Bilkent University

Department of Computer Engineering

CS319 PROJECT – GROUP #2

Analysis Report - Completed

CS 319 Project: Bombalamasyon

Oğuz Demir – 21201712

Anıl Sert – 21201526

Kaya Yıldırım – 21002071

Kaan Kale – 21000912

Course Instructor: Uğur DOĞRUSÖZ

Analysis Report

Mar 6, 2016

This report is submitted to the GitHub in partial fulfillment of the requirements of the Object Oriented Software Engineering Project, course CS319.

Table of Contents

Table of Figures

1 Introduction

We decided to create a game like the classic arcade game called Bomberman and our games' name is Bombalamasyon. It is a game that basically player/s try to reach the other players by exploding the walls and try to kill other bombers to be the last man standing.

This report contains an overview of the game, basic gameplay and rules of the game. Then it describes the requirements of the project which are functional and non-functional requirements. Lastly, system models of the game will be described and Glossary&references will be given.

2 Overview

Bombalamasyon is a strategic, maze-based game like the other alternatives such as Bomberman and Fireman. It is an easy and fun game to play but it is hard to master it. Player/s play the game in a labyrinth shaped map which consists of different kinds of walls and paths. The player/s objective is the reach and kill the other bombers and be the last man standing in a competitive killing race in a limited time.

2.1 SinglePlayer Game

The SinglePlayer mod of the game is a level based bomberman game which player needs to complete 5 levels to finish the game. Every level is different than each other and levels are become harder as player continues. At each level, the player's objective is the same as the whole game's which is kill the other bombers in a limited time. If the player is killed or time elapses, the user loses that level and move on the next one if there is any. At the end of the SinglePlayer game if the player takes greater score than computer opponents and earn better point than the lowest of the high score list, s/he deserves to enter her/his name to the high score list.

2.2 Multiplayer Game

The multiplayer mod of the game is a race between two players and the computer opponents. There is three different possible results in a multiplayer game's levels. The first one is both players are dead and this will end the level, second is one of the players be the last man standing in the maze and wins that level, the last one is time elapses and highest scored

bomber becomes the winner of the game. If one of the players wins the game and earn better point than the lowest of the high score list, s/he deserves to enter her/his name to the high score list.

2.3 Gameplay

The player/s play the game from the keyboard. There is specific keys for each action in the game like moving the bomber, drop the bomb, or pause the game and see the pause menu. If users play a multiplayer game because they will use the same keyboard to prevent hand conflicts we assign these action buttons separate as much as we can. To interact with the menus of the game like main menu and pause menu, player/s need to use the mouse buttons to choose their selections.

2.4 Bomberman and Bombs

The main character of our game is the bombermen and to differentiate each bomberman from the others we are going to use colors. Every bomberman will have a different uniform color and their bombs color will also be same with their uniforms. The player/s control the bombermen and interact with game with these characters. They can run the bomberman in four different directions and can place a bomb to their current position. Bombs are the weapons of the bombermen and have explosion magnitudes. At the start of every game bombermen can put only one bomb at the time. However, the explosion magnitude and the bomb quantity can be increased with the powerups. Bombs can break some kinds of walls and kill the opponents, so they are essential tool for every bomberman.

2.5 Walls

Since players will play the game in a maze, they are surrounded with the walls. However, players can use their bombs to break the walls and approach their opponents in different ways. Breaking a wall may also drop some powerups randomly that players can use to reach their goal faster and easier. However, not every wall is breakable. There are 3 types of the walls:

Brick Walls: They are the classic wall that easily can be exploded by the bombs.

Strong Brick Walls: They are stronger walls than the bricks walls and players need to explode it two times to fully break the wall. In the first explosion, it transforms to normal brick walls.

Steel Wall: They are the strongest walls of the game and cannot exploded by the bombs which means they are permanent in the game area.

2.6 Powerups

The powerups are the doping of the bombermen and they increase the abilities of bombermen. Powerups can be dropped by the breaking walls and players can pick them by walking over them. Powerups make the game more funny and unpredictable because it becomes hard to guess the explosion size of a bomber's bomb or guess how many bomb they can drop. There are 4 kinds of powerups:

Increase Bomb Quantity: If player takes this powerup, the number of consecutive bomb drops by the player increased by 1, but it can be maximum 5.

Increase Bomb Magnitude: If player takes this powerup, the bomber's bombs explosion magnitude will increase by 1 square.

Increase Bomberman Speed: If player takes this powerup, their characters walk speed will increase 1 unit.

Shield: If player takes this powerup, they are protected for the next damaging explosion.

3 Functional Requirements

3.1 Playing SinglePlayer Game

Users who wants to play Bombalamasyon game by only one player can select "SinglePlayer" option from main menu and when they choose this option the game will start with default settings from first level. There are 5 levels with increased difficulties. They can have 3 opponents by their own choices all of these players will different colors and the last man standing will win the game. When all levels finished a screen will occur and if the player makes a high score then this score will be added to high score list. Then system will open main menu screen.

3.2 Playing Multiplayer Game

If users want to play this game with two players then they select “Multiplayer” option from main menu. Players will try to be the last man while they are playing with other player and computer opponents.

3.3 Pause Menu

Users can stop the game by pressing “Pause Menu” button to change settings, get help, or exit from the game. However if they don’t want to exit game and just pressed this button to change settings or get help then they can press continue game button for returning to play. Also in this screen, players can see the remaining time and scores of all players.

3.4 Settings

To increase game experience, players can change game configurations by achieving “Settings” from main menu or pause menu. Users can change:

- Game speed according to their own skills and experience in game playing (playing in high speed is usually harder)
- Change volume of sound effects
- Change volume of background music

Then they can click “Save and Exit” or “Exit without Saving” button to exit from “Settings”.

3.5 Help

Player can be informed about rules, game information or keys by entering to “Help” option with using buttons in Main Menu or Pause Menu. The help menu contains the following information:

- The purpose of the game
- Rules of the game
- Control keys
- Wall types and powerups

The gaming experience can be increased by learning more about the game and playing better.

3.6 Credits

General information about developers of the game included in Credits for users to allow them contact with developers to express suggestions and their own experience about the game. With the feedback from the users, game can be improved.

3.7 High Scores

Players can select “High Scores” from Main Menu to look best scores of the players. Also, at end of the game if the player pass one of the high scores, they enter this list. This section helps people to compare their skills and scores with other people who play this game and it raises the competition in the game which makes it more enjoyable.

4 Non-functional Requirements

1.1. Performance

The Bombalamasyon is an interactive game for 1 or 2 players. For a good game experience, players should see their commands’ effects directly on screen. Moreover, this game is an arcade game which includes explosions, powerups and objects that can change states so that the view on the screen should be changed constantly without any delays. So system requirements will be tried to minimized as much as possible to make that game work in every computer since it is a basic and old type game.

4.1 Game Graphics

In order to satisfy a good gaming experience to users, the game graphics should be good looking. Also, there are different type of objects and players in the game area so that the graphics should prevent users from get confused, in other words, it should be plain and effective.

4.2 Useful User Interface

The graphical user interface should be simple and well-organized to welcome new users successfully and make them comfortable in the game quickly. A better user interface make

the accessibility to game sections simple and faster so that it increases the gaming experience of users.

4.3 Improvability and Extendibility

With the feedback of the users, the game should be opened to be enhanced and be fixed for problems that users encounter in order to prevent them from abandon playing the game.

Furthermore, the game should be able to be extended with the new ideas of the developers of users to attract more users to play this game and compete more with other games.

5 System Models

5.1 Use Case Model

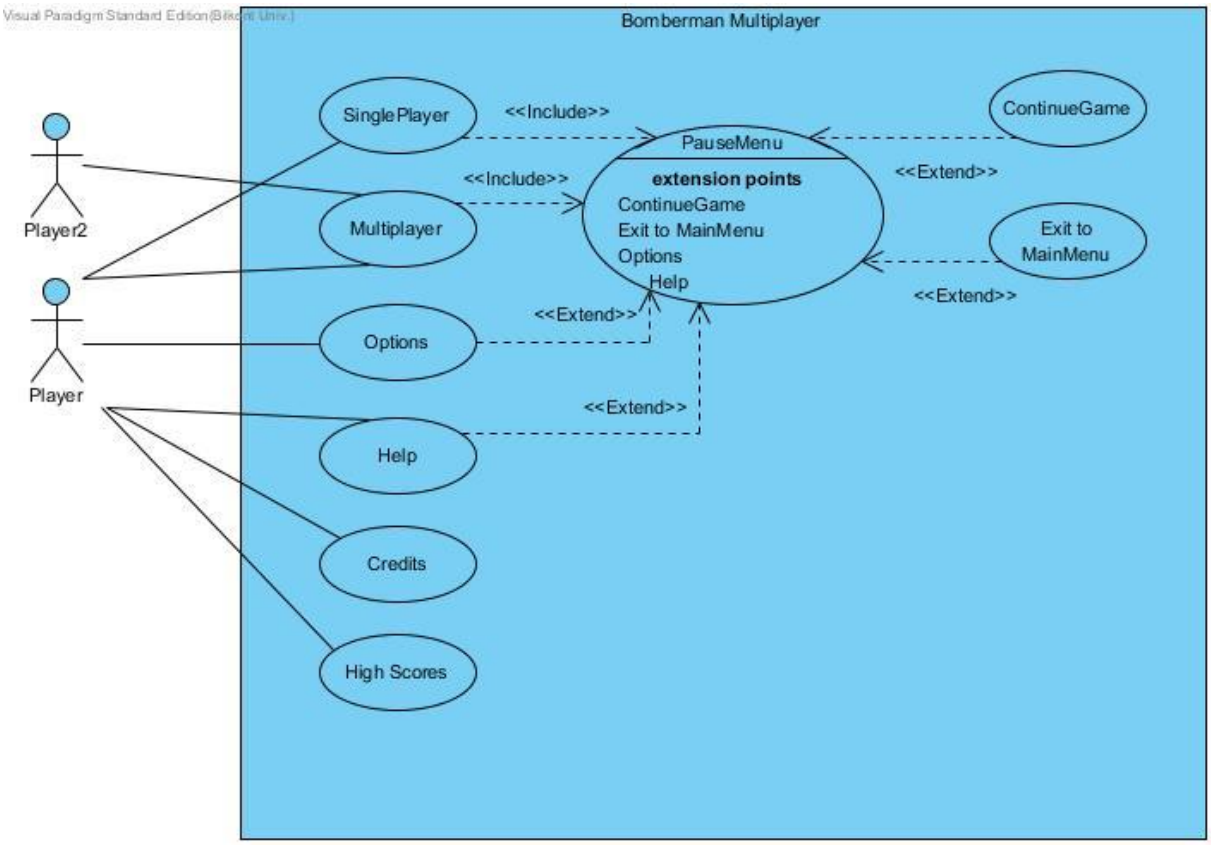

Figure 1: Use Case Diagram

1. **Use case:** SinglePlayer

   Participating actors: Player

   Interests: Player aims to be the last man standing with a high score.

   Entry condition: Player had executed the game and he is on the main menu, presses the “SinglePlayer” button.

   Exit condition: Player presses the “Pause Game” button or game is over.

   Pre-condition: The default or last saved settings is applied.

   Post-condition: The score of the player is added to high score chart if he wins and makes enough score.

   Main Flow of Events:

   1. Player starts the game and clicks “SinglePlayer” button.
   2. System creates the game area for the first level.
   3. Player starts playing corresponding level.
   4. The game continues until only one player remains. If the player is died, the level directly ends without waiting computer opponents.
   5. At the end of level, next level is loaded.

      Steps 2-5 is repeated until all levels are completed or player exits.

   6. At the end, the System brings the game over screen front and if the player has a high score around previous high scores, its score is written on high scores chard with the Player’s name.
   7. The player is brought back to main menu screen.

   Alternative Flows:

   A. More than one players are still alive when time limit is reached.

      A.1. Bomber with highest score so far is selected to be winner.

      A.2. System brings the game over screen front and if the player is winner same procedure is applied.

   B. User clicks the “Pause Game” button.

      B.1. Pause menu is brought front.

      B.2. Case 3 is applied.

2. Use case: Multiplayer

Participating actors: Player1 & Player2

Interests: Players aims to be the last man standing by defeating both other player and computer opponents with a high score.

Entry condition: Player had executed the game and he is on the main menu, presses the “Multiplayer” button.

Exit condition: Player presses the “Pause Game” button or game is over.

Pre-condition: The default or last saved settings is applied.

Post-condition: The score of the winner player is added to high score chart if he makes enough score.

Main Flow of Events:

1. Player starts the game and clicks “Multiplayer” button.
2. System creates the game area for the first level.
3. Players start playing corresponding level.
4. The game continues until only one player remains. If both 2 players are died, the game directly ends without waiting computer opponents.
5. At the end of level, next level is loaded.

Steps 2-5 is repeated until all levels are completed or player exits.

6. At the end, the System brings the game over screen front and if one of the 2 players has a high score around previous high scores, his score is written on high scores chard with that Player’s name.
7. System displays the main menu screen.

Alternative Flows:

A. More than one players are still alive when time limit is reached.

A.1. Bomber with highest score so far is selected to be winner.

A.2. System brings the game over screen front and if winner is one of the players, same procedure is applied.

B. Users click the “Pause Game” button.

B.1. Pause menu is brought front, is controlled by only one player.

B.2. Case 3 is applied.

3. Use case: PauseMenu

Participating actors: **Player**

Interests: Player aims to pause the game

Entry condition: A single or multiplayer game is being played and user clicks the "Pause Game" button.

Exit condition: Player presses the "Continue Game", "Settings", "Help" or "Exit to Main Menu" buttons.

Pre-condition: Current game time and scores are shown in screen.

Post-condition: -

Main Flow of Events:

1. User clicks "Pause Game" button from game screen.
2. System freeze the game timer and displays the game timer and current game scores.
3. If user clicks "Continue Game" button, previous game screen is brought back and continues.
4. If user clicks "Settings" button, case 4 is applied.
5. If user clicks "Help" button, case 5 is applied.
6. If user clicks "Exit to Main Menu" button, the Main Menu is displayed.

4. Use Case: Settings

Participating actors: Player

Interests: Player aims to change the game configurations.

Entry condition: Player is in main menu or pause menu and clicks the "Settings" button.

Exit condition: Player presses the "Save & Exit" or "Exit without Saving" buttons.

Pre-condition: The default or last saved settings is shown in configurations.

Post-condition: User's changes is saved.

Main Flow of Events:

1. User clicks "Settings" button from main menu or pause screen.
2. System displays the Settings screen where the adjustable settings with current configurations are shown.

3. User configures the game speed and volume settings of sounds and music according to his desire.

4. User clicks “Save & Exit” button and System brings user to Pause Menu or Main Menu according to the state before the Settings.

Alternative Flows:

A. User goes back without any change.

A.1. User clicks “Exit without Saving” button and System brings user to Pause Menu or Main Menu according to the state before the Settings.

5. Use Case: Help

Participating actors: Player

Interests: Player wants to be informed about game rules and playing.

Entry condition: Player is in main menu or pause menu and clicks the “Help” button.

Exit condition: Player presses the “Back” button.

Pre-condition: -

Post-condition: -

Main Flow of Events:

1. User clicks “Help” button from main menu or pause screen.

2. System displays the Help screen where user can find game information and instructions.

3. User clicks “Back” button and System brings user to Pause Menu or Main Menu according to the state before the Help.

6. Use Case: Credits

Participating actors: Player

Interests: Player wants to be informed about the developers of the game.

Entry condition: Player is in main menu and clicks the “Credits” button.

Exit condition: Player presses the “Back” button.

Pre-condition: -

Post-condition: -

Main Flow of Events:

1. User clicks “Credits” button from main menu.

2. System displays the Credits screen where user can find information about the developers.

3. User clicks “Back” button and System brings user to Main Menu.

7. **Use Case:** High Scores

Participating actors: Player

Interests: Player wants to see game’s high scores records.

Entry condition: Player is in main menu and clicks the “High Scores” button.

Exit condition: Player presses the “Back” button.

Pre-condition: -

Post-condition: -

Main Flow of Events:

1. User clicks “High Scores” button from main menu.

2. System displays the sorted high scores chart.

3. User clicks “Back” button and System brings user to Main Menu.

5.2 Object and Class Model

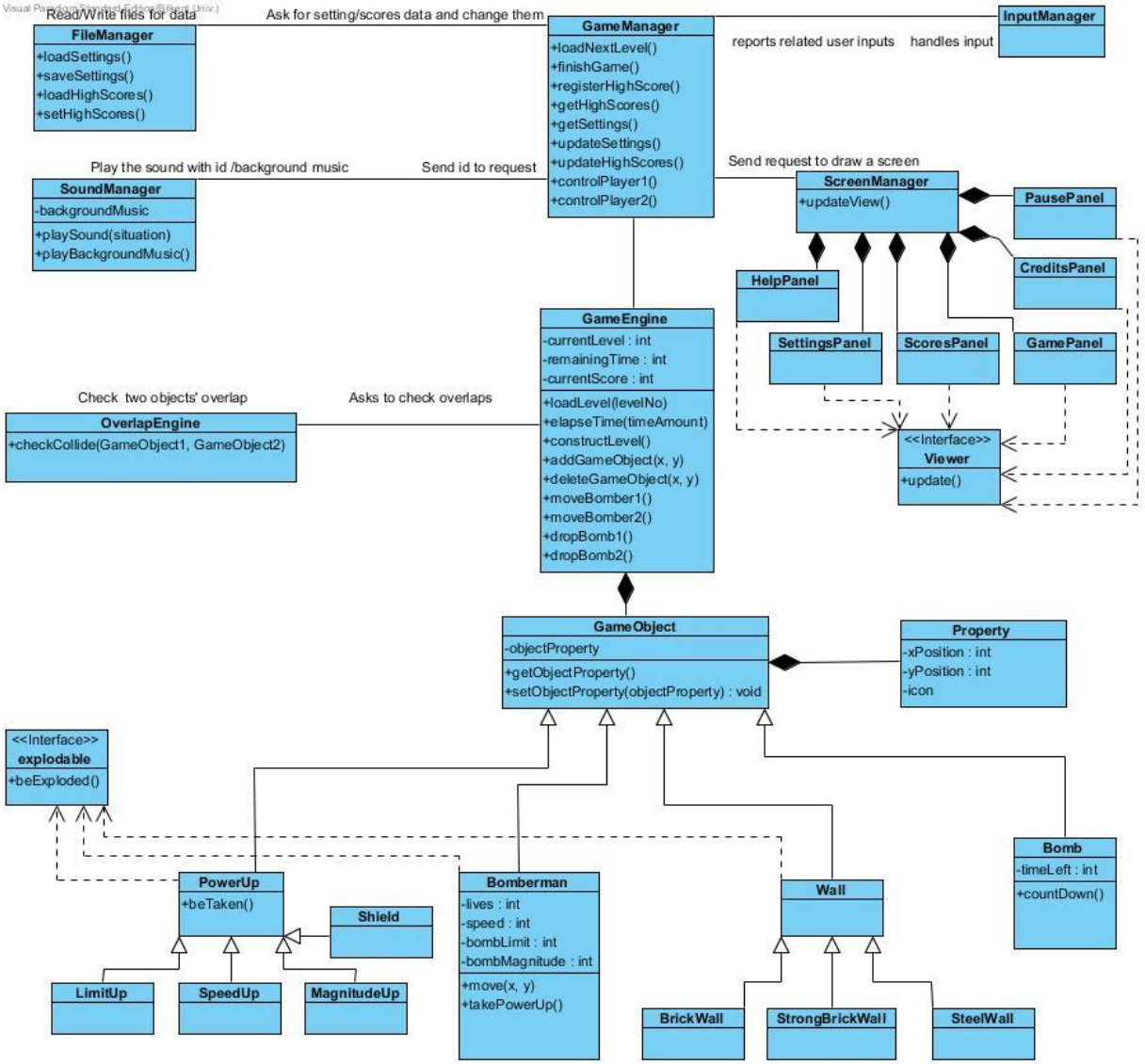

Figure 2: UML Class Diagram

The **GameObject** represents the entity object's abstraction. The instances of the game objects (powerup, bomberman, wall, bomb) are held by the **GameEngine**. With the advance of the game, engine can add or remove game objects from game area. Also it can manipulate the objects' properties such as moving them, with the help of **overlap engine**. The objects that can be affected by a bomb is put together with an interface named **explodable.**

GameEngine can initialize a level and hold data for current level. Also, it process the times and holds a timer. With the data that engine holds, the **GameManager**, the control object, can finish game, load another one, and update high scores. **GameManager** is in

communication with other 4 managers that establish the part of I/O of the game. The **ScreenManager**, manages the view of the game with 6 boundary objects that are together within a **Viewer** interface. The user's desires are passed from **InputManager** to **GameEngine** through **Game Manager** with control methods. These methods calls the moveBomber and dropBomb methods of engine.

5.3 Dynamic Models

5.3.1 Activity Diagram

Activity diagram show the main flow of the game. There can be two events in the game loop. One of them is player presses the keys and move the character or drop a bomb and other event is the dropped bombs' explosions. According to the collision information our game updates the map, make the player win or lose.

The player inputs are needed for game to progress. If player drops a bomb its timer starts, and if the player moves, overlay engine decide the collisions and let the player move or not according to the path. All activities are merged at the end and game loop looking the inputs and explosions again.

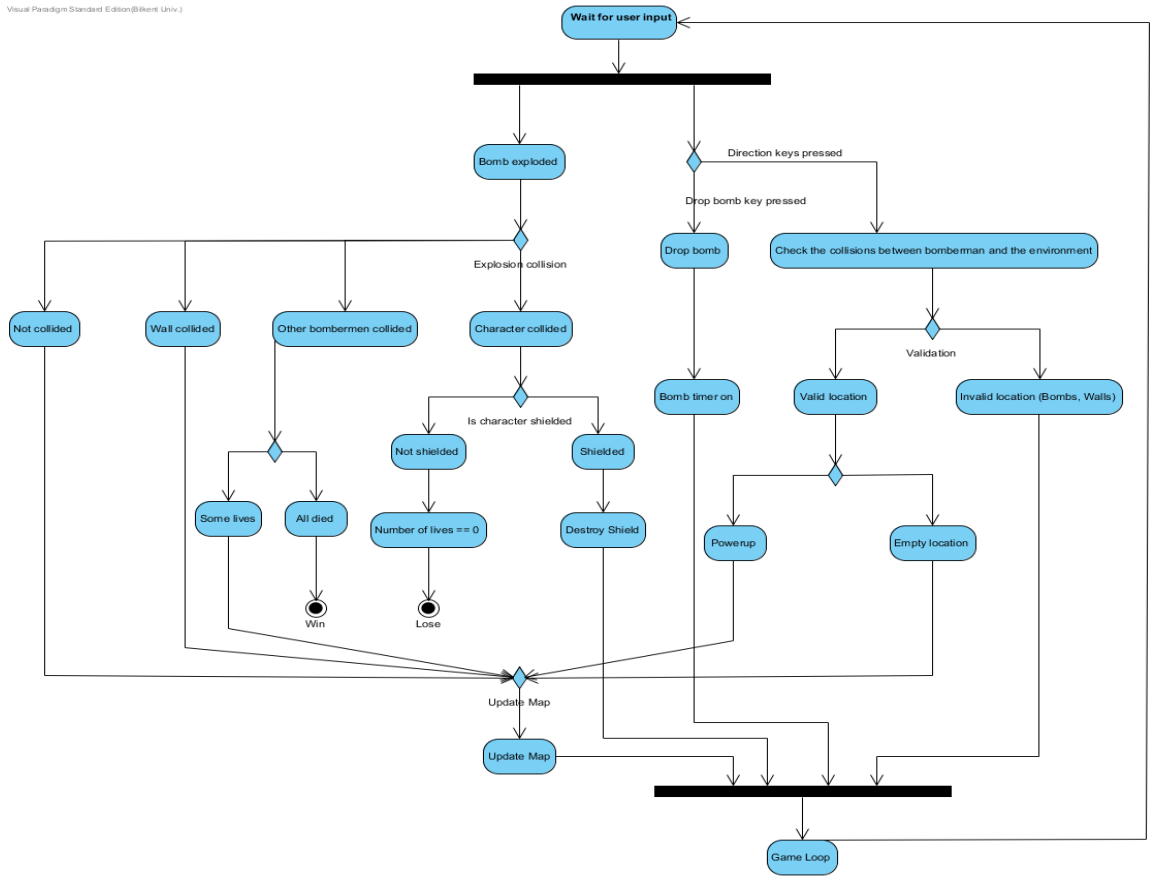

Figure 3: Activity Diagram

5.3.2 State Diagrams

Game State

Game is generally on the default state and if any one of the bombermen or walls are exploded it will go to the update state which updates the map. However, if all opponent bombermen or player is dead or the time is up game will go to the level and state and it ends the level.

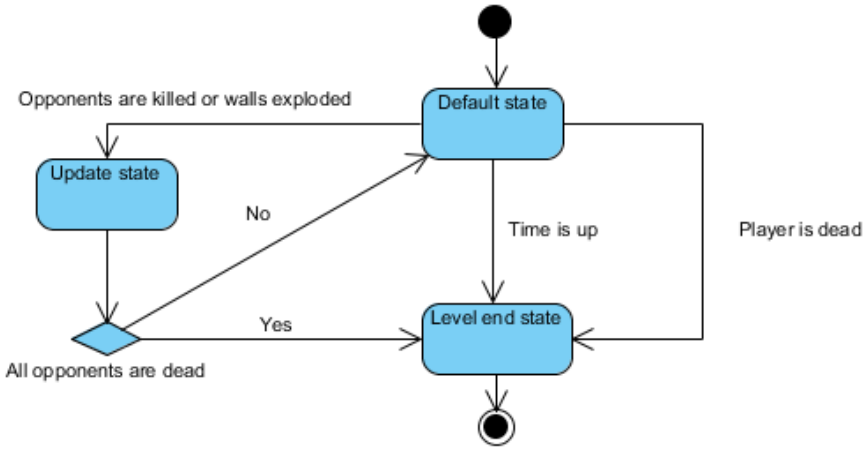

Figure 4: Game State Diagram

Bomberman State

The bomberman's general state is idle which is doing nothing and every other state is reached by the idle state. Drop bomb state is a state that bomberman drops a bomb and become idle again. Move state is a state that bomberman is moving along the map and it can take a power up and becomes idle again. Exploded state can be reached by both the bomberman is idle and moving and if it does not have any life remaining the game will end.

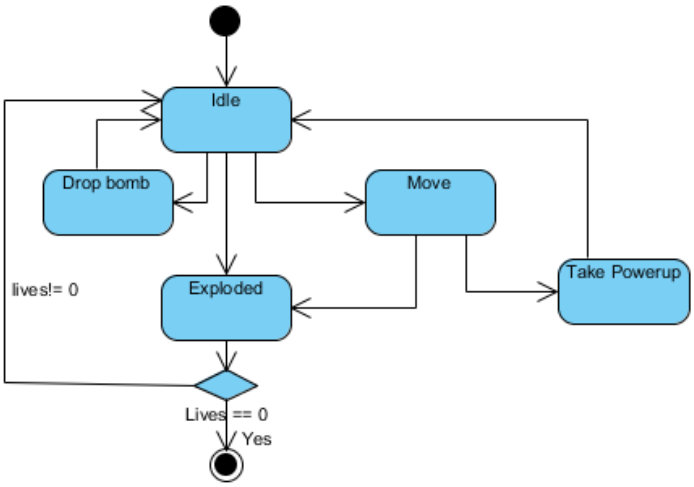

Figure 5: Bomberman State Diagram

5.3.3 Sequence Diagrams

Scenario 1: Start a level

Ahmet stars a single player game from main menu, first level is loaded.

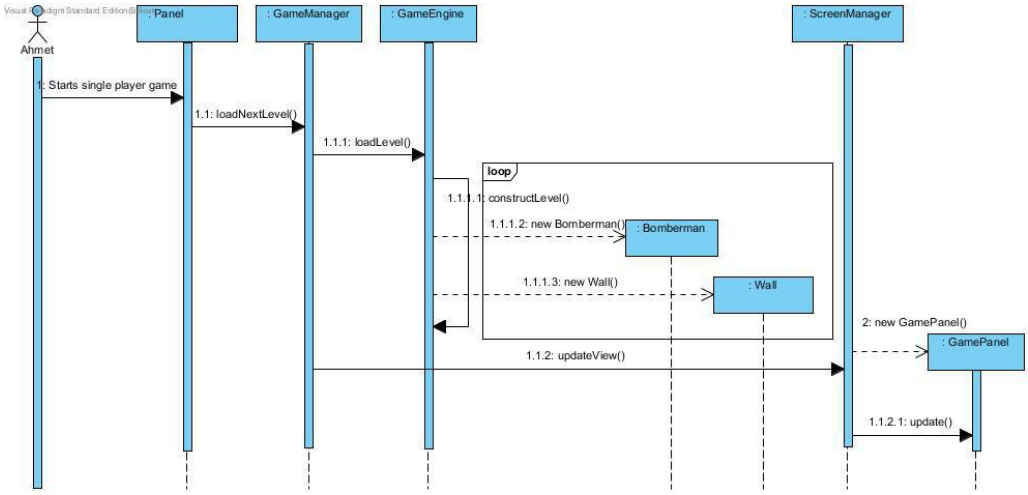

Figure 6: Sequence Diagram for Starting a New Level

In this diagram, input taken from user in current viewed panel and passed to **GameEngine** with its loadLevel() call, and the first level is constructed because the current next level is the first level. The contruct level method is the game engine's call for itself to create **bombermen** and **walls** within a loop. After creation, **GameManager** send a request to **ScreenManager** for creating a GamePanel and updating its view.

Scenario 2: Change Settings

Ahmet executes the game, main menu of the game appears, Ahmet wants to change settings of game. He clicks "Settings" button in the main menu. He turns off the music and he increase the game sound then he goes back to the main menu via "Save & Exit" button.

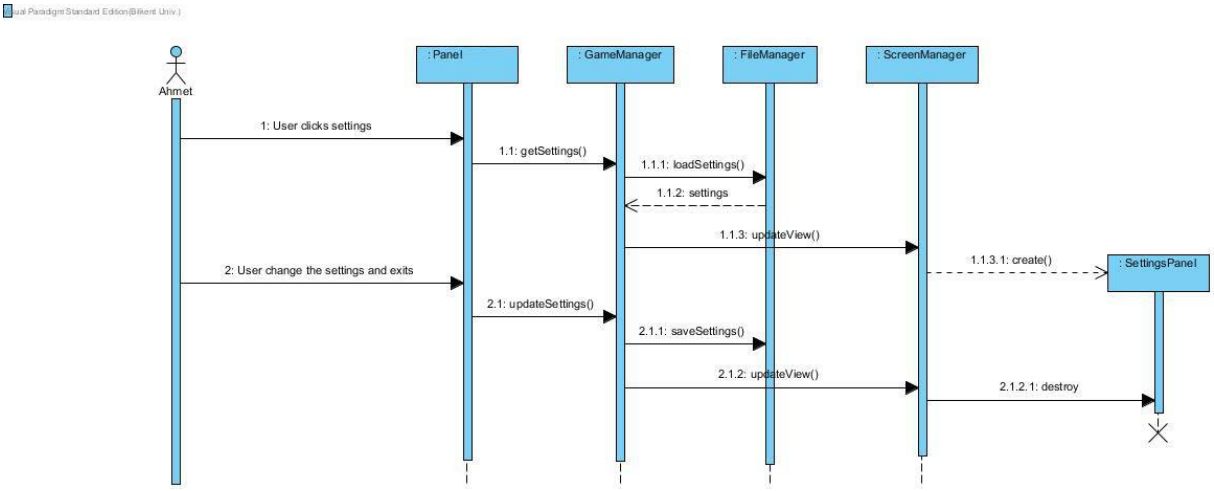

Figure 7: Sequence Diagram for Change Settings

In this diagram, the actor wants to change settings and this desire of actor is passed to **GameManager** by the current viewed panel (it can be Pause or Main Menu). The getSettings() method of **GameManager** is called, which firstly take the current settings(load or user modified) from **FileManager** and send a request to **ScreenManager** to open settings screen with current ones. **ScreenManager** creates a new Settings Panel, and user modifications are passed to GameManager again. **GameManager** send a request to **FileManager** to save the settings on the file and the Settings panel is destroyed.

Scenario 3: Brick wall

Ahmet moves Bomberman, he puts a bomb next to “Brick Wall”, after bomb explosion “Brick Wall” is also exploded and his score is updated.

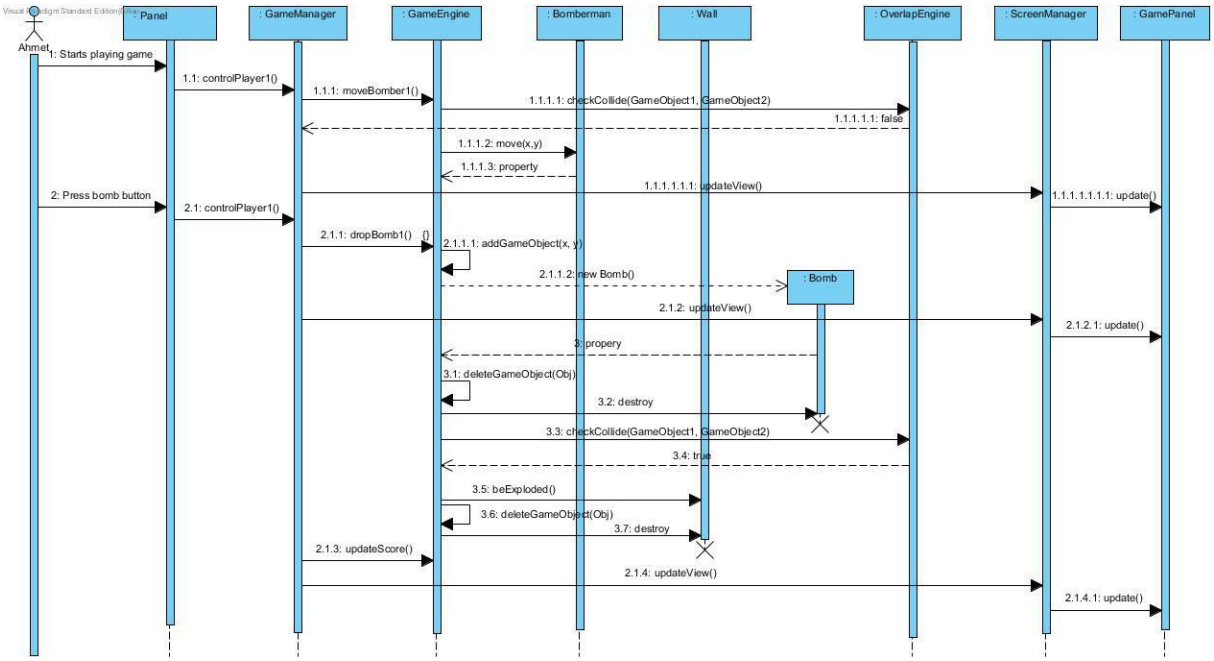

Figure 8: Sequence Diagram for Break a Wall

In this diagram, user’s moves on bomberman is shown and they are passed through panel with controlPlayer methods of **GameManager**. The move of bomberman is checked by **OverlapEngine** because go on top of walls or bombs are not allowed. With the users request for dropping bomb, **GameEngine** call its method for itself to add new object on collection. When the property (timeLeft) of Bomb reaches zero, it is removed by **GameEngine** again and its explosion is checked with the **OverlapEngine** again. The collided wall (if it is breakable of course) is deleted from the collection and view of boundary object,**GamePanel** is updated accordingly. The score of the player is also updated.

Scenario 4: Take PowerUp

While Ahmet is playing, he moves his Bomberman onto a powerup. When it goes up on the power up, the power up is taken by Ahmet's bomberman. The properties of bomberman is changed.

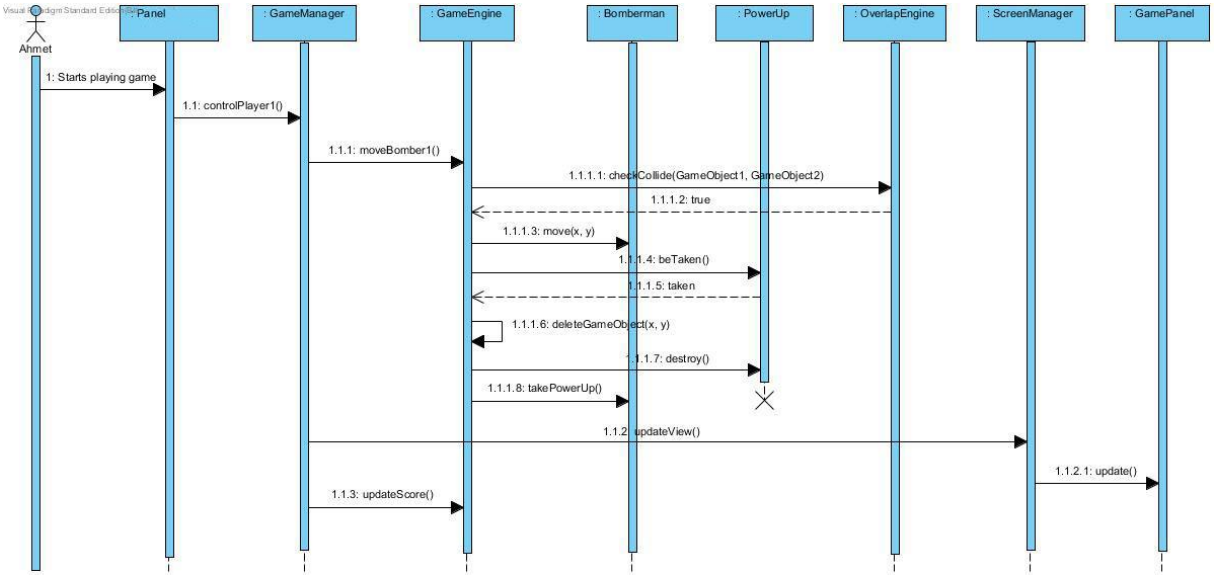

Figure 9: Sequence Diagram for Taking PowerUp

In this diagram, when Ahmet moves his bomberman, the commands are passed through panel to **GameManager**, the **GameEngine** checks for overlapping with any powerups with the help of **OverlapEngine**. When the check returns true, the powerup is deleted from collection and destroyed, also the properties of bomberman is changed with **takePowerup** method. After that, boundary object **GamePanel** and score data is updated as in the brick wall sequence.

Scenario 5: Kill Bomber, End Game and HighScore

While Ahmet and Mehmet is playing last level, all computer opponents had been eliminated. Ahmet drops a bomb and kill Mehmet whose bomber is out of lives. So the game ends, Ahmet has a high score, he enters his name and new highscore table is displayed.

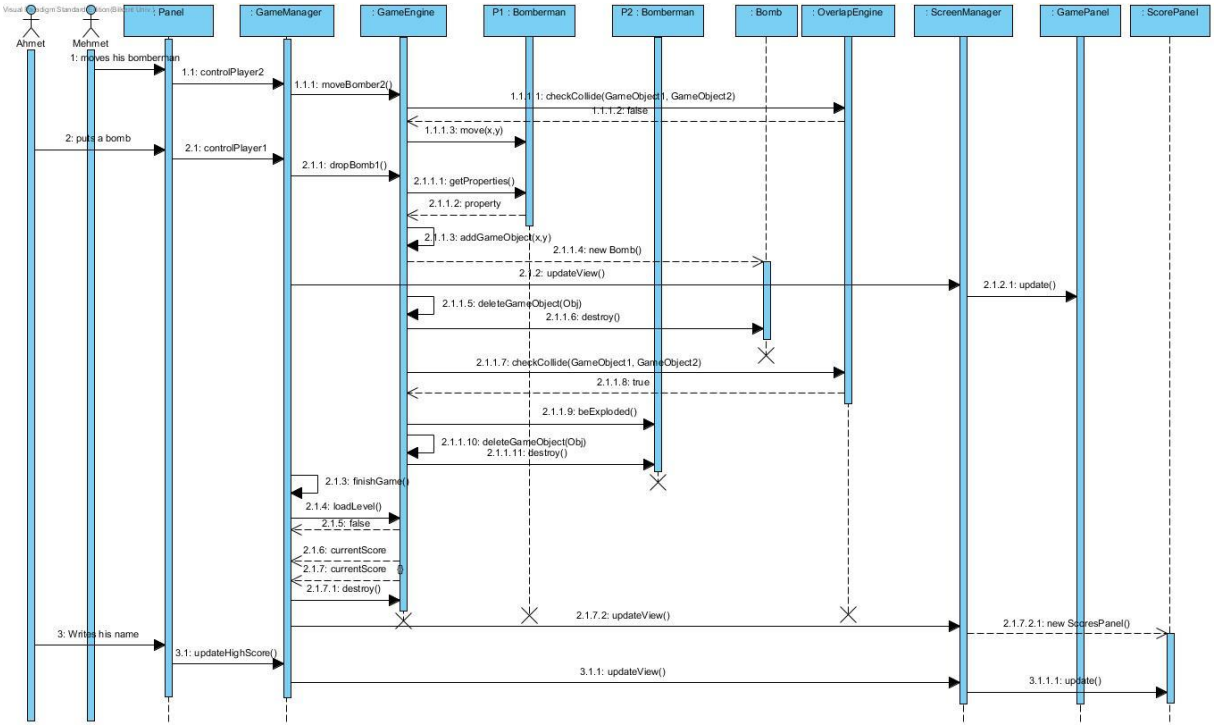

Figure 10: Sequence Diagram for End of Multiplayer Game with Highscore

In this diagram, as in the previous sequences, inputs are taken from current panel. Ahmet drops a bomb and this bomb is collided with Mehmet's bomber. After **OverlapEngine's** check about this collision, **GameEngine** removes the Mehmet's bomber. Since there is only one bomber left, the **GameManager** tells itself to finish game and load the next level. Since there is no next level, the game should be ended so scores is taken from **GameEngine**. The **GameManager** compares the scores with the highscores, and ask the Ahmet for his name to add entry to highscore table with the help of InputManager. **GameManager** updates the highscore data and request a screen update from ScreenManager. **ScreenManager** calls the update method of ScoresPanel which is a boundary object.

5.4 User Interface

5.4.1 Navigational Path

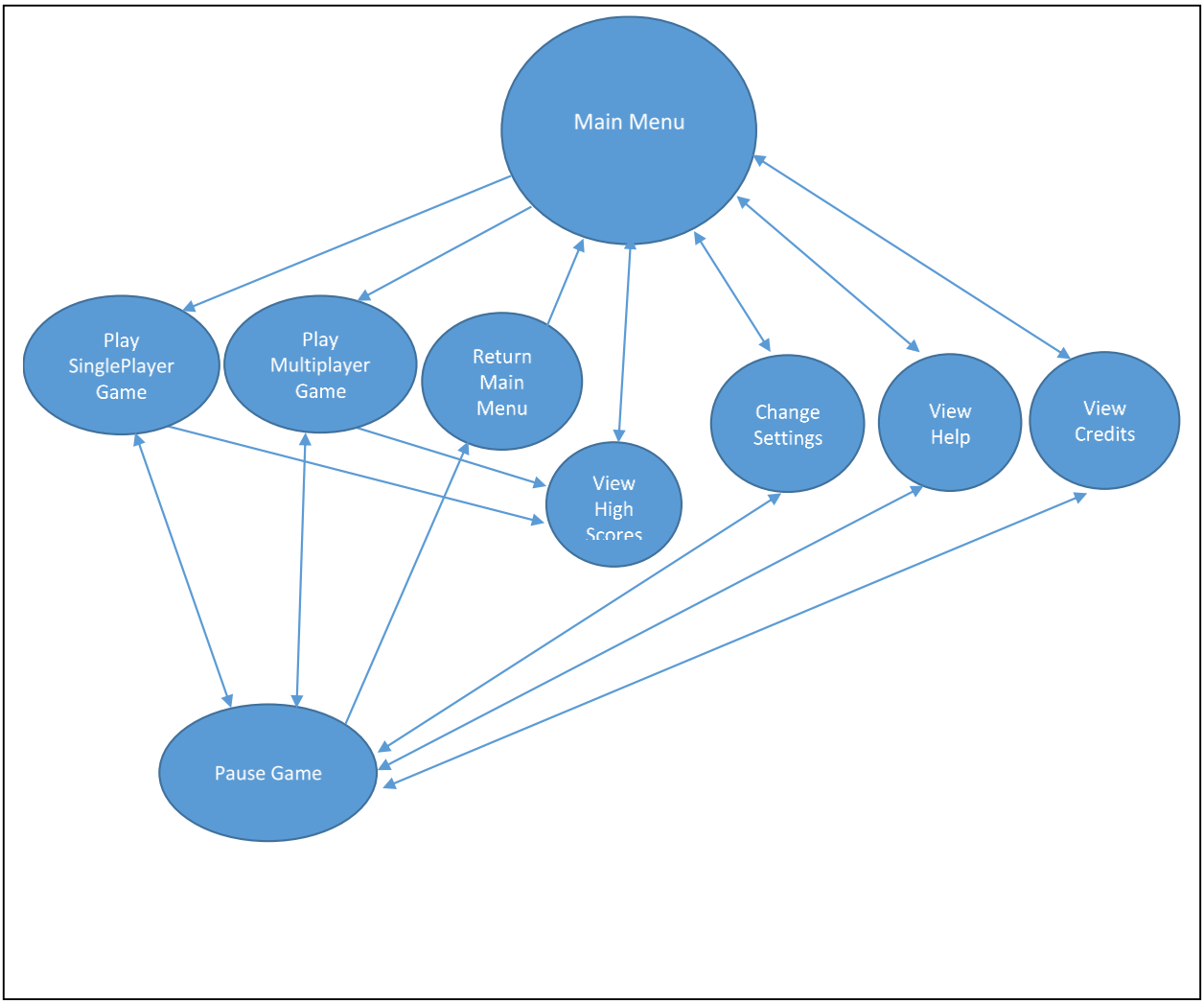

Figure 11: Navigational Path Diagram

5.4.2 Main Menu

Main menu screen is the screen that welcomes user. So it needs to be attractive and it should serve enough guide for users to satisfy better gaming experience.

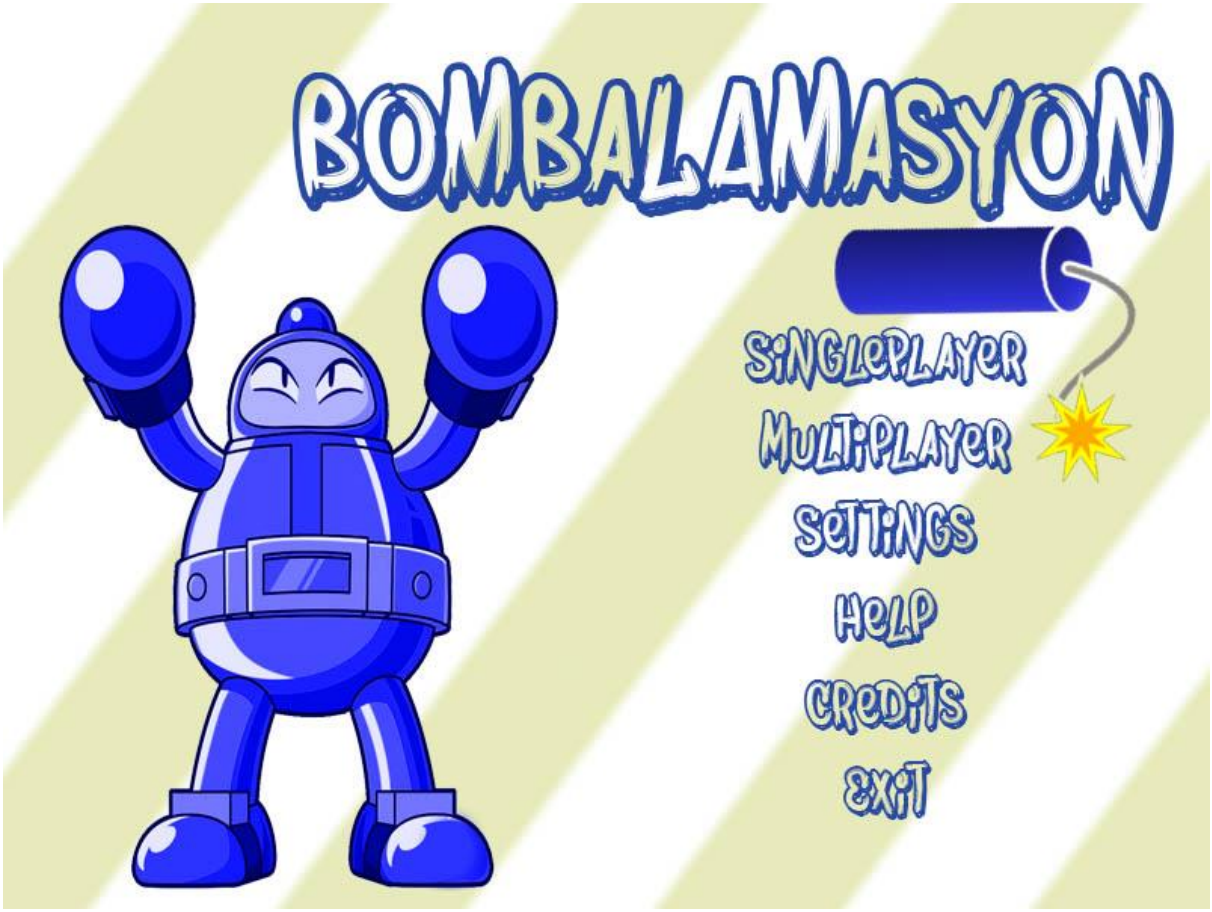

Figure 12: Main Menu Screen [1][2][3]

5.4.3 Game Panel

This game is inspired by an existing game called “Fireman”. So, the game panel of the game will also be imitated. The layout is the same, but the images for walls,bombs,power ups and bombermen will be changed.

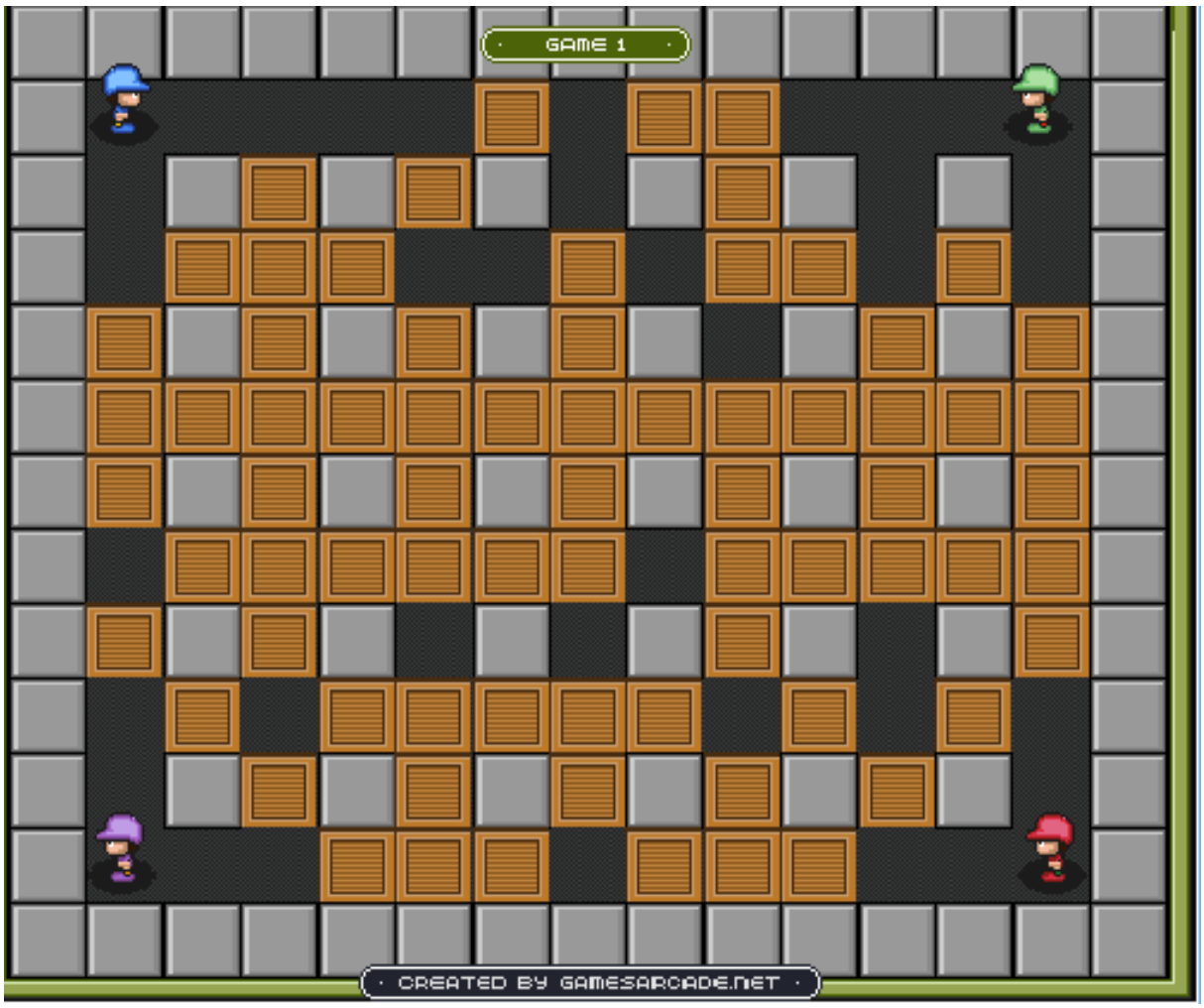

Figure 13: The inspired game panel, from “Fireman”[4]

5.4.4 Settings Panel

The settings panel is designed simple for usability. User can open or close background music and sound effects by putting a tick or removing it. The master volume can be configured with the help of slide bar.

Figure 14: Settings Panel with 3 options [3]

5.4.5 Help Screen

The help screen will also be inspired by the “Fireman”. There will be key instructions to guide the users and explanations about the aim and process of the game. The layout will be same with settings screen. This design will also be applied to credits and highscore screens.

Figure 15: The desired help screen [4]

5.4.6 Bomberman

The bomberman is the main character of the game and there is 4 in each game. To distinguish bombermen from each other, 4 different colors are used. A plain image is selected and the image will change when bomberman goes left or right.

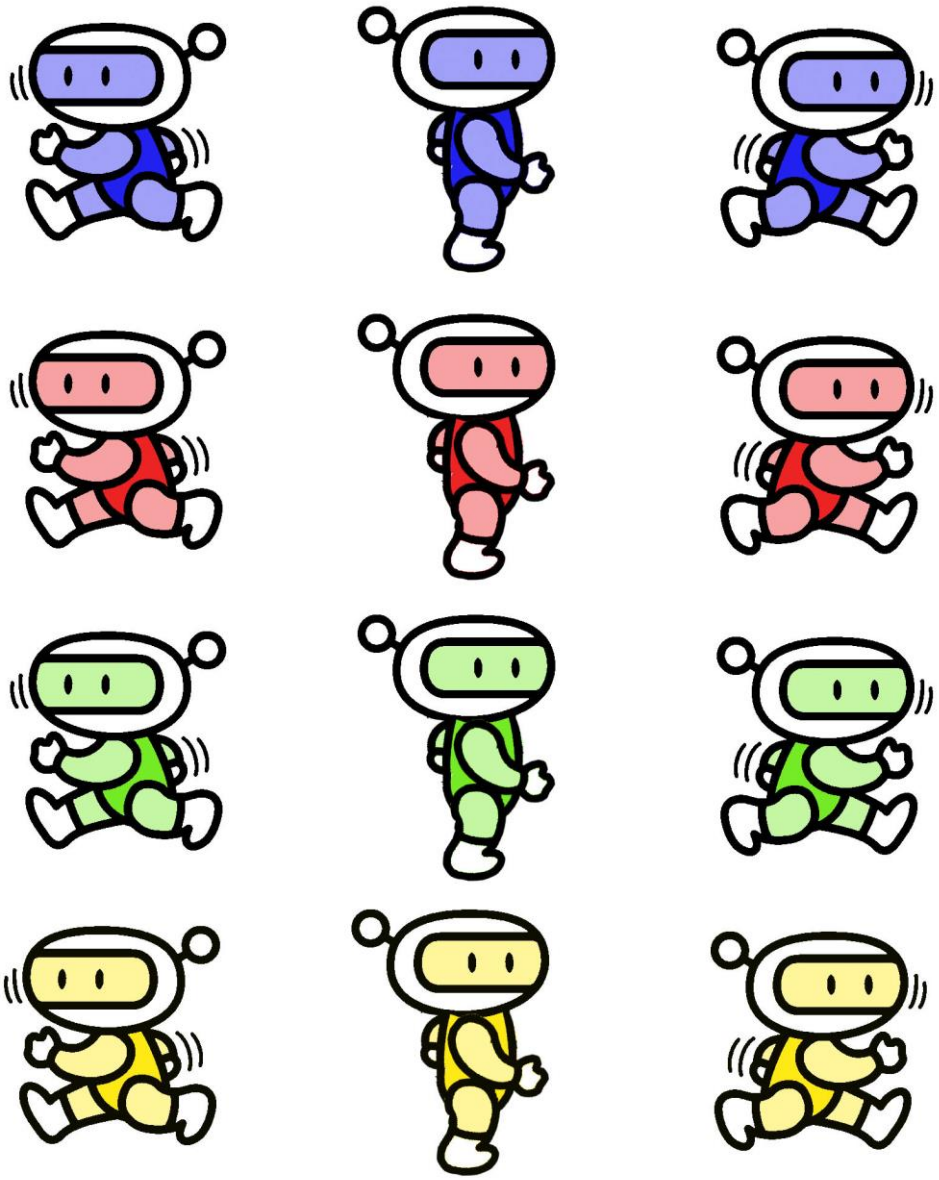

Figure 16: The images of the bombermen [5]

5.4.7 Bombs

Bombs are designed to be same color with whoever drop that bomb.

Figure 17: The bombs for each bomberman[2]

5.4.8 Walls

In this game, there are 3 types of walls and they are distinct by their images. These are, brick wall(weak), brick wall(strong), and steel wall(non breakable). Since main type of the first two walls are same(brick), a light colored image with a crack is chosen for weak one.

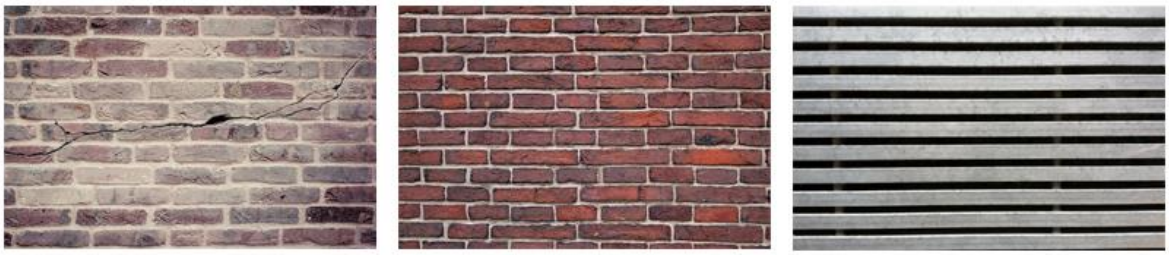

Figure 18: Wall types: 1. Brick Wall 2. Strong Brick Wall 3. Steel Wall [6]

5.4.9 Powerups

There are 4 powerups in this game and their pictures are chosen according to their function.

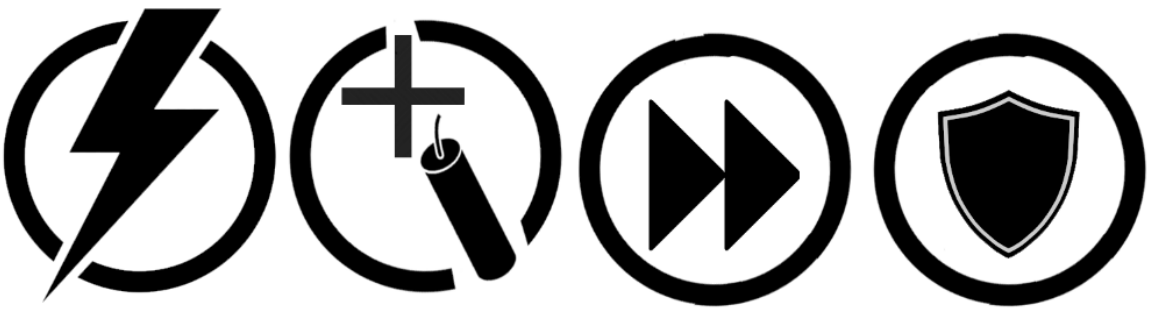

Figure 19: Powerup symbols for bomb magnitude, bomb count, moving speed and shield.

6 References

[1]: Main screen image the character “Gold” in “Bomberman” game.

http://bomberman.wikia.com/wiki/File:Gold_2.jpg, 5 March 2016.

[2]: Bomb image.

http://www.icone-png.com/theme-dynamite.php, 6 March 2016

[3]: Font named “bombing”.

http://www.dafont.com/bombing.font, 4 March 2016

[4]: Game panel and help screen from “Fireman” game.

http://www.kraloyun.com/Oyun/Fireman, 6 March 2016

[5]: Bomber image.

http://randomhoohaas.flyingomelette.com/bomb/nes-1 , 6 March 2016

[6]: Wall images.

https://pixabay.com, 6 March 2016