These components are the main pieces of the game which are in user's control to place in the empty areas on the map in order to solve that level's puzzle. The user is able to rotate these blocks clockwise or counter-clockwise to fit them to the right places.

2.5. Buildings

These components are placed in different orders on the map and they are intact components, meaning the user is not able to move or manipulate them whatsoever. Both the Police Cars and The Thief is not able to move through these blocks. In other words, they are used to borderline the playing area of that specific level.

2.6. Park

This is one of the additions that we have added to the traditional game. It serves as a easy getaway path to the thief and is planned to make certain levels more difficult for the user.

2.7. The Clock

The other additional component we will be adding to the traditional game is "The Clock". The Clock serves as a self competition component for the user to earn score points accordingly.

Meaning: The faster the user completes the level the more points he/she will earn.

2.8. High Score and Time

Each level will have its own high score and it will be displayed on the game screen while the user is playing that level. The high score will be calculated based on how fast the user finishes that level. The mentioned speed will be calculated by the “The Clock” component.

2.9. Thief’s House

Additionally to the traditional game, we will be implementing a safe house for the thief. If the thief reaches this house, the game will be lost just as in scenario of the thief escaping the map.

2.10. Tunnel

Tunnel will also be an additional component to the traditional game that we implement. It will work as a teleportation point for the thief. The thief will be able to move across the two tunnel symbols on the map.

3. Functional requirements

3.1. Play Game

3.1.1. Select Difficulty

The game will be consist of 3 difficulty levels : Easy, Medium, Hard. Each of these difficulties offer 10 games to the user : Level 1, Level 2, Level 3…… Level 10.

3.2. Settings

The user will be able set their preferences for the in-game effects voice level and music volume.

3.3. About Us

This section will allow user to view information about the developers that participated in creation of the game and also about the references that were used to obtain useful information during the development of the game.

3.4. Help

Through this option, it is possible to learn about the objective of the game and how to play it. The instruction manual(text) will be provided via “Help” button.

3.5. Exit Game

“Exit Game” button is provided for the user to be able to exit and close the game when desired.

4. Non-functional Requirements

4.1. Game Performance

Since our game is a simple Java application it does not require relatively high system specifications :

RAM: 128 MB;

Disk Space : 56 MB

Graphics Card : 64MB RAM or higher

Windows operating system is necessary to run our application.

4.2. User Friendly

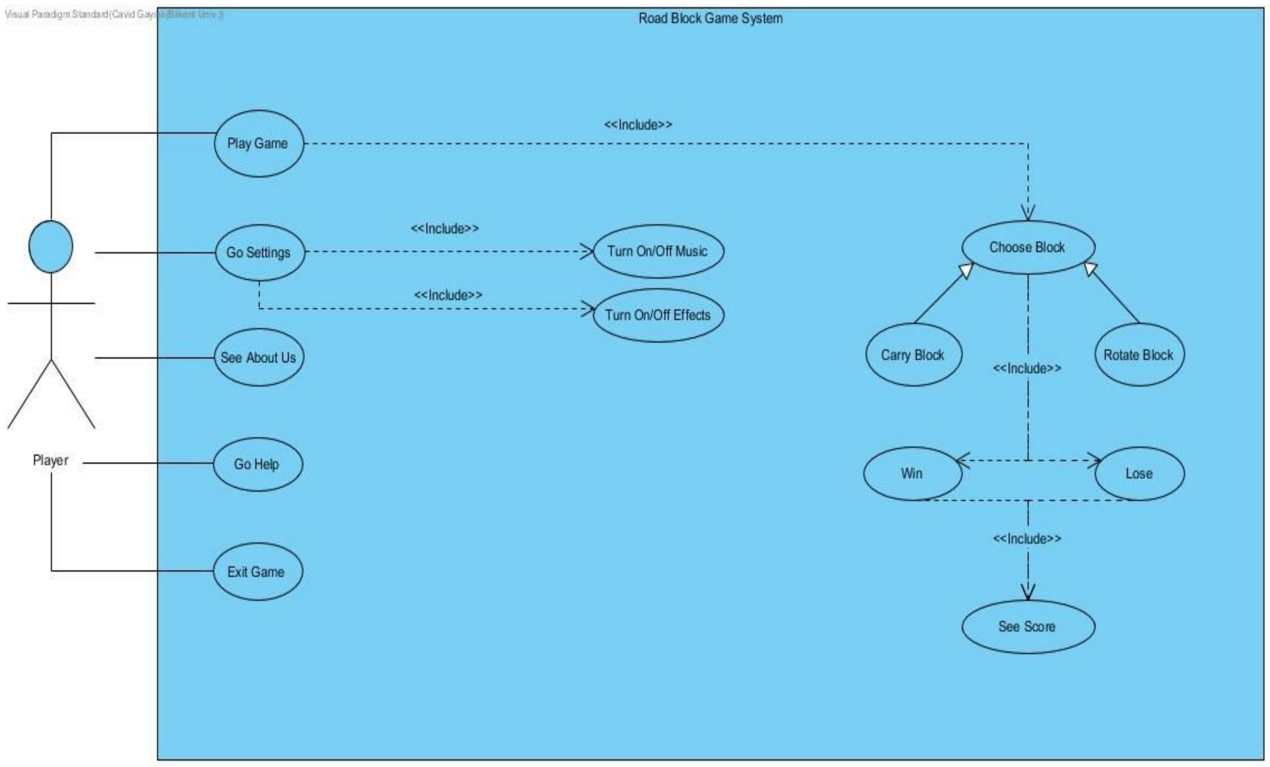
Our game's traditional version is recommended for 7 years or older old users. Therefore our application's features are designed to be easily understandable for above 7 years old users. We will have children users along with the adults to test our game in order to see whether they can comprehend the UI’s difficulty or not.

4.3. Response Time

The game should be able to respond to the user input almost instantly. Therefore, our images have to update as in the instant that user provides changes. After we have calculated, all of our game's actions' response time is 40-50 ms. By this way, a smooth gameplay experience is obtained.

5. System models

5.1. Use case model



Use Case For Play Game

Use case name: PlayGame

Participating actor/s: Player

Entry condition: Player should open the main menu by clicking the “PLAY” button.

Exit condition:

1. Player has to pass the current level by carrying blocks OR,

2. Player has to exit the game by clicking the exit button.

Main Flow of Events:

1. Player clicks the "Play Game" Button

2. System comes up with selection page that asks if player wants to select the difficulty or return back to menu.

3. Player plays the game and returns to main menu at the time player wants.

Alternative Flow of Event:

1. When the player wins and passes the current level, the player’s score will be shown as stars (out of 5).

2. Player can exit the game any time.

Use Case For Choose Block

Use case name: Choose Block

Participating actor/s: Player

Entry condition: Player should click the Play Game button and there will be a map and some blocks to be chosen.

Exit condition:

1. Player has to click the “Return Menu” button OR,

2. Player has to click the exit button at the right top of the screen.

Main Flow of Events:

1. Player chooses blocks in order to place them into the map.

2. Player can either carry the block without rotating it, or first rotating it and then carrying it in to the map.

Use Case For Win

Use case name: Win

Participating actors: Player

Entry condition: Player has to finish the game successfully.

Exit condition:

1.Player has to click return menu button OR,

2. Player has to click the exit button at the right top of the screen.

Main Flow of Events:

1. Player finishes the game successfully.

2. Score will be displayed depending on the time.

Use Case For Lose

Use case name: Lose

Participating actors: Player

Entry condition: Player has to finish the game unsuccessfully.

Exit condition:

1.Player has to click return menu button OR,

2. Player has to click the exit button at the right top of the screen.

Main Flow of Events:

1. Player finishes the game unsuccessfully.

2. Score will be displayed (in this case, score will be always 0).

Use Case For See Score

Use case name: See Score

Participating actors: Player

Entry condition: Player should finish the game.

Exit condition:

1.Player has to click return menu button OR,

2. Player has to click the exit button at the right top of the screen.

Main Flow of Events:

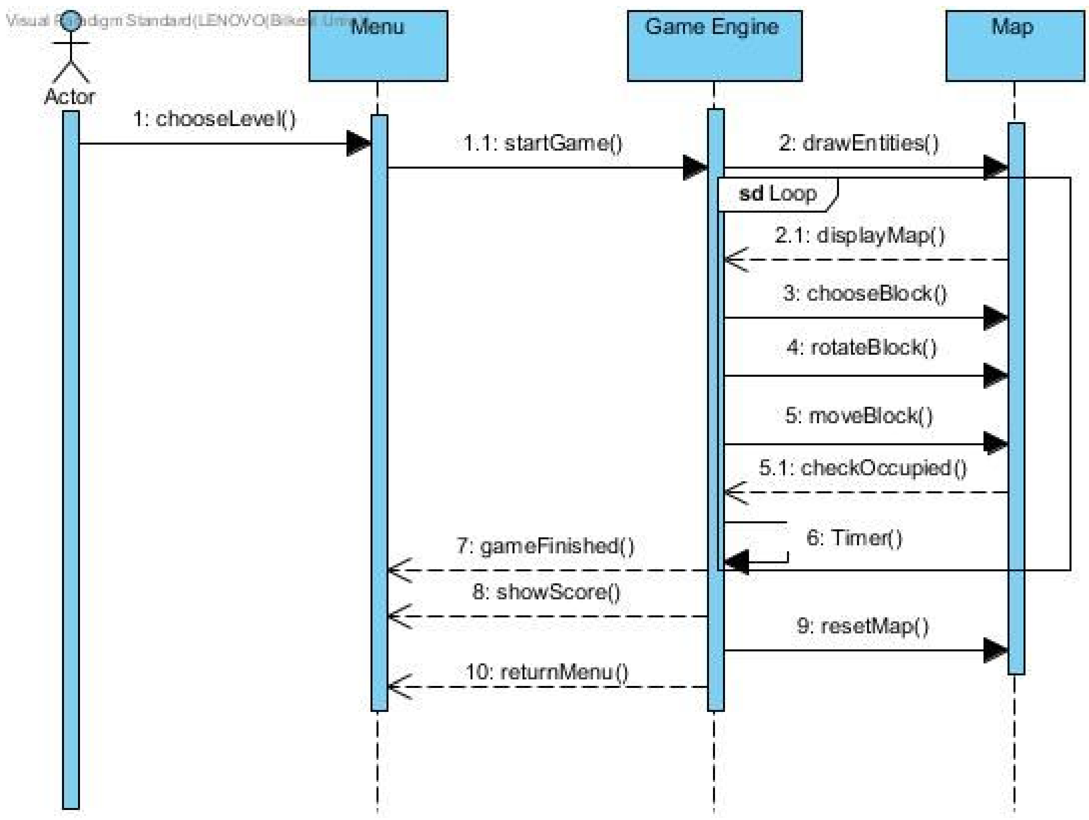
1. Game is finished whether successfully or unsuccessfully.

2. Score will be 0 if the game is finished unsuccessfully (lose).

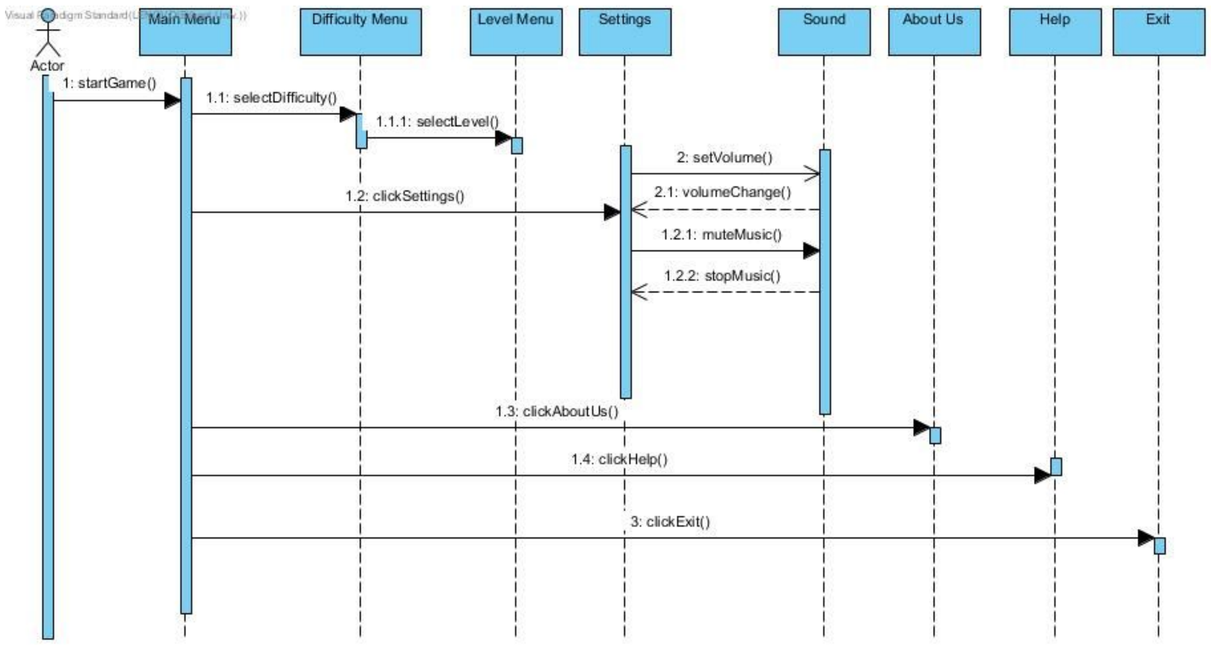
3. Score will be calculated depending on the time if the game is finished successfully (high score will be depending on less time, low score will be depending on much time).

5.2. Dynamic models

5.2.1. Sequence Diagram For Game Play and Interfaces

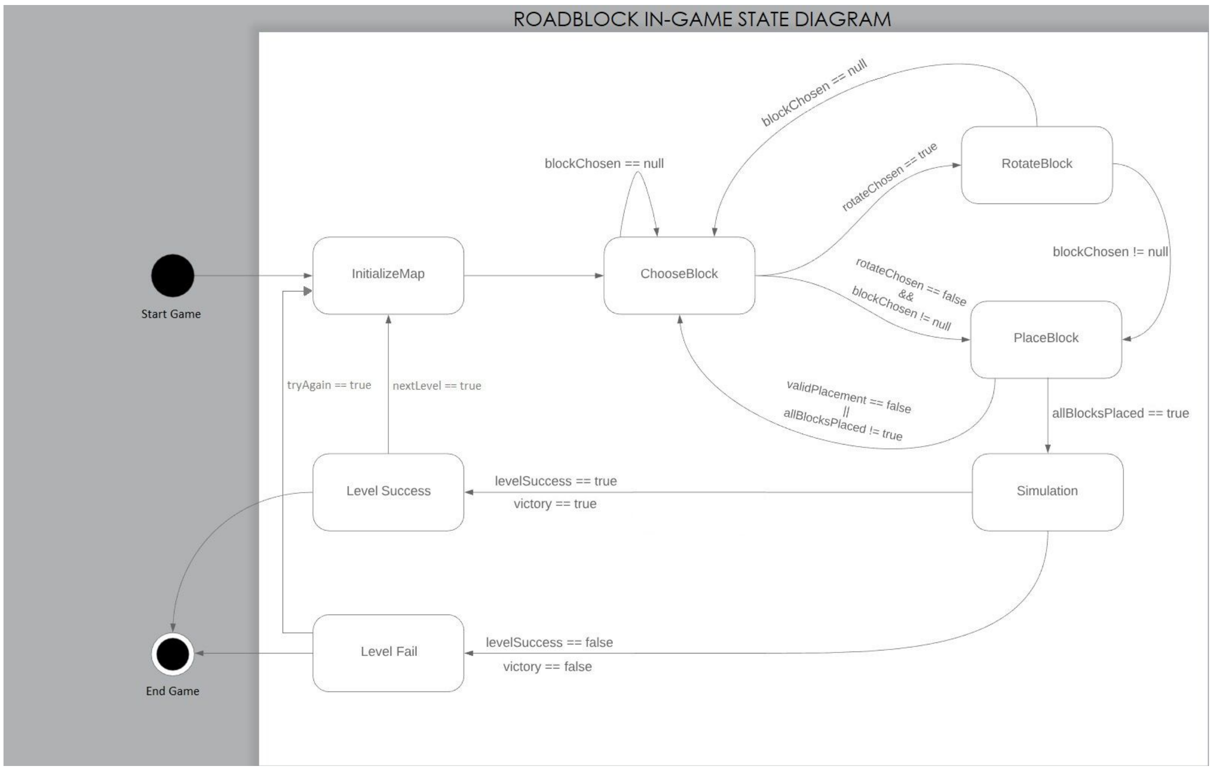


In this diagram, basically how the application works through classes and how they interact between them. All these processes are controlled by Game Engine and Map. Moreover, it will interact with the other classes when it is needed. Firstly, Game Engine first creates the map and entities in terms of input information, difficulty and level that user want to play. Manager recursively takes the inputs from mouse and according to inputs it will interact with entities. After putting entities, map will display game screen and according to mouse actions of users, blocks will be chosen, rotated and moved by Game Engine. When these chosen block moves, Map will check whether the destination position of the block is free or not. Secondly, Game Engine will count the time to calculate score of player. These processes will be repeatedly occur while Game Engine decides whether game is finished. If game is finished, the score of player will be shown. Furthermore, user can return to menu or restart game by clicking corresponding buttons.



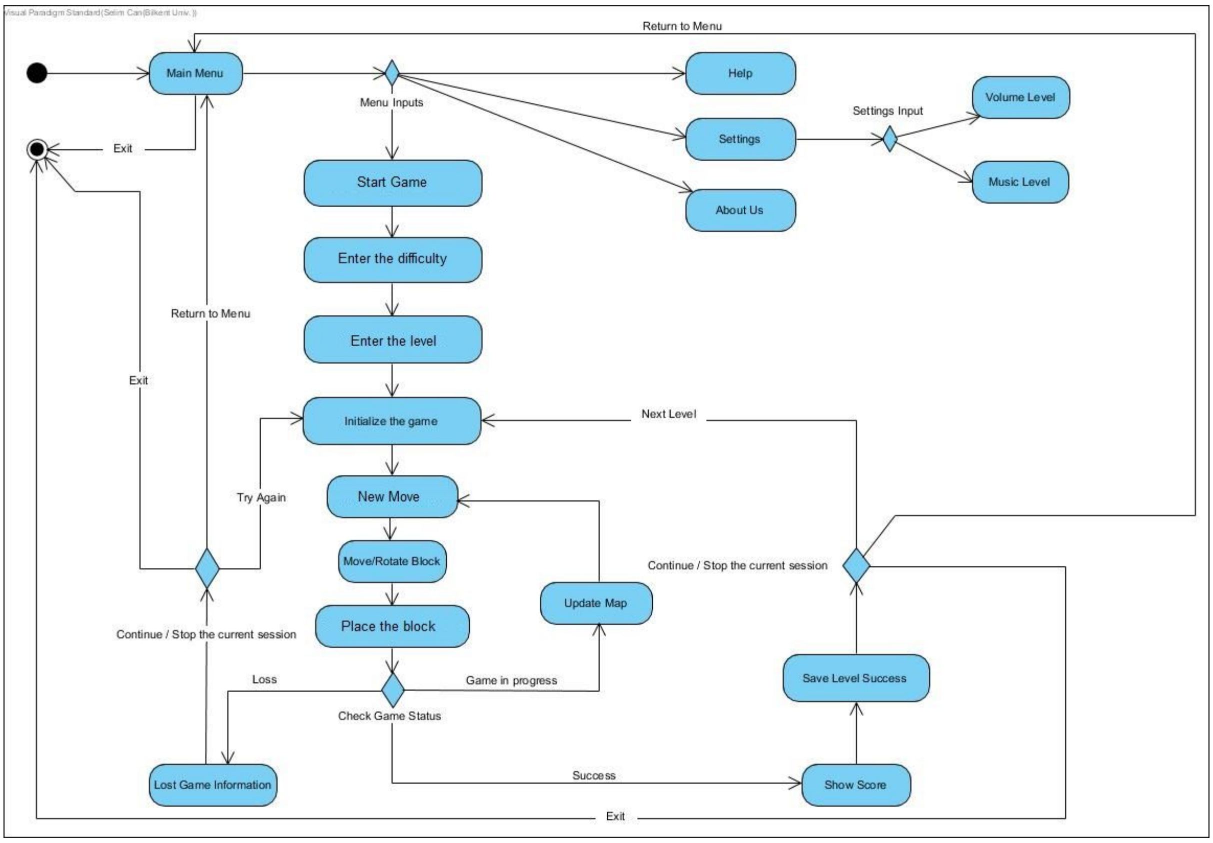
This diagram shows that how main menu interacts with other menus and screens. Firstly, actors interact with main menu when they start the game. Then, they select the difficulty of the game, after they determine which difficulty they will play, they need to choose level in that difficulty. Users go to the setting from main menu. What they can do in setting menu, is shown in diagram. They can change volume of attempts in the game, turn off /on background music . Furthermore, players can see informations about creaters of the game and learn how to play with the help of help screen. Lastly, user can quit the game by clicking exit button. In addition, user can quit by closing the window.

5.2.2. Statechart Diagram



In our game, when user enters the game through GUI, the system initializes the map. Then, there is a choose block statement that is for choosing which block to place. Then, when the user chooses the block that will be placed, the program waits for the user to rotate the block or place it. When the user chooses to rotate the block, the state changes to “RotateBlock” and waits for the user to place or drop it. Unless all blocks are placed, the system return to “ChooseBlock” state and expects for a new block placement. If all blocks are placed, the current state goes to a “Simulation” state. In this state, the program decides whether the current game is a victory or not. If the thief can escape from the current map, the user loses the current level and the program returns either to the current map’s first initialization, by trying again or to the main menu. If the user wins, by placing the blocks properly to stop the thief, (s)he can either continue with the next level or exit to the main menu.

5.2.3. Activity Diagram



The activity diagram represents the movements, inputs and the game management when interacting with the user. It is a proper way to introduce to the user how to play or how to use the application if the person has no information about the program.

The initial start of the application is the opening of the program and a little after, there is a menu which is to start the game or in order to help the user, there are information and settings about the game itself. If the user wants to start a level, then program waits for an input from the user to take an action.

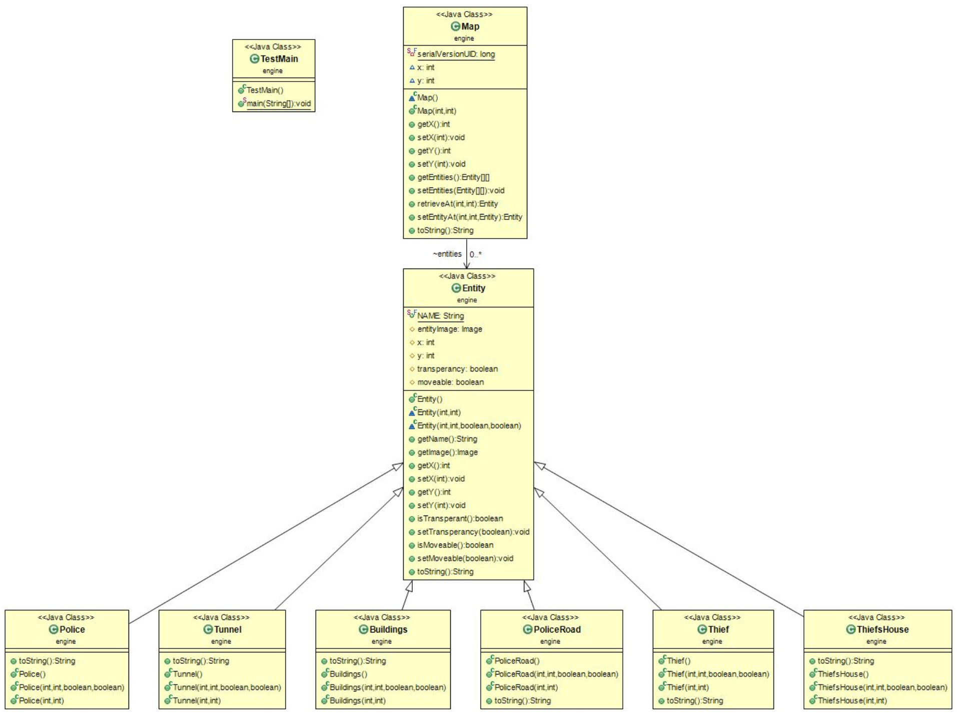
When the input needed to start the game is taken from the user, there are certain decisions which are mandatory to be chosen such as difficulty of the game and current level phase of the exact difficulty. When the requirements are satisfied, the user can start a new level.

During the level, the game itself, player has no restrictions such as move counts or health, but a clock that is independent from the player's moves. The blocks can be placed or taken from the location they are found, or they can be rotated to block the available roads. Every active movement of the player, meaning that if the person places a block other than taking or replacing it, will be checked by the game management to decide whether the game is finished. If it is not, the block that is chosen will be placed to the desired location if the location is available.

When it comes to the end of the game, there are two ways of exit: one wins or loses. Lost can occur when the time given to the player ends. In this case, the player can try again the exact level or return to menu to choose another difficulty or level.One of the exits in the game is in the loss menu when a player loses the current game. If the player wins, by placing the correct blocks, then the player can continue to the next level or return to the menu to change the current game if there is need for more challenging levels. If the person doesn't want to continue to the game, another exit in the game is in this menu.

5.3. Object and class model

Below is the object and class model of the game. The mechanics of the game will depend on this model.



Classes:

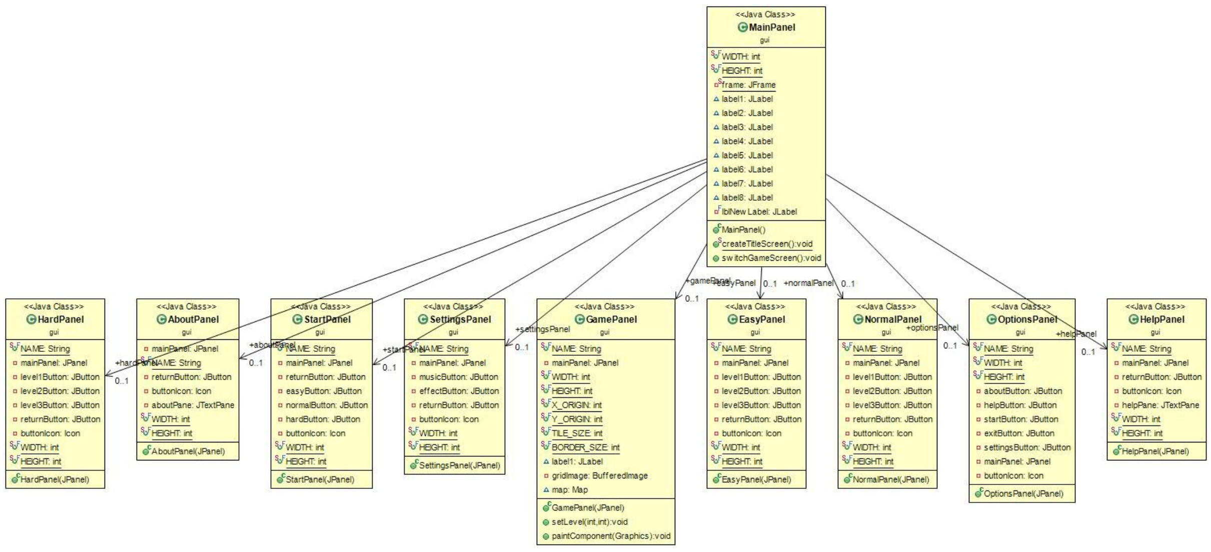
- Level : Its instance hold the current the current level, a map, and scores etc.
- Map : A map holds each 2D world object and their place on the 2D World.
- Entity : Objects of 2D world.
    - Police : A 2D object that represents a police of real world.
    - Tunnel :A 2D object that represents a tunnel of real world.
    - Buildings: A 2D object that represents a home of real world.
    - PoliceRoad : A 2D object that will move with the police in order to make the square non-empty.
    - Thief : A 2D object that represents a thief of real world.
    - ThiefsHouse : A 2D object that represents a house that belongs to a thief in the real world.

We are going to implement our game through the Level Class. When the game starts, the system initializes the level and associated other objects in it.

Every Object Class that occupies a space in this 2D World is a subclass of the Class Entity. They may be transparent or not. Transparent object such as Tunnel, lets the Thief pass. Others does not.

In each level, there will be a [x][y] Map that holds every entity object.

Below is the object and class diagram of outside of the game. To adjust the settings of the game and to initialize the game, we're going the use a GUI based on this object and class diagram.



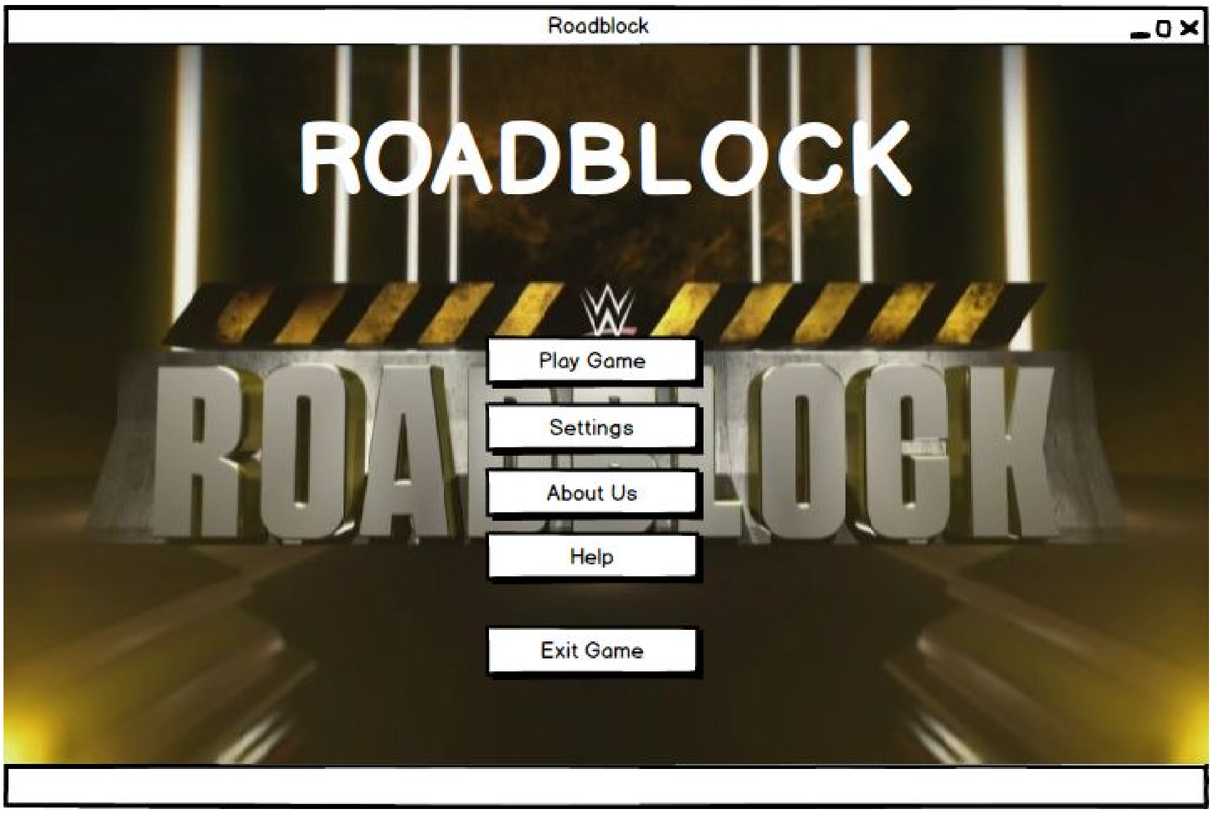
The main panel will be the initial frame. It will stay until the program exits. Each other panel will come to front when they are called through card layout.

We are going to use SingleTon Pattern on Main Panel in order to not lose the access from these panels to the Main Panel.

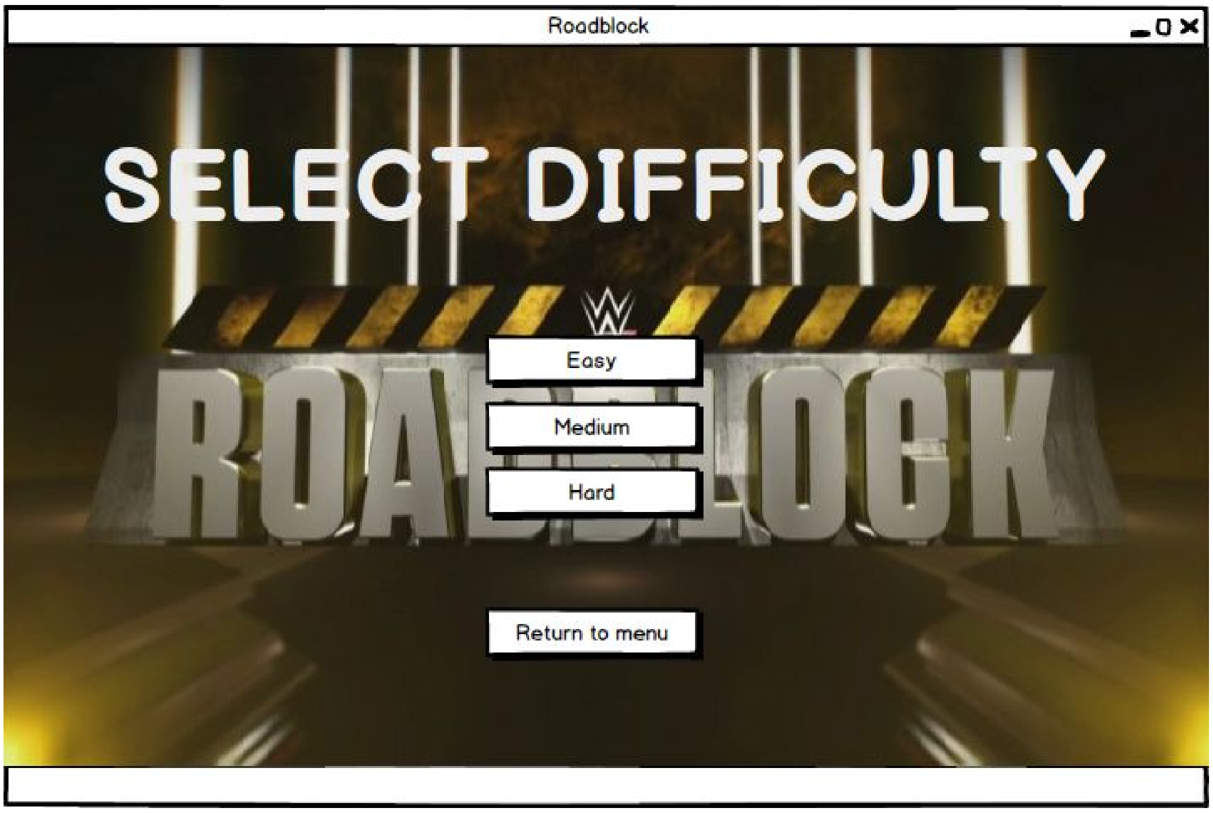
- Easy, Normal & Hard Panels are going to be initialized after the difficulty is selected. They are interpanels that will lead the user to select the level.
- Game Panel: It is the panel the game will be shown on. The player will use this panel in order to see the game.
- Options Panel: The user will make adjustments through this panel(turning on the sound, opening the effects etc.)

- About Us Panel is for introducing the game maker(us) to the user.
- Help Panel is going to be a short guide for user to learn how to play.

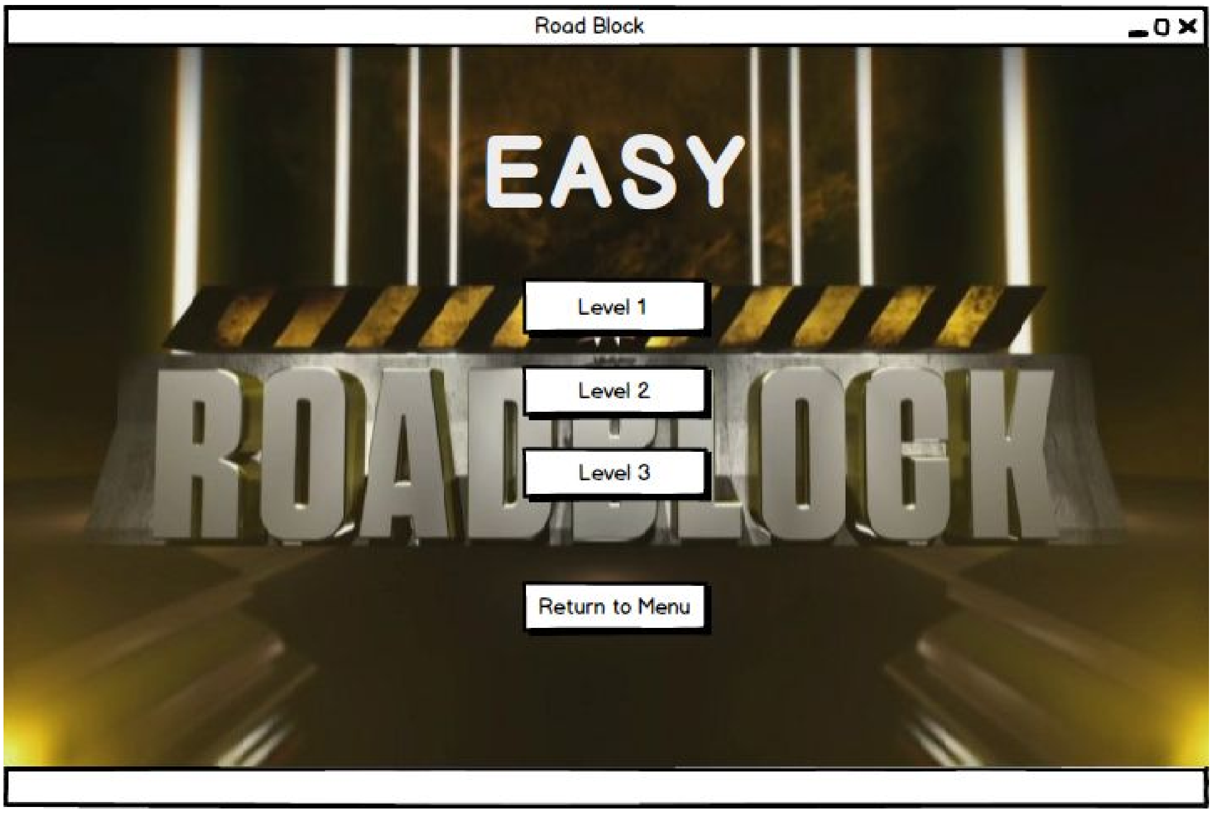
6. User interface - navigational paths and screen mock-ups



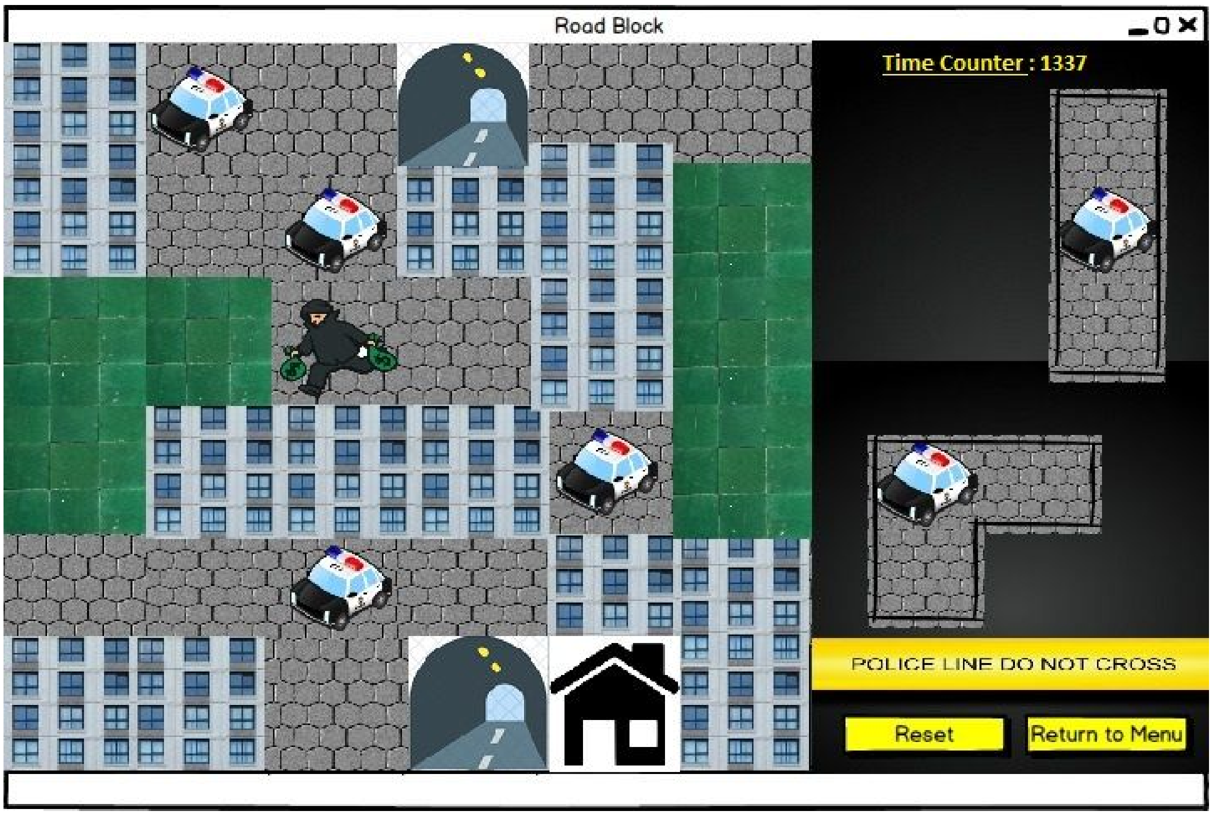
This is the entrance interface of the RoadBlock game. Using this interface it is possible for user to start a new game, set their setting preferences, learn about us,developers of the game, and get some useful information about how to play the game. This interface also offers the user to exit and close the game.



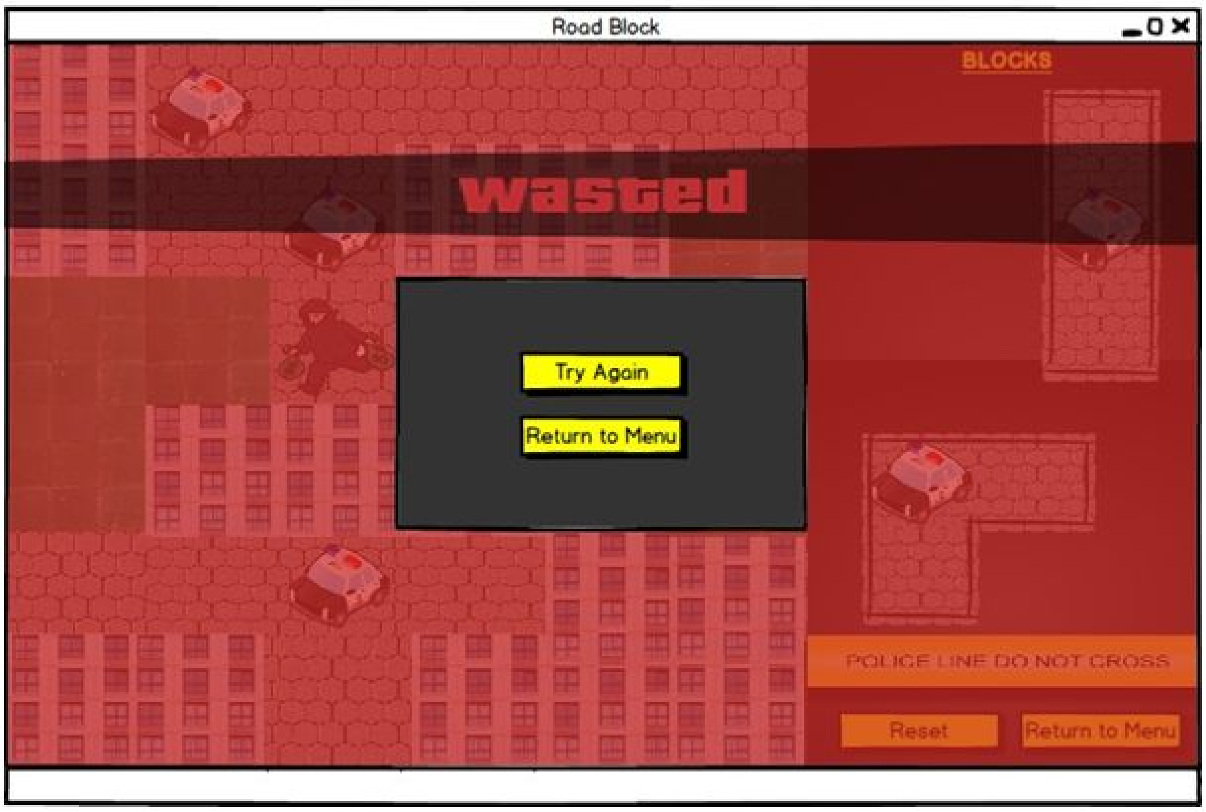
This page allows the user to choose a desired difficulty level of the game. It also has the option to return the previous menu.



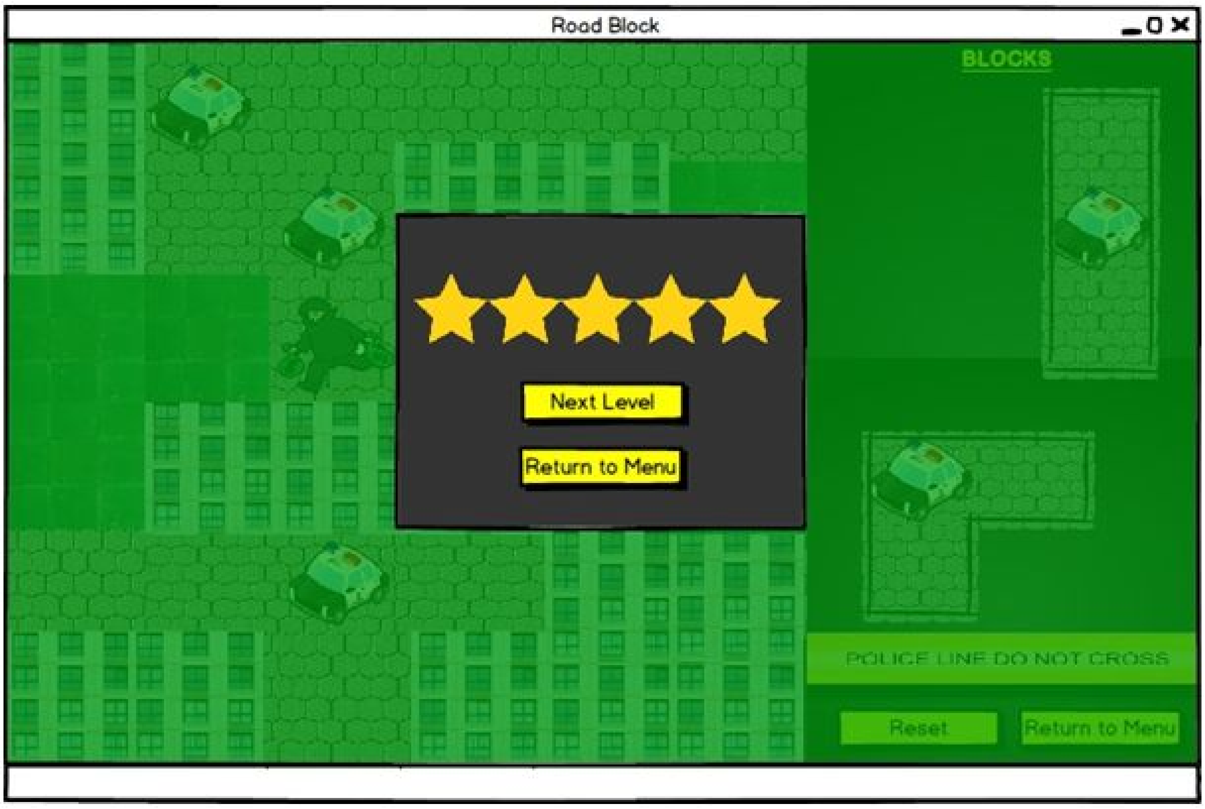
In this page,User may select 3 different levels of the Easy difficulty level.User may also return to the menu by clicking Return to Menu.



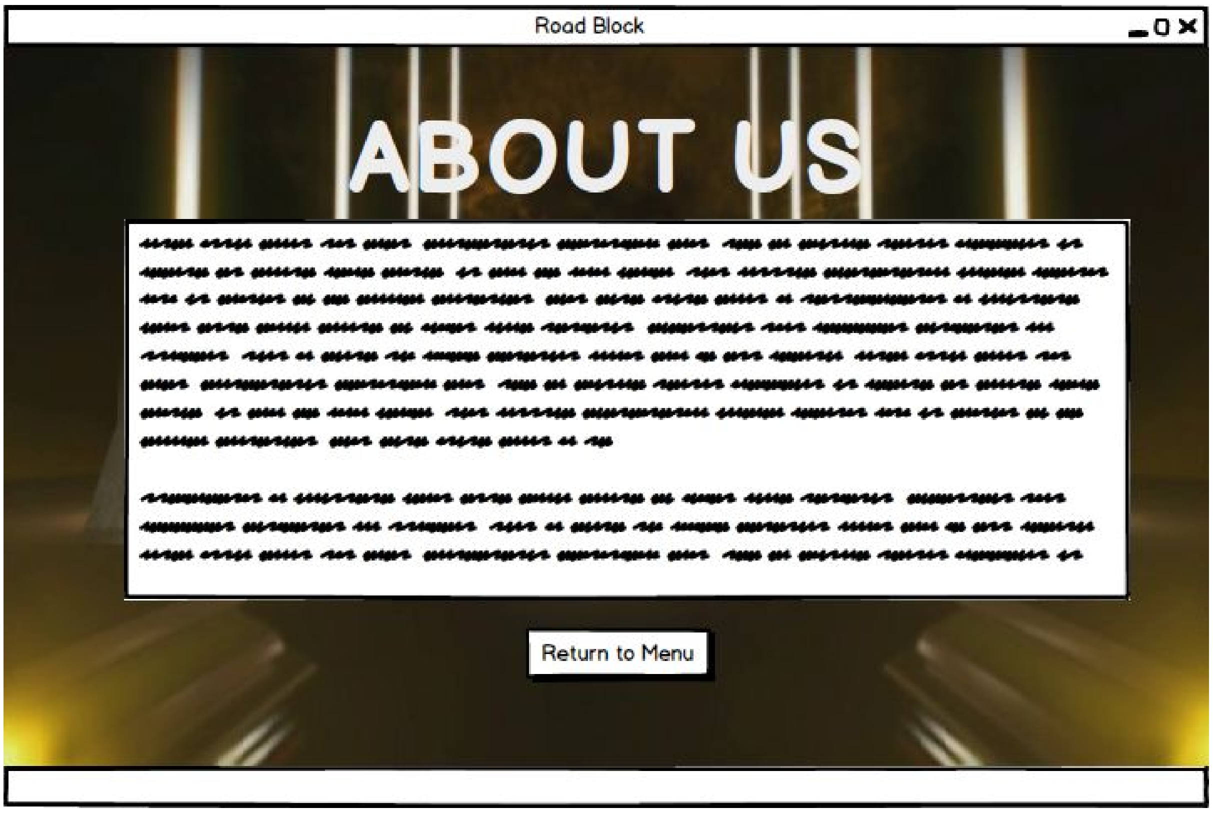
This is the main screen of the game. Player plays the game in this screen and places the police cars in order to block the paths which the thief may use in order to get away from the police.



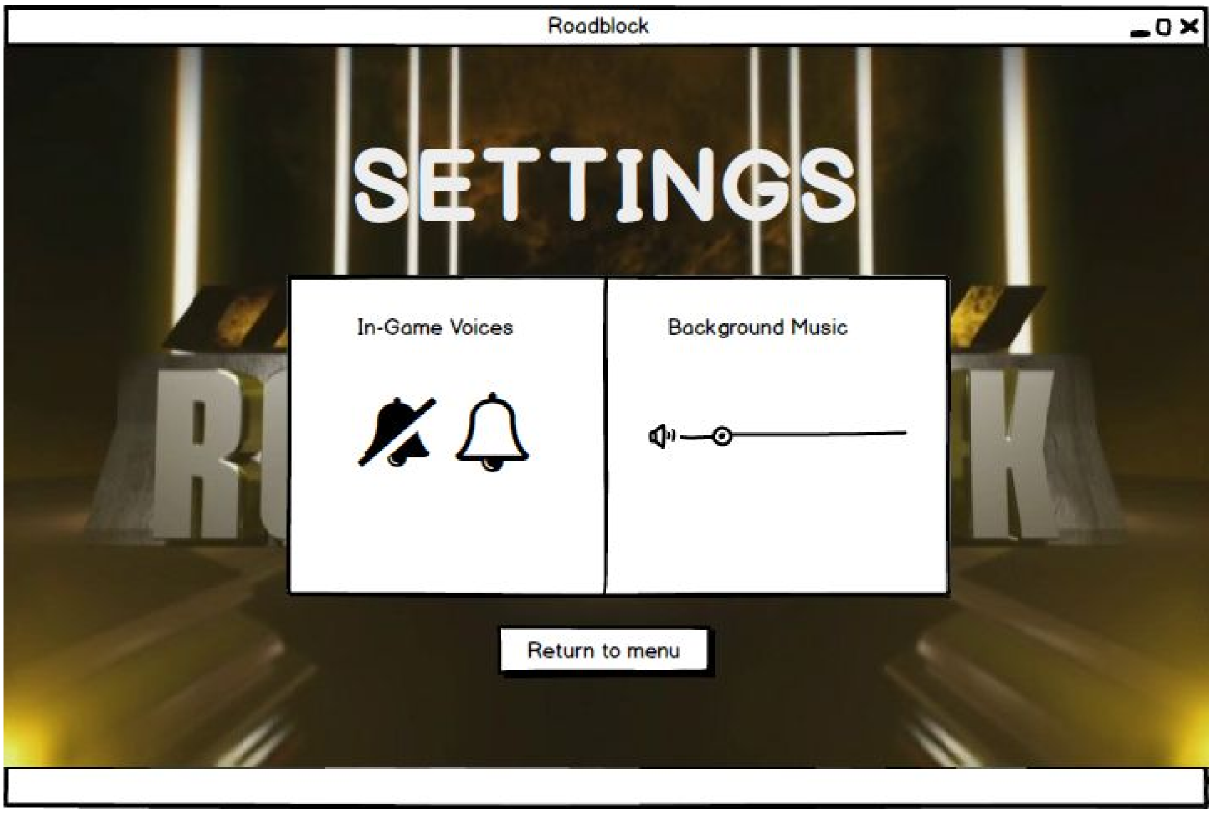
This page occurs when the player loses the game. The player can try it again or return back to menu.



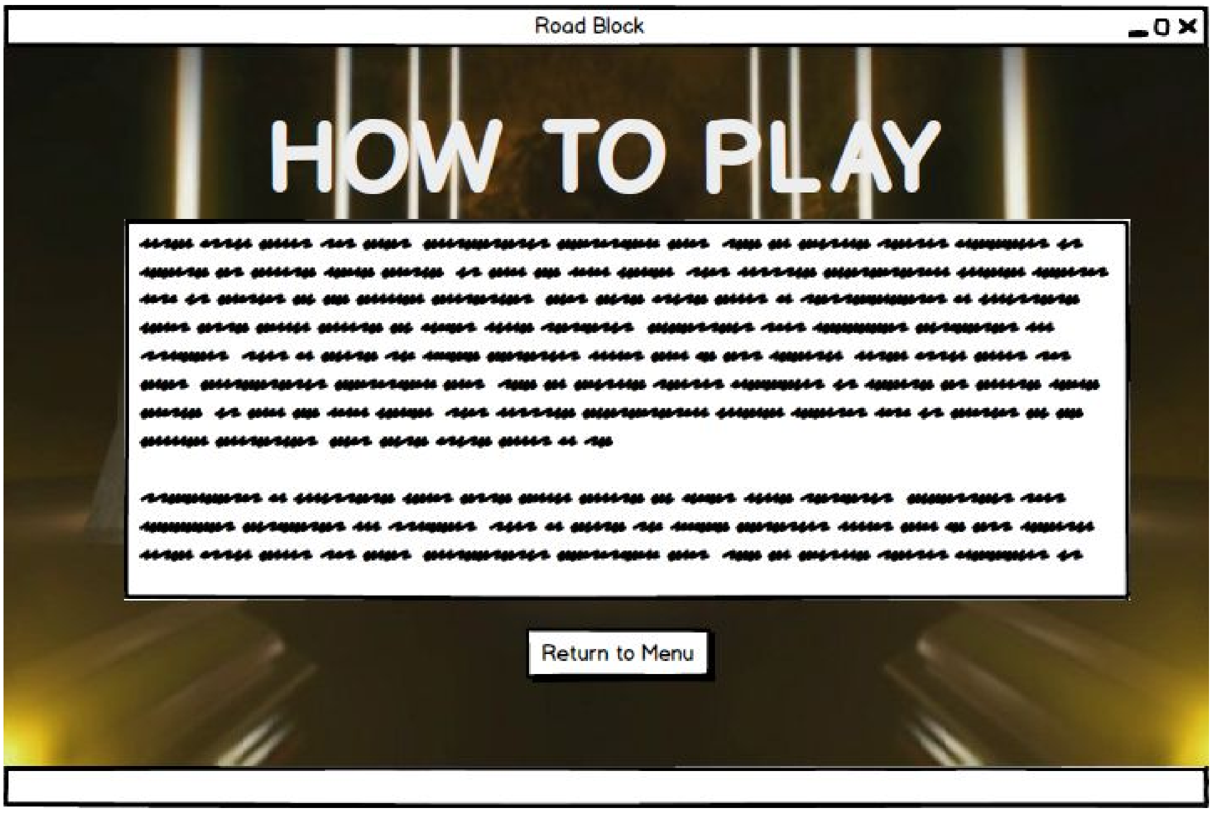
This page occurs when the player wins the game. The player can play the next level or return back to menu.



In this page, it is possible learn about the developers of the game and it also provides the option to return to the main menu.



These settings allows the user to turn on or turn off in-game voices. Also it is possible to adjust the background music volume.



In this page,user can learn the basics of the game.This page enables users to Return to Menu.

7. Glossary & references

Lethbridge, Timothy C., and Laganière R. *Object-Oriented Software Engineering: Practical Software Development Using UML and Java*. McGraw-Hill, 2002.