



# Cs319 Term Project

Section 3

Group 3D

Nomanleft

Analysis Report

Group Members:

Ahmet Akif Uğurtan

Berkay Karlık

Can Kaplan

Teymur Bakhishli

Eren Aytüre

Supervisor: Eray Tüzün

<b>1.Introduction</b>	<b>3</b>
<b>2.Overview</b>	<b>4</b>
2.1.Gameplay	4
2.2.Characters	4
2.3.Map and Zones	4
2.4.Boosters	5
2.5.Point System	6
<b>3.Requirements</b>	<b>6</b>
3.1.Functional Requirements	6
3.1.1.Play Game	6
3.1.1.1. Play Normal Game	6
3.1.1.2. Time Trial	6
3.1.1.3. SandBox	7
3.1.2.Shop	7
3.1.3.Options	7
3.1.4.How To Play	7
3.1.5.Credits	8
3.2.Non-Functional Requirements	8
3.2.1.User Interface	8
3.2.2.Customization	8
<b>5.System Models</b>	<b>9</b>
5.1.Use Case Model	9
5.2.Dynamic Models	26
5.2.1.Sequence Diagram	26
5.2.1.1.Play Game Sequence	26
5.2.1.2.Play Time Trial Sequence	27
5.2.1.3.SandBox	28
5.2.1.4. How to Play	29
5.2.1.5 Options	29
5.2.2.Activity Diagram	30
5.3.Object and Class Model	31
5.4.User interface	36
5.4.1.Navigational Paths	36
5.4.2.Mockups	37
5.4.2.1. Main Menu	37
5.4.2.2. Choose a Level	38
5.4.2.3. Play game	39
5.4.2.4 SandBox	40
<b>6.References</b>	<b>43</b>

# 1.Introduction

In our project, we will implement Walls & Warriors board game to a PC environment. The purpose of the game is completing different puzzle configurations. On each challenge, the player has to place differently shaped walls on the board in such a way that all blue knights and tower will be inside the castle. 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; the variety of characters with distinct attributes such as giant which covers two unit ground, battering-rams that requires more unit of walls to complete will save gameplay from monotony by enforcing player to think new ways of solving the puzzle. 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. Besides the gold, depending on the number of steps taken towards the completion, a level can be completed with up to three stars, which shows player's puzzle solving skills.

## 2.Overview

### 2.1.Gameplay

The aim of the game is keeping all friendly warriors and the tower inside the walls and all enemy warriors outside of the walls by placing walls in a way to accomplish this purpose. Walls can be rotated to try different possibilities. Boosters can be used to make walls put easily by changing the type of the ground or moving a warrior or hiding a warrior. The user will earn more gold if he/she finishes the game by trying fewer actions.

### 2.2.Characters

A soldier can be enemy or friendly. All friendly soldiers must be inside walls while all enemy soldiers must be outside of walls.

Peasant is friendly only. Peasant does not have to be inside walls, however, if it is inside walls more gold will be earned to award protecting peasants.

Battle Ram is enemy only. In the direction of ram's head, there must be two walls, otherwise, the ram will break it so the puzzle cannot be completed.

### 2.3.Map and Zones

A map is a  $n \times n$  matrix consisted of different zones. A map can be in different shapes by making some zones unavailable or void.

Inside the map, ground zones are just normal zones where any character or wall can be there.

Lava zones are zones which nothing can be put, however, lava zones can be transformed into a ground by using fill booster.

Mountain zones are again zones which nothing can be put there, however, they can be transformed into a ground by using dig booster.

Bush zones are zones that soldiers can hide in them so they do not have to be inside walls since they hide. Bush zones are not initially existed but can be created by using hide booster.

Tower zones are similar to soldiers actually. They must be inside the walls in order to solve the puzzle. Also, nothing changes tower zones.

Wall zones are zones that user has put walls. They are not initially existed but they exist when a user put a wall. Walls can be put only on the ground zones.

## 2.4.Boosters

There are four boosters, two of them can be applied to characters and other two of them can be applied to zones.

Hide booster can be used only on a friendly soldier in order to hide it so it does not have to be inside walls. When a hide booster is used a bush zone appears on the zone that soldier stays.

Move booster can be used on every character to move them only one unit next so that putting walls and keeping friendly soldiers inside, enemy soldiers outside becomes easier.

Dig booster can be used only on mountain zones to transform them into ground zones.

Fill booster can be used only on lava zones to transform them into ground zones.

## 2.5.Point System

The user will gain points as he/she completes the level by trying fewer actions or saving peasants too. The user will gain golds according to his/her points. The highest score of the user on a specific level will be displayed while choosing a level. Golds can be spent on the shop in order to buy boosters.

## 3.Requirements

### 3.1.Functional Requirements

#### 3.1.1.Play Game

There will be 3 different game modes as follows:

##### 3.1.1.1. Play Normal Game

The player can play this mode by clicking Play Game from the menu. When clicked a panel that contains all the level will appear. Then the player can choose the next one or one of the previous challenges to play. The selected challenge will get loaded.

In the challenge, the player must complete a puzzle which requires him to build walls that leave all the enemy knights out while keeping his knights in. The player may need to consider the properties of different characters and elements while solving the challenge.

##### 3.1.1.2. Time Trial

This mode can be opened from the menu by clicking the Time Trial button. Although logic for passing the challenges are the same here, in this mode player has to solve as many

challenges as possible within a time limit. Unlike the normal mode, the order of levels is randomized here.

#### 3.1.1.3. SandBox

The third mode SandBox allows players to design their own levels. In the menu, upon clicking sandbox option player encounters two options, create a level or play level. When create level is selected. The player places the enemy knights, different tile types and boosters in a valid combination to make a custom-made level. The second option play level allows the player to play the levels he/she made.

#### 3.1.2.Shop

Upon completing a challenge in the normal mode player is rewarded with some amount of gold which can be used to buy boosters from the shop. Different kind of boosters helps the player to complete the level easier. The shop can be opened from the main menu.

#### 3.1.3.Options

The player can access this screen from the main menu. In this panel, the player can adjust sound levels and mute or unmute them. Also, different themes are available for the player to change the appearances of the in-game characters.

#### 3.1.4.How To Play

From the menu, the player can open this panel by clicking how to play button. This panel contains information about the basics of the game such as:

- the purpose of the game, win/lose conditions,
- features of different boosters,
- attributes of different characters and tiles,

- instructions for different game mods.

### 3.1.5.Credits

This panel can be opened from the menu and it contains information about the developers and maybe the artwork used (if any).

## 3.2.Non-Functional Requirements

### 3.2.1.User Interface

The user interface plays a big role in our game since the crucial parts of the game such as placing walls, using boosters, customizing maps will be heavily dependent on the user interface. The user interface should work smoothly and be easily accessible to allow the player to focus on the puzzle aspect of the game.

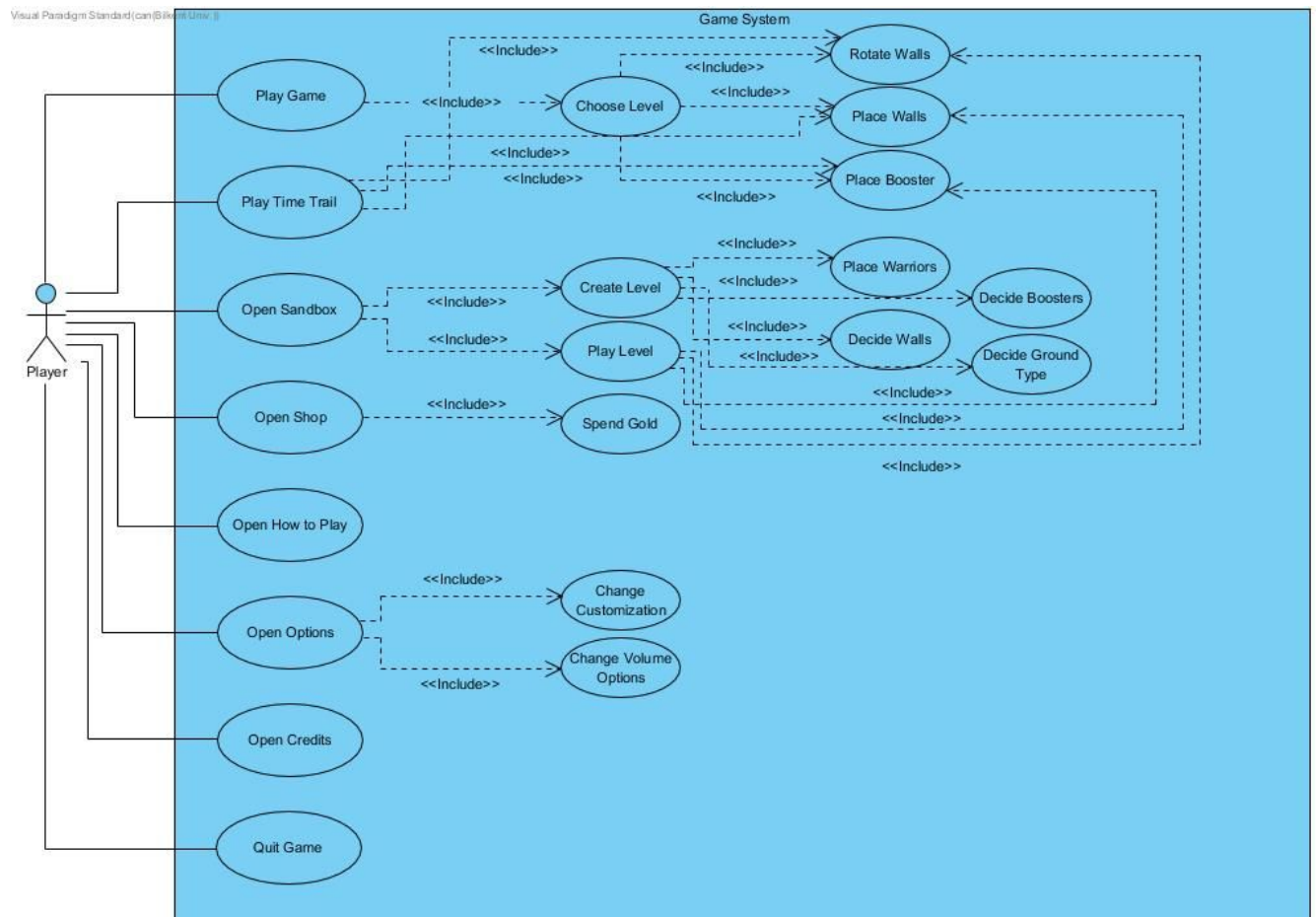
### 3.2.2.Customization

To increase the immersion to the game we thought of adding a character customization option which will allow the player to choose a theme for the character visuals. Each different character type will have its own image.



## 5. System Models

### 5.1. Use Case Model



#### ➤ Use Case #1

Use Case: Play Game

Primary Actor: Player

Stakeholders and Interests:

- Player selects play the game
- System shows levels

Pre-conditions:

- Player must be in the main menu.

Post-conditions:

Entry-conditions:

- Player should see any available level.

Exit conditions:

- Player selects back button.

Success Scenario Event Flow:

1. Player chooses Play game button.

Alternative Event Flows:

1. Player chooses back button to return menu.

## ➤ Use Case #2

---

Use Case: Choose Level

Primary Actor: Player

Stakeholders and Interests:

- Player selects level.
- System opens that level and lets player to play.

Pre-conditions:

- Player must be clicked play game button.

Post-conditions:

Entry-conditions:

- Player should choose any available level.

Exit conditions:

- Player selects back button to exit level panel.

Success Scenario Event Flow:

1. Player chooses Play game button.

Alternative Event Flows:

1. Player enters the level.
2. Player completes level.
3. Player gets point according to his/her performance.

### ➤ Use Case #3

---

Use Case: Rotate Walls

Primary Actor: Player

Stakeholders and Interests:

- Rotation of wall changes.

Pre-conditions:

- Player must be clicked to the level which he want.
- Level should be available.

Post-conditions:

Entry-conditions:

- Player can rotate walls by the menu at right.

Exit conditions:

- Player selects back button to exit level.

Success Scenario Event Flow:

1. Player decide which wall to rotate.

2. Player rotates wall 90 degree at the direction he want.

Alternative Event Flows:

1. Player exits level without finishing it.

#### ➤ Use Case #4

---

Use Case: Place Walls

Primary Actor: Player

Stakeholders and Interests:

- Place of wall changes.

Pre-conditions:

- Player must be clicked to the level which he want.
- Level should be available.

Post-conditions:

- Player must drag the wall to the available ground.

Entry-conditions:

- Player can drag the wall the any place.

Exit conditions:

- Player selects back button to exit level.

Success Scenario Event Flow:

1. Player decides which wall to move.
2. Player changes the location of wall by dragging.
3. Player puts the wall to the available place.

Alternative Event Flows:

1. Player exits level without finishing it.

➤ Use Case #5

---

Use Case: Use Boosters

Primary Actor: Player

Stakeholders and Interests:

- Tile type changes.
- Soldier moves

Pre-conditions:

- Player must be clicked to the level which he want.
- Level should be available.

Post-conditions:

- Player must drag the booster to the available ground or select soldier to move.

Entry-conditions:

- Player can apply the booster to the tiles or the soldier..

Exit conditions:

- Player selects back button to exit level.

Success Scenario Event Flow:

1. Player decides which booster to use.

Success Scenario Event Flow A:

1. Player changes the tile according to booster.

Success Scenario Event Flow B:

1. Player moves the soldier by the booster.

Alternative Event Flows:

1. Player exits level without finishing it.

➤ Use Case #6

---

Use Case: Play Time Trial

Primary Actor: Player

Stakeholders and Interests:

- The aim is to finish levels by solving puzzles as much as player can.
- Next levels are uploaded as player passes.

Pre-conditions:

- Player must be on main menu:

Post-conditions:

- Time has to end.

Entry-conditions:

- Player must click to the play time trial button on the menu.

Exit conditions:

- Player selects back button to exit level.
- Time ends so game ends.

Event Flow:

1. Player chooses Play Time Trial button.
2. Player passes levels in time.

Success Scenario Event Flow A:

- 3A. Time ends.

Alternative Event Flow B:

- 3B. Player chooses back button to return menu.

➤ Use Case #7

---

Use Case: Open Sandbox

Primary Actor: Player

Stakeholders and Interests:

- The aim is opening SandBox mode.

Pre-conditions:

- Player must be on main menu:

Post-conditions:

Entry-conditions:

- Player must click to Sandbox button on the menu.

Exit conditions:

- Player selects back button to exit sandbox mode.

Success Scenario Event Flow:

1. Player chooses SandBox button.

Alternative Event Flow A:

- 2A. Player chooses back button to return menu.

➤ Use Case #8

---

Use Case: Create Level

Primary Actor: Player

Stakeholders and Interests:

- The aim is to create a new level to be solved.
- Map must have a solution.

Pre-conditions:

- Sandbox button must be clicked.

Post-conditions:

Entry-conditions:

- Player must click to create level button.

Exit conditions:

- Player selects back button to return the main menu.

Success Scenario Event Flow:

1. Player chooses to create level button.
2. An environment to create a level show up.

Alternative Event Flow A:

- 3A. Player chooses back button to return menu.

## ➤ Use Case #9

---

Use Case: Play Level

Primary Actor: Player

Stakeholders and Interests:

- The aim is to play a level which is created by users.

Pre-conditions:

- Sandbox button must be clicked.

Post-conditions:

Entry-conditions:

- Player must click to play level button.

Exit conditions:

- The player selects back button to return the main menu.



Success Scenario Event Flow:

1. Player chooses to play level button.
2. A level shows up.

Alternative Event Flow A:

- 3A. Player chooses back button to return menu.

➤ Use Case #10

---

Use Case: Open How to Play

Primary Actor: Player

Stakeholders and Interests:

- The aim is to open and learn how to play.

Pre-conditions:

- How to play button must be clicked.

Post-conditions:

Entry-conditions:

- Player must click to how to play button.

Exit conditions:

- Player selects back button to return the main menu.

Success Scenario Event Flow:

1. Player chooses to how to play button.
2. A description of the game with images and texts shows up.

Alternative Event Flow A:

- 3A. Player chooses back button to return menu.

➤ Use Case #11

---

## Use Case: Place Warriors

Primary Actor: Player

Stakeholders and Interests:

- Player places warriors to make a custom level.

Pre-conditions:

- Sandbox button must be clicked.

Post-conditions:

- Placement must be in a valid tile.

Entry-conditions:

- Player must select and drag a warrior from the panel to the map.

Exit conditions:

- Player places the warrior to a valid tile.
- Player releases the warrior to a invalid place or tile.
- Player releases the warrior back to the panel.

Success Scenario Event Flow:

1. Player selects and drags the warrior to map.
2. Player releases the warrior to a valid tile.
3. Warrior is placed to the tile successfully.

Alternative Event Flows:

if player releases the warrior to a invalid place or puts it back to the panel:

1. Warrior returns to the panel.

### ➤ Use Case #12

---

## Use Case: Decide Boosters

Primary Actor: Player

Stakeholders and Interests:

- Player choses boosters to use in the custom made level.

Pre-conditions:

- Sandbox button must be clicked.

Post-conditions:

Entry-conditions:

- Player must click to the boosters from the panel to include them.

Exit conditions:

- Player checks boosters.
- Player unchecks boosters.

Success Scenario Event Flow:

1. Player checks boosters.
2. Boosters gets included to the level when custom level making is completed.

Alternative Event Flows:

if player doesn't complete the custom level:

1. Level and booster choices are not saved.

## ➤ Use Case #13

---

Use Case: Decide Walls

Primary Actor: Player

Stakeholders and Interests:

- Player chooses different shaped walls to use in the custom made level.

Pre-conditions:

- Sandbox button must be clicked.

Post-conditions:

Entry-conditions:

- Player must click to the walls from the panel to include them.

Exit conditions:

- Player checks walls.
- Player unchecks walls.

Success Scenario Event Flow:

1. Player checks walls.
2. Walls gets included to the level when custom level making is completed.

Alternative Event Flows:

if the player doesn't complete the custom level:

1. Level and wall choices are not saved.

## ➤ Use Case #14

---

Use Case: Decide Ground Type

Primary Actor: Player

Stakeholders and Interests:

- Player places different tile types to customize the level map.

Pre-conditions:

- Sandbox button must be clicked.

Post-conditions:

Entry-conditions:

- Player must drag tiles from the panel to the map.

Exit conditions:

- Player places the tile to the map.
- Player places the tile back to the panel.

Success Scenario Event Flow:

1. Player places the tile to the map.
2. The map is saved when the customization complete.

Alternative Event Flows:

if player doesn't complete the custom level:

1. Level and custom map are not saved.

## ➤ Use Case #15

---

Use Case: Open Shop

Primary Actor: Player

Stakeholders and Interests:

- Player enters the shop to buy boosters.

Pre-conditions:

- Shop button in the menu must be clicked.

Post-conditions:

Entry-conditions:

Shop button in the menu must be clicked.

Exit conditions:

- Player clicks back button in the shop panel to return to the menu.

Success Scenario Event Flow:

1. Player clicks the shop button while in the menu.
2. Shop panel appears.

Alternative Event Flows: if player does not complete the process of spending gold and presses the back button,

1. The amount of gold will not be deducted from the inventory of the player.

➤ Use Case #16

---

Use Case: Open Options

Primary Actor: Player

Stakeholders and Interests:

- Player selects Options button
- System shows Customization and Volume Options

Pre-conditions:

- Player must be in the main menu.

Post-conditions:

Entry-conditions:

- Player should see current volume settings and default customization settings.

Exit conditions:

- Player selects back button.

Success Scenario Event Flow:

1. Player chooses Options button.

Alternative Event Flows:

1. Player chooses back button to return menu.

➤ Use Case #17

---

Use Case: Change Volume Options

Primary Actor: Player

Stakeholders and Interests:

- Player scrolls the volume bar of system music.
- System gets the given volume and adjusts the game music accordingly .

Pre-conditions:

- Player must be clicked Options button.

Post-conditions:

Entry-conditions:

- Player should choose the appropriate music volume.

Exit conditions:

- Player selects back button to exit Change Volume panel.

Success Scenario Event Flow:

1. Player chooses Options button.

Alternative Event Flows:

1. Player chooses Options button.
2. Player chooses desired music volume.
3. System changes the music volume accordingly.

➤ Use Case #18

---

Use Case: Change Customization

Primary Actor: Player

Stakeholders and Interests:

- Player selects Change Customization button.
- System displays the available customization options.

Pre-conditions:

- Player must be clicked Options button.

Post-conditions:

Entry-conditions:

- Player must select a customization option.

Exit conditions:

- Player selects back button to exit Change Customization panel.

Success Scenario Event Flow:

1. Player presses Options button.
2. Player presses Change Customization button.
3. Player chooses the desired customization option.

➤ Use Case #19

---

Use Case: Open Credits

Primary Actor: Player



#### Stakeholders and Interests:

- Player selects Credits button.
- System shows credits about developers.

#### Pre-conditions:

- Player must be in the main menu.

#### Post-conditions:

#### Entry-conditions:

- Player should see the credits text content.

#### Exit conditions:

- Player selects back button.

#### Success Scenario Event Flow:

1. Player chooses Credits button.

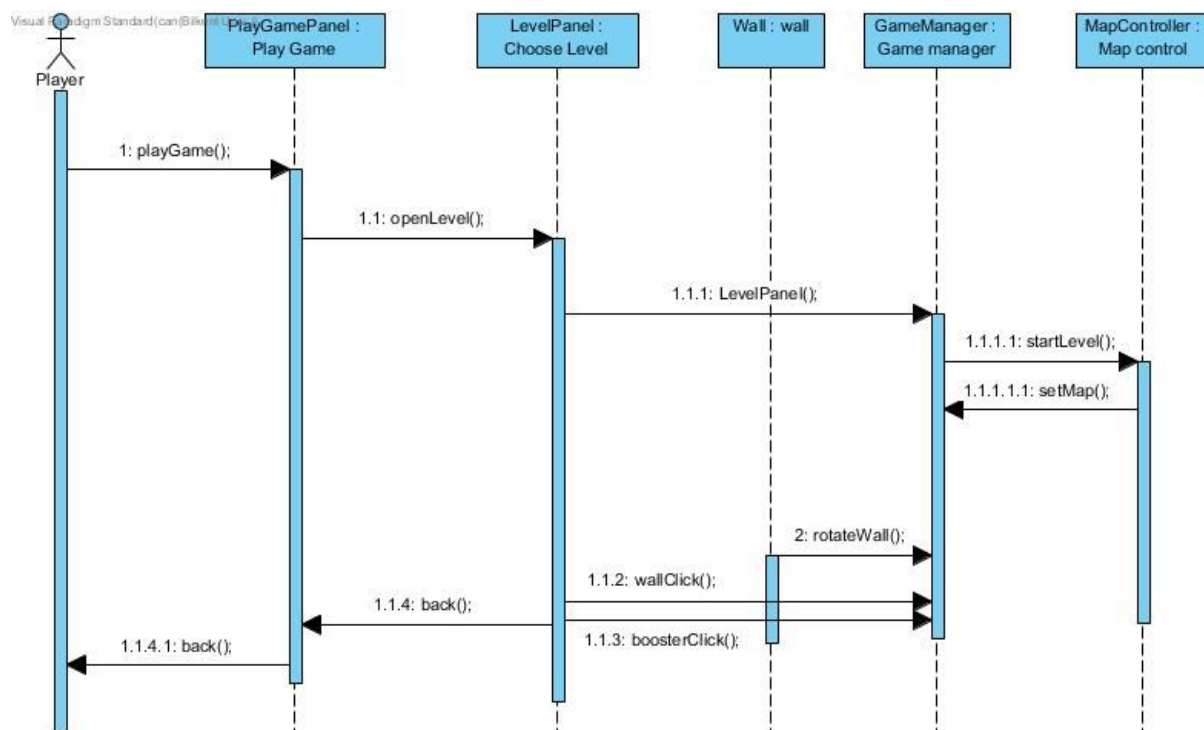
#### Alternative Event Flows:

1. Player chooses back button to return menu.

## 5.2.Dynamic Models

### 5.2.1.Sequence Diagram

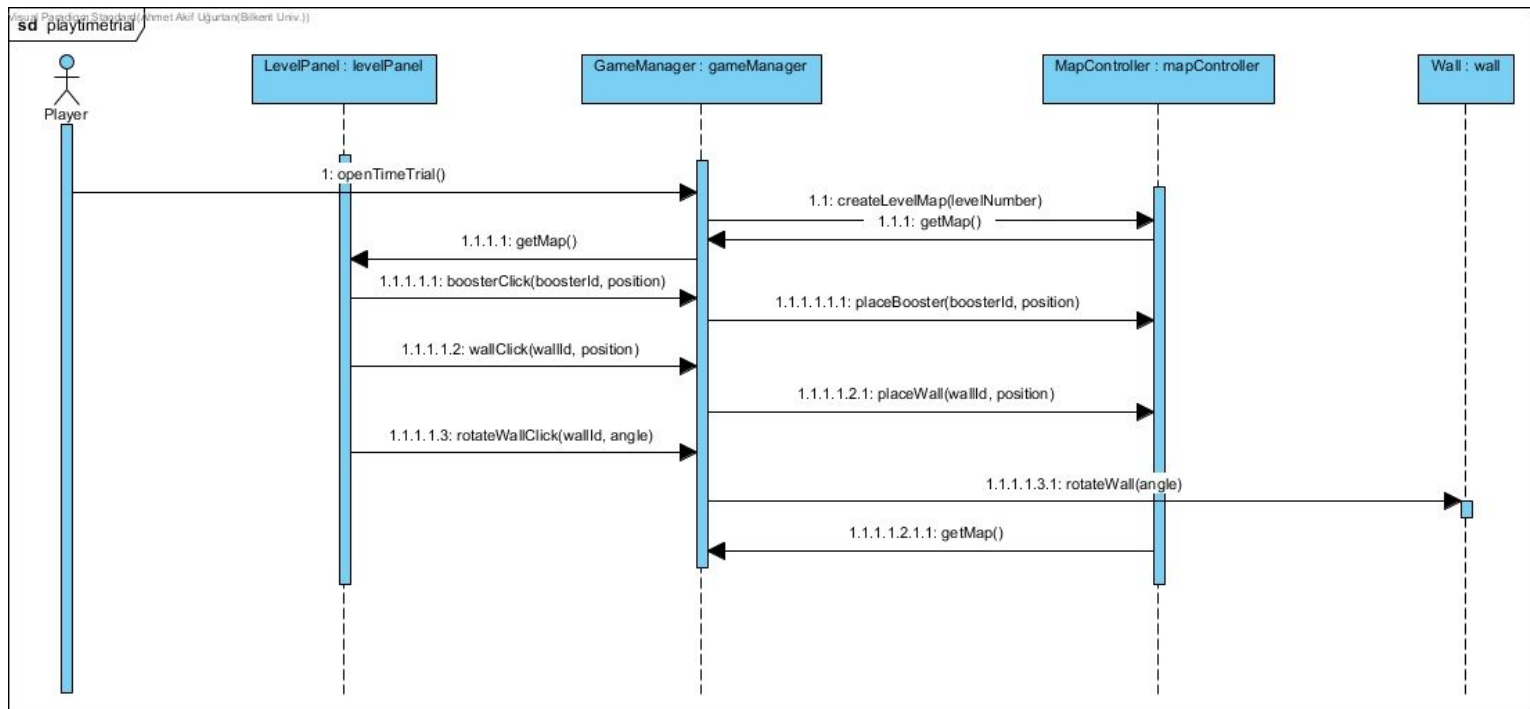
#### 5.2.1.1.Play Game Sequence



**Scenario:** User wants to start the game. He/she opens the play game panel from the menu panel. After opening play game panel he/she is able to choose levels but these levels will be available when you pass previous levels. With the level panel, the user selects the level he wants and the game will be initialized by the game manager. After that game, GameManager sends the level's information to the map controller and map will be set. The user can play the game afterward. User able to rotate walls from the UI. In order to place walls, he/she has to place available place. There are two types of boosters: 1) Human

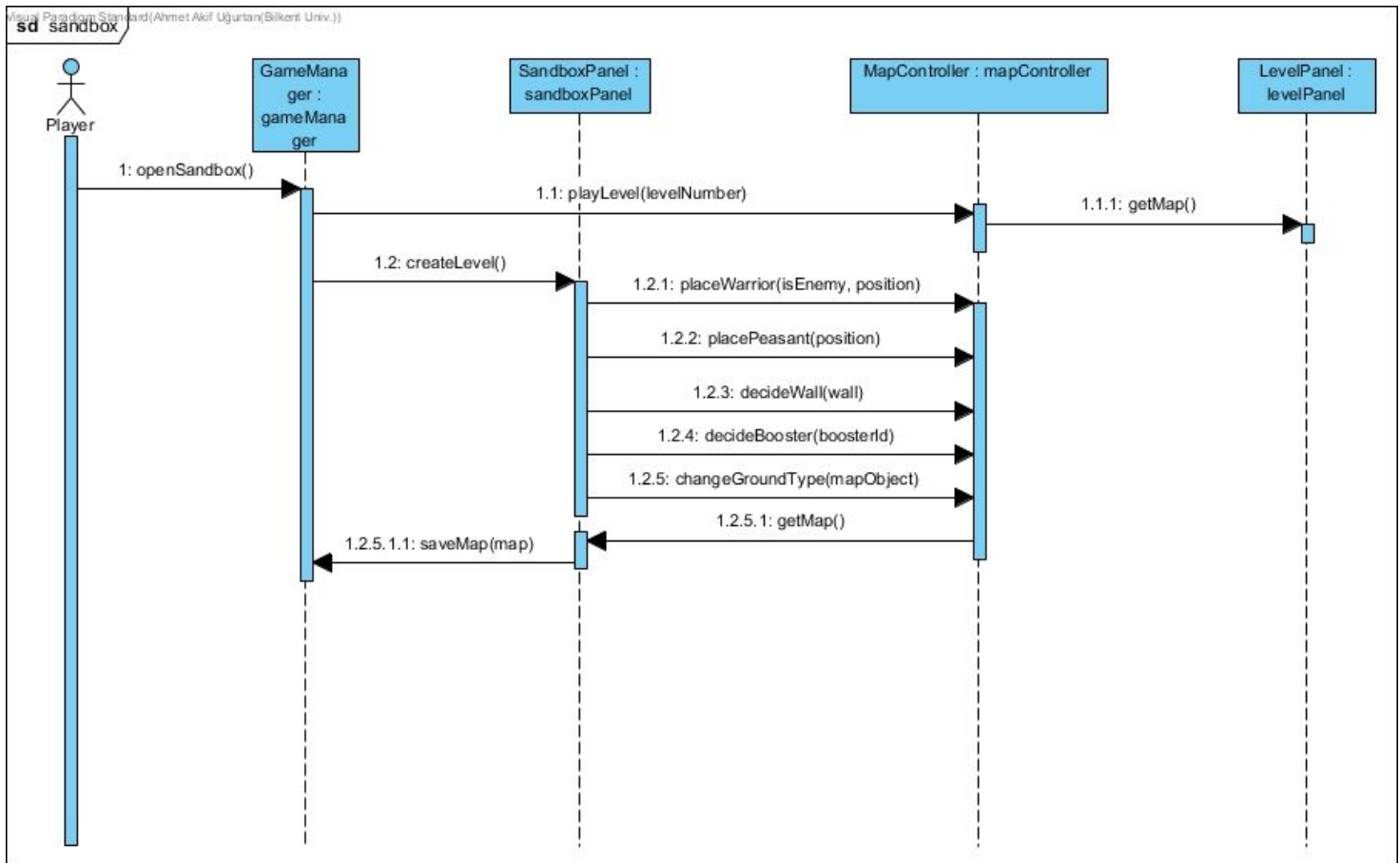
Booster 2)Tile Booster. The user needs to use these boosters to the correct places to advance the game. When the user finishes the level he/she returns to the level panel.

### 5.2.1.2.Play Time Trial Sequence



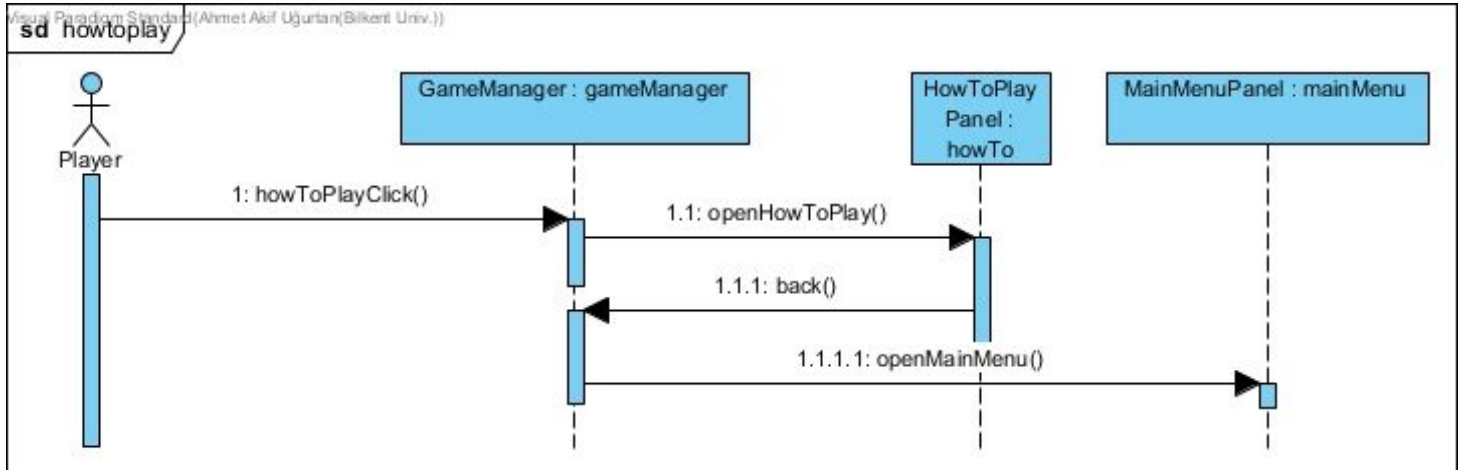
**Scenario:** User wants to play time trial challenge. He/she clicks time trial button on the main menu. GameManager takes the request and chooses a random level and sends a request to the mapController to create the randomly chosen level. Then, MapController returns the current map to the LevelPanel via GameManager. LevelPanel displays the map, the boosters, the point etc. to the user. User clicks a booster then a zone on the map to user or user chooses a wall then a zone on the map to put the wall centered on that zone or user chooses a wall and clicks the rotate button on the wall to rotate wall. All these actions with id and position first go to GameManager then MapController. MapController performs necessary modifications and returns the current map. LevelPanel displays the current map.

## 5.2.1.3.SandBox



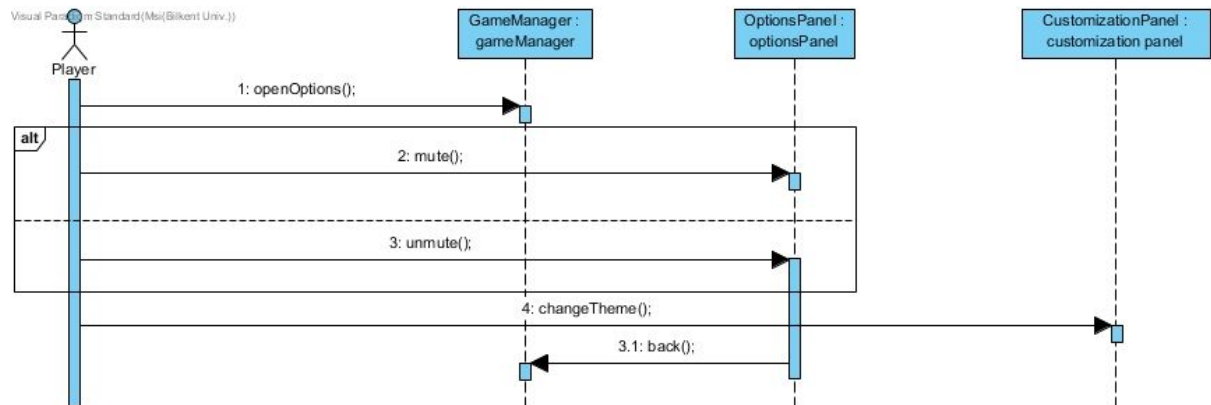
**Scenario:** User wants to create or play a custom level. User clicks sandbox button on the main menu. GameManager opens a new panel for the user to choose to create or play. If the user chooses to play a level, the level map appears and the user chooses which map to play and plays the custom level map. If the user chooses to create a new level, SandboxPanel opens. The user can decide which type of walls will be used, or which boosters will be available or where soldiers will stay or where peasants will stay or the type of zones. All these actions will be done through onclicks and panels in the SandboxPanel and all changes will be saved on MapController class.

## 5.2.1.4. How to Play



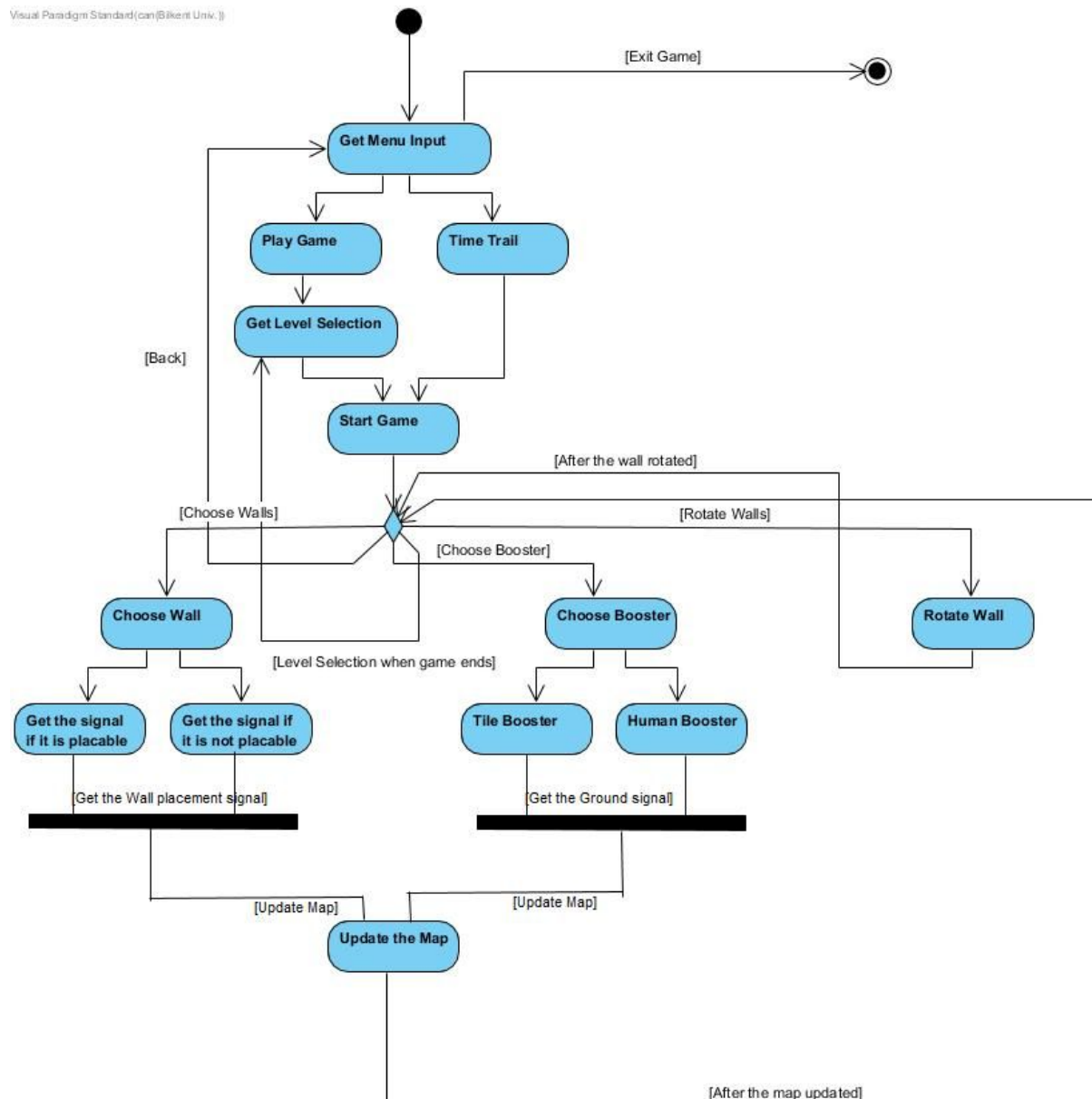
**Scenario:** The user wants to learn how to play the game. The user clicks howToPlay button on the main menu. GameManager opens HowToPlayPanel. The user observes the information with images and texts about the game. The user clicks back button and returns to main menu.

## 5.2.1.5 Options



**Scenario:** User wants to change options so he clicks options button. Game Manager opens the options panel. Here the user can mute the sounds or if it's muted he can unmute it by clicking the checkbox. Player can also change the theme by clicking any themes button. Player can go back with or without making any changes by clicking back button.

## 5.2.2. Activity Diagram



Initially, the system waits for the user input. The user might start the game, or exit the game. If the user exits, the system goes to the final node. If the user starts the game, he/she can start either normal game or time trial challenge. In normal game option, the user selects a level to play. In time trial, the game chooses the level automatically. Both ways continue with starting the game. Inside the game, the user can rotate walls to try different solutions.

The user can try to place walls, if the zone is placable then the game updates the map with walls. The user can use boosters by choosing which type he/she is gonna use then after that the game updates the map. When the game ends, the user can choose another level to play. The user can go back and exit the game anytime.

### 5.3.Object and Class Model

**Booster:** Booster class is a subclass of MapController class and it represents the boosters in the game. In this game, we provide some boosters via shop system. The player can buy and use boosters in the game. These boosters divide into 2 subclasses.

**HumanBooster:** This booster is a subclass of Booster class and it represents the boosters which are only for human objects. This class divides into 2 subclasses.

**Hide:** This booster is a subclass of HumanBooster and it allows the player to hide a soldier with bush.

**Move:** This booster is a subclass of HumanBooster and it allows the player to move a soldier for one tile.

**TileBooster:** This booster is a subclass of Booster class and it represents the boosters which are only for tile objects. This class divides into 2 subclasses.

**Dig:** This booster is a subclass of TileBooster and it allows the player to dig a single mountain tile.

**Fill:** This booster is a subclass of TileBooster and it allows the player to fill a single lava tile.

**Tile:** Tile class is a subclass of MapObject class and it represents the tiles in the map. It has two operators in order to return whether a tile can be filled or dug. Tile class divides into six subclasses.

**Ground:** Ground class is a subclass of Tile class and it represents a zone on the map as a ground.

**Lava:** Lava class is a subclass of Tile class and it represents a zone on the map as lava.

**Mountain:** Mountain class is a subclass of Tile class and it represents a zone on the map as a mountain.

**Bush:** Bush class is a subclass of Tile class and it represents a zone on the map as bush.

**Tower:** Tower class is a subclass of Tile class and it represents a zone on the map as a tower.

**WallTile:** WallTile class is a subclass of Tile class and it represents a zone on the map as a wall if some part of wall stays in the zone.

**OptionsPanel:** Information about the sound settings kept in this class. Player can adjust volume level or can mute/unmute all sounds all together by this class. CustomizationPanel class is instanced in this class.

**CustomizationPanel:** Images and the buttons of the themes are kept in this class. Selected theme is also kept here. Player's theme selection is done through this class.

**GameManager:** GamaManager class is responsible for transitions between the panels and creation of the new game. Depending on the user input it might get the level choice from the PlayGamePanel and start the chosen level or TimeTrial class calls playTimeTrial function to initialize the time trial mode.

**ShopPanel:** This class keeps the gold amount player has. When player clicks to a booster to buy in the shop panel this class handles it.

**MainMenuPanel:** This class handles all the button clicks in the main menu. When a player clicks any button in the menu MainMenuPanel instance invokes the GameManager instance to load the panel associated with that button.



**PlayGamePanel:** The panel that appears after clicking the play game button associated with this class. When player clicks a level button in the level choice panel this class invokes GameManager to initialize it.

**TimeTrial:** This class keeps the information about the timeTrial mode such as, high score, levelCount and time. Changing level during the gameplay handled here as well.

**HowToPlayPanel:** The panel that appears after clicking the How to Play button in the Main Menu. It keeps the instructions and tips about how to play the game and the aim of game with image file.

**CreditsPanel:** The panel that appears after clicking the Credits button in the Main Menu. It keeps and displays the data about the developers of the game.

**LevelPanel:** This class keeps the data about the levels panel that appears to the user after clicking the Play Game button in order to determine which level he/she wants to play. It keeps the levels that are available to the user, the list of all levels and the collection of different combinations of walls for different levels in order to display as Levels panel.

**Wall:** This class keeps variations of different types of walls and rotateWall(angle: int) method which does the rotation operation in game.

**MapController:** This class keeps the array of all default maps which are different in each level. It holds the map and modifies in the change of positions of objects.

**MapObject:** This class keeps the data of all objects including human characters and tiles. It holds the position of each object on the map and it sends the data of position of objects to MapController class in order to modify the display of the game. isGround() method of this class determines whether the given position Point is ground or consists of any object.

**Resources:** This class keeps the ArrayList of all image sources that are used inside the map display of the game.

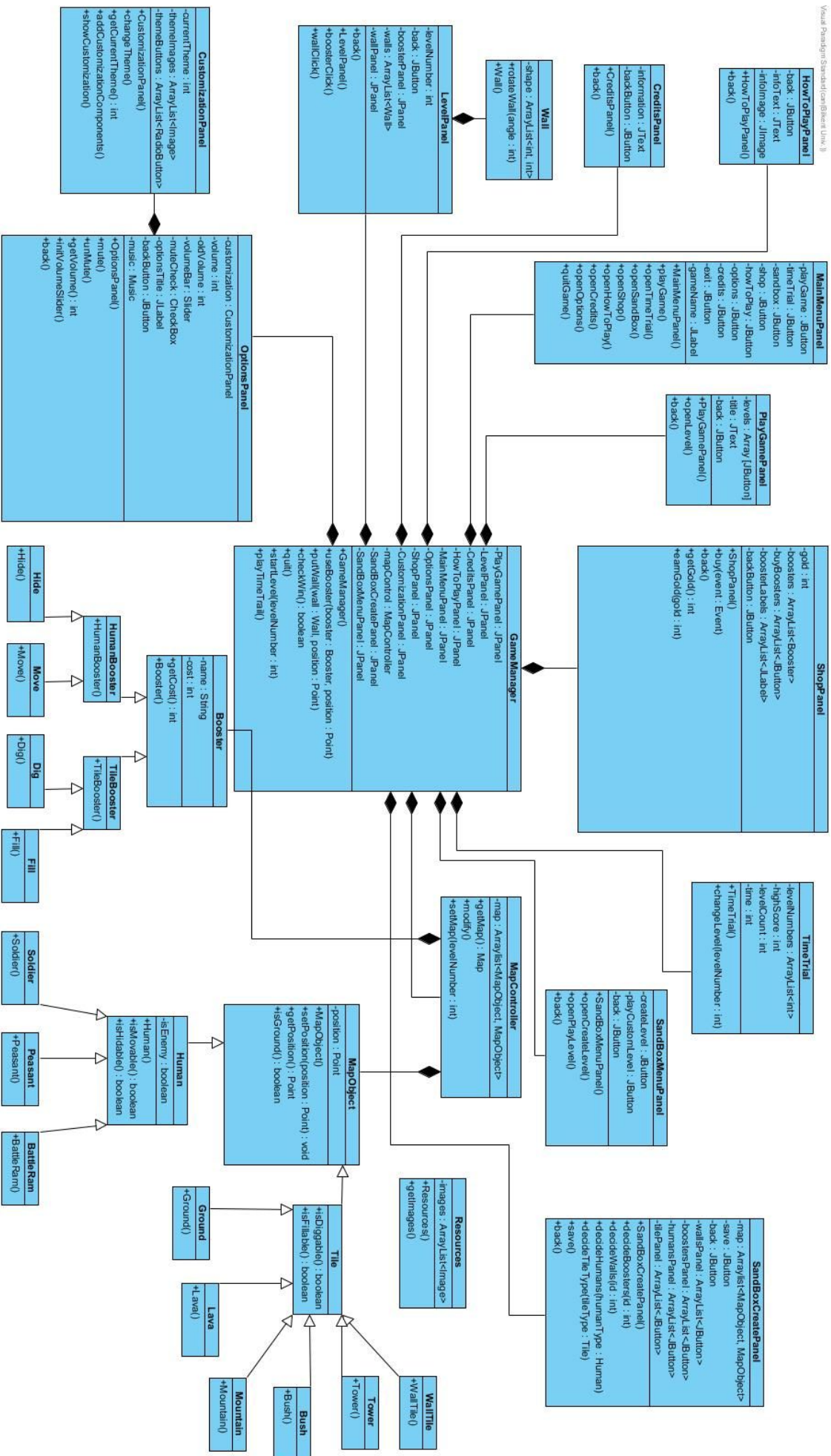
**Human:** This class is a subclass of MapObject class and it represents the human characters inside the map. It keeps the data about whether the human character is movable

(or hideable) or not. It has isEnemy attribute in order to determine the human character is whether enemy or not.

**Soldier:** Soldier class is a subclass of Human class and it represents the soldier characters on map of the game.

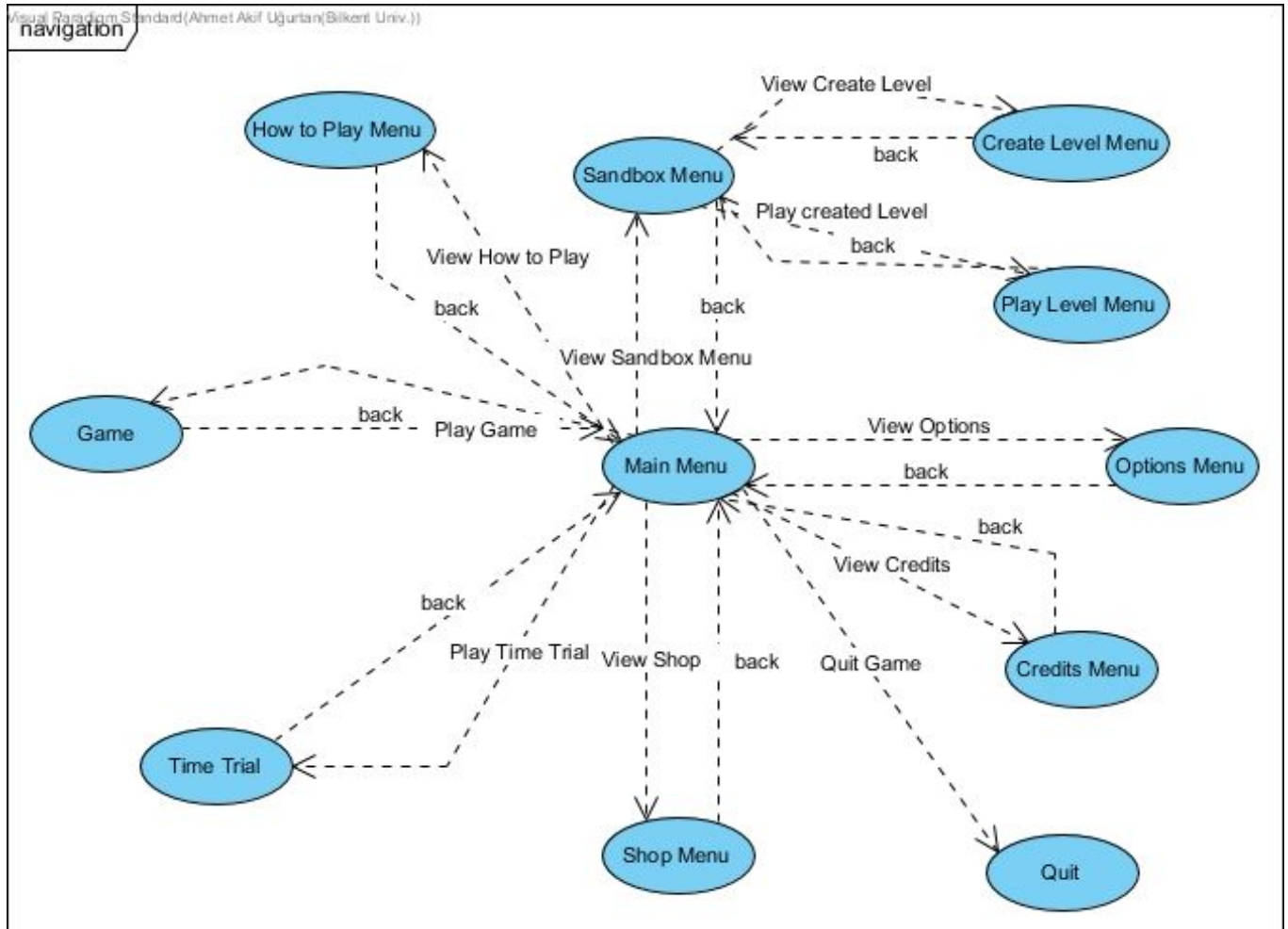
**Peasant:** Peasant class is a subclass of Human class and it represents the peasant characters on map of the game.

**BattleRam:** BattleRam class is a subclass of Human class and it represents special character which requires double wall in order to border it and complete the level.



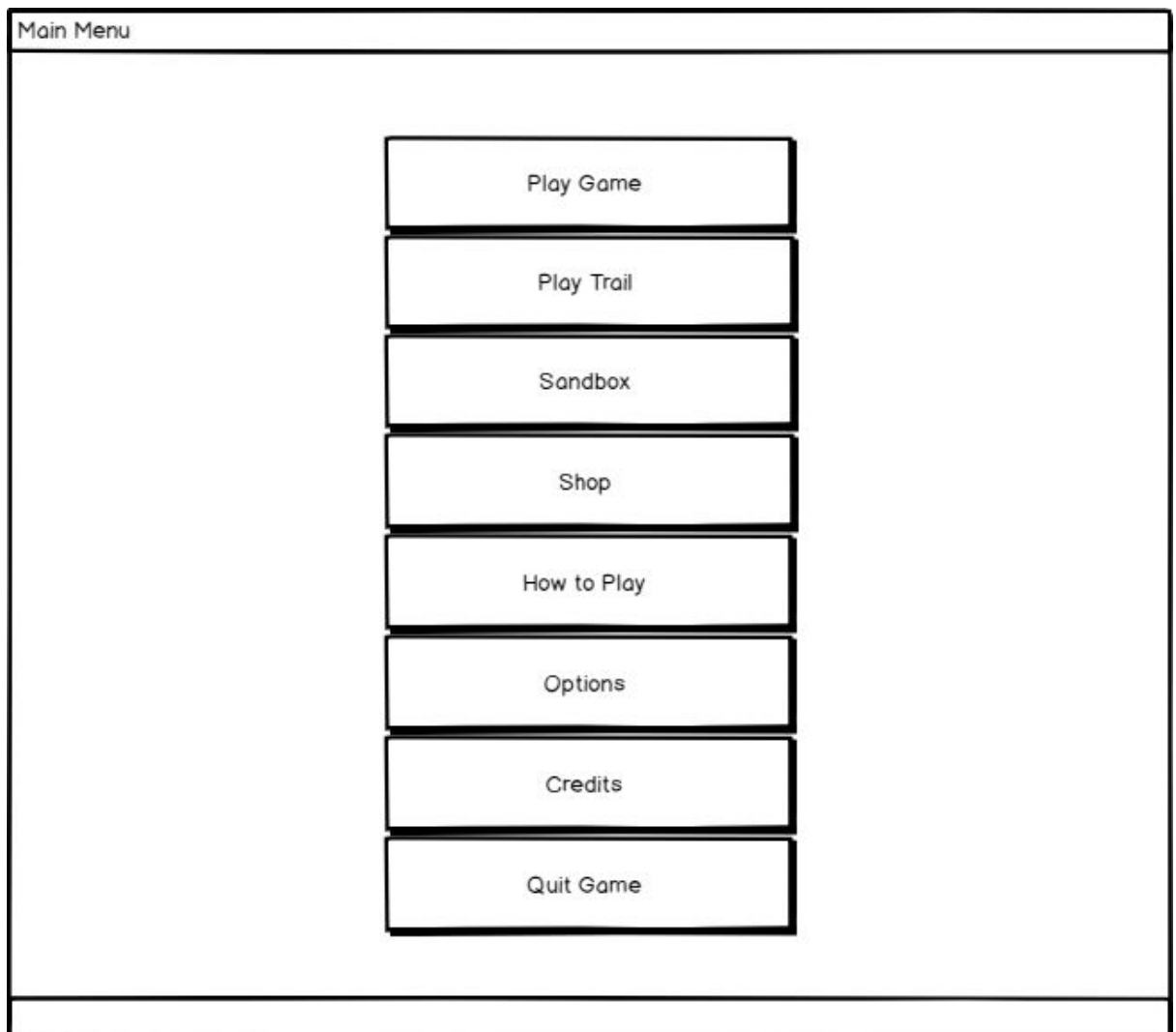
## 5.4. User interface

### 5.4.1. Navigational Paths



## 5.4.2.Mockups

### 5.4.2.1. Main Menu



5.4.2.2.

Choose

a

Level

Choose Level

Level 1

Level 2

Level 3

Level 4

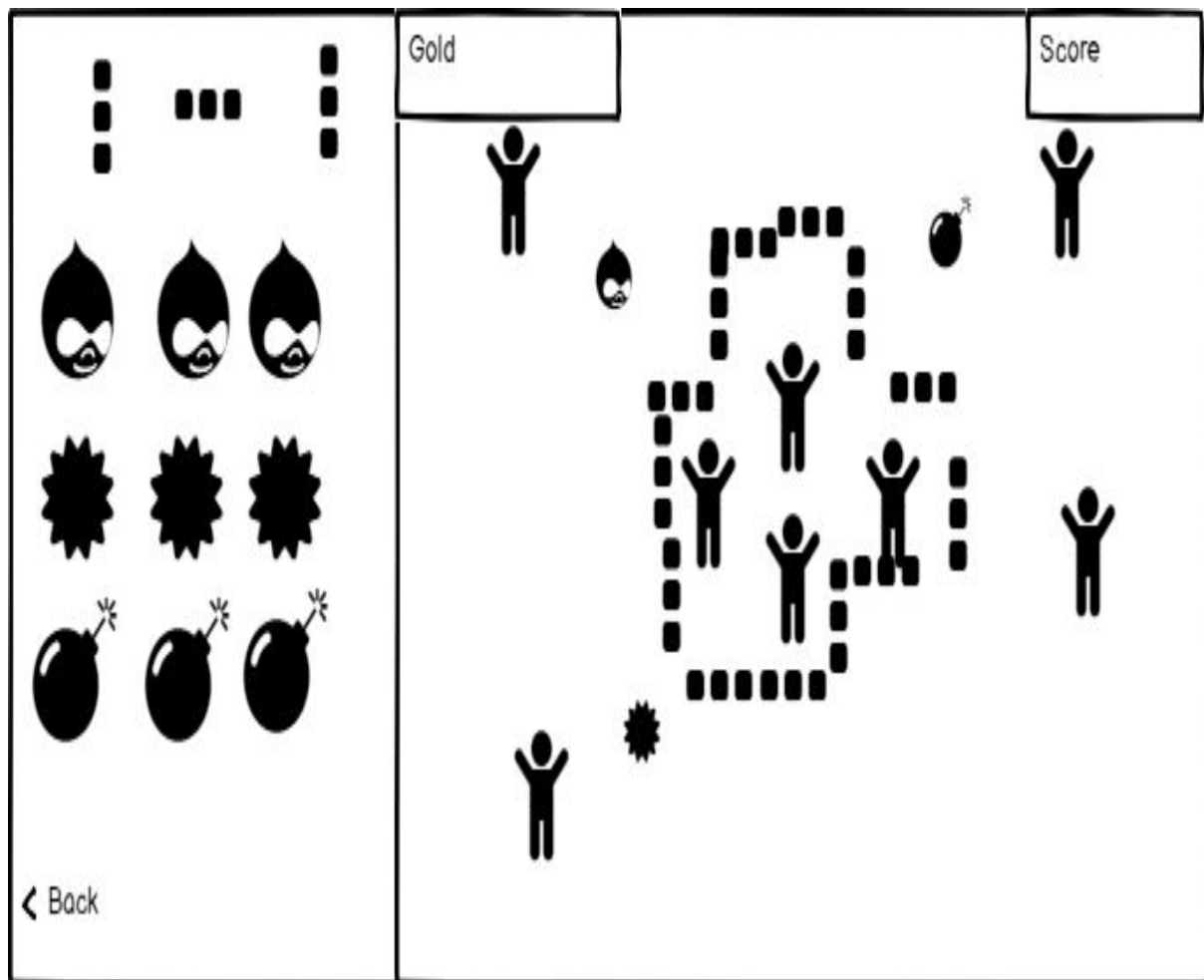
⌵

⏪ Back

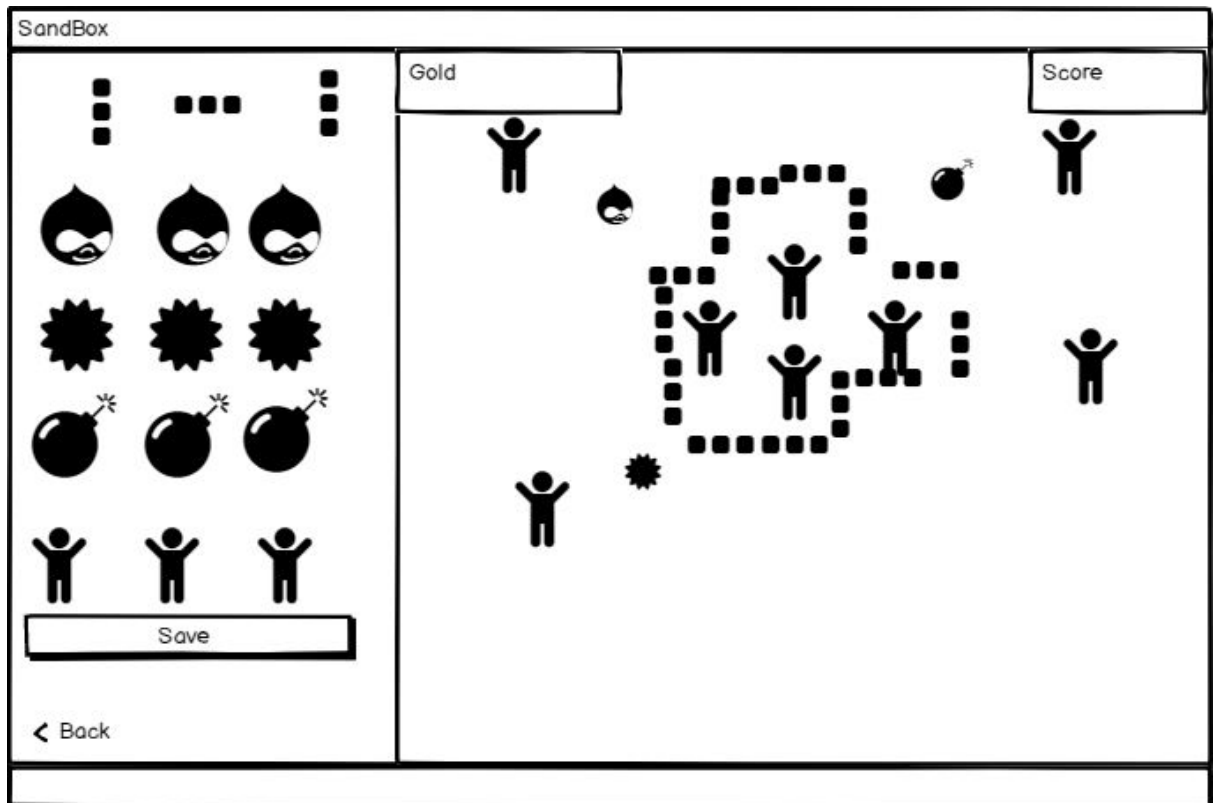
5.4.2.3.

Play

game



## 5.4.2.4 SandBox







## 5.4.2.5 Options

Options

Soldier one (o)	Pleasant one (o)	Wall one (o)
Soldier two (o)	Pleasanttwo (o)	Wall two (o)
Soldier three (o)	Pleasantthree (o)	Wall three (o)

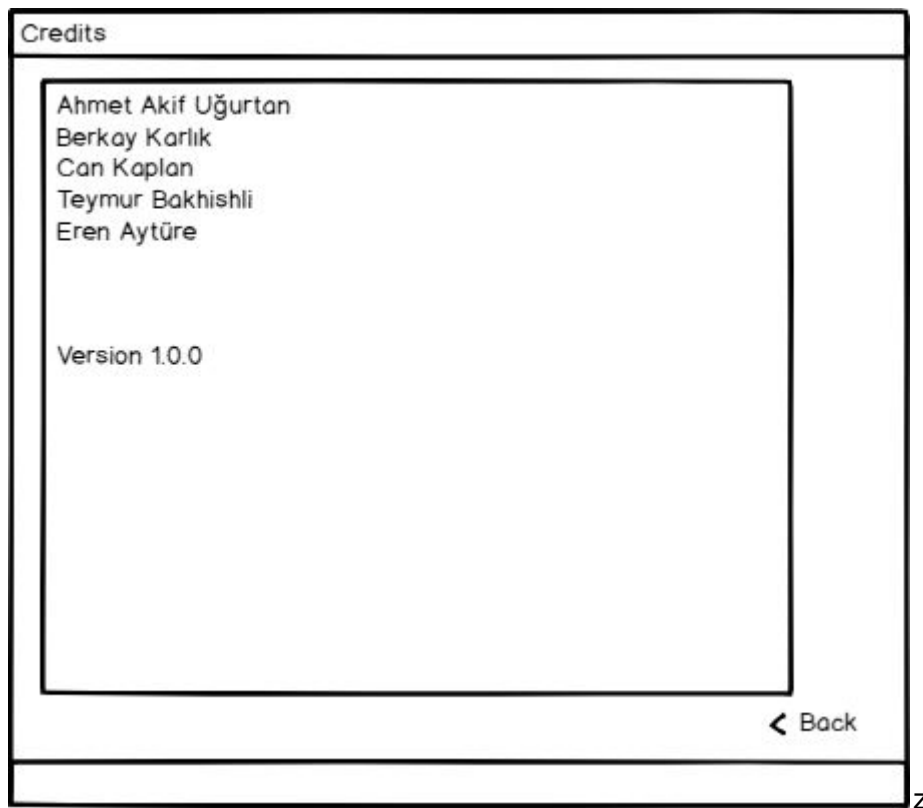
 

Save Changes

< Back



## 5.4.2.8 Credits



## 6. References

Original game:

<https://www.smartgames.eu/uk/one-player-games/walls-warriors>

Project repository:

<https://github.com/ahmtakf/nomanleft>