



CS 319 - Object-Oriented Software

Final Report

# Colony Wars

Supervisor: Bora Güngören

İbrahim Taha Aksu

Ahmet Emre Nas

İzel Gürbüz

Kaan Çakmak

## **Table of Contents**

### **1. Introduction**

### **2. Application Setup**

### **3. The User's Manual**

### **4. Changes Done During The Implementation Stage**

### **5. Incomplete Parts**

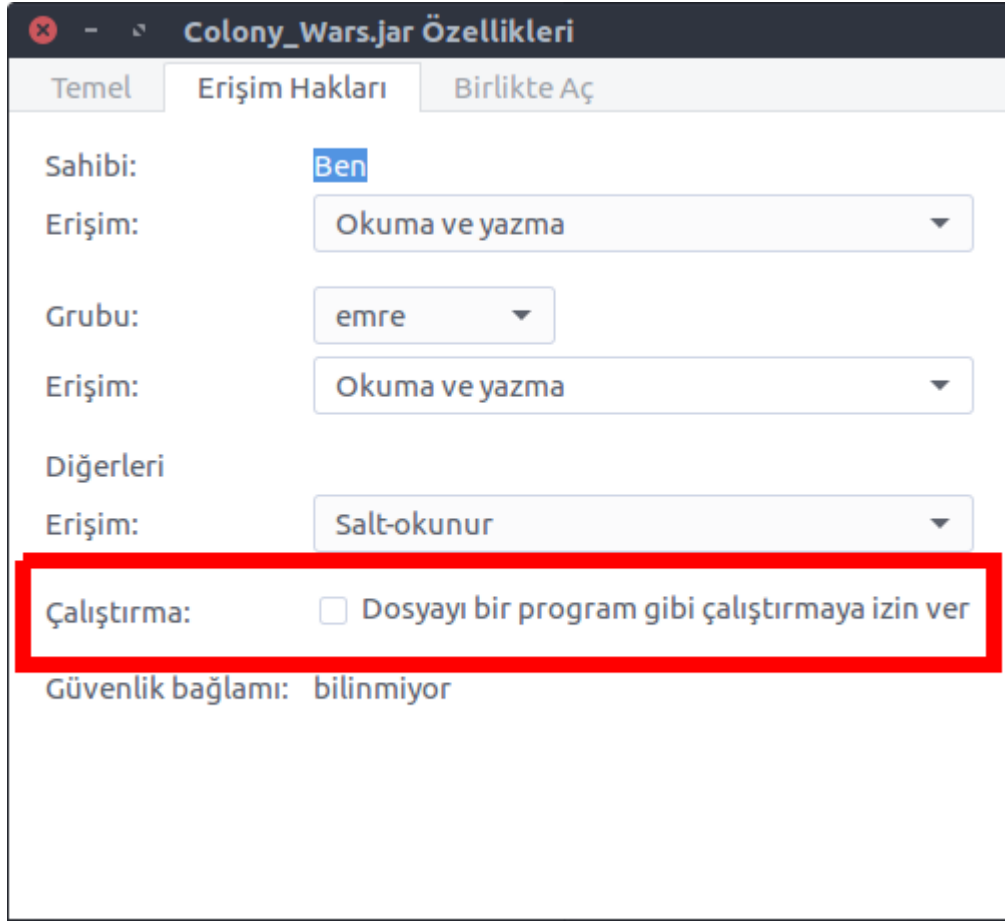
## 1.Introduction

Colony Wars is a time based strategy game that is visualised on 2D map with several game objects represented as icons. Game is played against the AI and could be played in various difficulty levels. Game map consists of base buildings that might belong to AI or the player. Game has two game modes that are campaign and skirmish. In the campaign mode the nation and difficulty levels are stable for every level whereas in the skirmish mode user chooses them himself/herself. The main reason in developing this game is to provide users with a time lesiure application which also requires strategical thinking and also improves this ability of users.

## 2. Application Setup

In order to run the game user must have Java in their computer. Game does not requiere any additional setup than java since all needed resources like (Images, Sounds, Fonts) are already in jar file. Simply user can open the jar file on github and start to play. Also user can download the source code of the project to customize game. But it is highly recommended to use Netbeans IDE if the user want to play with the source code. Game can be downloaded from (<https://github.com/emrenass/CS319-Project>) .

**Important Note:** Linux users need to right click to jar file select Properties->Permission-> Check Allow executing file as program



Colony\_Wars.jar Özellikleri

Temel Erişim Hakları Birlikte Aç

Sahibi: Ben

Erişim: Okuma ve yazma

Grubu: emre

Erişim: Okuma ve yazma

Diğerleri

Erişim: Salt-okunur

Çalıştırma: ☐ Dosyayı bir program gibi çalıştırmaya izin ver

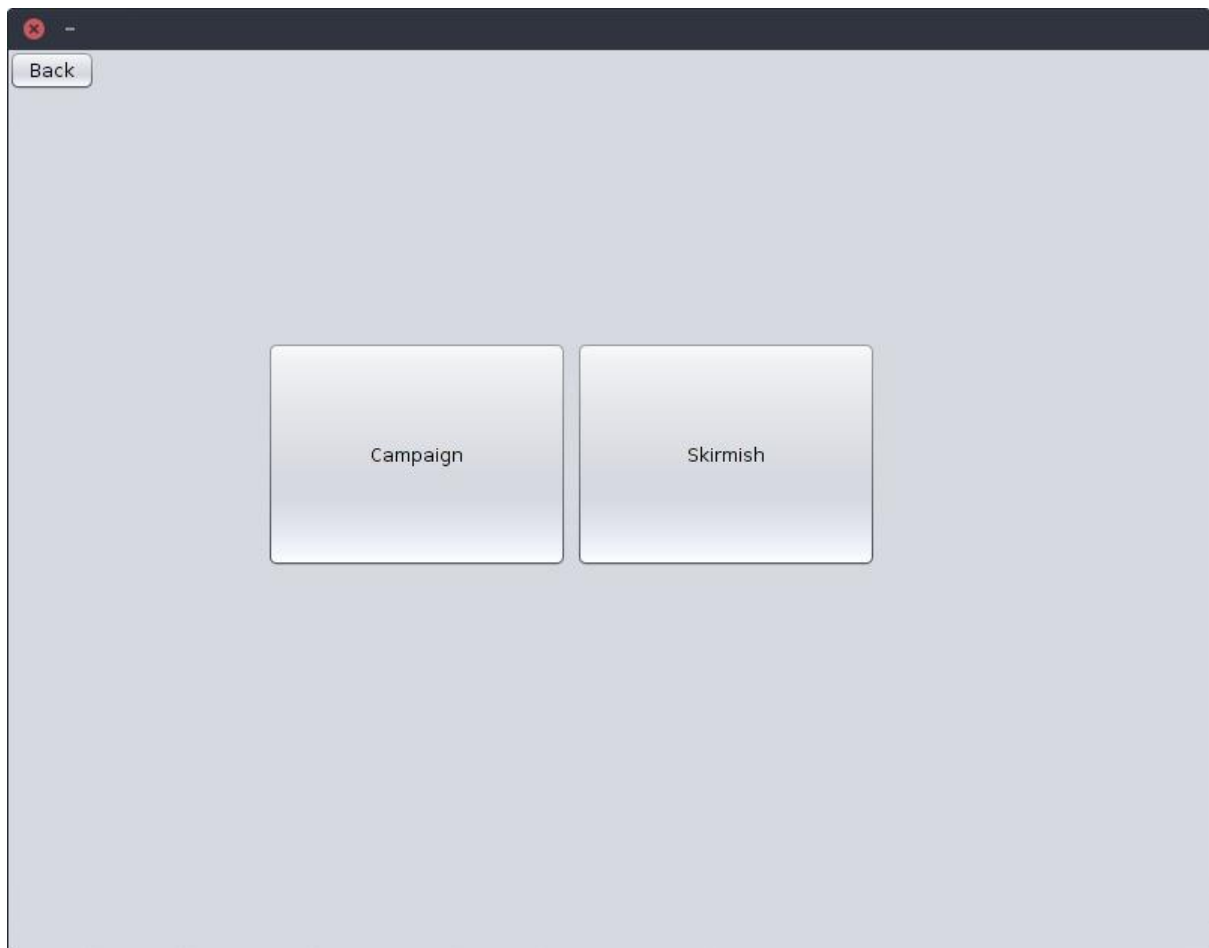
Güvenlik bağlamı: bilinmiyor

### 3.The User's Manual

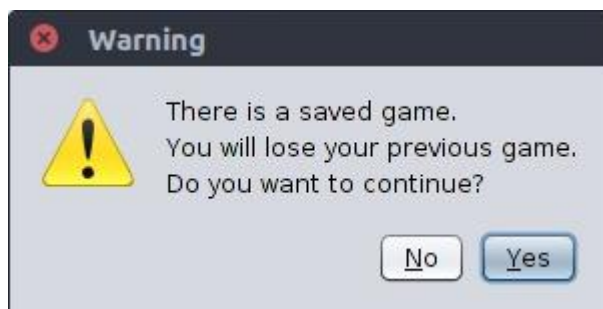
With the execution of the application the first thing that user encounters is the main menu window. This window includes several functionalities.



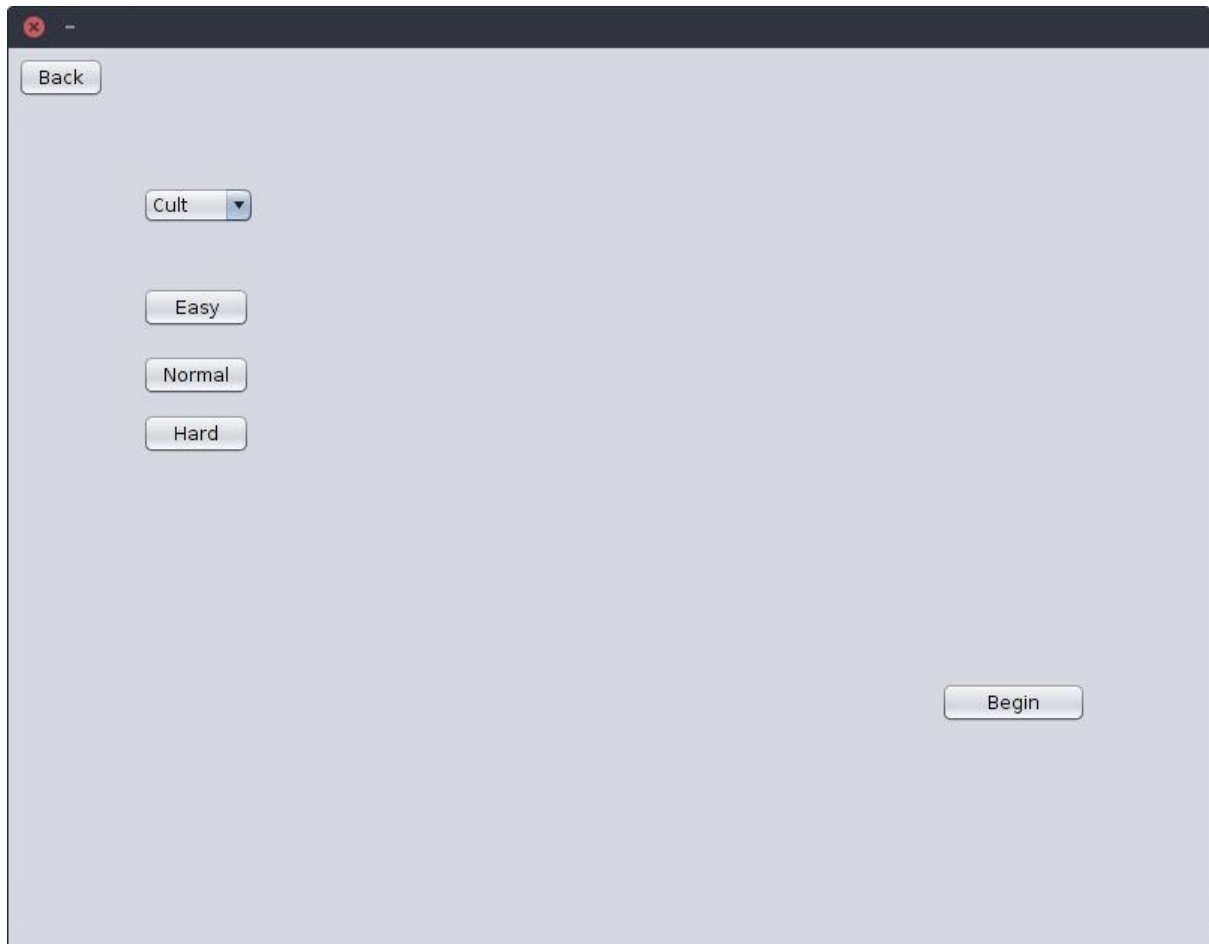
- The user can choose to play a new game which directs him to the game mode window and skirmish and game windows accordingly. The back button when pressed directs the user to back to main menu.



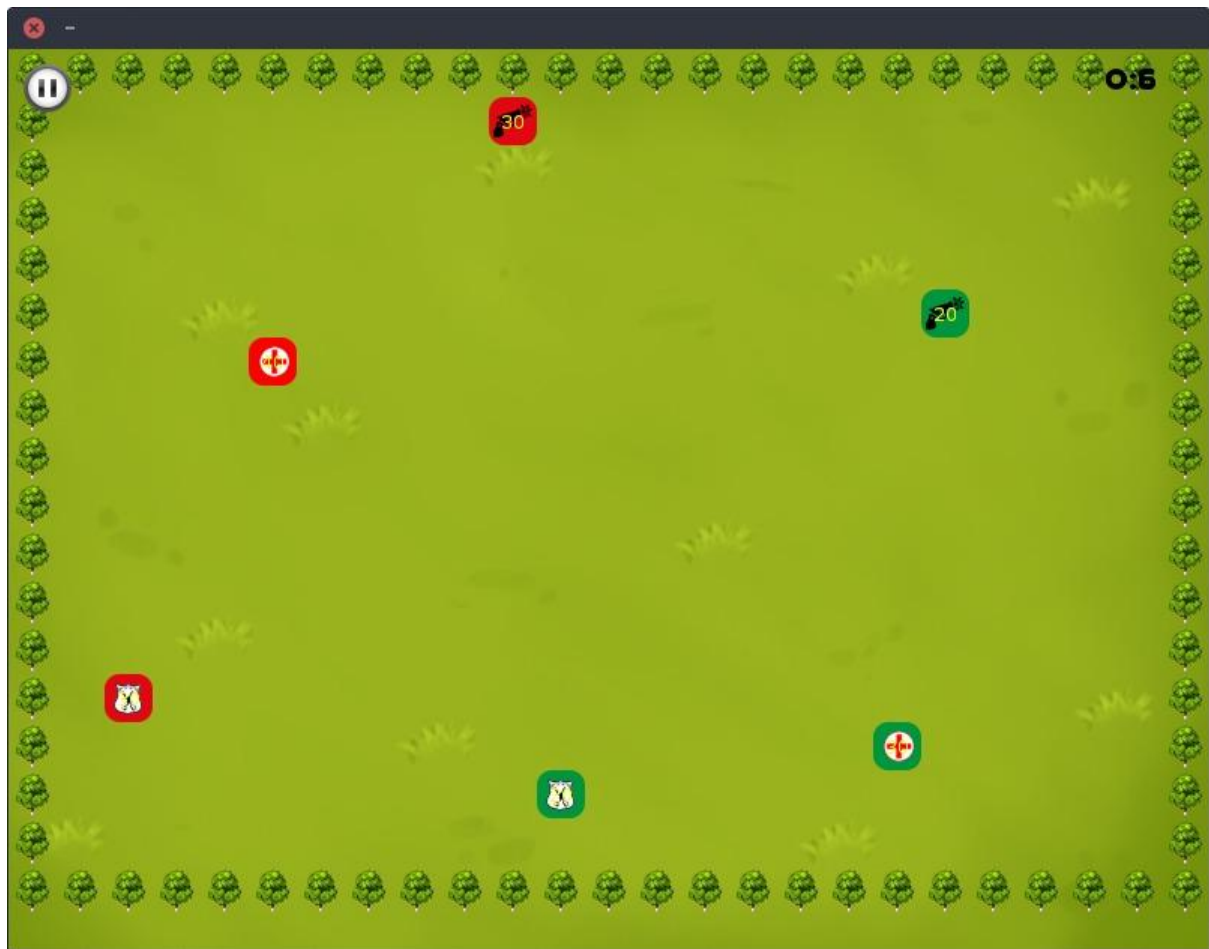
- The player can choose to play campaign mode which would pop up a dialog box indicating the possible loss of previous saved levels. If the user continues the levels s/he already played would be erased and he would start the campaign mode from scratch.



The player may as well choose to play the skirmish mode in which case they would need to choose a nation and the difficulty level( whose scope varies from 0 to 3). The skirmish game mode includes only one map available in it since this mode is to practice various nations against AI with different levels. The back button would take the user back to game mode window when pressed.

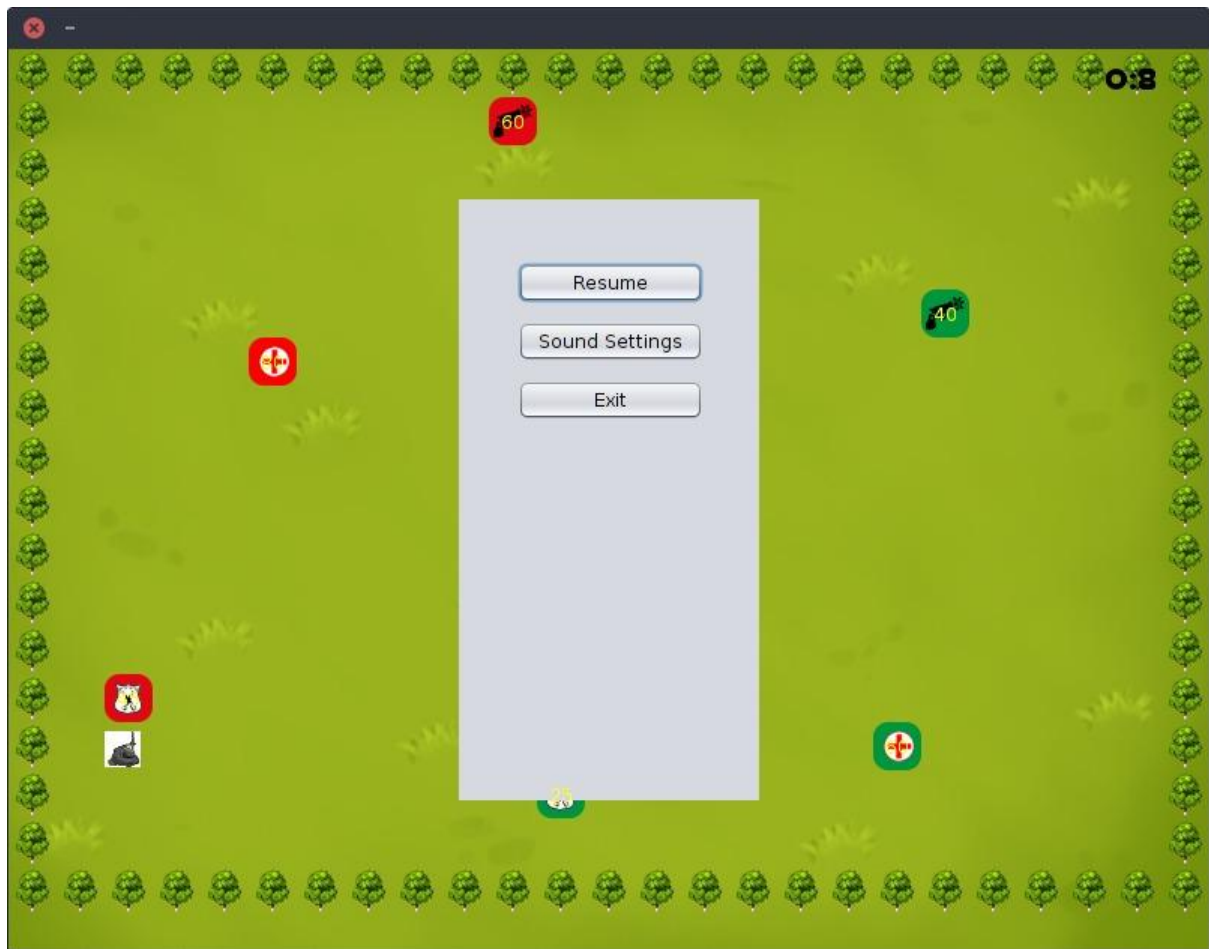


- Choosing any game mode and attributes the player is then directed to the game window where s/he actually is in interaction with the game through graphics. During the gameplay user might choose to pause the game using the pause button on the upper left corner of the game window. this would pop the pause window up.



- Pause window has 3 options in it, these being the resume button which resumes the game, sound settings button which opens the sound settings and exit button which exits the game.

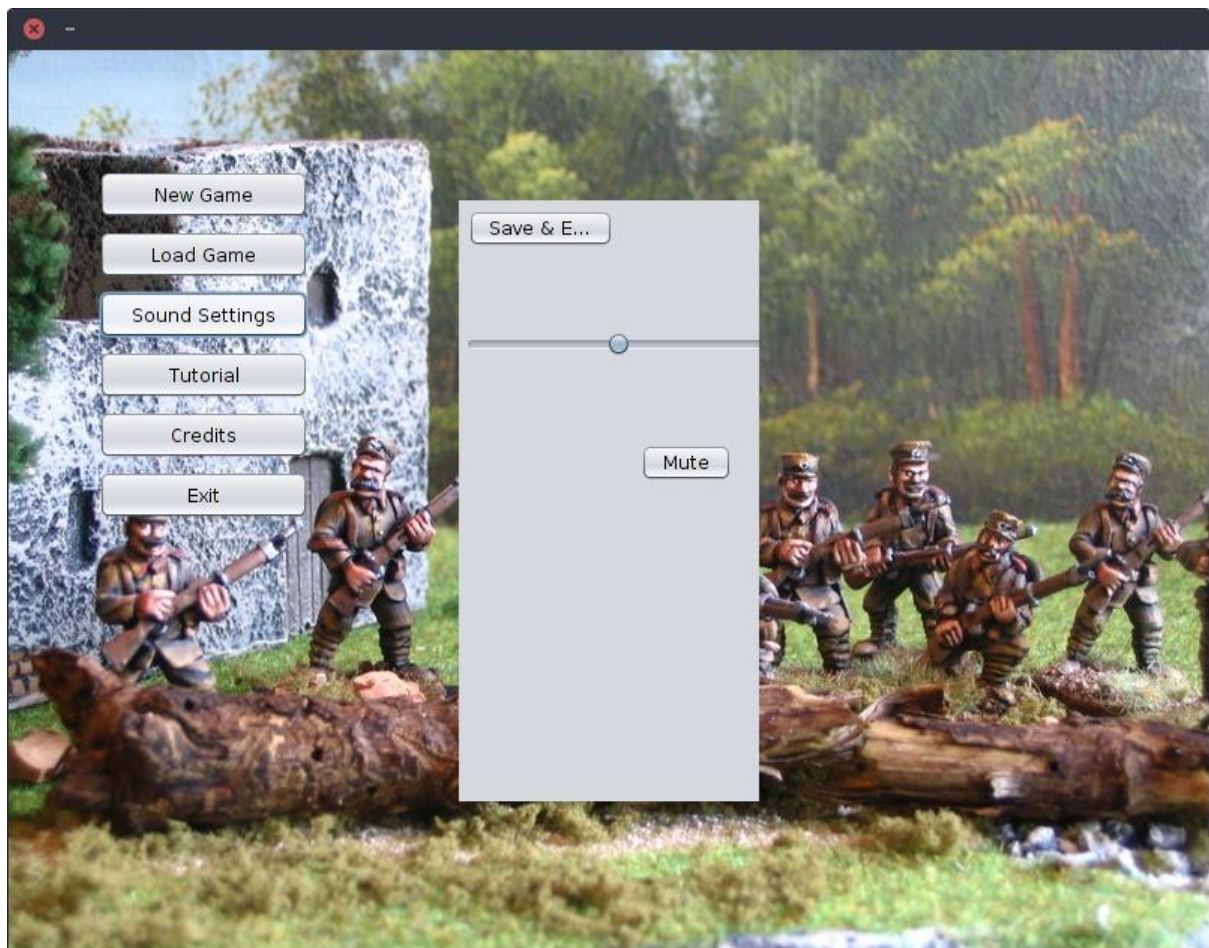




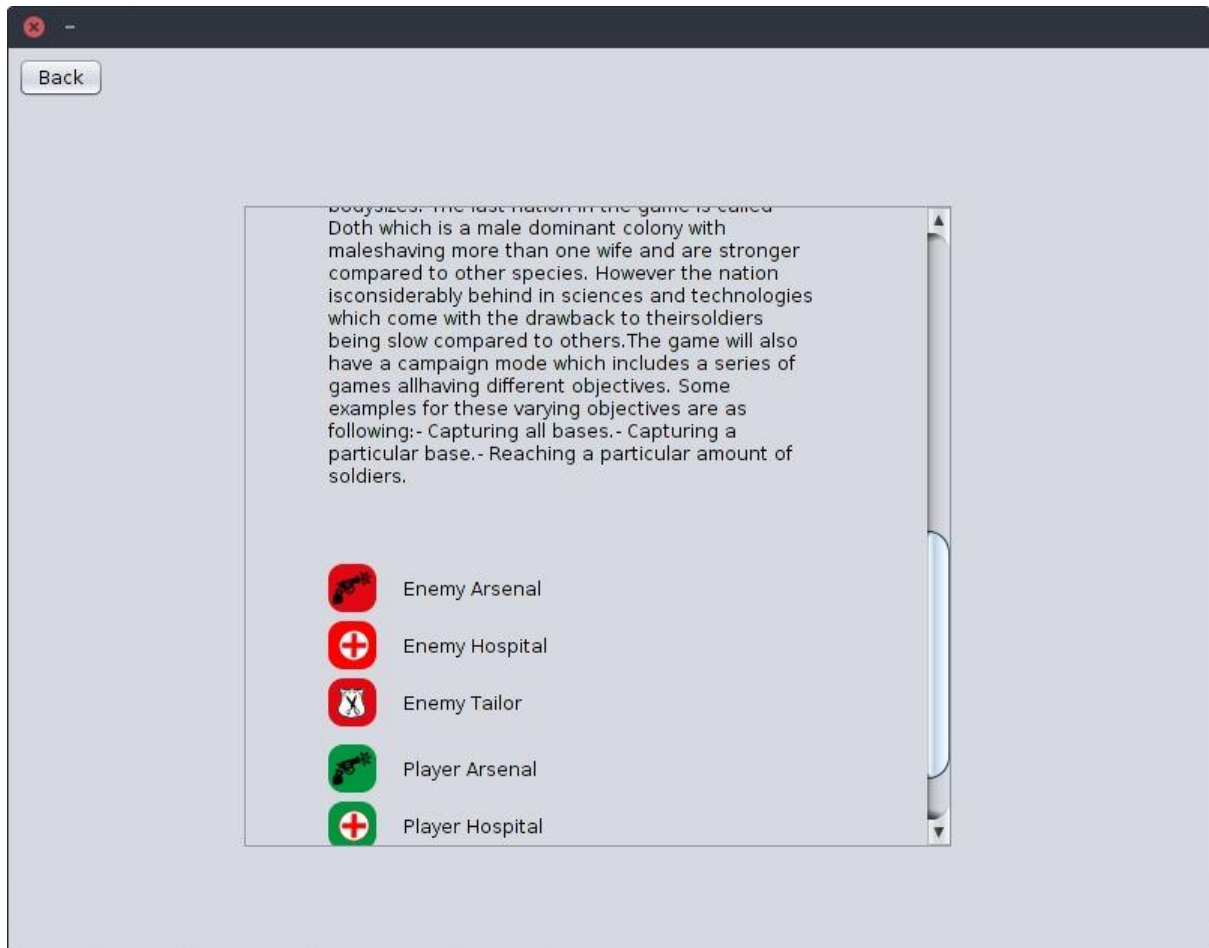
- The player can choose to load a previous game if s/he already played the campaign mode before and managed to win the game at any levels. The games that are won are saved so that the user can continue playing from the next level when they come back. However if the user begins a new Campaign the data saved would be lost. The saved games are highlighted with green color whereas the games that are not achieved to be won yet are colorless. So in the load game screen player can choose to play any highlighted level, or those that he won already or the next level he supposed to win. Back button would take the user back to the main menu.



- Another functionality in the Main menu window is the sound settings. The game has a background music added into it, however, the user is provided with the ability to un/mute or increase and lessen the volume of the music if they wanted to. The mute button in the right corner of the sound settings would shut the music off completely and the scroller above it would increase the sound if it was to be scrolled right and vice versa. Save&exit button would save the changes made and get the user back to the main menu window.

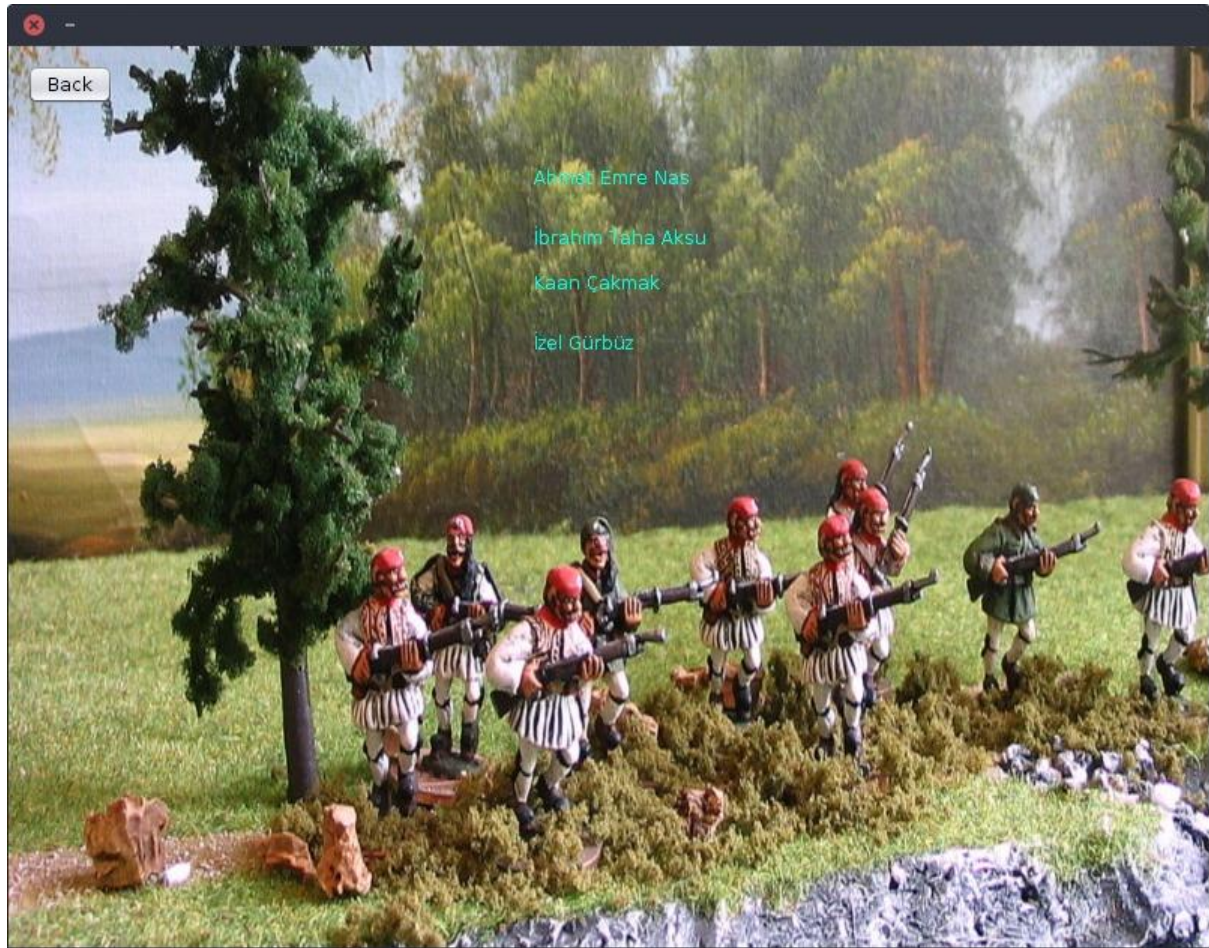


- The next button inside the main menu is the tutorial window where the user can learn about the story behind the game and also have an idea about the basic game mechanics. The tutorial window also introduces the user with all bases and elements used in the game using the actual icons. The back button would take the user back to the main menu window.



- User, if s/he wants to can reach the developers' information through credits window where the names of all developers are indicated. The back button would take the user back to the main menu window.





- Finally the last thing included in main menu is the exit button which exits the application when pressed.

#### 4.Changes Done During The Implementation Stage

We went for a considerable change in the design character of our GameView and GameObjects. In the design stage we thought of our Game class as the main processor of the game where all the calculations regarding the game were to be made and the changes were to be feed to user interface. However, we changed it in a way that made the implementation more manageable where we made our GameView(We also changed the name of the class as it was beter fit for the updated usage) class extend JPanel, and our GameObject class extend JLabel. This way our GameView Class represents the main window where the game is player and the GameObject class is for all the elements that are included in the map. Our Map class

is for initializing the first View of the map since we have various maps in our game. With this new update GameView (Game class with its old name) moved to View package from Controller since now it feeds the user with Graphical output.

In the design stage we had two different classes regarding the game modes those being: Skirmish and Campaign. During implementation we realized that we do not need to have a class for the gameMode since Campaign relies on skirmish but just includes several of games of skirmish mode. Thus we implemented our game logic based on skirmish mode and added several modifications for the campaign game mode. Thus campaign mode is just an instance variable of GameView fed by the user in the window where they choose game mode.

During the design stage we only had mute and unmute buttons in options menu but we added also the functionality to lessen or increase the sound of the music during gameplay.

In the design stage we decided on 3 different Windows after the user chooses to play a new game. First window where he chooses the game mode and then accordingly directed to the related window. However since in the campaign mode all the attributes such as the difficulty the nation and map is specific there is no need for an additional CampaignWindow. We rather added a dialog box where the user is warned if they choose to play Campaign that their saved data might be lost. The skirmish window is designed to be played on one map to practice several nations and difficulty levels which user chooses in the SkirmishWindow.

In the design stage Map class was determined to be child class of Game and GameObject to be a child class of Map. The design and actually functionality of Map class was changed in the implementation phase, which lead us to changes in the hierarchy also. The Map class thought be including all the game objects inside it in the design phase, however, it is changed to hold the initial map data inside in the implementation phase. Thus it does not have to have GameObject as its child class any more. The GameObjects are held in a list inside the Game class now (GameView with its new name).

In the design since we were building the system on Game class to make the calculations given the user input we designed a class called inputManager which would realize the input from the user and interpret it in an appropriate way for the Game class. However, making the Game class GameView extend JFrame we now have access to user input inside it own. Thus we do not need to take user input from other classes which also lessens the coupling between the classes.

## **5.Incomplete Parts**

We as developers were planning on having neutral bases in the game map rather than only bases that belongs to players, however, we are posponing it to a further upgrade as the game might go over complicated with the interference of more bases in the window.