CS 319 - Object-Oriented Software Engineering

Analysis Report

Panzer17

Group 2-C

Ndriçim Rrapi

Burak Sibirlioğlu

Pınar Bayata

Ferhat Serdar Atalay

Table of Contents

1. Introduction

As the name of the game implies, Panzer17 will be the game with the tanks. We are planning to give life to the old arcade game Tank90 with some modifications. The game starts with a map which is formed by various types of bricks. The excitement of the game is oriented from game's enemy features. There are two castles in this game. One castle is owned by the player while the other one belongs to the enemies'. Main aim of this game is to defend the castle from the enemies. Enemies' first place will be generated randomly and all the tanks including the players can move in four directions, up, down, right and left. Game will be controlled by keyboard. Restriction of the game is that an enemy can shoot the player 3 times and at the end of the third time, player gets killed, meaning the game is over. On the contrary, every time player shoots an enemy, he/she gains points. Therefore, there will be a Point Counter and a Heart Counter at the top of the game.

In Panzer17, we are planning to add new features to Tank90 such as maps, tank types, level displayer, sound effects, heart and point counter.

The game will be built using Java. All the features of the game, Panzer17 are analyzed in details in the upcoming parts of the report.

2. Game Overview

Panzer17 starts with the top view of a map and tanks. The map consists of various bricks such as one brick can be collapsed with one shoot while the other one should be shot three times. There are two castles, one for the enemies and one for the player. The castles are surrounded by the easily destroyable bricks. There will be 3 levels and in each level, map's design is changed so that it will be hard for the player to move and catch an enemy. Enemies are classified in themselves. Like the bricks, some of them have to be shot more than the other ones. Heart counter starts to count from 3 and decreases the value whenever player is shot by an enemy. Apart from that, point counter will start from 0 and adds the points as the player shoots enemies. An enemy will not be able to shoot other enemy. An enemy's tank will be different from the player's. Enemy's position will be generated randomly in every level. The goal is to play all the levels and reach the castle in every levels.

2.1.Brick Types

There will be three types of bricks in the game. The union of the bricks provides a map. Enemies and the player will move around this map. The map mainly consists of Type-1 Brick. This Brick will be collapsed in one shoot. The other type of brick is hardly destroyable. It is distinguishable as its color is red.

Type-3 Brick can be seen in Level 3. As the players jump to the levels, game becomes much harder. As a consequence of this, Type-3 Brick disables the user to pass through it or shoot. This break cannot be collapsed.

2.2. Tanks

Enemies' tanks can be distinguished from the player's tank. Tanks can throw a ball and move around. We created four types of enemy tanks. For the first one, we can say that it is a normal tank, it moves around and shoots as expected. Second tank is speedy but it shoots normally. In second level, player encounters with the enemy which can be killed at the end of the third shoot. We named this tank as "strong tank". For the third level, we created "ice ball shooter tank". This "ice ball shooter tank" throws ice balls which will make the player tank freeze. To sum up, the game consists of different types of tanks and tanks will show up depending on the level.

2.3.Bullet

The bullet is the way to tell the player that he/she is shooting or is going to be shooted by the enemies. Bullet starts to appear when the player presses Space bar button, and goes directly in x or y axis without a deviation. Enemies are shooting all the time, wherever they are. If a bullet confronts with an obstacle like a brick, it will destroy the brick and it will disappear. (If the conditions are satisfied; the brick should be destroyable and so forth).

2.4. Levels/Map

The levels determines the maps. Difficulty of the levels are determined by the bricks which construct a map, and enemies. As the player successfully finishes a level, he/she will encounter with a much more complicated map. There will be a Level Displayer at the beginning of the levels to demonstrate which level is the player is going to play. As stated above, in Level 3, the map will contain the non-destroyable brick which makes the game harder.

2.5.Castles

There are two castles in this game. Main objective of the game is to defend the player's castle. However, if the player reaches to the enemies' castle and destroys it without getting killed by the enemy tanks, player wins the game and he/she will gain points. Castle is not able to move and is surrounded by the breakable bricks.

2.6.Ice Ball

Ice ball is shot by the ice ball shooter tank. Its facility is that it freezes the player's tank when confronted. Its color will be different from the bullet.

2.7.Coins

Coins are supposed to give the player a large amount of points upon collection. Like the pacman eats the coins, our player can collect coins. The coins will be waiting for the player in some parts of the map. Yet, the coins will be placed in dangerous areas and it will be hard to collect them since we do not want everyone to easily go on top of the high scores list. An enemy cannot collect coins, so when an enemy passes the road with coins, nothing will happen to the coins. There will be a high score recorder, so the coins will be beneficial for the ambitious players.

3. Requirement Specifications

3.1. Functional Requirements

3.1.1. Play Game

Panzer17 is an implementation of a classic console game Tank90 which was very popular around 90s. The main aim of the game is defending the castle from enemy tanks. Player has a tank and the tank starts from beside of the castle. At the same time, three enemy tanks appear on the upper side of the map. They attack through the castle and when the player destroy one of them a new enemy tank appears. There are four types of enemy tanks which are normal ones, fast ones, freezer ones and the ones that are destroyed by three shots. There are also 4 types of walls; normal brick which is destroyed with one shot, white bricks which can't be destroyed, blue bricks which represent water and only bullet can pass the water bricks, and green bricks which represent forest and can hide tanks and bullets and tanks can pass through it. Water bricks and forest bricks can't also be destroyed. If the player tank gets a shot and destroyed, one of the three life of the player disappears and when all of the three life disappears the game is over. If the player achieves to destroy all enemy tanks, or destroy enemy castle level become cleared and next level starts.

There are different power-ups which will increase the speed of the tank or increase the endurance of the tank. Also there are several levels which can increase variation and enjoy.

3.1.2. Change Settings

Apart from the default settings, player can change the settings of the game. As a basic console game, the game hasn't got many settings.

- Player can enable or disable the sound of the game.
- Player can change the color of the tank.
- Player can disable shooting effects

If the player wants to turn to original settings of the game, player can choose the default settings.

3.1.3. *Pause Game*

The game can be paused during gameplay. When the player tries to pause the game, all tanks and bullets stay in the same place and game progress won't be lost. Also when player start again the game will continue immediately. Apart from that player can reach help panel, highs scores table and settings panel from pause menu. Also player can quit the game by this menu.

3.1.4. *View Help*

Game will have a help panel to give information about gameplay and features. Player can get tips about types of enemy tanks and types of walls. Also player can be informed about rules and game controls. Player can reach the help panel from the main menu of the game.

3.1.5. View Credits

After the game is completed, user can see the credits panel and this panel have the information about, developers which can be used to communicate with, designers to give some new ideas which can develop the game.

3.2. Non-Functional Requirements

3.2.1. Game Performance

As a classic console game, requirements of Tank90 was not high. We also aim to keep the requirements in Panzer2017 as low as possible to have an efficient performance on every system that the game is played. Game have sound and some graphic features like moves of tanks, bullets and destroy of tanks or walls. These features should not decrease the speed of gameplay, to guarantee the enjoyment of the game.

3.2.2. User-Friendly Interface

Panzer17 is a basic game therefore graphical user interface should be easy to understand. Player should use the controls and menus of the game easily and also player should understand the map as fast as possible to start the game. Therefore we will try to make the game user friendly.

3.2.3. Extendibility

We will try to have many simple classes to have a chance to improve the game after the demo according to tests and user ideas. Having such kind of implementation will allow us to change attributes of main items without changing the main classes. This will make the game extendable and reusable. Also according to our opinion maintenance and solving some runtime errors will be easier.

4. System Model

4.1. Use Case Model

The Use Case Diagram named Panzer17 as in the Figure 4.1 demonstrates player's actions.

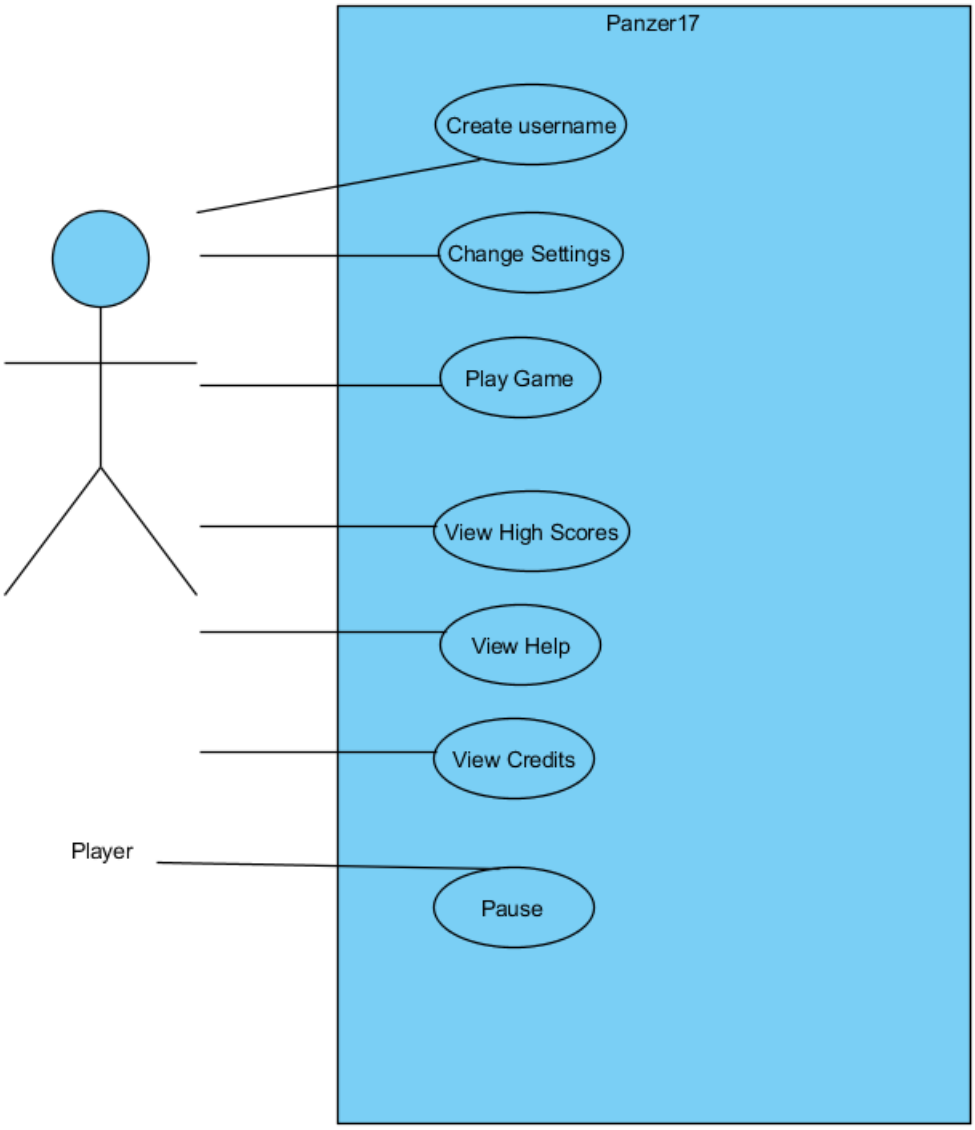


Figure 4.1

4.1.1. Create or Pick Username

Use Case Name: Create or Pick Username

Primary Actor: Player

Stakeholders and Interests:

-Player is forced to create or pick a username.

-System shows the username window.

Pre-conditions: System keeps records of usernames.

Post-condition: System will pick a username randomly.

Entry Condition: Player selects "Play" from Main Menu.

Exit Condition: Player selects "Random" to start the game.

Success Scenario Event Flow:

1.System saves the username player wrote.

Alternative Flows:

A. If player desires to pick a random username:

A.1. Player selects "Random" button to start the game.

A.2. System picks a random username.

4.1.2. Change Settings

Use Case Name: Change Settings

Primary Actor: Player

Stakeholders and Interests:

-Player wants to change settings.

-System opens the "Change Settings" window.

Pre-conditions: System keeps records of the current settings.

Post-condition: System will save the changed settings.

Entry Condition: Player selects "Change Settings" from Main Menu.

Exit Condition: Player selects “Back” to return to the Main Menu.

Success Scenario Event Flow:

1.System shows the Change Settings window.

Alternative Flows:
A. If player desires to return main menu at any time:
A.1. Player selects “Back” button to return main menu.
A.2. System displays Main Menu.

B. If Player requests to return previous menu at any time:

B.1. Player selects “Back” button from “Change Settings” screen.
B.2. Game settings are updated by System.
B.3. Player returns previous menu.

4.1.3. View High Scores

Use Case Name: View High Scores

Primary Actor: Player

Stakeholders and Interests:

-Player wants to see top nine scores with player names.

-System shows the list containing top nine scores with player names.

Pre-conditions: System keeps records of top ten scores.

Post-condition: -

Entry Condition: Player selects “View High Scores” from Main Menu.

Exit Condition: Player selects “Back” to return Main Menu.

Success Scenario Event Flow:

1.System displays top ten scores with player names.

Alternative Flows:

A. If player desires to return main menu at any time:

A.1. Player selects “Back” button to return main menu.

A.2. System displays Main Menu.

4.1.4. View Credits

Use Case Name: View Credits

Primary Actor: Player

Stakeholders and Interests:
-Player wants to see who have implemented the game.
-System shows the list who contributed to the game.

Pre-conditions: System keeps records of the names.
Post-condition: -

Entry Condition: Player selects “View Credits” from Main Menu.
Exit Condition: Player selects “Back” to return Main Menu.

Success Scenario Event Flow:

1.System displays the names who implemented the game.

Alternative Flows:
A. If player desires to return main menu at any time:

A.1. Player selects “Back” button to return main menu.

A.2. System displays Main Menu.

4.1.5. Play Game

Use Case Name: Play Game
Primary Actor: Player

Stakeholders and Interests:

- The objective of the player is to eliminate every enemy in each level and reach to the end of the game.

- Tracking of the player’s score will be kept

Pre-conditions: The player should open the game, he/she should type a username. After that he/she can play the game.

Post-condition: If the player prefers, he/she can press “ESC” or “P” from the keyboard while playing the game. The pause menu will be opened and the player can pause,view help,

settings, high scores and exit. If Player gets a score which is high enough to be in high score list, high score list will be updated by System.

Entry Condition: The player interacts with “Play Game” button which is in the main menu.

Exit Condition: Player selects “Exit” to exit the game.

Success Scenario Event Flow:

1.Player opens the game.

2. System asks to create a username or pick one.

3. Player types a username or let the system choose one.

4. After the username window, system loads the game.

5. The player manages to eliminate all the enemies and reaches to the castle.

6. Player advances to the next level until he/she reaches to the end.

7. The system records the player’s score.

8. Main Menu appears on the screen.

Alternative Flows:

A. If Player wants to exit:

A.1. Player presses “ESC” or “P” from the keyboard.

A.2. Player selects “Exit” button from “Pause” screen.

A.3. Player exits the game.

4.1.6. View Help

Use Case Name: View Help

Primary Actor: Player

Stakeholders and Interests:

- Player wants to learn purpose of game and how to play it.
- System shows a text document explaining purpose of game, player controls, power ups and brick types.

Pre-conditions: Player should be in Main Menu or Pause Menu.
Post-condition: -

Entry Condition: Player selects “View Help” from Main Menu or Pause Menu.
Exit Condition: Player selects “Back” to return previous menu.

Success Scenario:
1.Player selects “View Help” from Menu.
2. System displays help document giving instructions about main purpose of game, game controls and power ups and brick types.

Alternative Flows:
A. If Player requests to return previous menu at any time:

A.1. Player selects “Back” button from “View Help” screen.
A.2. Player returns previous menu.

4.1.7. Pause

Use Case Name: Pause
Primary Actor: Player
Stakeholders and Interests:
- Player wants to pause the game by pressing “ESC” or “P” from the keyboard.
- System pauses the game and opens the “Pause” window.

Pre-conditions: Player is already playing the game.
Post-condition: -
Entry Condition: Player presses “ESC” or “P” from the keyboard.
Exit Condition:

-Player can choose to replay, OR

-Player can view “Help”, OR

-Player can view “Settings”, OR

-Player can view “High Scores”, OR

-Player can exit the game.

Success Scenario:

1.Player presses “ESC” or “P” from the keyboard.

2. System displays the pause window.

3. System pauses the game.

3. Player chooses replay from window.

4. Player continues the game.

Alternative Flows:

A. If Player requests to view help:

A.1. Player selects “Help” button from “Pause” screen.

A.2. Player returns to the “Pause” screen.

B. If Player requests to view settings:

B.1. Player selects “Settings” button from “Pause” screen.

B.2. Player change the settings.

B.3. System saves the changed settings.

B.4.Player selects “Back” button.

B.5. Player closes the “Pause” window.

B.6.Player continues to play.

C. If Player requests to view high scores:

C.1. Player selects “High Scores” button from “Pause” screen.

C.2. Player returns to the “Pause” screen.

D. If Player wants to exit:

D.1. Player selects “Exit” button from “Pause” screen.

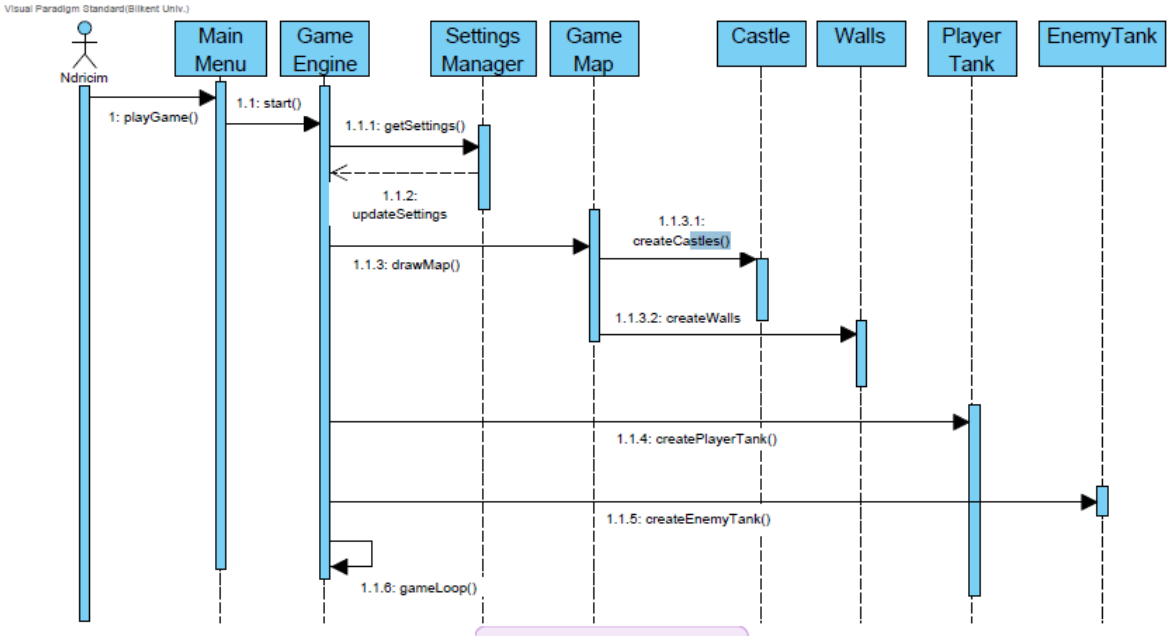
D.2. Player exits the game.

4.2.Dynamic Models

4.2.1. Sequence Diagrams

4.2.1.1. First Scenario-Play Game

First Scenario



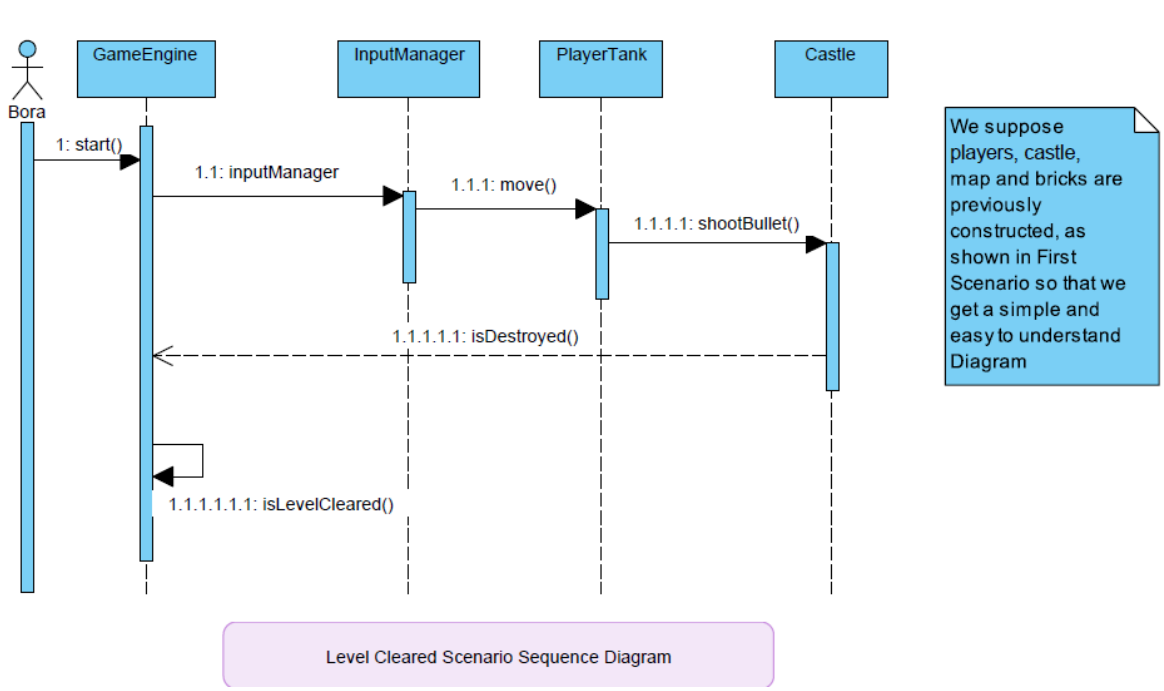
Primary Actor: Ndricim

Scenario: In this scenario Ndricim chooses to play the game from the Main Menu button. He faces the buildup of the gameplay as he presses button play.

Description:

Ndricim wants to play the game. He presses the "Play" button from main menu. Now Ndricim can see the map with the bricks, he can spot the enemy tanks and enemy castle as well as his tank and his castle. The map being built now Ndricim is ready to start playing by using arrow keys to move and spacebar to shoot bullets.

4.2.1.2. Level Cleared Scenario



Level Cleared Scenario Sequence Diagram

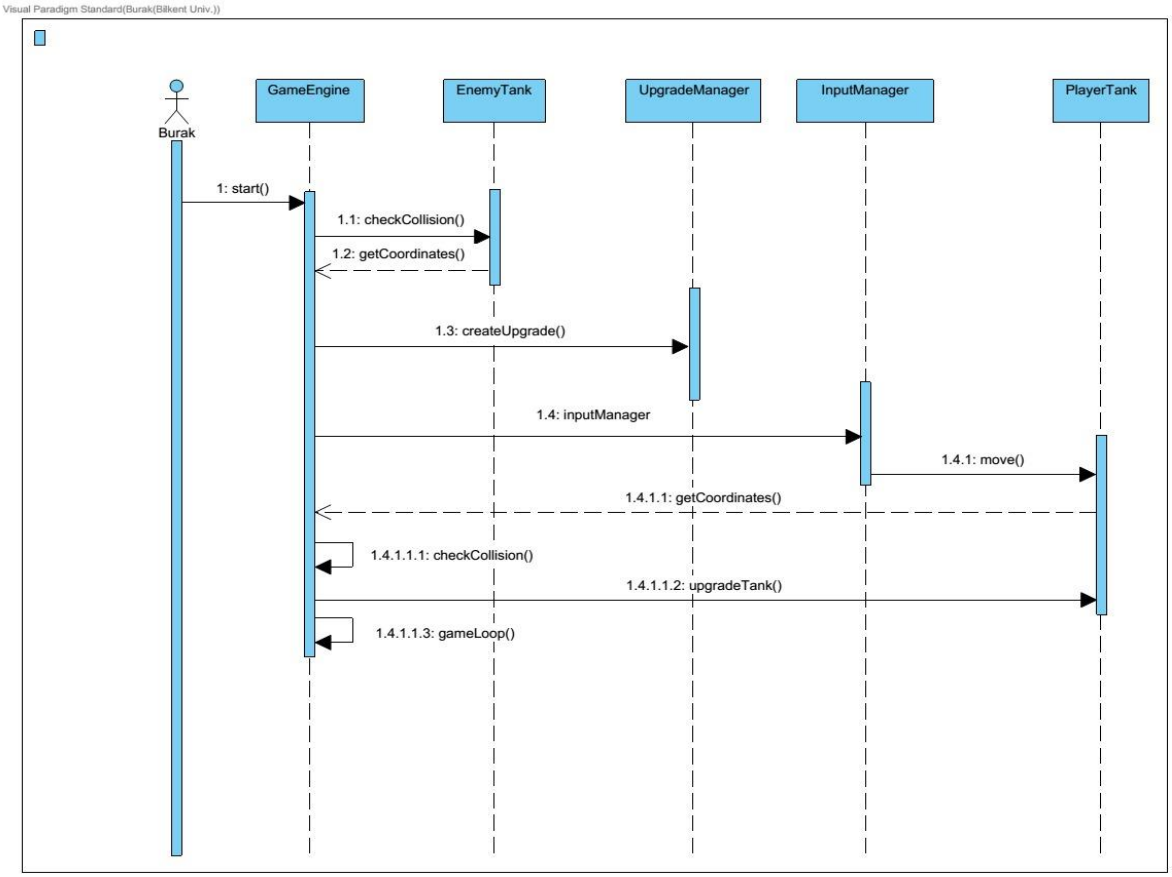
Primary Actor: Bora

Scenario: In this scenario Bora is already moving the tank around the map, in an attempt to shoot the enemy castle.

Description:

As the game starts Bora starts moving around the map and manages to avoid the enemy tanks and shoot at the enemy castle. As Bora shoots the enemy castle gets weakened and after several shots the castle is destroyed. Bora than clears the level and continues to the next one.

4.2.1.3. Power-Ups/Upgrades



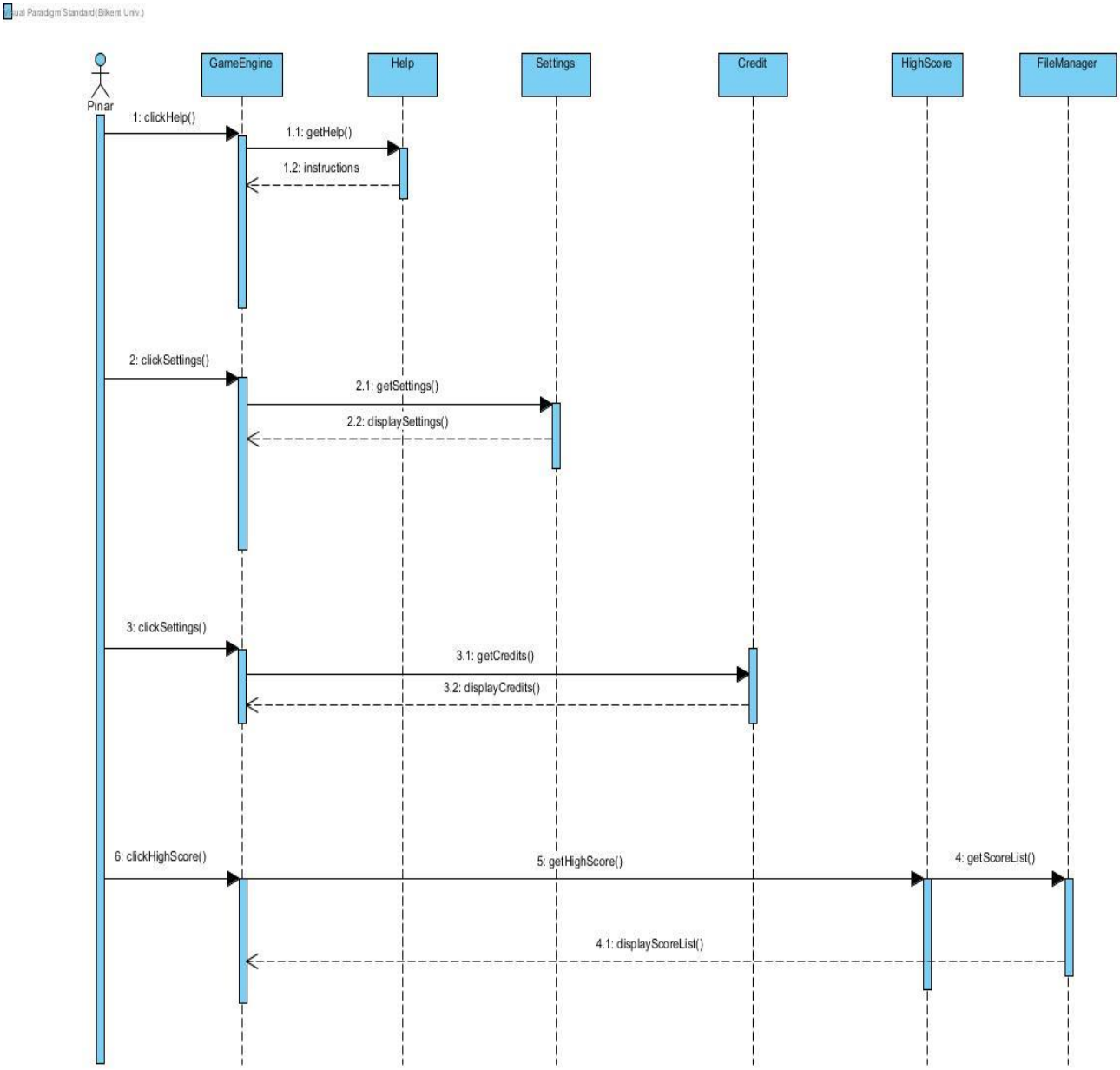
Primary Actor: Burak

Scenario: During gameplay, Burak faces with coins or upgrades that appears suddenly on any place of the battle arena. He directs his tank through the upgrade to take and make his tank more powerful or faster. He managed to get the upgrade before it disappears, and his tank become faster than the original tank.

Description:

At start of the game, gameEngine who controls the game entities and gameplay, creates upgradeManager object and upgradeManager who controls the upgrade objects and control their attiributes, creates upgrades and emplaces them random places on the map. If the timer become zero, upgradeManager deletes the upgrade. At the same time gameEngine is connected with inputManager to make the player tank move. playerTank object sends its coordinates to gameEngine and gameEngine class control the collision of tank and upgrade. If checkCollision function returns true, gameEngine upgrades player tank. During whole gameplay gameEngine will be in gameLoop to continue the original functionality of the game.

4.2.1.4. Game Menu



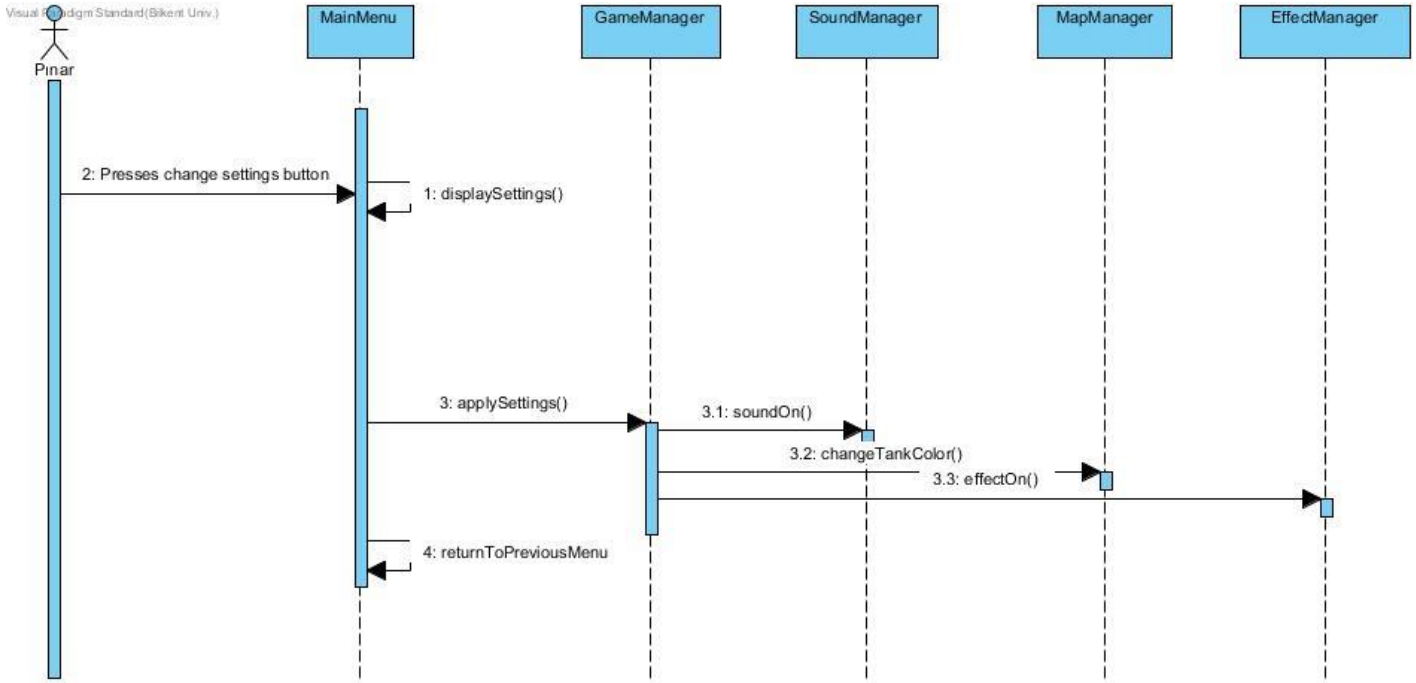
Primary Actor: Pınar

Scenario: Before the game starts, Pınar encounters with a menu. Pınar does not know how to play the game so she presses “Help”. After she learns, she clicks on to the “Back” to turn back to the Main Menu. She is curious about the high scores and the implementers so she clicks on “Credits” and “High Scores” one by one. In every case, she returns to the Main Menu with a click on “Back”.

Description:

When the game is opened, there are options to select in main menu. If the “Help” button is clicked, gameEngine interacts with the help class and help class returns a string of interactions. On the other hand, if settings are clicked as an option, gameEngine connects to the settings class with the help of getSettings() function and settings class returns the settings. If the user wants to see the credits, he/she interacts with the credit class and the class returns the credits as strings. For the high score list, fileManager stores the data to show so when the High Score option is selected, highScore class interacts with the FileManager class and returns the score list.

4.2.1.5. Change Settings



Primary Actor: Pınar

Scenario: Pınar wants to change settings. She clicks on the Main Menu and presses “Change Settings”. She can either turn on or off the sound, select a color for the player's tank or she can turn on or off the sound of the shooting effects.

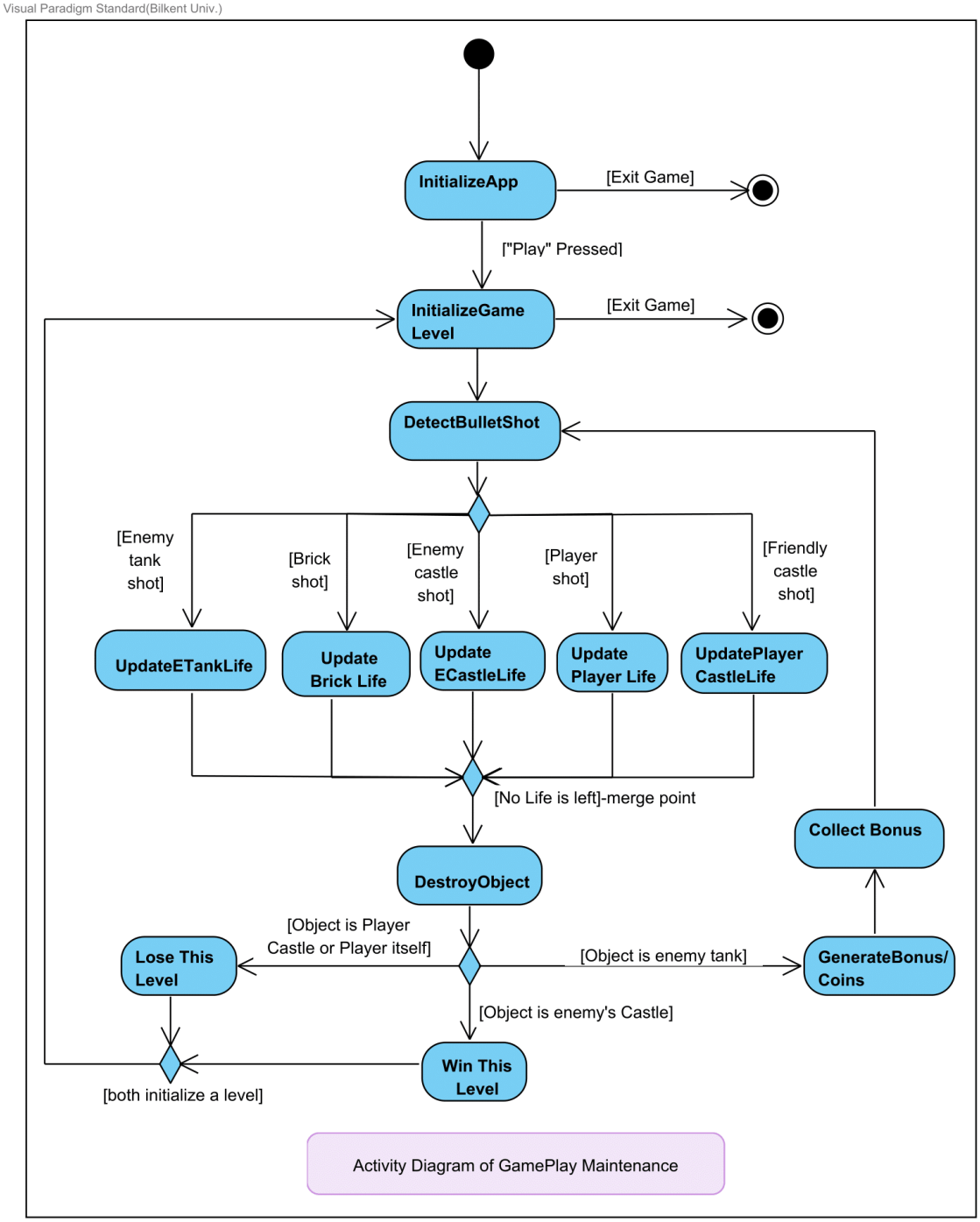
Description:

Player presses change settings button from a menu object (since both PauseMenu and MainMenu provides this functionality), the system displays settings, and player changes the settings according to his desire. After that, when player presses back button, system updates new settings by GameManager.

GameManager sends tank color which is a common property for all tank objects to MapManager class. GameManager also sends the new effect settings to EffectManager to apply. New sound settings are sent to SoundManager to apply, by GameManager again.

4.3.Activity Diagram

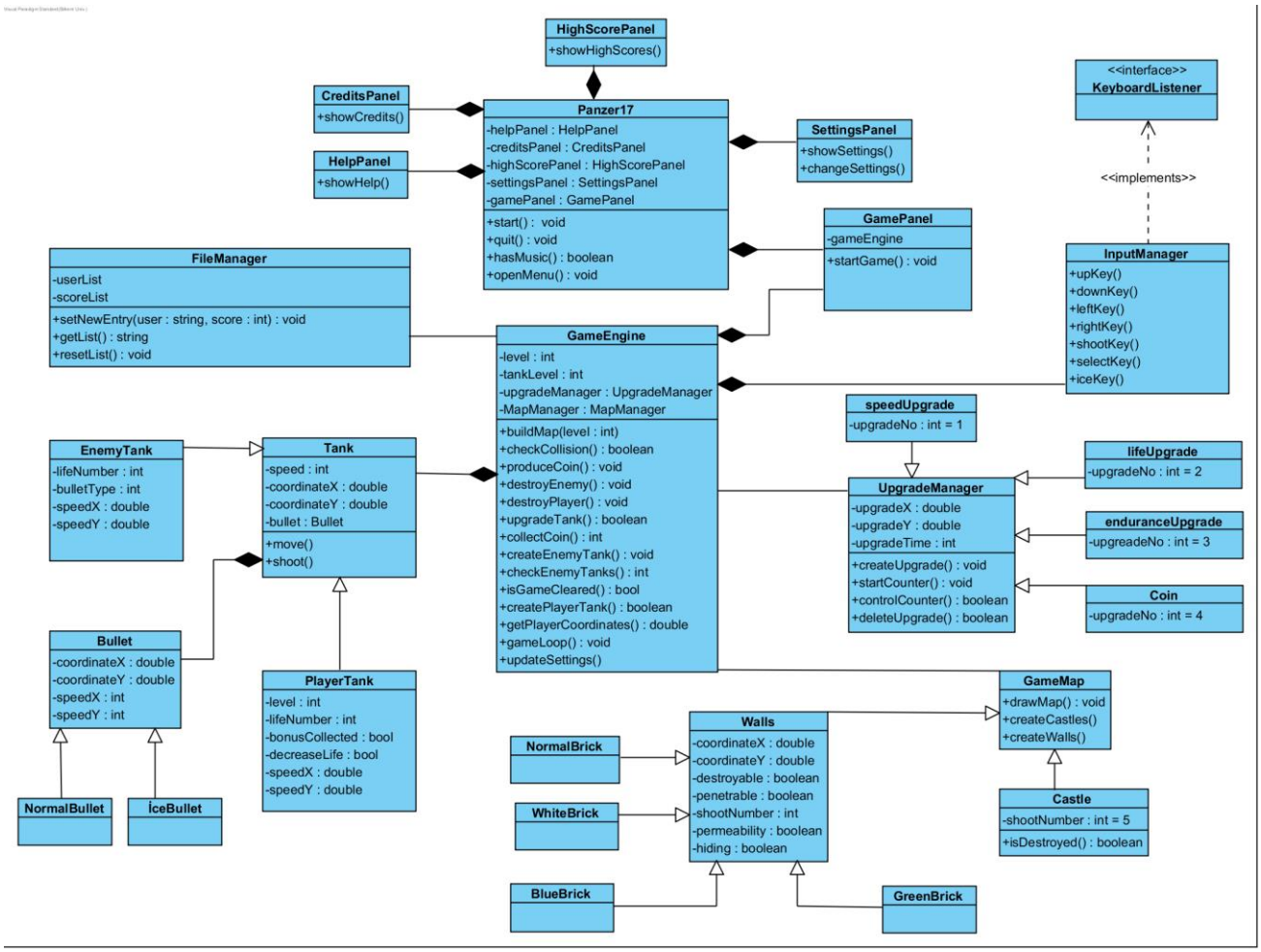
The diagram below demonstrated the gameplay maintenance of our Panzer17 game.



As the user opens the game he presses “Play” button from the Main Menu which initializes the map with all the tanks, castle’s and bricks in this case at first Level. The user may optionally chose to exit the game from this point on during gameplay. Then the user will move around the map and shoot bullets towards the bricks, enemy tanks and enemy castle.

The enemy tanks shoots as well. Each of the shots are then detected and identified. Bricks, tanks and castles can be shot continuously. As they have different life points their life points will be updated after each shot they get and if they have no life points left they are then destroyed. Yet, a brick destroyed means a new path opened, but if the user himself or his castle get destroyed it means game over, the user loses and the level restarts. In case an enemy tank gets destroyed bonus will come out at the coordinates of destruction, which might be coins or power upgrades. For the user to win the game he needs to shoot the enemy's castle and take out all of its life points. As the user wins he gets to the next level and the cycle continues.

4.4. *Object and Class Model*



The class diagram of Panzer2017 consists of 28 classes and the model is shown above.

Pazner17 class is the main class that holds and intializes the main menu represented by the panels **HighScorePanel, CreditsPanel, HelpPanel, SettingsPanel** and **GamePanel**. The game starts through **GamePanel** which uses **GameEngine** as the main engine that performs all of the main functionalities of the gameplay. **GameEngine** in itself combines all of the objects in the game and maintains the games core visuals as well as handling the game logic which includes object collision and movement. Yet, in order to perform all of the above functionality **GameEngine** uses other management classes which are explained below:

-FileManager is a class which retrieves the local data saved locally for the game settings and player preferences

-InputManager class controlling the connection between keyboard and the system. Input manager using keyboardListener to detect any pressed keys.

-UpgradeManager class controlling upgrades on the map. This class creating those upgrades and control the image and specialty of player tank after each upgrade. Also upgradeManager creates coins and emplaces them.

-MapManager class draws the map of each level by using walls and castle classes. According to level number this class chooses right structure of map and walls.

Movement of objects will be controlled by class itself. coordinateX and coordinateY attributes holds the place data of the object and speed data for movement. When game is paused the coordinate data will be stored and play function will again start the game. Other parts of the game is controlled by the gameEngine, all entity objects created and controlled by the gameEngine.

5. *Screen Mock-ups*

5.1.Main Menu

The main menu consists of 5 options to choose from as shown in the Figure 5.2.1. User navigation between options will be set-up using an icon moving up and down according to the option on focus. Users may use arrow keys to select the preferred option.

Figure 5.2.1 Screen shot of Main Menu

5.1.1. Play

-Play Game: This option redirects the user to the main screen of the gameplay and prompts the user for some information as shown in Figure 5.2.1.1.

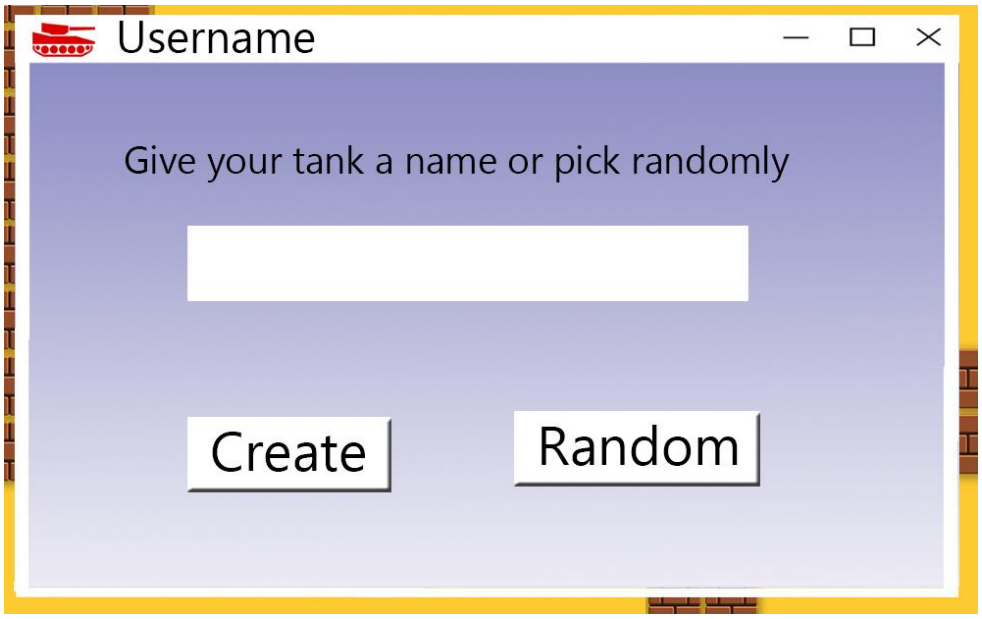


Figure 5.2.1.1 Screen shot of gameplay username prompt

As shown in the picture the user is prompted to enter a user name needed for the highs score table. If the user does not wish to give a name he/she may choose a random user name as well.

-Change Settings: This option redirects the user to the settings window. The user may then choose sound options, player's tank colors and shooting effects or may optionally reset the settings to default as shown in Figure 5.2.1.2. User may get back to the Main Menu through the "Back" button.

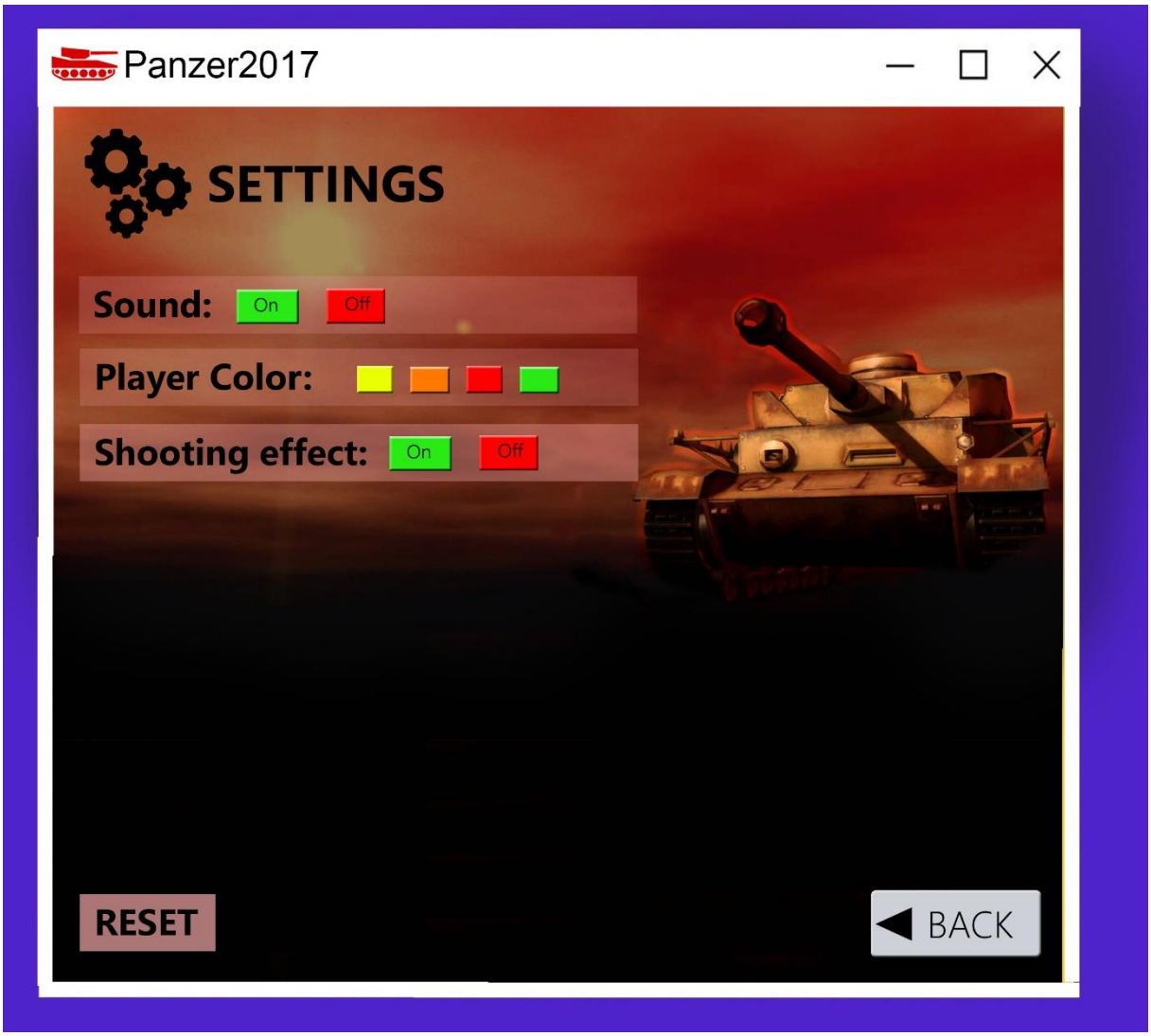


Figure 5.2.1.2 Mock-up of Settings

-View High Scores: When high scores button is pressed, system will show the list of top ten scores with player names. If player can make a score which is higher than the 10th score, his score is displayed at this list and he will enter his name to high scores list. (Figure 5.2.1.3)

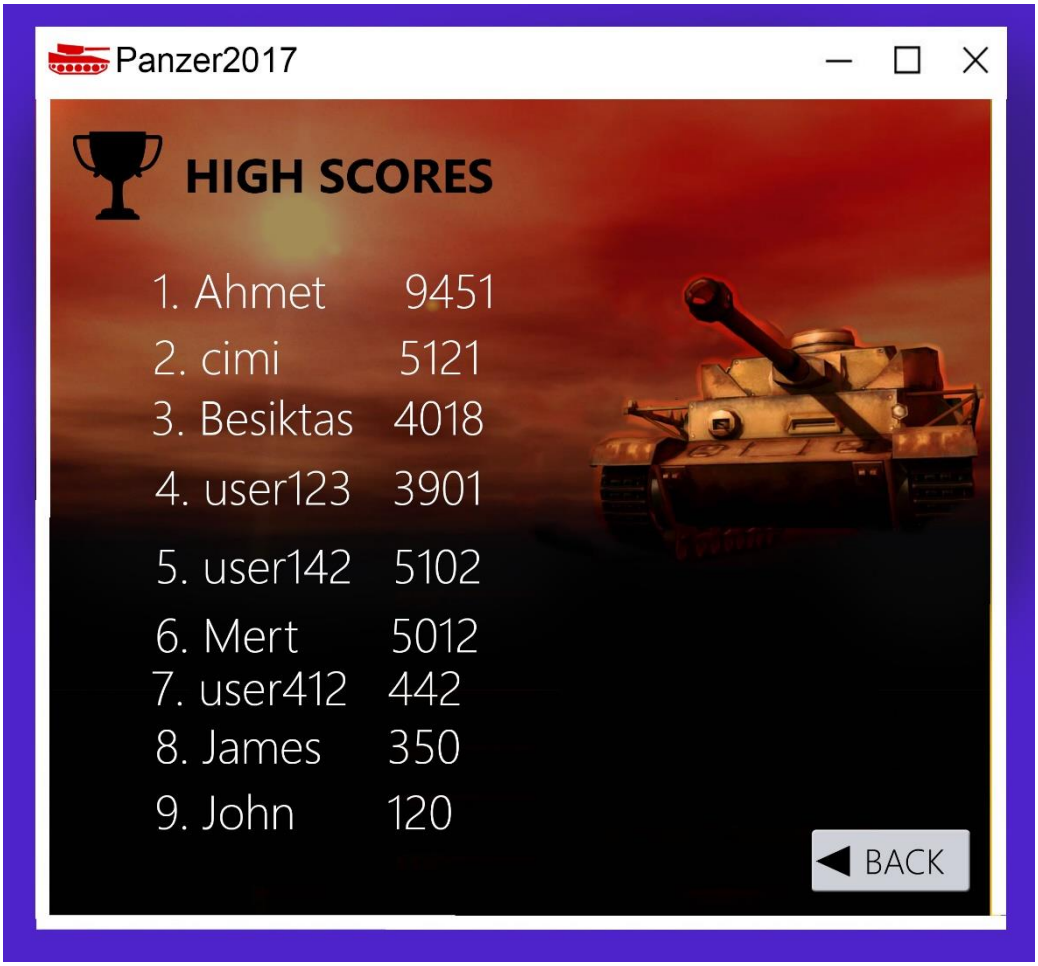


Figure 5.2.1.3 Mock-up of High Scores

-Help: This button redirects the user to the help window as shown in Figure 5.2.1.4. Again user may exit to main menu through the “Back” button on the bottom.

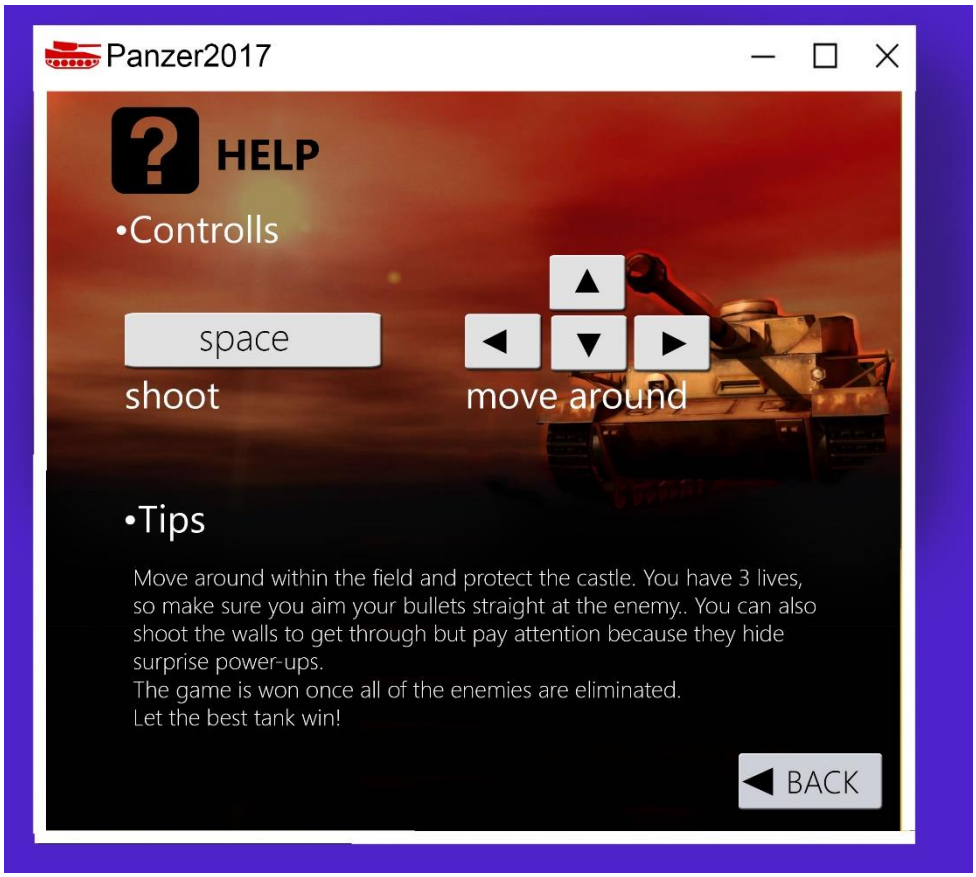


Figure 5.2.1.4 Mock-up of Help

-Credits: When Player selects Credits, contact information and names of game developers are shown on screen. Player can return to previous menu by clicking “Back”. (Figure 5.2.1.5)

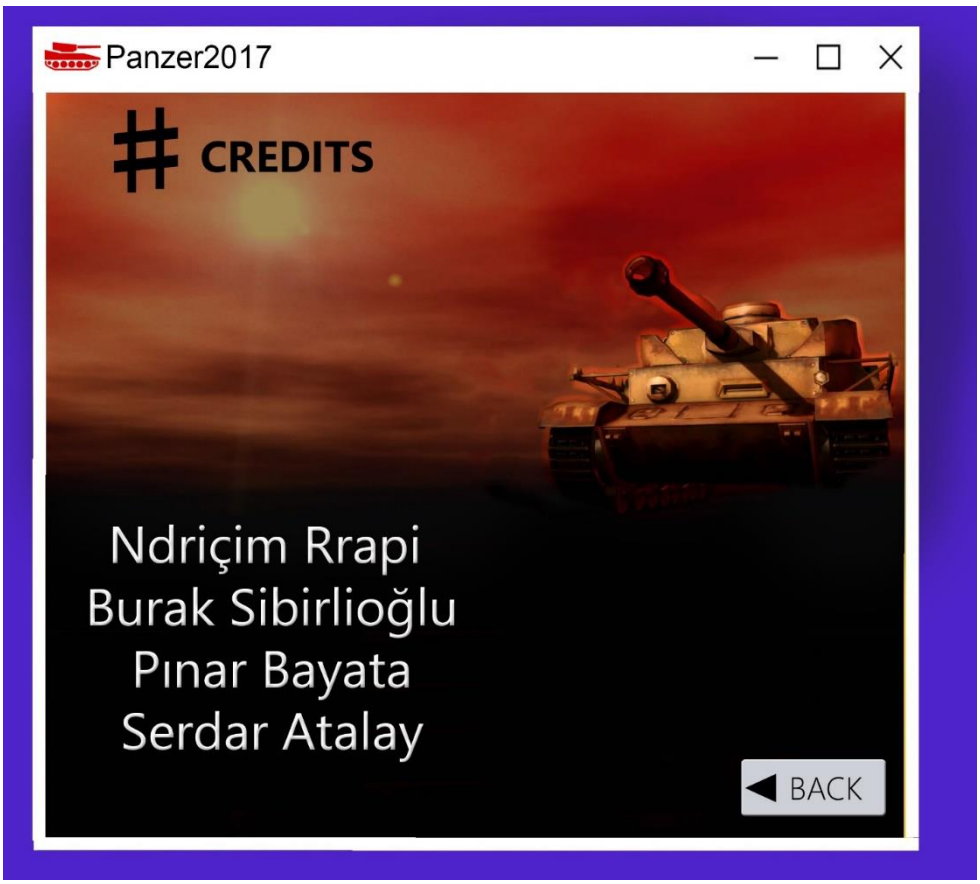


Figure 5.2.1.5 Screen shot of Credits

5.1.2. Pause Menu

The Pause Menu can be activated by pressing “ESC” or “P” from the keyboard. This short menu contains 5 options such as “Play”, “Help”, “High Scores”, “Settings”, and “Exit” as shown in figure Figure 5.2.2. So except for returning to the game and accessing game options the user may as well choose to quit the game as well by pushing the “Exit” button.

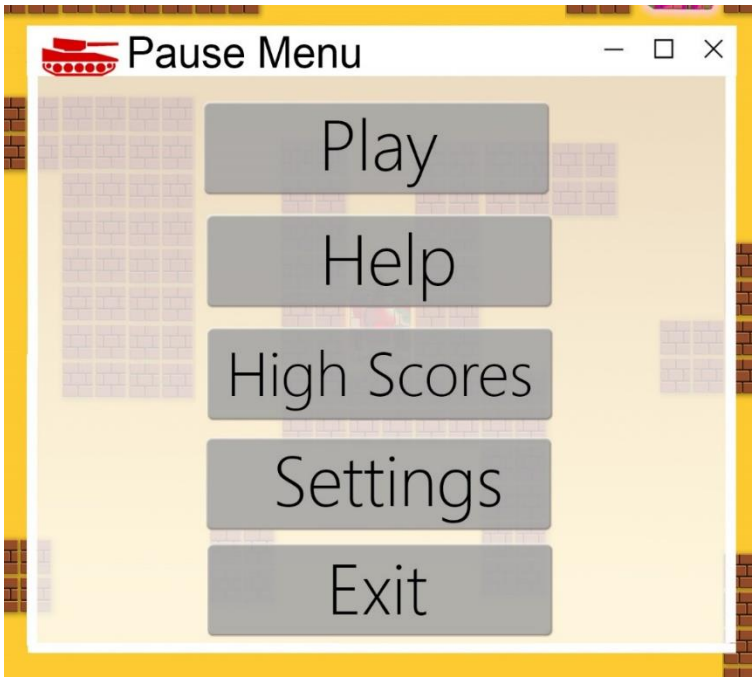


Figure 5.2.2 Screen shot of Pause Menu

5.1.3. Power Ups

The game will also consist in surprise power ups which will pop-up during gameplay shown in the pictures below:

Time Freeze

Explosion – All current enemies die

Protection Shield – Protects for limited time from shots

Gives the user fast bullets

User may shoot multiple bullets faster

Extra life +1

5.1.4. Bricks

The bricks in this game are going to be obstacles that can be shot by both the enemies and the user. We have decided that there will be 4 types of bricks

-**Brown Brick**: Brown bricks are destroyed by one hit.

Figure 5.2.4.1 Brown Bricks

-Green Bricks: The green bricks are destroyed by 2 hits.

Figure: 5.2.4.2 Green Bricks

-**Red Bricks**: The red bricks are destroyed by 3 hits.

Figure 5.2.4.3 Red Bricks

-Gray Bricks: The gray bricks are indestructible.

Figure 5.2.4.4 Gray Bricks

6. Conclusion

After an intense work, we provided requirements, system models and some mockups for the game "Panzer17". In the introduction part, we described our game briefly, we told about our inspirations and the properties of our game. The requirements part was necessary because the games functionality is investigated. These listed functionalities were beneficial for the system model part that we can easily identified the classes in sequence diagrams.

Our System Model consists of four parts:

- Use Case Model: We thought a lot about the cases that the player can encounter and because our game is user friendly, we realized that the user can reach to any menu or list he/she wants at anytime.
- Dynamic Model: Sequence diagram was the hardest part for us because it briefly describes the interaction between classes and objects. However, when we start implementing, these will benefit
- Activity Diagram: This diagram summarizes the game, what will happen if a tank is shooted and so forth.
- Object and Class Model: Thanks to this model, it will be more eager for us to implement this game.

For the last part, we provided screen mock-ups so that it will be easier to understand the game's functionalities.

For this analysis report, we investigated the features of our game and we added some original modifications to the game Tank1990. We hope this analysis design report will help us in the implementation and we will try our best to implement what we said in the functionalities.