CS 319 - Object-Oriented Software Engineering

Analysis Report – Revised

Bombplan

Group 1

Asena Rana Yozgatlı	*21000132*
Berk Yurttaş	*21200581*
Mehmet Furkan Şahin	*21201385*
Saner Turhaner	*21100475*

Deadline: 06/03/2016

Course Instructor: Uğur Doğrusöz

Table of Contents

Table of Figures

1. Introduction

Bombplan is a brand-new version of classical Bomberman game. The main aim in Bomberman is to reach to exit door without being killed by the monsters in the game. The main character has the power to plant the bombs in different locations on the map. The map is in form of a maze and there are some wall blocks that can be destroyed by the bombs and some others that can never be destroyed. The character starts from one end of the maze and by destroying the suitable walls and running away from the monsters, tries to reach to the exit door. The bombs are not only affective on the walls, if there is the main character or one of the monsters in the range of the bomb, they can also be destroyed.

In this report, the main design of the Bombplan will be discussed. We will provide Bombplan with an OOP implementation. The following sections will lead to discussions related to requirement analysis, and the analysis with object models and dynamic models. During the requirement analysis part we will examine functional and non-functional requirements, constraints, scenarios, use case models, and user interface of Bombplan. At the end, we will conclude with a general revision of the whole report.

2. Overview

Bombplan is a desktop application that will be created with Java. Game instructions will be as following;

- The main character (from now on, it will be referred as bomber) will have only the ability to plant a bomb.
- A bomb has a range of one wall block from 4 sides. The waiting time for the explosion of the bomb is 3 seconds, default. The bomb features can be changed by the taken bonuses that are

randomly distributed inside of the walls on the map. The features that can be changed are as following;

The bomb will be as the following:

Figure 1: Bomb image [1]

When bomb explode explosion will be as the following:

Figure 2: Fire image [1]

- Bomb explosion can be made up to the user. By the help of a taken special bonus, bombs do not explode themselves in 3 seconds; instead, they wait for the user to explode it.
- The range of the bomb is 1 wall block from 4 sides -default-, but it can be extended by a special bonus. After the bonus, the range increases 1 more wall block in depth from 4 sides. Since, user can take from that special bonus more than once, at some point, range can be 4 wall blocks in depth from 4 sides.
- Normally, user cannot plant one more bomb if there is already a planted bomb. However, by the help of another bonus, user can plant multiple bombs. Each bonus increases 1 the maximum number of the bomb that can be planted at once.
- There is a special bonus that different from the other three bonuses. After this special bonus is taken, it gives a random bonus to player.
- There is also a bonus that does not affect bomb. It resets the timer after player takes it.

- Bomber has 3 lives at starting. This can be increased with hearth shaped bonuses by one and of course, it decreases by dying.

  The bomber, hero, will be as the following:

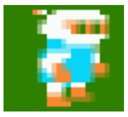
  

  Figure 3: Hero image [1]

- Bomber can die by being in the range of an exploding bomb, fail to kill all of the monsters and to reach the door in the given time or touching to one of the monsters.
- There are 2 types of monsters -slow and fast- in the game. The number of the monsters from these types changes according to the level. Total number of monsters also changes according to the level.

  The slow monster will be as the following:

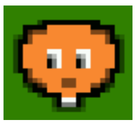
  

  Figure 4: Slow monster image [1]

  The fast monster will be as the following:

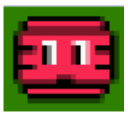
  

  Figure 5: Fast monster image [1]

- There are 2 types of walls in the game. One of them can never be destroyed by bombs, the other can. Their design changes according to the type of the wall and of course all of the bonuses are in the walls that can be destroyed.

The destroyable wall will be as the following:

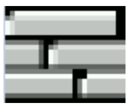


Figure 6: Destroyable wall image [1]

The non-destroyable wall will be as the following:

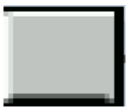


Figure 7: Non-destroyable wall image [1]

- Number of blocks will be definite for each level. However, the blocks will be distributed randomly on the map. Of course there will be some constraints for the distribution of the walls to provide a more homogenous map.
- To pass a level, bomber need to kill all of the monsters and find the secret door inside of one of the wall blocks in the given time.
- To finish the game, bomber should pass 3 levels of maps or spend all of his lives.
- During the game, user earns some points with destroying walls, killing monsters, taking bonuses or passing levels. At the end of the game, the collected point is put in high-score list if it is one of the highest 10 scores ever played on that computer.

3. Functional Requirements

New Game

The player will start to play a game by choosing “New Game” from the main menu of the game. The game will start from the very first level that is already prepared by the developers.

Play Game

The game starts by locating the main character to the top left corner of the map to each level. The map is in form of a maze and its difficulty changes according to the level. The monsters in the field distributed and move randomly. Speed of the monsters again changes according to the level and different type of monsters will be seen in high levels. Their only aim is to touch to the main character and kill him. The main character should kill all of the monsters in the map before losing all of his lives. By default, the main character will start to the game with 3 lives that will give him a chance to die 3 times before loosing the game. To kill a monster, bomber should plant a bomb to a location that possibly a monster will be in the range of it. With only one bomb, it is possible to kill multiple monsters if they are all in the range of the bomb. Additionally, bomber can be stuck in an area but there will be some walls that can be destroyed by bombs and some others that cannot. By destroying the walls, bomber can open a way for himself. In the same time, bonuses will be hidden in these destroyable walls randomly. To reach a bonus, bomber should first destroy the wall. A bomb can both destroy a wall and kill some monsters if there is any in the range of it in the same time. To finish a level of the game successfully, bomber should both find the secret door that is hidden in one of the destroyable walls and kill all of the monsters. The crucial point is that, he door is placed under one of the destroyable walls after the user kills all of the monsters. After passing from the door, the game will load the next level if there is one. If the user passes the last level, or die before succeeding in the game, he will see a menu if his score is one of the highest 10 and he will be able to save his score with a name.

Save Game

In any time of the game, user will be able to save the current state of the game with a name so that he can continue after a while if he needs to quit the game in that moment.

Load Game

The user will be able to see his saved games by choosing “Load Game” from the main menu and we will list his saved games. After he choses one of the saved games, we will load it to the game.

Pause Game

The user will be able to stop the game in anytime and do his urgent job if he should. The game will be paused if he pushes the button “Esc” from his keyboard or choose pause button from the screen of the game. He will see a push menu after pause there will be the list of Save Game, Continue Game, Help, Return to Main Menu, and Quit.

Help

For a user who is completely new to the game, help menu is crucial. Thus, it should be visible to him before starting to play a game. Additionally, it should be reachable while the user playing the game since he might want to check what a bonus is or simply how to move the character.

In the help menu, we will provide the information on the main aim generally, abilities of the character, bonuses, how to use them, and how to move the character.

High Scores

The user will be able to see the high scores in the game by choosing High Scores from the main menu. High scores will be given in form of a table with their names. If there is no saved high score, it will give an appropriate message.

Options

Options menu will be reachable from both main menu and pause menu. Through the options menu, user will be able to set character type, music (on/off), and effect sounds (on/off).

Credits

Credits on the game will be presented in “Credits” that is available on the main menu. It will include developer information, library names and version number.

4. Nonfunctional Requirements

4.1 Performance Requirements

- In order to have a smooth game flow, the software will handle the amount of throughput within the size 1 MB in 1 second, that is necessary for continuity of the game.
- Response time will be low enough so that it will never surpass 1 second.
- Highscores will be saved (written) to the txt file. Reading this data and creating highscore table will be fast, it will not surpass 1 second.

4.2 Reliability Requirements

- The game will be robust. Empty highscore list will not be a problem while displaying highscore list.
- Option preferences, listed highscores and saved games will be stored in files.

4.3 Supportability Requirements

- The game will be executable for every platform that Java works.
- The software will not require any installation process.

4.4 Usability Requirements

- The game rules will not be hard to understand, user shall be able to learn game rules within 2 minutes by reading help instructions.

- Game controls will be easy to understand such that user shallbe able to learn game controls within 1 minute by reading help instructions.

4.5 Constraints

- Bombplan is going to be implemented using Java programming language.
- Game graphics will be drawn by Paint.Net.
- Game language will be English.
- Space that is occupied by the Bombplan on the computer will not exceed 20 MBs.
- It will be an open source software licensed.

5. System Models

5.1 Scenarios

Scenario 1: Play Game

Player Ilkay wants to play a game and starts the game. Since she is pretty new to the game, she wants to get information and sees the help option in the main menu. After clicking the “Help”, she gets enough information on the game. She comes back to the main menu and chooses “New Game”. Default level 1 is loaded and she starts to play the game. Although, she learns the game from “Help”, she dies 3 times and loses all of her lives. Since she is the first player, her score is the top score. Thus, she sees the high scores menu and saves her score. After saving her score, Ilkay comes to the main menu and chooses quit.

Scenario 2: Change Settings

Player Osman enters a new game by pressing “New Game” from Main Menu. Osman plays the game without his earphones, and then he thinks that he can mute the sound since he already does not hear anything. In order to do that Osman stops the game by pressing "Esc" key. Then game pauses and a pause menu shows up. Osman change sound and the music on to off. Then he wants to continue to play game, so he presses "Return to Main Menu" button from the pause menu. After a while he is just bored and he wants to quit. From the pause menu he presses "Quit" and exit from the game.

Scenario 3: Load and Save Game

Player Cenk starts the game. After main menu appears, Cenk clicks "Load Game" button. He chooses last game from all saved games. Game initializes game board and settings according to the last game which Cenk saved. After playing a while, Cenk stops the game and clicks to "Save Game" button in the pause menu. Game saves last settings successfully and returns to pause menu again. Then he presses "Quit" button and exits the game.

5.2 Use Case Model

Use Case Descriptions

Use Case 1

Name:	Play Game
Participating Actors:	Player

Main Flow of events:

- Player starts the game by selecting 'new game' option.
- The system loads the default Level 1 map and s bonusesand monsters randomly through the destroyable walls in the map. The system gives 3 lives to the user and locates the character in top left corner of the map. Also the time counter starts to countdown.

- Player starts to play first level by using keyboard control buttons, he or she destroys all monsters on the map and find door and passes the level 1 within given time.
- Player passes all three levels of the game in the same way and finishes the game successfully.
- System asks player to enter his/her name since its score is among the top ten scores.
- Player enters his name and then high score list shows up.

Entry Condition: Player should start the application, and then press the new game button. The map is loaded with all monsters, walls and bonuses successfully.

Exit Condition: Player finishes all levels and he enters his name for high score list.

Alternative Flow of Events:

- Player finishes all levels successfully but since his score is not among the top ten scores game over message appears at the end of the game.
- Player has lost all of his lives and game over message is displayed on the screen.
- Player stops the game while he is playing and from the pause menu he decides to save game and exit from the game.

Use Case 2

Name: Load Game

Participating Actors: Player

Main Flow of events:

- Player chooses 'load game' choice from the main menu of the game.
- Saved games are listed on a table on the screen.
- The player chooses one of the pre-saved games from the table to start game.
- Game is started by the system successfully.

Entry Condition: Player chooses 'load game' choice from the main menu panel of the game.

Exit Condition: Saved game that is chosen is started by the system successfully.

Alternative Flow of Events:

- Player drops the idea of loading game and he backs to the main menu.
- There is no pre-saved game on the list, and then player backs to the main menu.

Use Case 3

Name: View Credits

Participating Actors: Player

Main Flow of events:

- Player chooses 'view credits' option from the main menu of the game.
- Credits panel is created by the system.
- Player views names, e-mails of developers and current version of the game.
- After player views the credits, he or she backs from this panel to the main menu.

Entry condition: Player chooses 'view credits' option from the main menu of the game.

Exit condition: Player backs from credits panel to the main menu.

Use Case 4

Name: Help

Participating Actors: Player

Main Flow of events:

- Player chooses 'help' option from the main menu to read instructions about the game.
- System shows information about the main aim, abilities of the character, bonuses, how to use them, and how to move the character.
- Player reads all instructions from this panel and backs to the main menu of the game.

Entry Condition: User should choose the 'help' option from the main menu.

Exit Condition: User should back from help menu to the main menu.

Use Case 5

Name: Change Settings

Participating Actors: Player

Main Flow of Events:

- The game is started and from the main menu player selects the ‘options‘ choice.
- Options panel is displayed on the screen by the system.
- Player mutes sound and the music from this panel.
- Settings will be saved by the system.
- Player returns to the main menu.

Entry Condition: The game is started and from the main menu player selects the ‘options‘ choice.

Exit Condition: Player backs to the main menu from options panel.

Alternative Flow of Event:

- Player does not change any settings from this panel and returns to the main menu.

Use Case 6

Name: View High Scores List

Participating Actors: Player

Main Flow of Events:

- The game is started and from the main menu player selects the ’view high scores’ option.
- High Scores panel is displayed on the screen by the system.

- Player looks over the top ten scores that are previously played.
- Player returns to the main menu.

Entry Condition: The game is started and from the main menu player selects the 'view high scores' option.

Exit Condition: Player returns to the main menu.

Alternative Flow of Event:

- High score list is empty, and player returns to the main menu.

Use Case Diagram

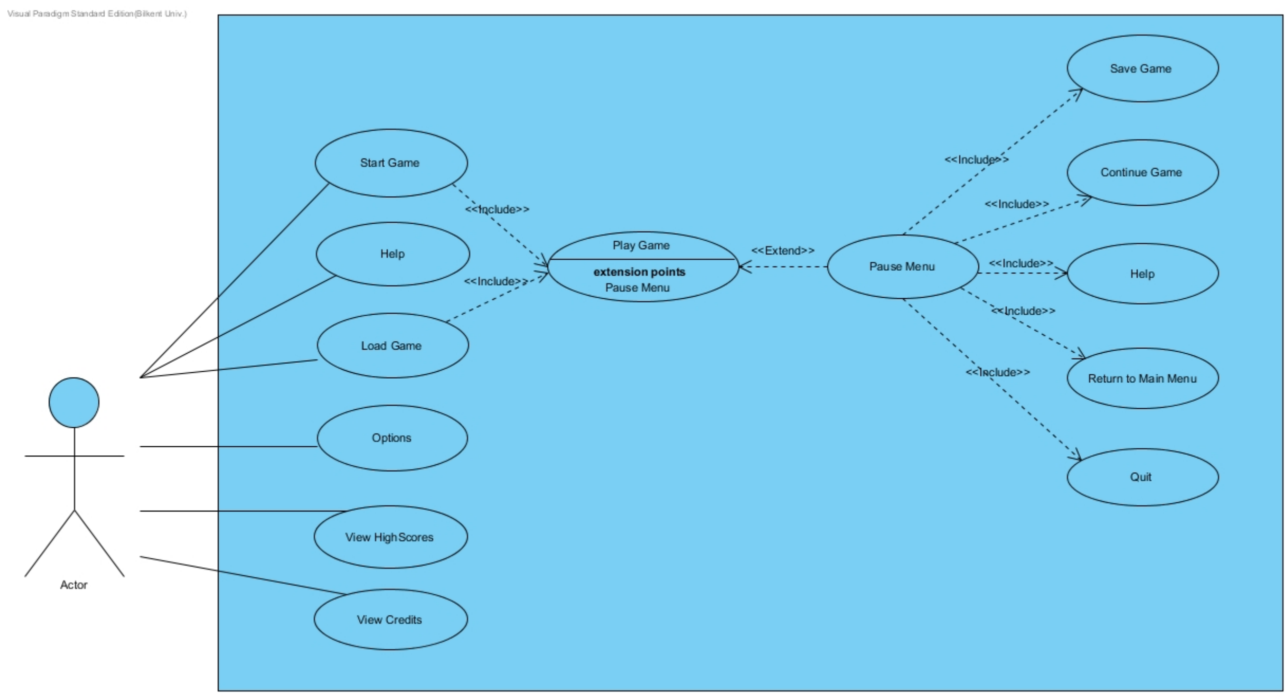


Figure 8: Use Case Diagram

Use case diagram is shown in above Figure 8.

5.3 Object Model

Data Dictionary

Player: Player is the main character of the game. He starts to the game with 3 lives and without extra abilities such as multiple bombing or control when to explode a bomb. The player class holds this information in itself. He is able to plant bombs, take bonuses to improve his abilities and kill the monsters with the help of his bombs. The class will also hold the location and icon information.

Bonus: There are 4 mainly different bonuses. We added one more that randomly selects one of the 4 bonuses and gives to the hero. Bonuses contain power-ups and timer reset. All of them increases the point user collected. Bonus class has a type variable indicating the bonus, location and icon information.

Bonus Type: It is an enumerator class and indicates the bonus type.

Monster: Two different monsters have two different speeds respectively. They move randomly through the map and they hold the speed information according to the type variable. The class also holds the icon information for itself.

Door: Door class has a location and a Boolean variable indicating if it is active or not. Since the door will be hidden under one of the destroyable walls, its activity will change according to the state of the wall. The class also holds the icon information for itself.

Wall: Wall class has location, icon, and destroyable data. Each wall is a square and they might be destroyed according to the location of the bomb. Walls hide bonuses and the door under themselves.

Destroyable wall: Has a destroy method and a specific icon.

Non-destroyable wall: Holds a specific icon. It does not have a destroy method.

Bomb: Bombs are planted by the hero to the specific locations and according to the bonuses the hero has, the effect and control system of them changes. They might be destroyed by a timer as well as a user. The class has location, icon and destroyable information.

Destroyable: Destroyable is an interface that should be implemented by all of the destroyable objects.

Movable: As an interface, implies for all movable objects that need the move method.

Map Object: Holds the location, icon and destroyable information and it is extended by all of the objects that will be in the map of the game.

Location: Holds x and y coordinates for the object. Its variables are mutable for movable objects.

Game Map: Game map holds the information on the objects existing in the map. It is responsible to do the changes demanded by the game engine on the objects. It manages collisions and handles the events happening after a request came from game engine. It holds a 2 dimensional map array to see the current locations and relations of the objects. It also holds the remaining time information.

Collision Manager: Provides a chance for game map to see the collisions through the method checkCollision.

Game Engine: Holds information on the current level and score. It is responsible from the general flow of the game. It manages the backend according to the user interactions.

Sound Manager: Responsible to play the game music.

Menu Panel: General flow of the menus and the functionalities are implemented in the menu panel.

Storage Manager: Handles to load and save actions through interacting with the game engine and fetching or writing the details of the variable and objects in the game.

Game Panel: General drawings of game screens and user inputs are handled by the game panel class.

Class Diagram

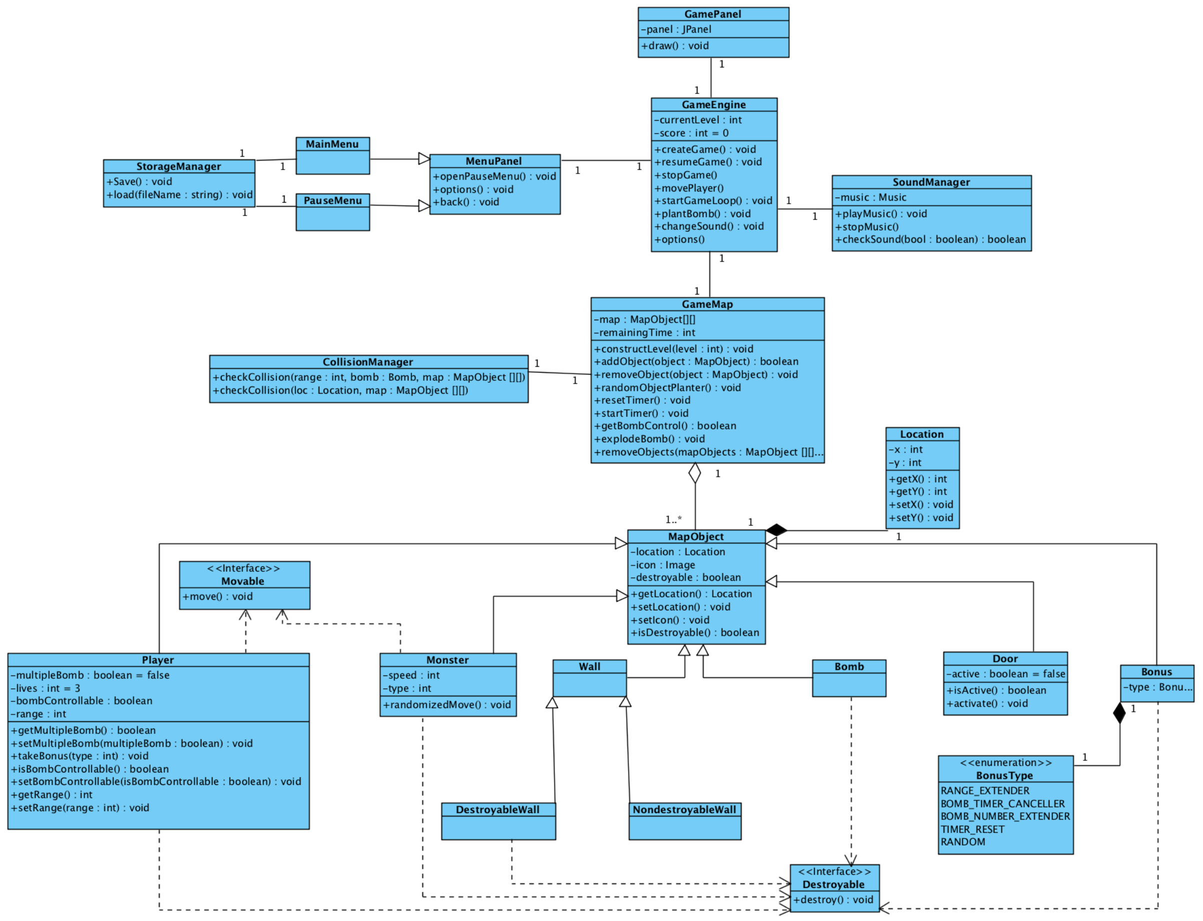


Figure 9: Class Diagram

GameEngine class is the core of the game. It holds 2 manager classes; SoundManager, StorageManager and controls them. It has also a relation with GameMap. GameEngine only access the entity objects through GameMap class. GameMap has a property of GameObject array. GameMap controls MapObjects according to this array. Class diagram is shown in above Figure 9.

5.4 Dynamic Models

Activity Diagram

The diagram in Figure 10 shows the general game flow of the game. Player can initialize the game in two different ways; Starting a new game or loading a previously saved game. After initializing it, general game loop starts. During the game flow, in each time unit, it is checked if there is any monster left. If there are some monsters and the user is dead -game is over-, we know that the game is lost. After this step, according to the score the user collected, either he goes to the high score panel and save his score or directly continues to the main menu. If there are some monsters left and the game is not over yet, he continues to try to kill the monsters, destroy the walls, or collecting bonuses. These events are all decided with check collision step. According to the collision, maybe there is no, game map is updated and monster number check happens and game loops back to the collision check step. After killing all of the monsters, the loop breaks and the door is placed in one of the available places randomly. After the hero reaches to the door successfully, game loops back to initialize game step. If he cannot reach to the door successfully in the given time period, the game is over and again high score check happens. In any stage, if the hero loses all of his lives, we loop back to high score check stage.

Activity diagram for the game flow is as the following:

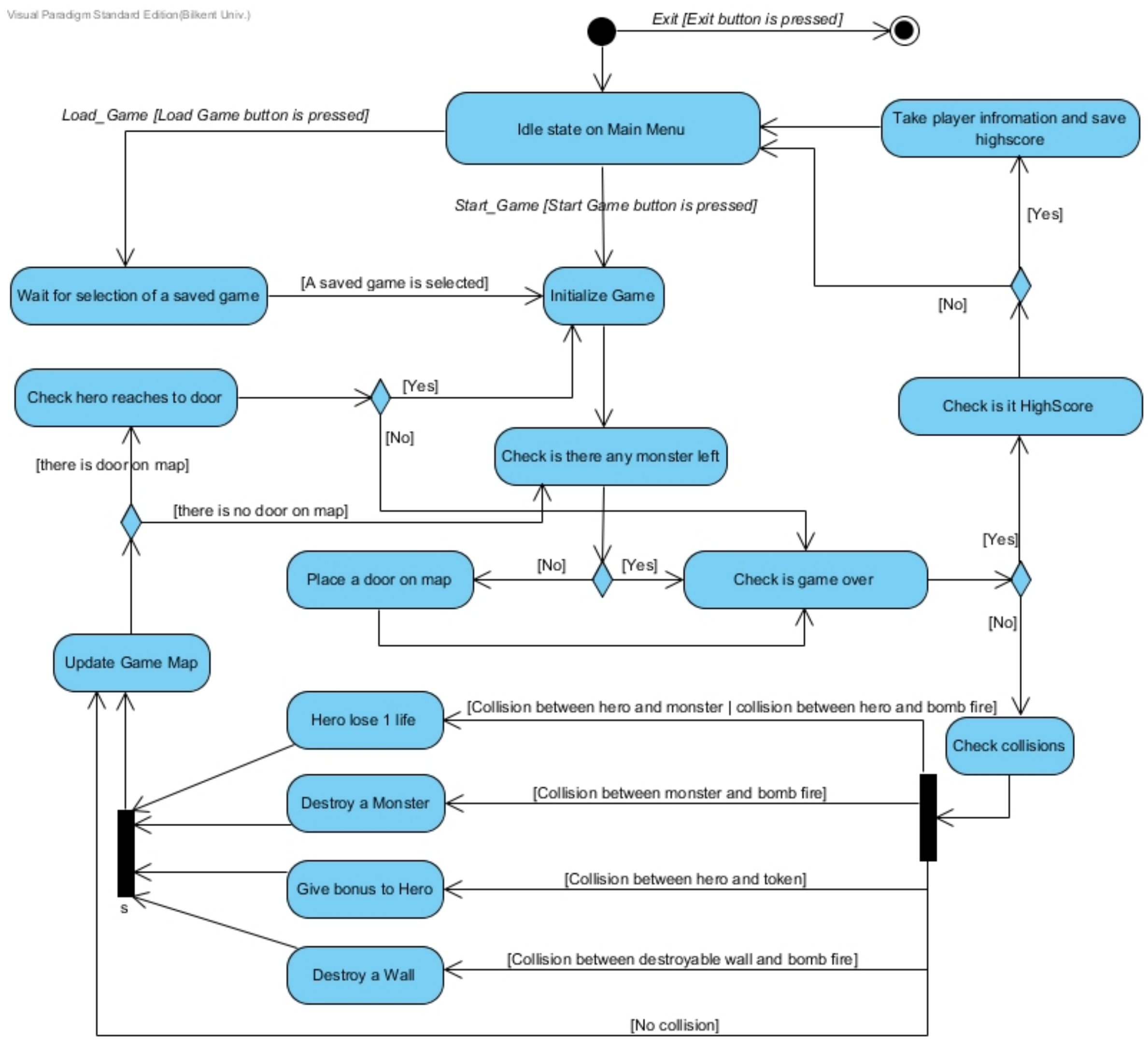

Visual Paradigm Standard Edition(Bilkent Univ.)
Exit [Exit button is pressed]
Load_Game [Load Game button is pressed]
Idle state on Main Menu
Take player infromation and save highscore
Start_Game [Start Game button is pressed]
[Yes]
Wait for selection of a saved game
[A saved game is selected]
Initialize Game
[No]
Check hero reaches to door
[Yes]
Check is it HighScore
[No]
Check is there any monster left
[there is door on map]
[there is no door on map]
[Yes]
Place a door on map
[No]
[Yes]
Check is game over
[No]
Update Game Map
Hero lose 1 life
[Collision between hero and monster | collision between hero and bomb fire]
Check collisions
Destroy a Monster
[Collision between monster and bomb fire]
Give bonus to Hero
[Collision between hero and token]
Destroy a Wall
[Collision between destroyable wall and bomb fire]
[No collision]

Figure 10: Activity Diagram for the Game Flow

Sequence Diagrams

Diagram 1: Start Game

This diagram in Figure 11 explains a scenario which the player starts a game. Actor chooses the "Start Game" from the Main Menu and GamePanel triggers the 'createGame' method of GameEngine. GameEngine creates ScreenManager and GameMap. GameEngine then triggers the 'constructLevel' method of GameMap with 'currentLevel' as the parameter. GameMap creates Player, Wall, Monster, Bonus, and Door. GameEngine creates CollisionManager and triggers the 'startTimer' method of GameMap. GameEngine enters the 'startGameLoop' loop of itself.

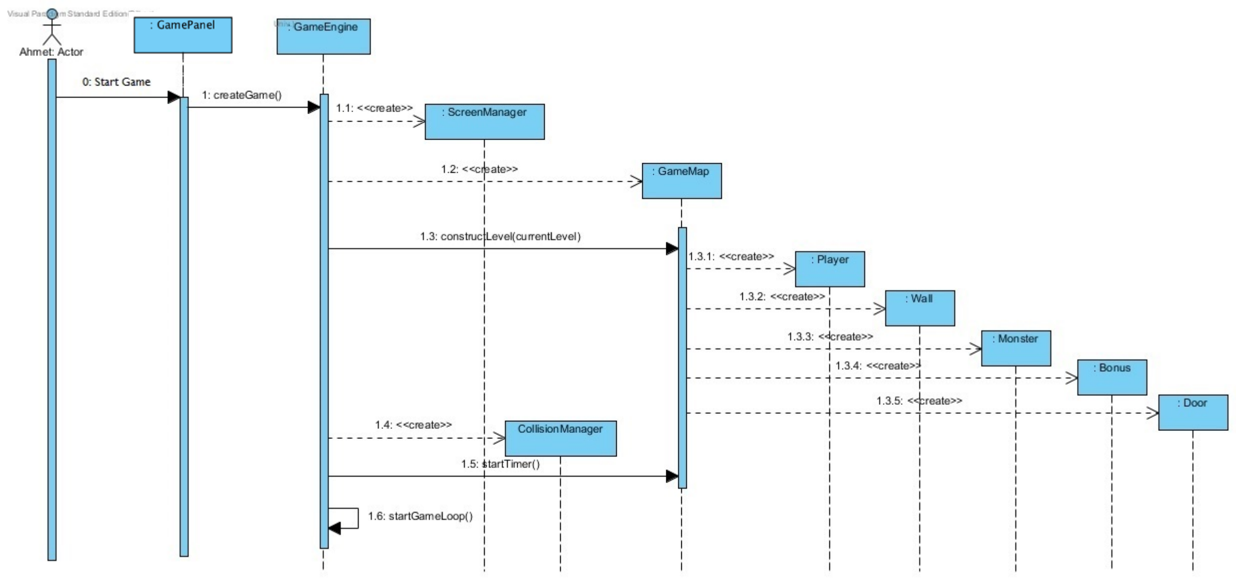


Figure 11: Sequence diagram of starting a game scenario

Diagram 2: Bomb Explosion

This diagram in Figure 12 explains a scenario which a bomb explodes and destroys a monster. Actor plants a bomb by pressing a specific button from the keyboard and 'plantBomb' method of GameManager is called by the GamePanel. This method gets the location of player, hero, and GameEngine creates a bomb in that location. Then GameEngine triggers the method 'addObject' method of GameMap class for that specific bomb. If that process is successful then GameEngine checks if bomb is controllable or not. In our scenario it starts timer since planted bomb is an uncontrollable one. After timer reaches the zero GameEngine triggers the 'explodeBomb' method of the GameMap. GameMap checks all of the possible collisions in the range of the bomb and CollisionManager returns all of the objects that should be exploded to the GameMap. After taking the objects from CollisionManager it calls 'removeObjects' method which destroys all objects in the list.

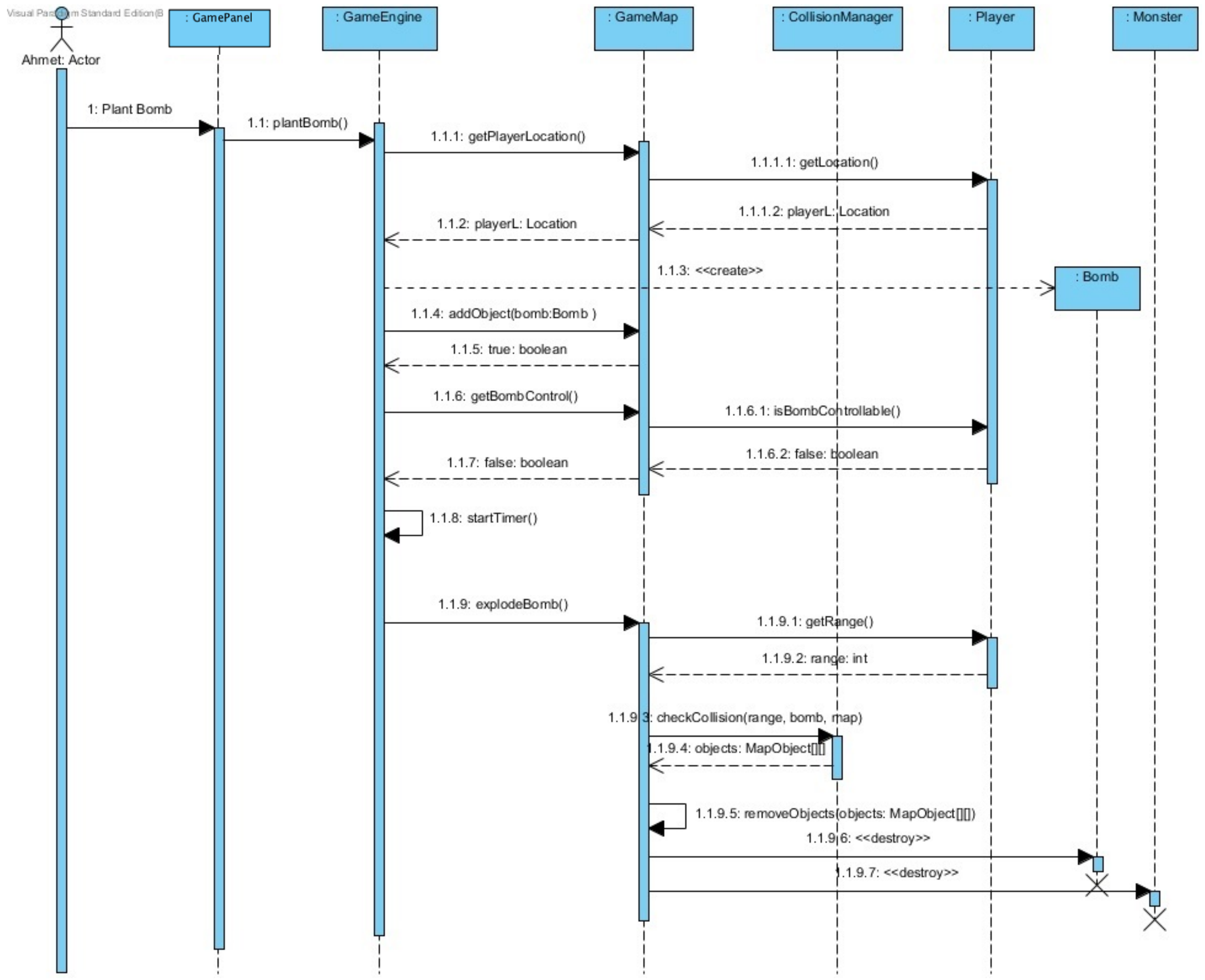


Figure 12: Sequence diagram of bomb explosion scenario

Diagram 3: Change Settings

This diagram in Figure 13 explains a scenario which the player mutes the music of the game from the settings. Actor stops the game by pressing a specific button from the keyboard and ‘stopGame’ method of GameEngine is called by GamePanel. GameEngine triggers 'openPauseMenu' method of MenuPanel. MenuPanel creates PauseMenu. Actor chooses the "Options" from the PauseMenu and GamePanel triggers the 'options' method of GameEngine. GameEngine triggers 'options' method of MenuPanel. Actor chooses "Mute/Unmute" from PauseMenu and GamePanel triggers 'changeSound' method of GameEngine. GameEngine triggers 'checkSound' method of SoundManager. In our scenario the SoundManager returns 'true' since the music is playing. GameEngine triggers 'stopMusic' method of SoundManager. Actor chooses "Back" from PauseMenu and GamePanel triggers 'back' method of GameEngine. GameEngine triggers 'back' method of MenuPanel. Actor chooses "Back" from PauseMenu and GamePanel triggers 'back' method of GameEngine. GameEngine triggers 'back' method of MenuPanel. MenuPanel destroys PauseMenu.

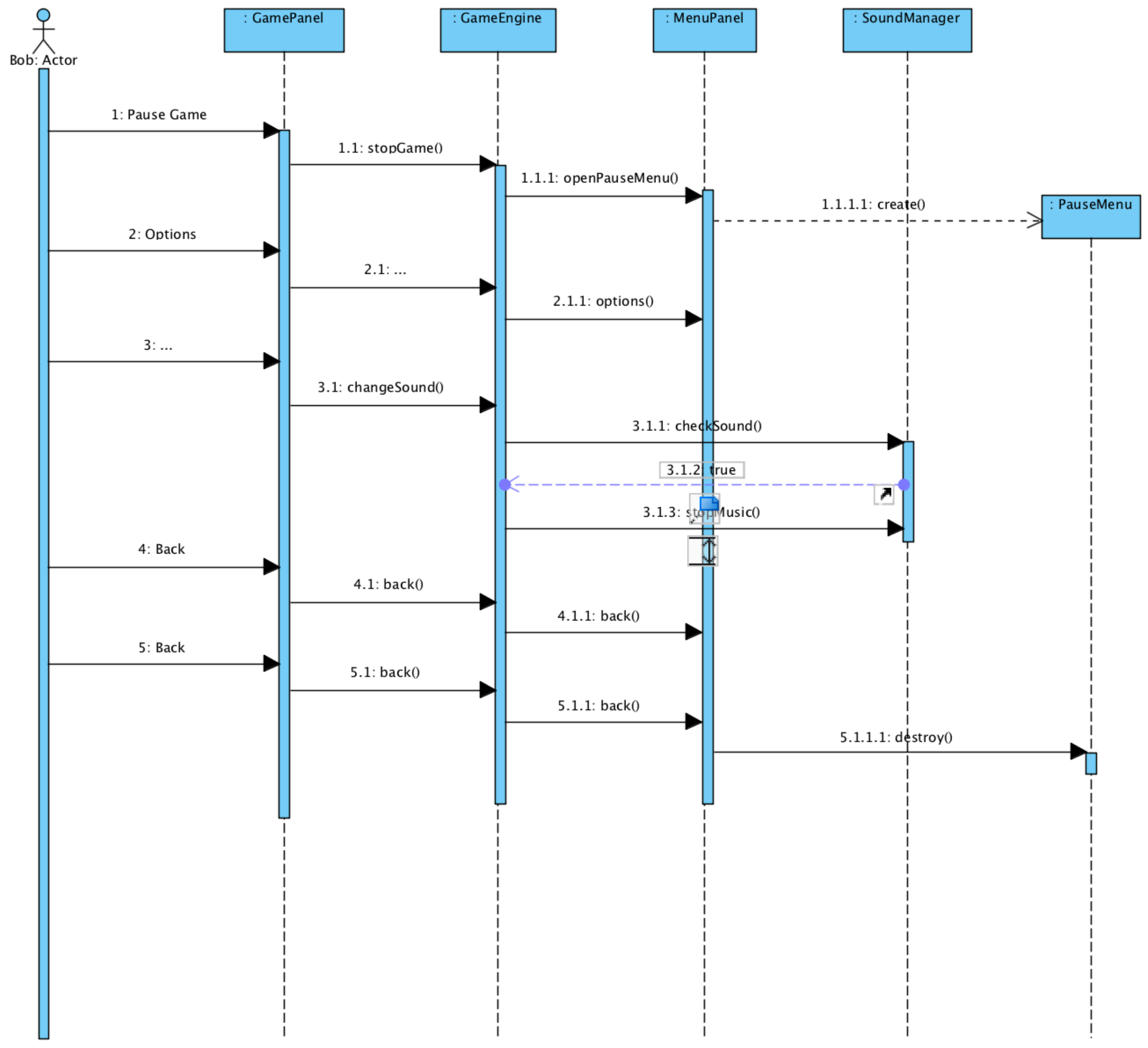


Figure 13: Sequence diagram of change settings scenario

Diagram 4: Take Bonus

Bob puts the bomb in order to destroy the wall. After he puts the bomb, it explodes and the bonus occurs. He takes the bonus and gain ability to put multiple bombs.

Diagram shows that player press the space button to put the bomb. GamePanel receives the key action and delivered to GameEngine. GameEngine executes the plantBomb() method. After the method executed, GameMap creates a Bomb object. GameMap also trigger putBomb() method of Player. Player returns necessary values to check collision in GameMap. CollisionManager checks the collision and confirm the collision than GameMap triggered explodeBomb() and destroy the Bomb object. Player presses the right button and Player takes the bonus.

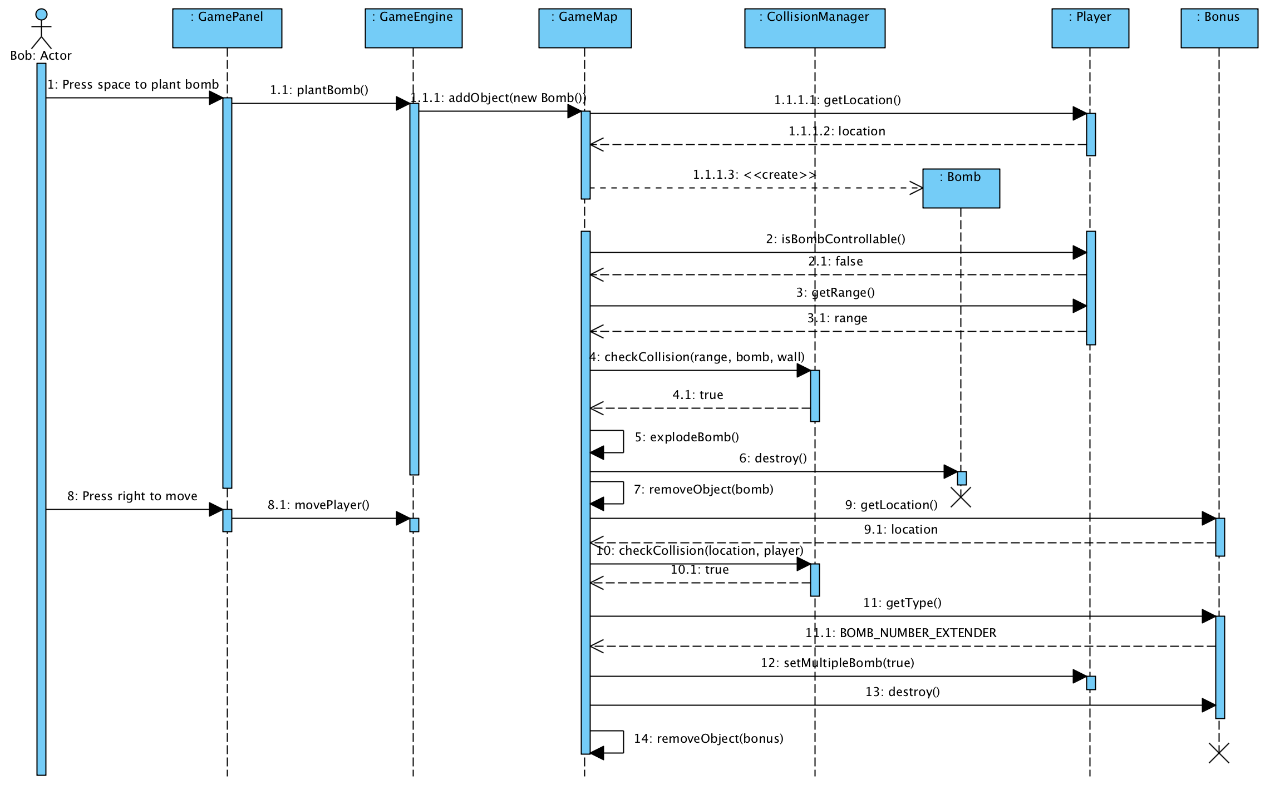


Figure 14: Sequence diagram of take bonus scenario

5.5 User Interface

In this part, user interfaces of the Bombplan game will be examined.

Main Menu

When player starts the game Main Menu shows up as the first screen of the game. Main Menu has Start Game, Help, Load Game, Options, HighScores, Credits and Exit options. Player can move any of these panels by pressing related button.

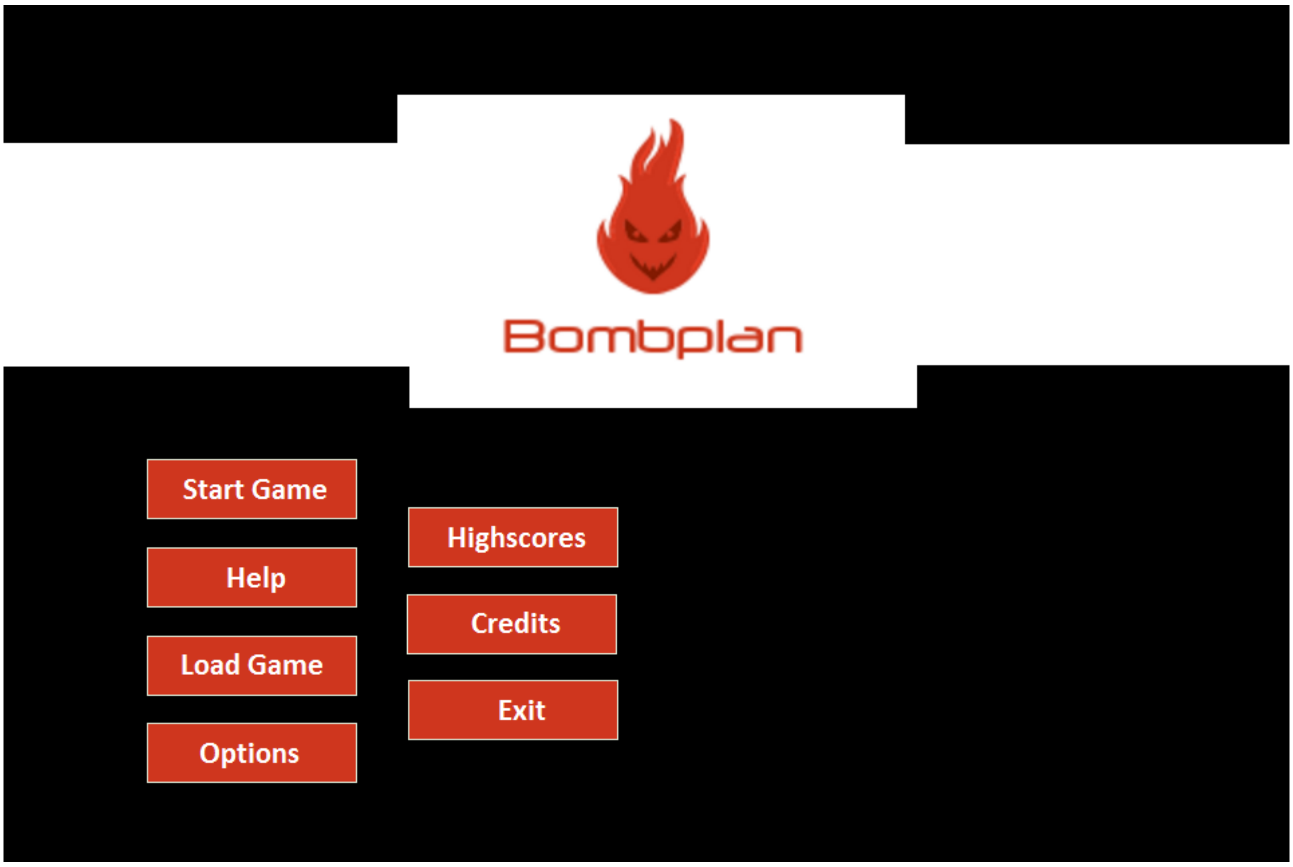
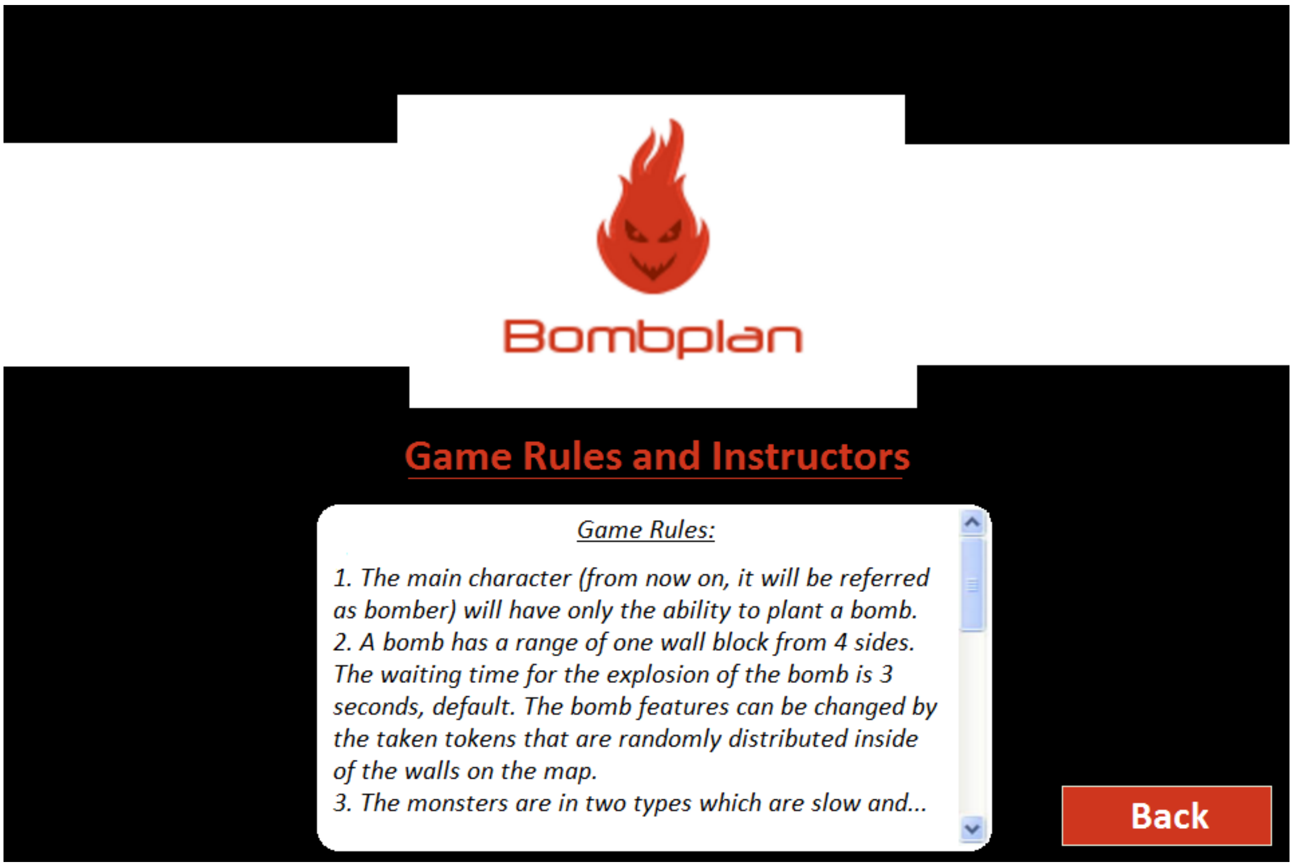


Figure 15: Main menu panel

Help

If player selects the Help option from the Main Menu, screen that has game rules and instructions will be shown. In the Help panel, player shall be able to read the game rules and instructions. In addition to that player can learn keyboard controllers. After player reads the information that he/she needs it can return to the Main Menu.



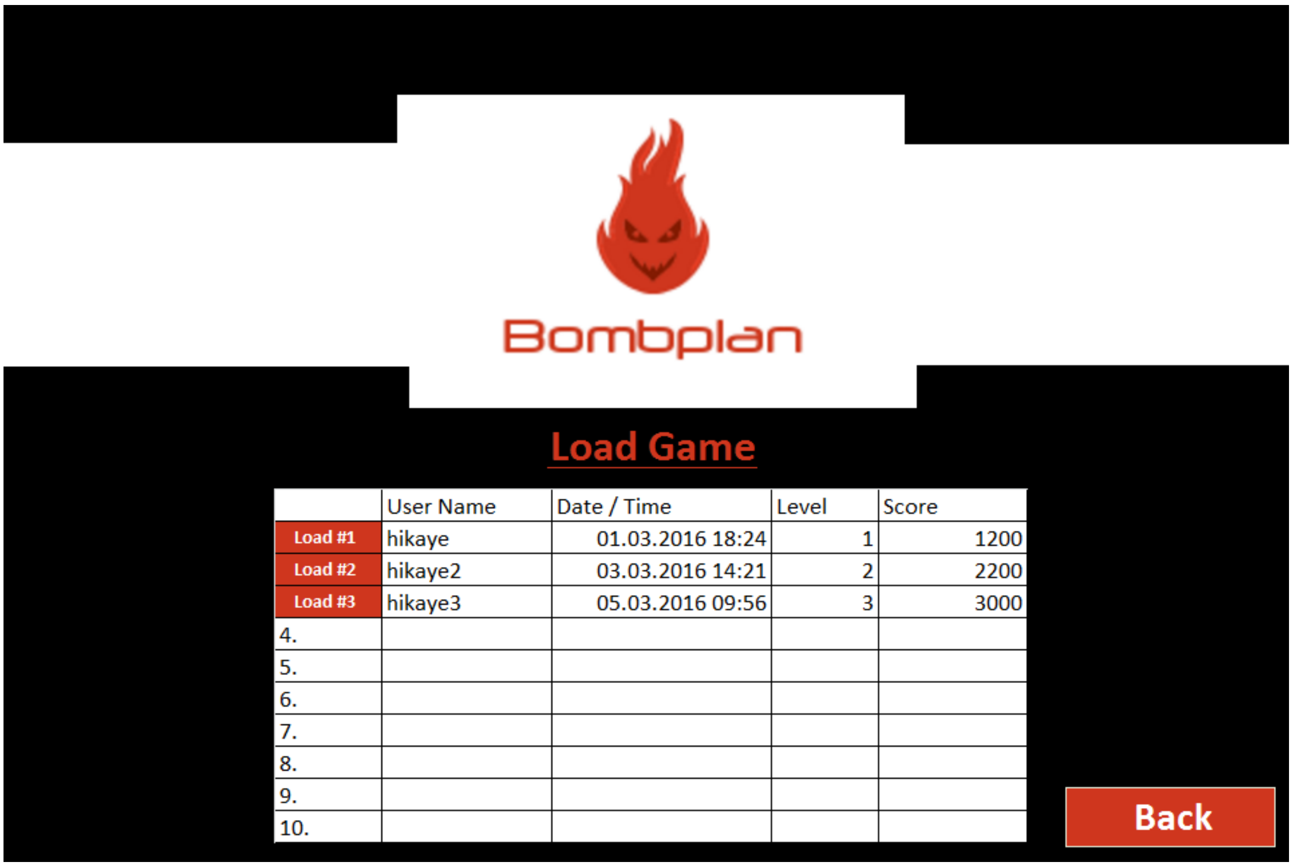
Game Rules:

1. The main character (from now on, it will be referred as bomber) will have only the ability to plant a bomb.
2. A bomb has a range of one wall block from 4 sides. The waiting time for the explosion of the bomb is 3 seconds, default. The bomb features can be changed by the taken tokens that are randomly distributed inside of the walls on the map.
3. The monsters are in two types which are slow and...

Figure 16: Help panel

Load Game

Player can load a game from this screen. If there is any saved game, these saves are going to be list in a table. If there is no saved game, then table will be empty. Saved games are stored with their user name, date, level and score information and this information is available in the table for player to find and continue a saved game easily. Player can return to the Main Menu by pressing Back button.



	User Name	Date / Time	Level	Score
Load #1	hikaye	01.03.2016 18:24	1	1200
Load #2	hikaye2	03.03.2016 14:21	2	2200
Load #3	hikaye3	05.03.2016 09:56	3	3000
4.				
5.				
6.				
7.				
8.				
9.				
10.				

Figure 17: Load game panel

Options

There is ‘Options’ choice on the Main Menu, if player clicks it this panel will be shown. Options panel will be a mini configuration management screen of the Bombplan game. Player shall be able to change the volume of the sound effects and the music of the game via this panel. When player finishes his/her changes, he or she can return to the Main Menu and changes of the settings are going to be saved by the system.

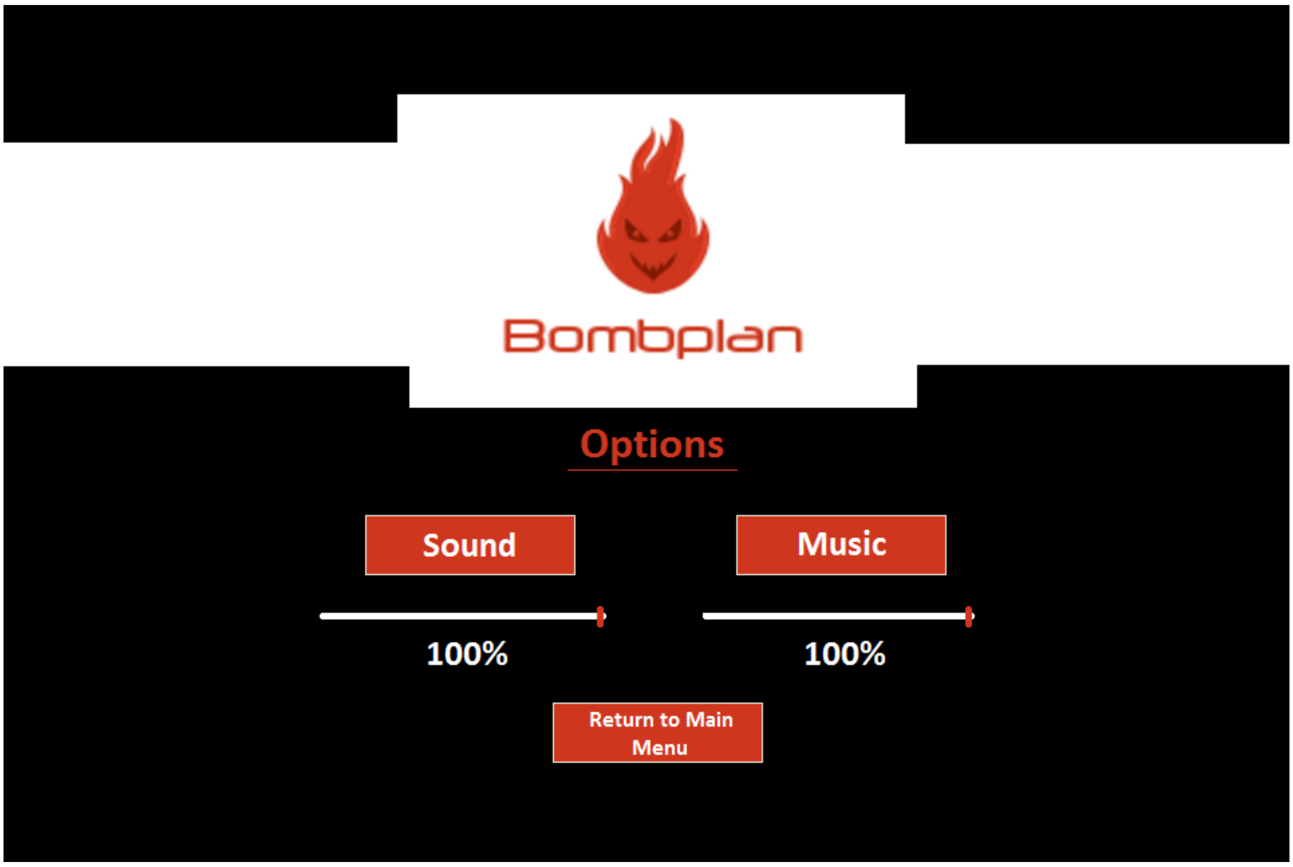
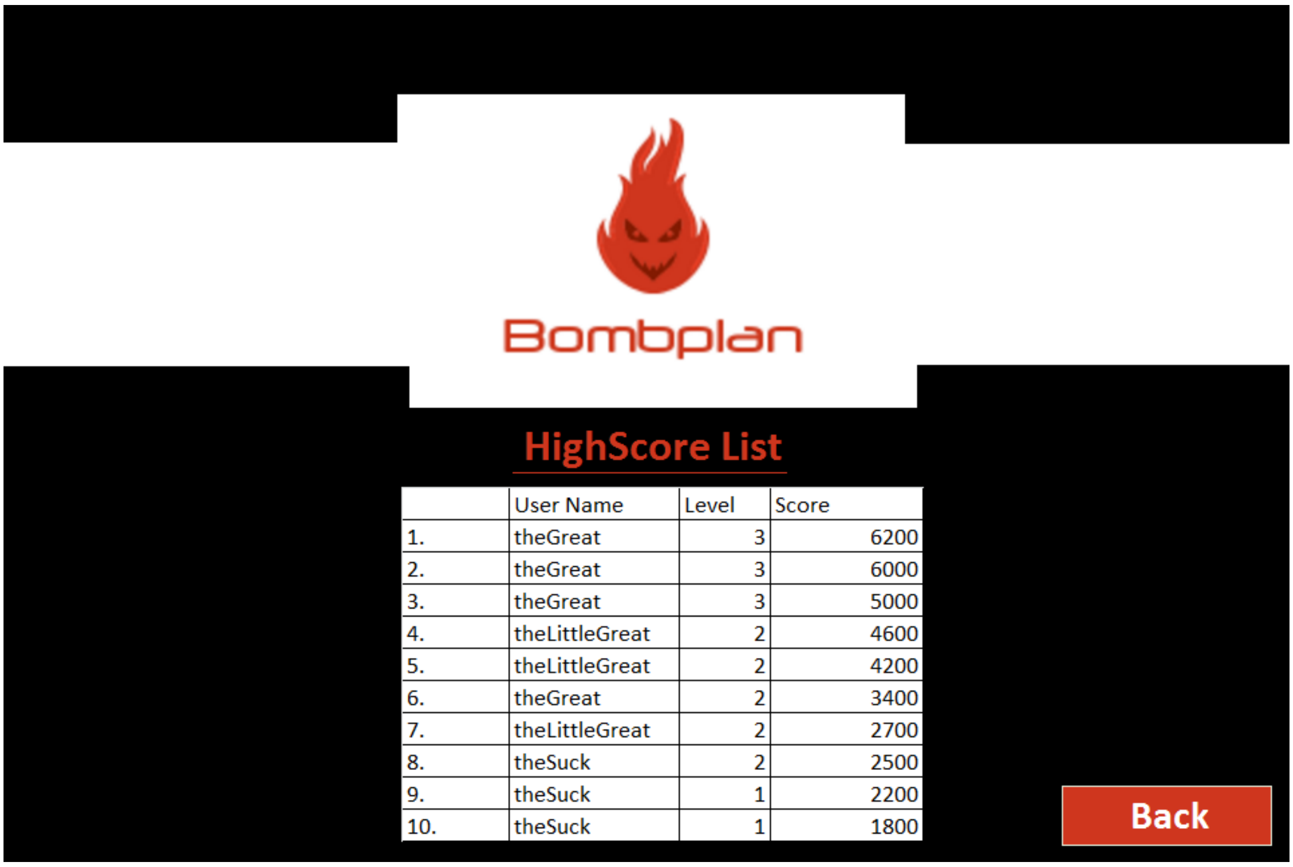


Figure 18: Options panel

HighScores

There will be 'HighScores' option on the Main Menu. When player chooses this option on the Main Menu, high score list panel is going to show up accordingly. On the HighScores panel there will be a list of top ten scores. User name and level information is available for each line of score. Player can returns to the Main Menu after he or she views the high score list. If there is no recorded score, high score list table is going to be empty in that panel.



	User Name	Level	Score
1.	theGreat	3	6200
2.	theGreat	3	6000
3.	theGreat	3	5000
4.	theLittleGreat	2	4600
5.	theLittleGreat	2	4200
6.	theGreat	2	3400
7.	theLittleGreat	2	2700
8.	theSuck	2	2500
9.	theSuck	1	2200
10.	theSuck	1	1800

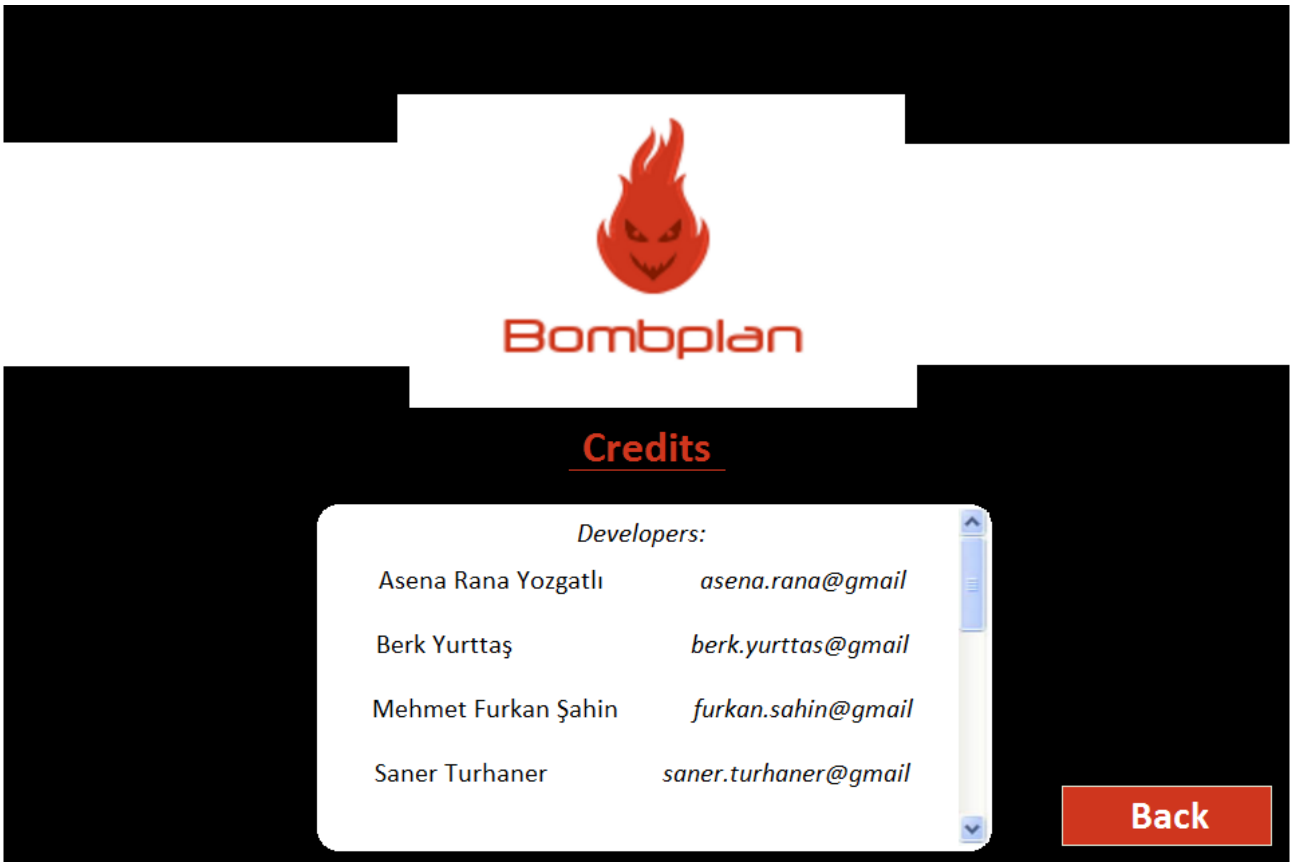
Figure 19: HighScores list panel

Credits

There is also Credits option on the Main Menu. Player can see the credits information of the Bombplan game using this option. In this panel there will be a text box which holds necessary information related credits and also a back button for player to return to the Main Menu. Credits panel will be as in the Figure 18:

Game Panel

On the Game panel there will be a timer on the left up corner of the screen. And number of the lives of the hero will be displayed on the right up corner of the screen. Game screen of the Bombplan will be as in the Figure 19:



Developers:

Asena Rana Yozgatlı	*asena.rana@gmail*
Berk Yurttaş	*berk.yurttas@gmail*
Mehmet Furkan Şahin	*furkan.sahin@gmail*
Saner Turhaner	*saner.turhaner@gmail*

Figure 20: Credits panel

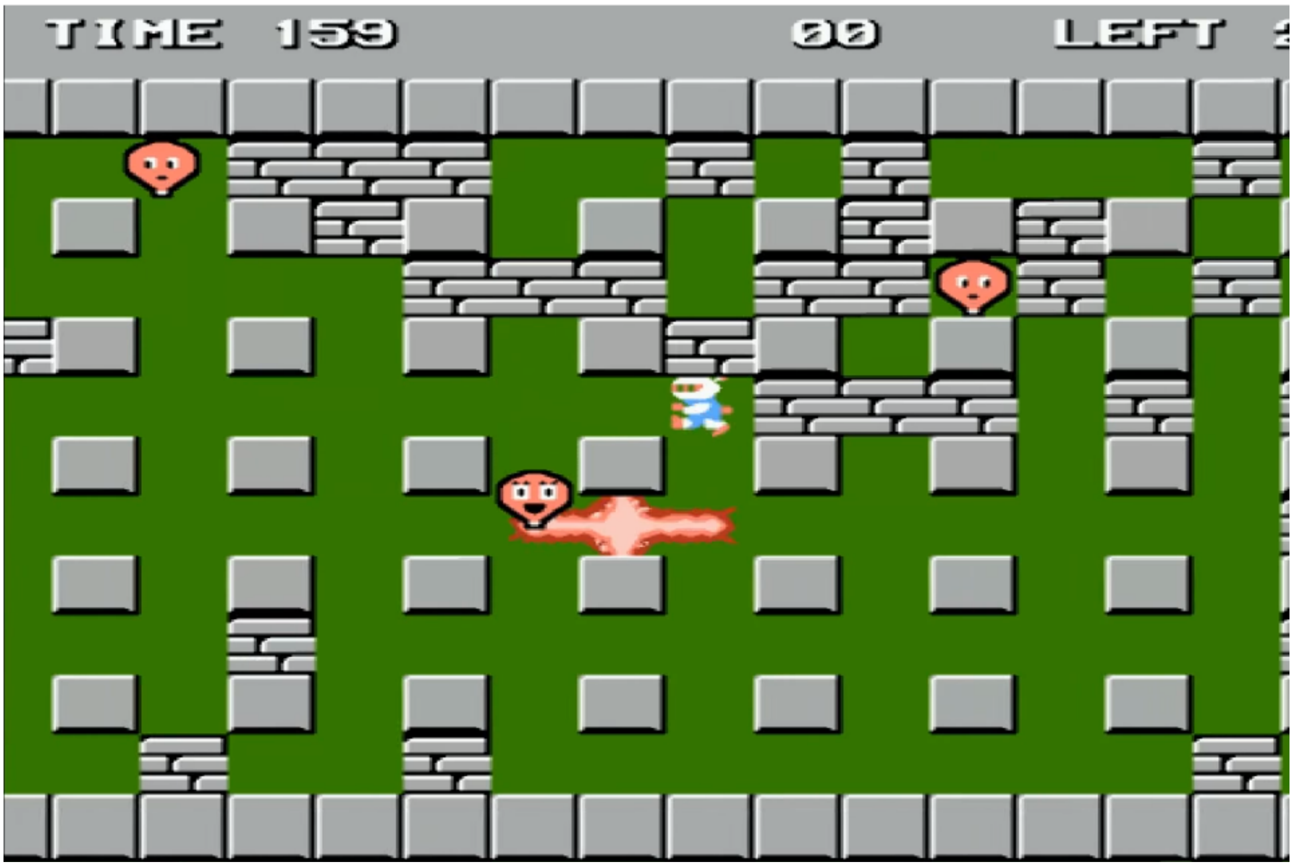


Figure 21: Game panel [2]

References

[1] Game hero, bomb, fire, monster types, and wall types images, accessed on 02 March 2016, http://eski.nintendocu.com/en-iyiler.php?subaction=showfull&id=1234094933&archive=&start_from=&ucat=20&

[2] Game panel, accessed on 03 March 2016, https://www.youtube.com/watch?v=EsIyPoKh0LM