Cs319 Term Project

Section 3

Group 3D

Nomanleft

Iteration 2 Final Report

Group name: Nullpointers

Group Members:
Ahmet Akif Uğurtan
Berkay Karlık
Can Kaplan
Teymur Bakhishli
Eren Aytüre

Supervisor: Eray Tüzün

1. Introduction **3**

2. Design Changes **4**

3. Lessons Learnt **6**

3.2. Work Allocation 7

4. User's Guide **8**

4.1. System requirements & installation 8

4.1.1 How to set-up 8

4.1.1.1 Playing via JAR file 8

4.1.1.2 Build&Play via sourceCode 8

4.2. How to play 9

4.2.1. Main Menu 9

4.2.2. Classic Mode 10

4.2.3 Time Trial Mode 11

4.2.4 SandBox Mode 12

4.2.5 Shop 12

4.2.6 Options 13

1. Introduction

In our project, we will implement Walls & Warriors board game to a PC environment. The original game settings include a game board, 4 walls of different shape, 1 high tower, 3 blue knights and 4 red knights. Players then place the knights and high towers to the game board in the way game's challenge booklet suggest. Once the setup is completed, players may start the game. The purpose of the game is placing the walls on the selected challenge in such a way that all the red knights are left outside the walls while all the blue knights and the high tower is inside. Once this achieved the given challenge is considered solved. The booklet has 80 of such challenges. Our implementation has many expansions and changes over the original one. A major difference is beside the original challenge solving mode, there will be time trial mode where the player will solve as many puzzles as he/she can against time and a sandbox mode where the player can design he/she's own challenge and then play it. Alongside the new mods, original gameplay's challenge design will be expanded as well. Different landforms such as hills or lava pits will be part of the game. A set of boosters will be available player's aid to complete the challenge rather easily. Each level completion will include some gold reward so that player can refill boosters from the shop ones they finish.

Addition to this, we have used IntelliJ IDE to manage our project. For the version control, we used Git with SourceTree, a program that provides GUI for Git which made everything user-friendly and easy to manage.

We divided the workload according to the MVC pattern and started with the important parts of each subsystem. In our controller subsystem, we have GameManager and FileManager.

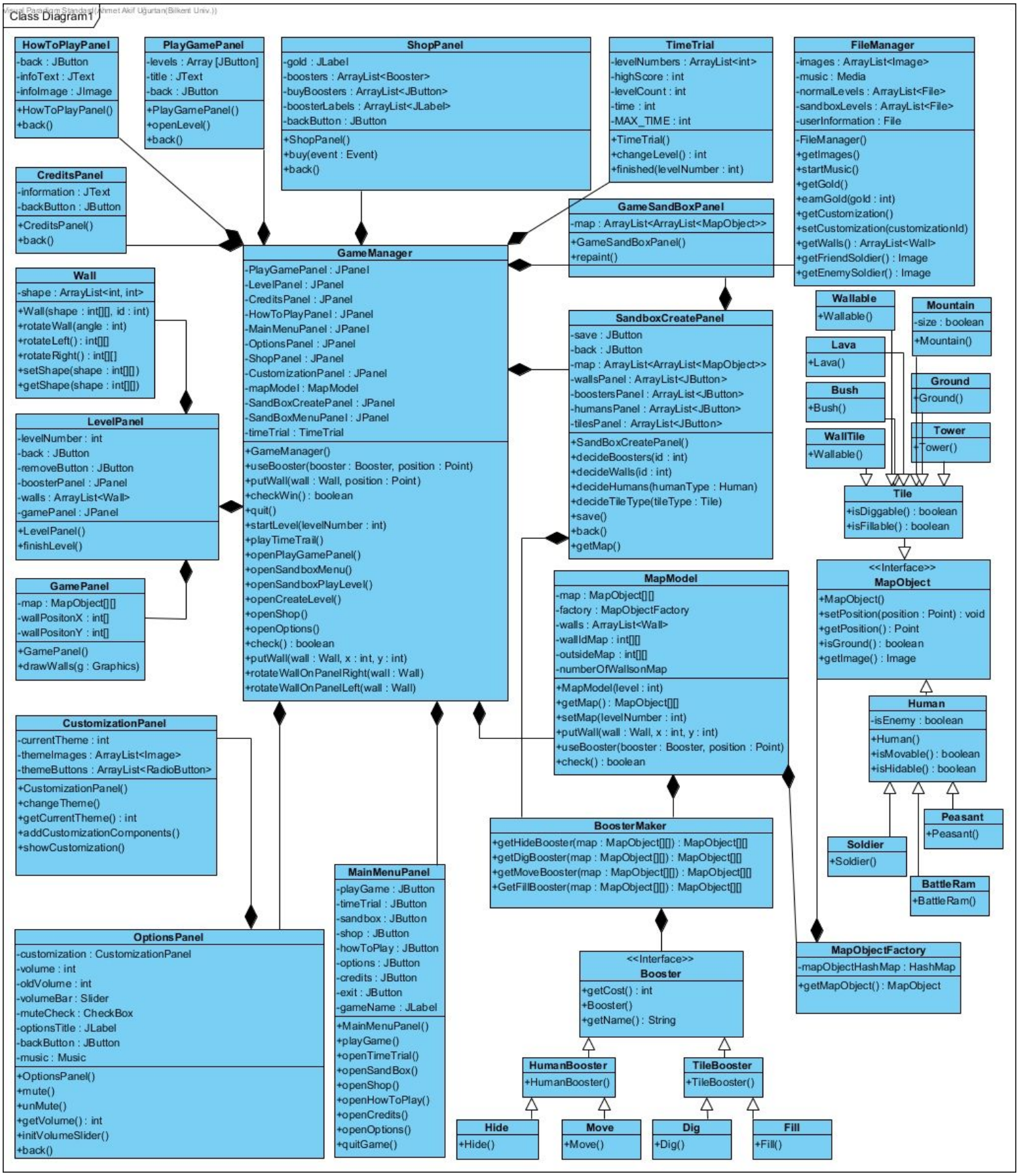
In our model subsystem, we have completed the MapObject classes and subclasses. As we stated in the Design Report, we implemented the FlyWeight design pattern to reduce memory usage. The booster classes are implemented according to Façade Design Pattern and we created BoosterMaker class to make design easier.

Lastly, in view subsystem, we have implemented the LevelPanel, MainMenuPanel, PlayGamePanel, ShopPanel, OptionsPanel, and SandBocCreatePanel. HowToPlayPanel and CreditsPanel are rather trivial so we didn't focus on them.

We have faced some problems due to the lack of knowledge about IntelliJ IDE and Version Control via Git which slowed us down. Also, we had to make some changes in our Class Diagram which we couldn't predict in the first iterations design process.

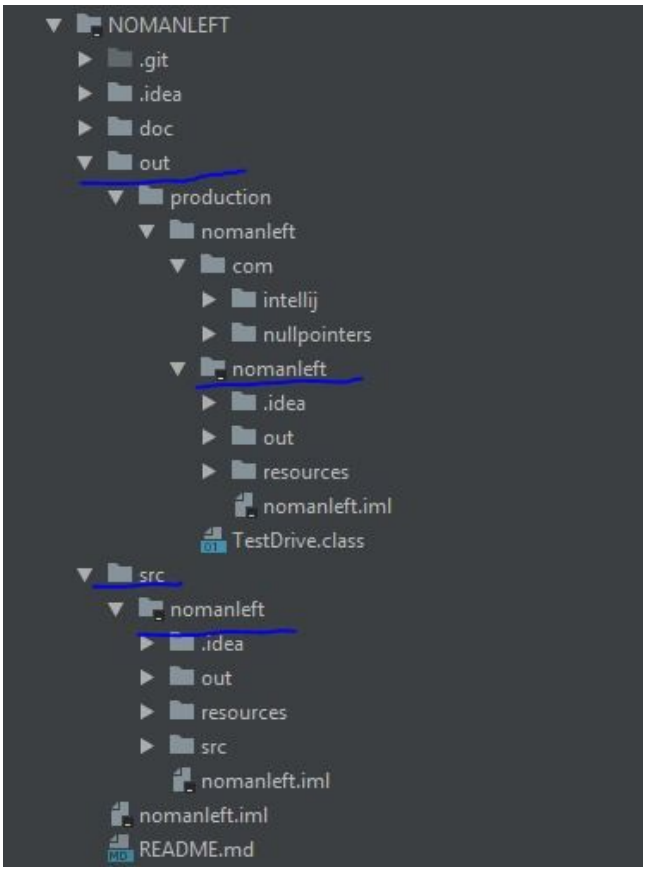
2. Design Changes

Generally, there were no substantial design change decision after the design report of the project. However, during the preliminary implementation of the project, we had few classes or method changes in order to make the whole project easy-to-code and understandable. In FileManager class, we added the setter and getter methods of all game objects (Wall, WallTile, Ground, Lava and etc.) to easily reach them from the FileManager class. We added WallPanel class to View classes which extends JPanel and uses the object of Wall class to repaint and update the map while the changes in wall structures in the game. Other than those, there have few changes in class methods. We changed that custom levels will be reached by PlayGamePanel instead of a new panel. In order to perform repaint functions, we created inner panels for SandboxPanel and LevelPanel which are GameSandBoxPanel and GamePanel. The updated form of project Class diagram is given below:


Class Diagram1
HowToPlayPanel
-back : JButton
-infoText : JText
-infoImage : JImage
+HowToPlayPanel()
+back()
PlayGamePanel
-levels : Array [JButton]
-title : JText
-back : JButton
+PlayGamePanel()
+openLevel()
+back()
ShopPanel
-gold : JLabel
-boosters : ArrayList<Booster>
-buyBoosters : ArrayList<JButton>
-boosterLabels : ArrayList<JLabel>
-backButton : JButton
+ShopPanel()
+buy(event : Event)
+back()
TimeTrial
-levelNumbers : ArrayList<int>
-highScore : int
-levelCount : int
-time : int
-MAX_TIME : int
+TimeTrial()
+changeLevel() : int
+finished(levelNumber : int)
FileManager
-images : ArrayList<Image>
-music : Media
-normalLevels : ArrayList<File>
-sandboxLevels : ArrayList<File>
-userInformation : File
-FileManager()
+getImages()
+startMusic()
+getGold()
+eamGold(gold : int)
+getCustomization()
+setCustomization(customizationId)
+getWalls() : ArrayList<Wall>
+getFriendSoldier() : Image
+getEnemySoldier() : Image
CreditsPanel
-information : JText
-backButton : JButton
+CreditsPanel()
+back()
GameSandBoxPanel
-map : ArrayList<ArrayList<MapObject>>
+GameSandBoxPanel()
+repaint()
GameManager
-PlayGamePanel : JPanel
-LevelPanel : JPanel
-CreditsPanel : JPanel
-HowToPlayPanel : JPanel
-MainMenuPanel : JPanel
-OptionsPanel : JPanel
-ShopPanel : JPanel
-CustomizationPanel : JPanel
-mapModel : MapModel
-SandBoxCreatePanel : JPanel
-SandBoxMenuPanel : JPanel
-timeTrial : TimeTrial
+GameManager()
+useBooster(booster : Booster, position : Point)
+putWall(wall : Wall, position : Point)
+checkWin() : boolean
+quit()
+startLevel(levelNumber : int)
+playTimeTrail()
+openPlayGamePanel()
+openSandboxMenu()
+openSandboxPlayLevel()
+openCreateLevel()
+openShop()
+openOptions()
+check() : boolean
+putWall(wall : Wall, x : int, y : int)
+rotateWallOnPanelRight(wall : Wall)
+rotateWallOnPanelLeft(wall : Wall)
Wall
-shape : ArrayList<int, int>
+Wall(shape : int[][], id : int)
+rotateWall(angle : int)
+rotateLeft() : int[][]
+rotateRight() : int[][]
+setShape(shape : int[][])
+getShape(shape : int[][])
SandboxCreatePanel
-save : JButton
-back : JButton
-map : ArrayList<ArrayList<MapObject>>
-wallsPanel : ArrayList<JButton>
-boostersPanel : ArrayList<JButton>
-humansPanel : ArrayList<JButton>
-tilesPanel : ArrayList<JButton>
+SandBoxCreatePanel()
+decideBoosters(id : int)
+decideWalls(id : int)
+decideHumans(humanType : Human)
+decideTileType(tileType : Tile)
+save()
+back()
+getMap()
Wallable
+Wallable()
Mountain
-size : boolean
+Mountain()
Lava
+Lava()
Bush
+Bush()
Ground
+Ground()
WallTile
+Wallable()
Tower
+Tower()
Tile
+isDiggable() : boolean
+isFillable() : boolean
LevelPanel
-levelNumber : int
-back : JButton
-removeButton : JButton
-boosterPanel : JPanel
-walls : ArrayList<Wall>
-gamePanel : JPanel
+LevelPanel()
+finishLevel()
GamePanel
-map : MapObject[][]
-wallPositonX : int[]
-wallPositonY : int[]
+GamePanel()
+drawWalls(g : Graphics)
MapModel
-map : MapObject[][]
-factory : MapObjectFactory
-walls : ArrayList<Wall>
-wallIdMap : int[][]
-outsideMap : int[][]
-numberOfWallsonMap
+MapModel(level : int)
+getMap() : MapObject[][]
+setMap(levelNumber : int)
+putWall(wall : Wall, x : int, y : int)
+useBooster(booster : Booster, position : Point)
+check() : boolean
<<Interface>>
MapObject
+MapObject()
+setPosition(position : Point) : void
+getPosition() : Point
+isGround() : boolean
+getImage() : Image
Human
-isEnemy : boolean
+Human()
+isMovable() : boolean
+isHidable() : boolean
Peasant
+Peasant()
Soldier
+Soldier()
BattleRam
+BattleRam()
CustomizationPanel
-currentTheme : int
-themeImages : ArrayList<Image>
-themeButtons : ArrayList<RadioButton>
+CustomizationPanel()
+changeTheme()
+getCurrentTheme() : int
+addCustomizationComponents()
+showCustomization()
BoosterMaker
+getHideBooster(map : MapObject[][]) : MapObject[][]
+getDigBooster(map : MapObject[][]) : MapObject[][]
+getMoveBooster(map : MapObject[][]) : MapObject[][]
+GetFillBooster(map : MapObject[][]) : MapObject[][]
MainMenuPanel
-playGame : JButton
-timeTrial : JButton
-sandbox : JButton
-shop : JButton
-howToPlay : JButton
-options : JButton
-credits : JButton
-exit : JButton
-gameName : JLabel
+MainMenuPanel()
+playGame()
+openTimeTrial()
+openSandBox()
+openShop()
+openHowToPlay()
+openCredits()
+openOptions()
+quitGame()
OptionsPanel
-customization : CustomizationPanel
-volume : int
-oldVolume : int
-volumeBar : Slider
-muteCheck : CheckBox
-optionsTitle : JLabel
-backButton : JButton
-music : Music
+OptionsPanel()
+mute()
+unMute()
+getVolume() : int
+initVolumeSlider()
+back()
<<Interface>>
Booster
+getCost() : int
+Booster()
+getName() : String
MapObjectFactory
-mapObjectHashMap : HashMap
+getMapObject() : MapObject
HumanBooster
+HumanBooster()
TileBooster
+TileBooster()
Hide
+Hide()
Move
+Move()
Dig
+Dig()
Fill
+Fill()

3. Lessons Learnt

While we trying to use GitHub we made some mistakes. For example, In our project, we used IntelliJ as IDE but some of us were using the older version of IntelliJ because of that pushing and pull to the Github caused some errors. Addition to that we made a mistake at creating a project clone in our computers. One of us created the project file from the outer folder which was changed the tree of the project. When he pushed that project to the GitHub, and we did not recognize at first and pulled it and because of that our project clones are messed as well.



On every push and pull, .class files and workspace.xml files were creating merge conflicts but they are unimportant for us since they get created every time the application runs. We just resolved these conflicts every time by selecting one of the conflicts since neither of them is important. However, we know that this is not the correct solution, the correct solution would be never pushing these files to the git remote.

In order to reach images from the project, we need to create an external folder within the project folder which is the resources folder from this picture.

We learned the use of GUI Forms in IntelliJ and we learned that how to use custom panels from objects.

In IntelliJ GUI forms, the createUIComponents function which is a function but more like the custom constructor for creating custom components works before default constructor. We were getting NullPointerException since we did not know this.

We made and trained some teamwork. For example, for implementing our levelPanel one of us need to communicate with the other who wrote the mapModel class since user interface part and map model part are done by different people.

3.2. Work Allocation

Ahmet Akif Uğurtan

- Implemented game map algorithms in MapModel class such as how to put wall, check if the game ends etc..
- Implemented drawing algorithm for a level map in repaint function of GamePanel class.
- Implemented FileManager class for uploading images to ram only once so game speed is not slow.
- Implemented Boosters and BoosterMaker class.
- Wrote parts on reports.

Can Kaplan

- Changed images in order to improve the user experience.
- Designed LevelPanel and PlayGamePanel forms and components where the game is played.
- Wrote parts on reports.

Berkay Karlık

- Implemented OptionsPanel, MainMenuPanel, ShopPanel, CustomizationPanel classes.
- Implemented music in OptionsPanel.
- Implemented TimetrialPanel and algorithms.
- Wrote parts on reports.

Eren Aytüre

- Implemented MapObjects and MapObjectFactory class.
- Implemented SandboxPanel to create a new level.
- Wrote parts on reports.

Teymur Baxisli

- Wrote parts on reports.

4. User’s Guide

4.1. System requirements & installation

The recommended system requirements for the game is having Operating System - Windows 7, 8 or 10. The game does not have special Memory or Graphics system requirements as it occupies very small storage and has basic music sounds.

Minimum Requirements:

OS	Windows 7
Processor	Core i3-6006U and above
Memory	2GB Memory
Graphics	Intel HD Graphics 520
Additional	1600*900 resolution

4.1.1 How to set-up

4.1.1.1 Playing via a JAR file

Requirements:

- Java Runtime Environment (JRE)

Anyone that wants to play the game can enter our GitHub repository and download the jar file alone and run it directly to play it. The only requirement is player should have installed at his/her computer.

4.1.1.2 Build&Play via sourceCode

Requirements:

- A version control tool (optional)
- An Integrated Development Environment (optional but for compatibility, IntelliJ Recommended)
- Java Runtime Environment (JRE)
- Java Development Kit (JDK)
- An adequate understanding of software

Alternatively, a player can download the source code or clone it via a version control tool.

To do that step by step with version control:

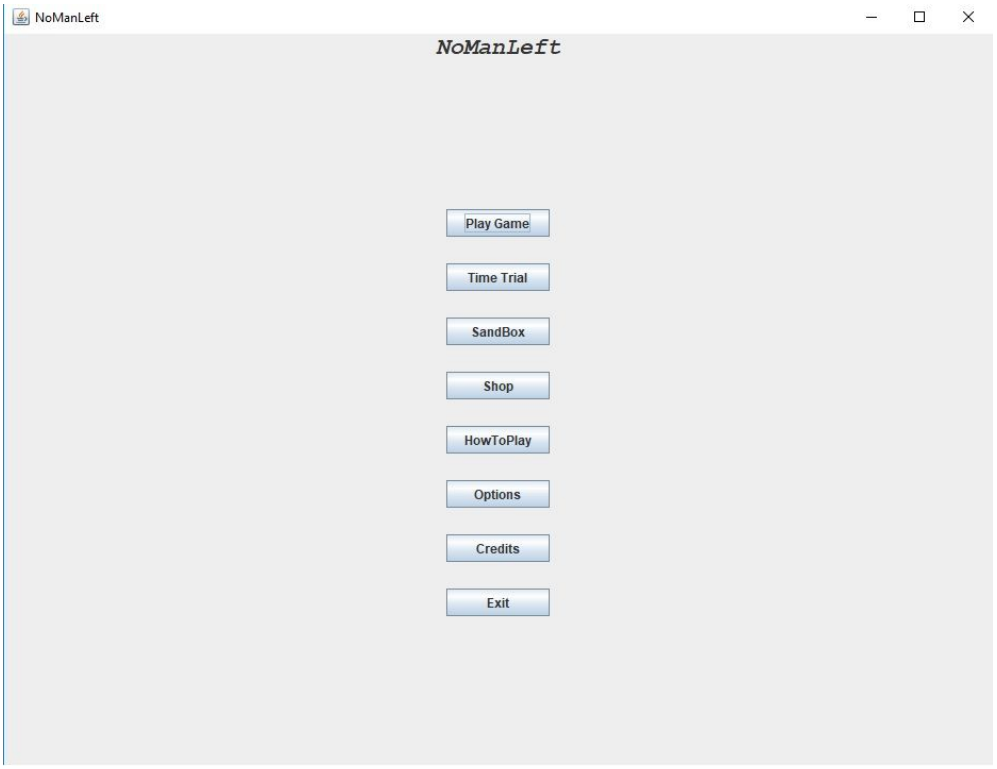
1. Clone the project from the project repository.
2. Open the project folder via IntelliJ or create a new project on the IDE you use. The project folder is src/nomanleft in the repository. If you use a different ide copy the files to your project excluding project related files.

3. Compile the project with Java 8 JDK.
4. Run it and have fun.

Doing it via downloading is pretty much the same. Instead of cloning download the project file and continue from step 2.

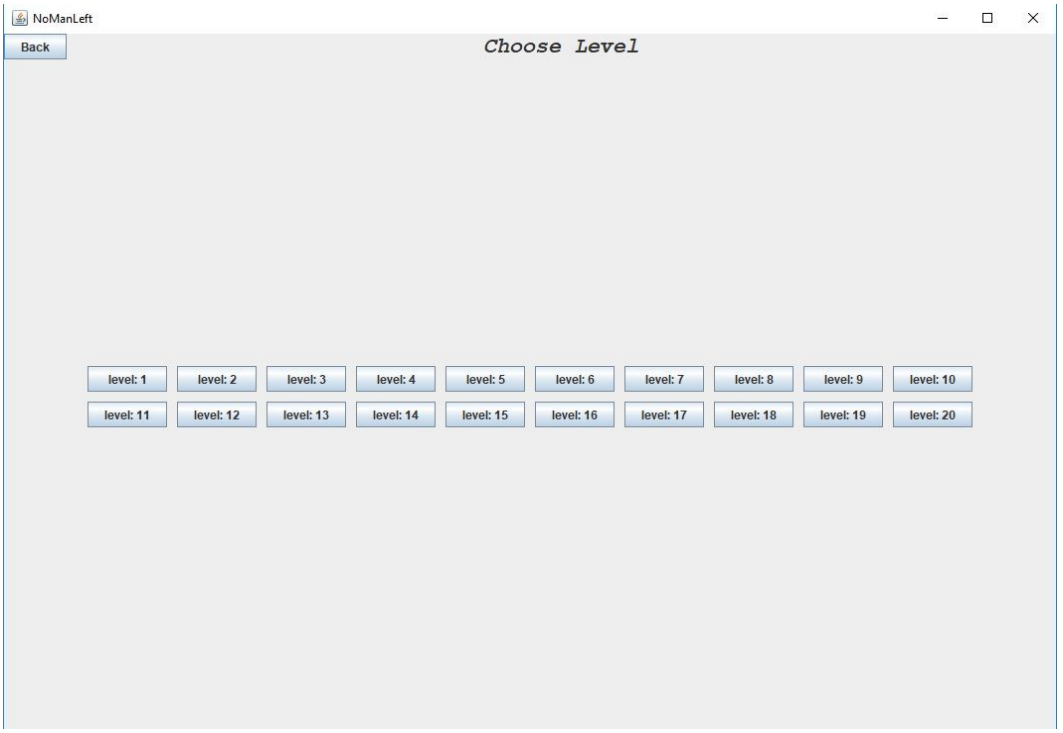
4.2. How to play

4.2.1. Main Menu

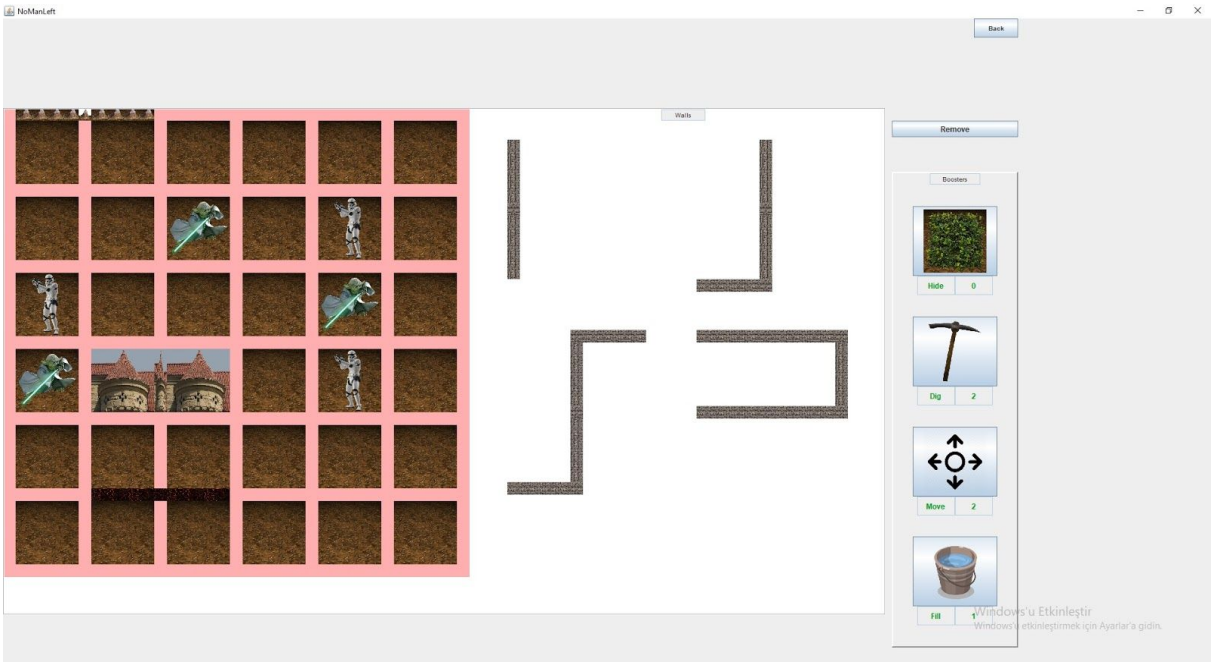


As soon as the game starts, the main menu greets the player. From here, the player can choose to play the game in the classic mode by clicking the Play Game button, play a time trial mode by clicking Time Trial button or can design his own level in the sandbox mode by clicking Sandbox button. Moreover player can enter the shop to buy boosters with the gold gained from the levels. Options menu allows to mute or change sound levels and customize the in-game pictures of the game. How to Play button opens a panel where the game instructions are written. Similarly, credits panel has information about us. Finally, exit buttons close the game.

4.2.2. Classic Mode



After clicking the Play Game button the level selection panel appears.From here the player can choose any available level.

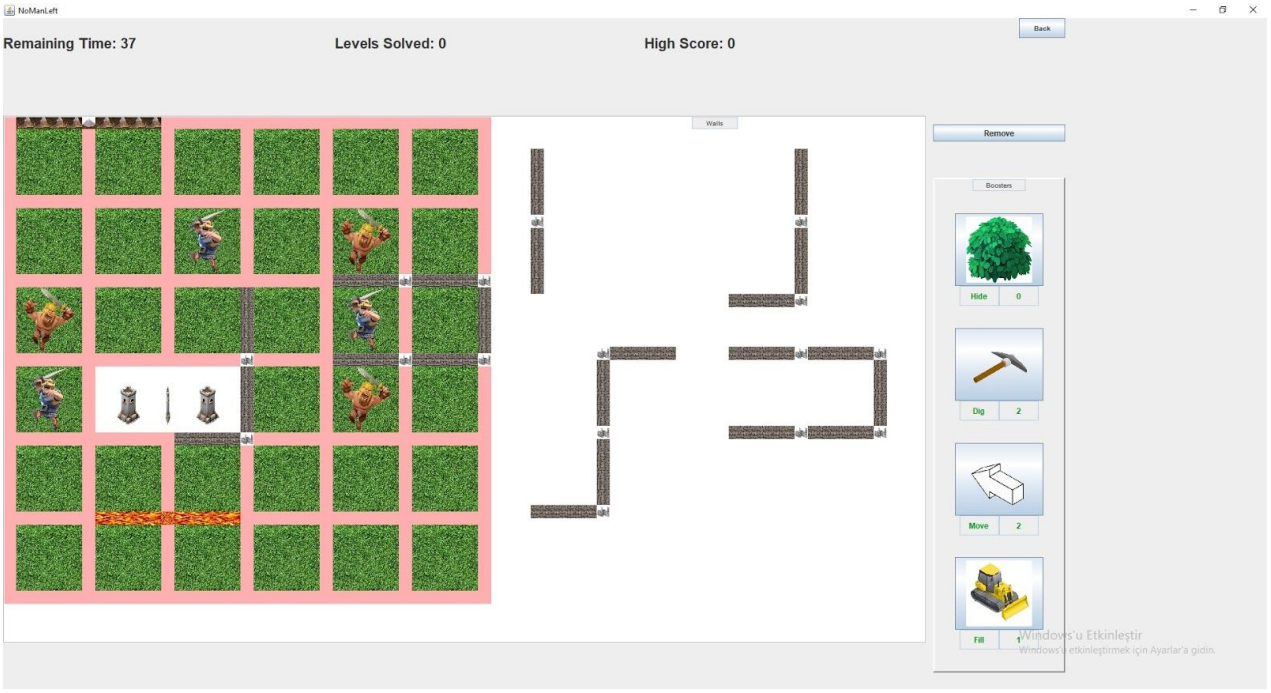


As soon as a level is chosen the game starts. The player can choose any one of the four walls available by clicking and dragging to the desired point of placement. If need to be walls can be rotated by a right click before dragging. Furthermore, if there are available boosters player can use them to gain the advantage. When applied to a friendly knight bush booster hides him and player can finish the level without including him inside the walls. Fill

booster removes one unit of lava from the map. Similarly, dig booster removes one unit of a mountain from the map. Player can move a friend or enemy knight by dragging one square after using move booster. To use a booster click the respective button and then click (for move booster its drag) the target structure(mountain, lava, knight). To remove a wall that is placed, the player has to click the remove button and then the wall that needs to be removed. Finally, if the player wants to quit the level without completion he can click back button which triggers a pop-up as follows:

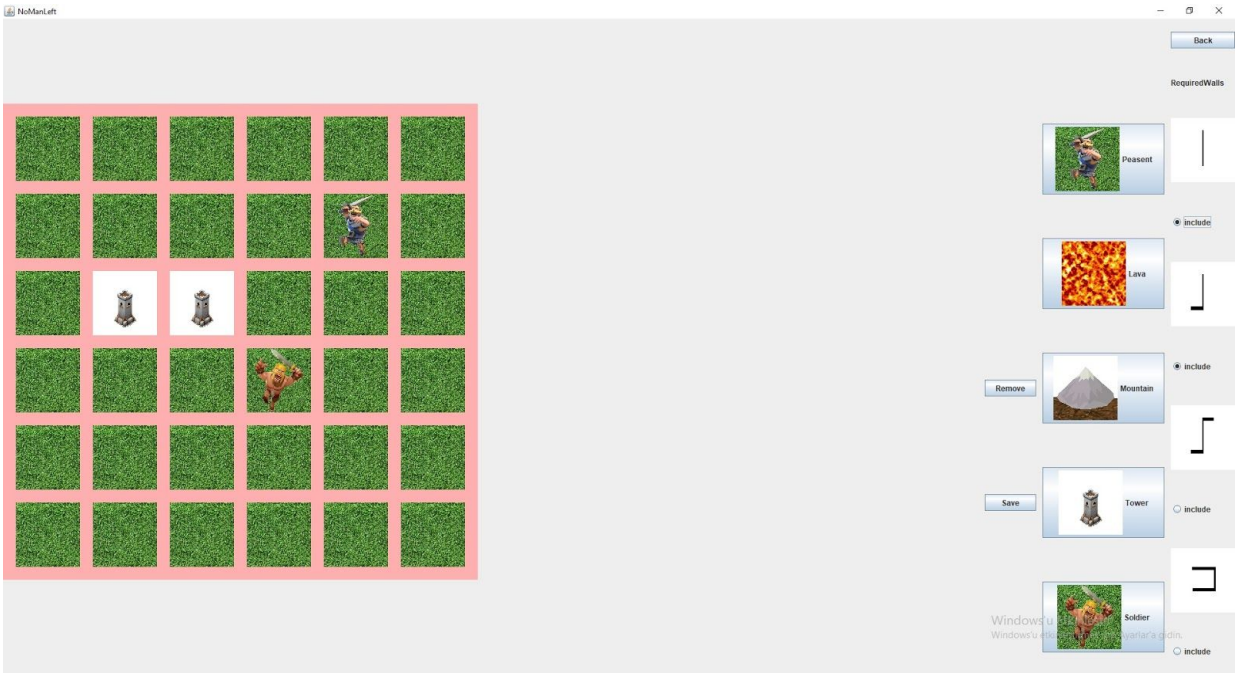
Upon clicking yes player goes back to the level selection panel. Each completed level rewards the player with 10 gold.Player is notified with a pop-up.

4.2.3 Time Trial Mode



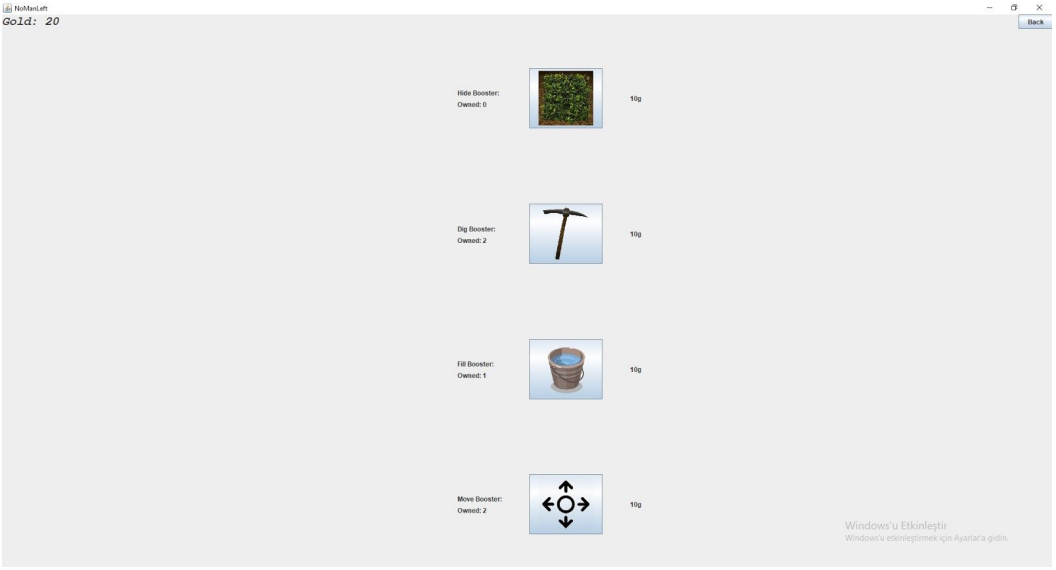
Time trial mode is the same as the classic mode in terms of controls and mechanics. However, in this mode player has to solve as many levels as possible in the given time frame. Upon completing one level, another level is loaded randomly. If the player finishes all of the levels available before the given time limit or time runs out the game is over. Each completed level adds 10 gold to the reward. The player can see the high score, remaining time and the solved level count above the panel while playing.

4.2.4 SandBox Mode



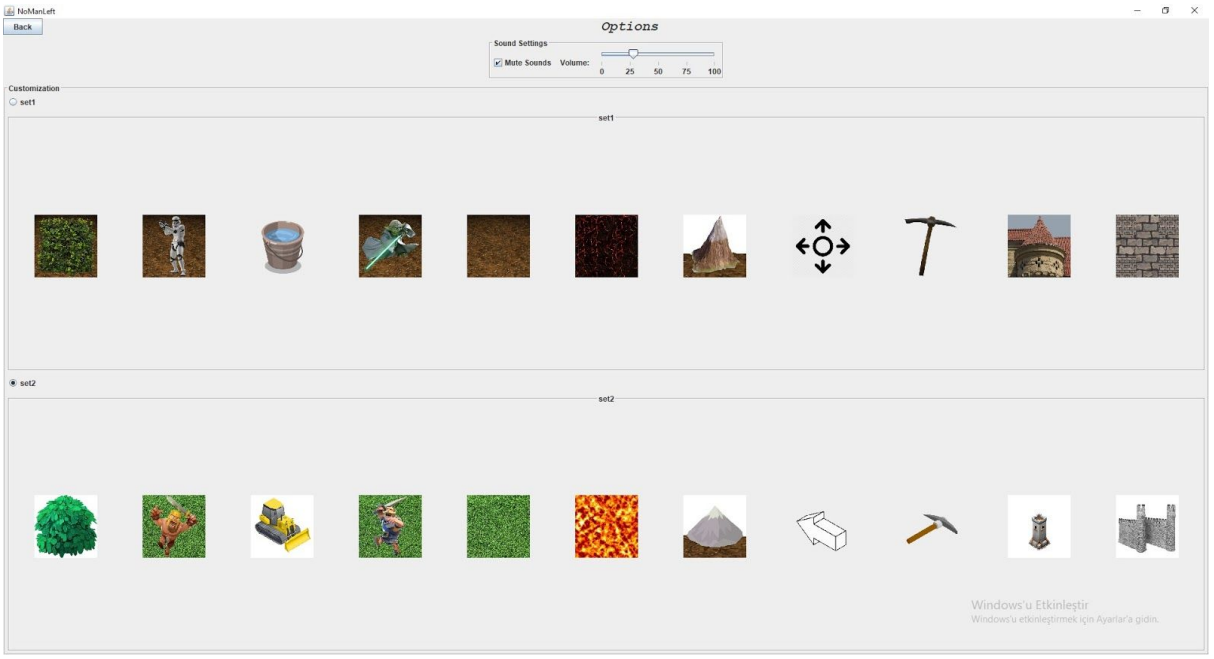
Our third mode allows the player to create levels of his own. The player can generate his level by dragging knights and obstacles to the available points on the map. Player can click the button of the object they want to place and then click to the destination point. The walls that needs to be used in the level can be changed as well.

4.2.5 Shop



The player can buy boosters from this panel if they have enough gold. If they don't a pop-up warns them.

4.2.6 Options



From the options menu, the player can adjust the sound level or mute it from the sound panel. The pictures of the game can be changed from the customization settings by selecting a predefined set of pictures via radio buttons.