Bilkent University
CS 319
Object-Oriented Software Engineering
Fall 2008

JAVA BOMBERMAN PROJECT

ANALYSIS REPORT
October 31st, 2008

GROUP #4 Members
Turgut İşık
Fakih Karademir
Burak Karakayalı
Uğur Kaya
Tevfik Eren Kuzucu
1. OVERVIEW

We are the members of Group #4 in CS 319 course. We are assigned to design and implement an object-oriented programming project. We have decided to work on a Bomberman Game Project with Java. There is a general definition of and brief information about Bomberman in “Introduction” part of this report. After short description of the game, there is an explanation of why Group #4 has chosen to work on Bomberman Game Project. In this report, requirements of the project are coming under two headings: Functional Requirements and Non-functional Requirements. System models of the game will be described in detail. System models section includes domain analysis, use-case analysis, sequence diagrams, state and activity diagrams. After System models section, detailed description of Bomberman game features will be deliberated. Then the report is concluded with evaluation of Game Project Analysis Period.

2. INTRODUCTION

2.1. General Information about Bomberman

Bomberman is a kind of arcade game. It is played in a labyrinth-based game area. Bomberman was a video game when it was developed by Hudson Soft in December of 1985. Hudson Soft is a Japanese electronic entertainment publisher. The hero of the game, Bomberman is a kind of robot. Bomberman starts the game in a maze. In this maze-based place Bomberman tries to find out a door. With the assistance of this important passing-door, Bomberman will be able to rescue himself from his enemies (monsters) and it will pass to another level, maybe relatively a more difficult level, then it will try to survive again in that new environment.

The game environment consists of free areas which make up ways for Bomberman and also for the enemies of Bomberman. There is a number of enemies, some of them are similar to each other. The powers and speeds of these enemy characters against Bomberman vary from stage to stage. The enemies are in groups, some of them are more effective against Bomberman. Bomberman starts each level in the upper left corner of every stage. The enemies are trying to touch Bomberman in order to kill him or eliminate him during the game. There are some structures or rocks that cannot be detonated by the bombs of Bomberman. There are some exceptional monsters that can pass through these structures in specific periods or stages of the game.

Bomberman starts the game with short-range bombs. He can explode some of the square-shaped rocks in the game area and after explosion he can find the door that will pass him to next level. He can also find out some bonuses such as range increasing bombs, extra bombs, time-adjustable bomb, speed-increasing power-ups etc. Bomberman must be careful when he is putting the bombs in some places. He is trying to kill the enemies. But the bombs have a time limit before explosion. So after
some plan, the bombs can be settled to strategic points in order to kill the enemies right away and find out the possible bonuses, weapons, and of course the exit door. The overall stages have a time limit, meaning that Bomberman must be as fast as possible to kill the enemies and to find out necessary bonuses and the exit door. If all enemies are not killed, it is not possible to pass the next stage from the door for Bomberman.

Bomberman should play carefully since he may explode himself carelessly and explode the door or the possible bonuses. Explosion of the bonuses or the exit door will result in a very bad situation. Then, there will appear a number of horrible enemies which are coming onto Bomberman so fast. At this time it is very difficult to survive for Bomberman. What about the points? Bomberman will get points according to his actions, such as killing enemies, getting bonuses, getting over levels etc. The number of lives that Bomberman has at the beginning of the game will decrease if it is killed at any level of the game. However, he can increase the number of the lives that he has by passing over the levels of the game. The last but not the least point that is to be emphasized, the enemies have no ability of having or exploding bombs. This can be perceived as an advantage for Bomberman.

2.1.1. Power-ups

The Bomberman games usually include many power-ups that are found beneath destructible blocks or rocks. Power-ups are important factors of the Bomberman games. Here are most common generic power-up styles. We do not exactly use this power ups in our project necessarily. We added some extra different power ups.

- **Roller-skates**: Gives extra speed to the player.
- **Bomb**: This allows Bomberman drop an extra bomb.
- **Triple bomb**: This lets Bomberman drop all his bombs at once by hitting the assigned button.
- **Flame**: Increases the size of Bomberman’s bomb’s explosion range.
- **Kick**: This lets Bomberman kick bombs away by walking into them.
- **Boxing Glove**: This lets Bomberman punch bombs away by hitting the assigned button from keyboard when standing next to them.

2.2. Why did we Choose Bomberman?

As a team, the main purpose of choosing Bomberman Game Project with Java is to develop an arcade-style game using the principles of general programming and Object-Oriented software techniques. By using the software techniques and Object-Oriented approach we can be able to design a simple, understandable and object-oriented game.

Bomberman is a simple game that can be easily understood. The elements that form out the characters, stages and levels of the game, power-ups of the game and time factors of the game can be carefully analyzed and simplified by just an Object-
Oriented view. In this CS 319 course, we should try to get a deep insight of the principles of the Object-Oriented Programming, UML Techniques and Software Engineering. In order to perform these acquisitions from this course, we thought that Bomberman is an appropriate game to work on.

3. REQUIREMENTS

3.1. Functional Requirements

3.1.1. Help

Help option provides the necessary information about the Bomberman game. Users can find the rules of the game. The keys and their assigned functions in the game are expressed clearly. The properties of the important parts of the game, the features of the levels can be accessed here.

3.1.2. Start

This function is required and the most basic feature of the game for the players is starting the game. The user selects the start game button and Bomberman tries to survive out of the bad monsters according to the commands from the user with specified actions.

3.1.3. New Game

This is a very basic option for computer games. The players of this game should be able to start a game from very beginning. The default settings will be applied and the game will take start from the very first level.

3.1.4. Pausing the Game

The players are given the chance of pause the game at any time. By this way, players will be able to do their other works and if they wish, they may continue the game from the level that the game was interrupted. The score and other necessary data will be recorded at the time of interruption in order to prepare a sustainable play stage to the user.

3.1.5. Continue Game

Continue Game option is a basic requirement of the game since the game can be interrupted by the player at any time after the game took start. This feature can be used if and only if there was a paused game beforehand. Then the user will be able to continue to his/her game from the level that he was playing at.
3.1.6. Level Selection

Level selection part is another requirement of this game. All players do not have the same playing abilities. Thus advanced users should be able to set the level of the game according to their experiences. If the user enters New Game, the game automatically starts from the first level. Level Selection option gives the user the chance of playing the any arbitrary difficult levels of the game.

3.1.7. Settings

Settings option is the part of the main menu where players can change the external effects of the game manually. For instance, a user may want to play under no sound, and then he/she changes the game mod to silent. Levels, sound effects and types of the game sounds can be set here. The color adjustments of different parts of the game (character views, stage views, tool views, etc.) can be made. Animation arrangements should be done at this section. Game view options can be specified.

3.1.8. Hall of Fame

Most computer games have a part that shows the highest points recorded at that game. And this section is a requirement of our game project since players may want to compare themselves to other players. Highest points and the record holders are registered to the Hall of Fame window. By this way, the quality of the competition and the real rivalry may be achieved by many players.

3.1.9. Credits

General information about the designers of the game should be included to the requirements since users may want to reach related documents about the game and project. Links to the project website and group will facilitate the communication between the game makers and game players. Requests and suggestions may be done by the users. Credits section is the official part of the program where owners of the game claim copyright.

3.2. Non-functional Requirements

3.2.1. Performance

Bomberman game is played by just one player. That's why the player will not wait for another player's game to be over. Some animations in the game such as the movements of the Bomberman and its enemies are important for visual appearance. The design, programming, and testing phases of the project will better tell us about the performance of the game under specific conditions. It is a requirement that game should be played comfortably when executed on a computer system.
3.2.2. Easy to Learn, Hard to be Master

    The graphical user interface (GUI) is not too complicated therefore the players of the game will be enabled about getting familiar with the game view. The game should be simple to execute and play. Users should be able to readily set up the game and execute it. It should not have too bad effects on checkout system of the computer. Also we will try to make the game as easy to learn as possible and it will be require some time to master the game. This motto is statistically proven for a game’s ultimate success. There are tons of beautifully crafted games but they never become a hit because of their hard learning curve. On the other hand original Bomberman’s success rests in this principle.

4. SYSTEM MODELS

4.1. Domain Analysis

    Bomberman game consists of a timer component, a well-designed and ordered game map, the hero Bomberman, the monsters that try to wipe out Bomberman and specific Power-up components which are given to Bomberman as prize. Power-ups may be considered as a part of Game map. Game gives Bomberman a certain time for killing the monsters and passing the levels. If Bomberman can’t finish the game in the given countdown time, the game is over.

    On the game map, the hero of the game, Bomberman, has a specified location which is defined by analytical coordinates. Bomberman has also a certain number of lives. The number of lives is a default value at the very beginning of the game. However, it can increase or decrease according to the actions, successes or failures of Bomberman changes the number of lives such that if Bomberman passes levels or gets important bonuses his lives increment. Bomberman may lose his lives if it is caught by any monster or if he bombs himself. And Bomberman has a score which is computed by game system. The score increases when Bomberman kills monsters and gets bonuses. When the score is getting higher, it is possible to be listed in the Hall of Fame section by that player. Bomberman gets at least 0 (zero) bonuses. There are many monsters that try to kill out and eliminate Bomberman. One or more creature may kill Bomberman at a time. So we can say that there is one-to-many relationship between Bomberman and monsters. Similarly, Bomberman kills zero or more monsters in the game.
As it is shown in Domain Analysis figure above, Bomberman and monsters are acting on the stage which is game map. The enemies of Bomberman, or the monsters, are also defined with some properties. Monsters are able to act according to their analytical coordinates on the game map. And we have different types of monsters. That is to say, monsters may have different physical appearances such as desperate faces, different colors, and various sizes. Just like Bomberman, monsters have a number of lives which provide the chance of maintaining the game for them. Most of them have only one live; meaning they are out of game if they are killed by the bombs of Bomberman. However, it is possible that Bomberman faces more powerful monsters that have more than one live chance. The speeds and acting mechanisms of monsters are also different. Some of them are faster and specifically tries to destroy Bomberman in the game map. In this game, one hero is playing in the game map. There is a one-to-one correspondence between Bomberman and Game map.

Game map is a very important component of this game. Almost all actions and interactions happen on this game maze. There are walls in the game map. These walls are hiding bonuses and the exit door. Bomberman tries to explode these walls and find out bonuses which will make him more powerful to struggle with monsters.
One of the walls is hiding the main exit door. By this exit, Bomberman will be able to pass the stage if there are no more monsters to kill out.

Game map includes transitions which make up possible directions and ways for Bomberman and the monsters. There are some bonuses hidden randomly in the game map. We call these bonuses as “power-ups” s. Power-ups have different locations meaning that Bomberman can’t find two power-ups in the same place of the Game map. These power-ups have different contributions to Bomberman. Some of them make him faster; some of them give Bomberman a break of having more bombs. Some power-ups make him stronger etc.

On the Game map, at any particular time, one or more monsters are trying to kill Bomberman. Therefore, there is a one-to-many relationship between the game map and the monsters. There is also a one-to-many relationship between the game map and the power-ups.

4.2. Use Case Analysis

![Figure 2: Main Use Case Diagram](image)
Main use case diagram of Bomberman game system is above. Through this system, the game players will be able to play the game, access the help documents, see the high scores, change game options, save game and load game. These 6 use cases will be analyzed and detailed in activity and state diagrams later. Playing game will start a new game session from the very beginning. Users will be able to get help about most important issues and instructions of the game especially when they got stuck. Change options part will allow the players to make changes in the general game settings such as sound and effects. Each use case is to be explained in detail in this report. Here is the use case analysis.

**Use Case UC1: Play Game**

**Scope:** The Bomberman  
**Level:** User Goal  
**Primary Actor:** Player  
**Stakeholders and Interests:**  
- Game: Wants to supply a nice and proper bomberman game to user with no errors.  
- Player: Wants to play and spend enjoyable time with the game “The Bomberman” without errors and inconvenience.  
- Hall Of Fame: Wants to hold a list of ten highest score that is made in “The Bomberman” game.  
- Save Game: Wants to save the current game session in order to provide the same scene of the game any time in future.  
**Preconditions:** Player has opened the game “The Bomberman”, s/he has selected “New Game” option from the main menu of “The Bomberman”.  
**Success Guarantee:** Player has played the game “The Bomberman”. If the score s/he has got from the game is high enough to enter “The Hall Of Fame”, which is a top 10 score list of the game, the score is recorded to “The Hall Of Fame”. Next, player is returned to main menu.  
**Main Success Scenario:**  
1. Player selects “New Game” option from the main menu of “The Bomberman”.  
2. A new game is started from first level.  
3. Player passes all levels of the game successfully.  
4. The score s/he has made is compared to the scores in Hall of Fame.  
   4a. His/her score is high enough to enter Hall of Fame  
      1. Player is informed that s/he has a high score.  
   4b. His/her score is not high enough to enter Hall of Fame.  
      1. Player is returned to main menu.  
**Extensions:**  
*a. At any time, player is out of lives.*  
1. A warning that indicating “game over” is showed up.  
2. The score of the player is compared with the scores in hall of fame.  
   2a. His/Her score is high enough to enter Hall of Fame  
      1. Player is informed that s/he has a high score.  
   2b. His/her score is not high enough to enter Hall of Fame.  
      1. A window that asks for a new game is popped out.
2-1a. If player selects “Play Again”, a new game is started.
2-1b. If player selects “Back to Main Menu”, s/he is returned to

*b. At any time, player wants to exit the game.
1. Player presses “ESC” button, a menu shows up and game is paused.
2. Player selects “Quit” button from the menu.
3. A confirmation window is popped out to confirm whether player is sure or not.
   3a. Player selects “Yes” from confirmation.
       1. Game ends and player is returned to main menu.
   3b. Player selects “No” from confirmation.
       2. Game continues from where it is paused.

Use Case UC2: See Hall of Fame

Scope: The Bomberman
Level: Subfunction
Primary Actor: Player
Stakeholders and Interests:
-Game: Wants to supply a nice and proper bomberman game to user with no errors.
-Player: Wants to see the 10 highest score and compare him/herself with them.
-Hall Of Fame: Wants to hold a list of ten highest score that is made in “The Bomberman” game.
Preconditions: Player has opened the game “The Bomberman” and the main menu is showed up.
Success Guarantee: Player saw the list of highest scores of the game “The Bomberman”. Next, player returned to main menu of the game.
Main Success Scenario:
1. Player selects “Hall of Fame” menu from the main menu.
2. In “Hall of Fame” menu, a list of 10 highest score is listed..
3. Player observes the list and compares him/herself with top scores. Next, s/he clicks the “Back to Main Menu” in order to return to main menu.
Extensions:
*a. At any time, player wants to clear the “Hall of Fame” list.
   1. Player clicks to “Clear” button.
   2. A confirmation is popped out to ask the player whether s/he is sure or not.
      2a. Player selects “Yes” from confirmation.
          1. List is cleared
      2b. Player selects “No” from confirmation.
          2. Confirmation is vanished.
   3. Player is back to “Hall of Fame”.
*b. At any time, “Hall of Fame” list warns the user if there is no list of scores because there is no completed game or list is cleared.
   1. A warning is popped out to inform that currently there is no list of top scores.
   2. Player returns to main menu.
Use Case UC3: See How to Play

Scope: The Bomberman
Level: Subfunction
Primary Actor: Player
Stakeholders and Interests:
-Game: Wants to supply a nice and proper bomberman game to user with no errors.
-Player: Wants to learn how to play the game. Wants to grasp basic information about controls and rules.
-How To Play: Wants to provide information about the controls and instructions of the game.
Preconditions: Player has opened the game “Bomberman” and main menu is showed up.
Success Guarantee: Player saw the instructions to play the game “The Bomberman”. Next, player returned to main menu of the game.

Main Success Scenario:
1. Player selects “How to Play” menu from the main menu.
2. In “How to Play” menu, controls and instructions of the game is introduced.
3. Player reads and comprehends the instructions and rules, s/he clicks the “Back to Main Menu” in order to return to main menu.

Use Case UC4: Change Options

Scope: The Bomberman
Level: Subfunction
Primary Actor: Player
Stakeholders and Interests:
-Game: Wants to supply a nice and proper bomberman game to user with no errors.
-Options: Wants to see or modify the settings of the game.
-Player: Wants to see or modify the settings of the game.
Preconditions: Player has opened the game “Bomberman” and main menu is showed up.
Success Guarantee: Options set the game according to requests of the player. After that, player returned to main menu of the game.

Main Success Scenario:
1. Player enters to “Options” menu from the main menu.
2. A menu containing all options and settings for the game is showed up.
3. Player arranges the settings and options according to his/her choice.
4. Player clicks to “Back to Main Menu” button and a confirmation is popped out.
   4a. Player selects “Cancel”.
      1. Confirmation window vanishes and player stays in Options menu.
   4b. Player selects “Save and Back to Main Menu”.
      1. All changes in settings is saved.
      2. Player is returned back to main menu.
   4c. Player selects “Back to Main Menu Without Saving”.
      1. Player is returned back to main menu.
Use Case UC5: Save Game

Scope: The Bomberman
Level: Subfunction
Primary Actor: Player
Stakeholders and Interests:
- Game: Wants to supply a nice and proper bomberman game to user with no errors.
- Player: Wants to stop and freeze the game s/he is playing in order to play that scene any time in future.
- Save Game: Wants to save the current playing session in order to provide that scene to player any time in future.
Preconditions: Player has opened the game “The Bomberman”, s/he has selected “New Game” option from the main menu of “The Bomberman”.
Success Guarantee: The current playing session is saved. Player returned to main menu of the game.

Main Success Scenario:
1. During the playing session, at any level of the game, player presses “ESC” button.
2. The game pauses and a window shows up with 4 options.
   2a. Player selects “Cancel” button.
       1. The window vanishes.
       2. The game continues from the moment it is paused.
   2b. Player selects “Save” button.
       1. Game is saved with the tag with date and time.
       2. The game continues from the moment it is paused.
   2c. Player selects “Save&Quit” button.
       1. Game is saved with the tag with date and time.
       2. The game ends and player returns back to main menu.
   2d. Player selects “Quit” button.
       1. The game ends and player returns back to main menu.

Use Case UC6: Load Game

Scope: The Bomberman
Level: Subfunction
Primary Actor: Player
Stakeholders and Interests:
- Game: Wants to supply a nice and proper bomberman game to user with no errors.
- Player: Wants to play the game s/he has saved before, properly and without inconvenience.
- Load Game: Wants to load the game the player has saved before.
Preconditions: Player has opened the game “Bomberman” and main menu is showed up.
Success Guarantee: The saved game that user wants to play is loaded, with the same scene and settings as the moment it was saved.

Main Success Scenario:
1. Player selects “Load Game” option from the main menu of “The Bomberman”.
2. A list of saved games show up.
3. Player selects the desired one.
4. The game loads the saved game that user wants.

**Extensions:**

*a.* At any time, the load game menu might be empty because of unavailability of any saved game.
   1. Player is warned with a message that currently there is no saved game.
   2. Player returns back to main menu.

*b.* At any time, player might want to erase a saved game.
   1. Player selects the “Delete a Saved Game” button.
   2. Another list of saved games is popped out in a different list format.
   3. Player selects the desired one.
   4. Game pops out a confirmation box to player in order to confirm deletion.
      4a. Player selects “Yes” from the confirmation.
          1. Saved game is deleted.
      4b. Player selects “No from the confirmation.
   3. S/he returns back to “Load Game” menu.

**4.3. Sequence Diagrams**

**Sequence Diagram 1:**

Fakih executes the program, he sees the opening window of the game, then he chooses the help document to read. He gets information about how to play the game (i.e. the instructions) in the game maze.
In this sequence diagram, player executes the Bomberman game. Then he requests to see the help document related to the instructions of the game. GUI Manager contains the main window of the game. It also contains the other important windows and their components visually.

GUI Manager is in interaction with the Help Window and its contents. The player sends a request for opening the help window. GUI Manager responds to the player and after opening the help documents window, player chooses the necessary helping documents via GUI Manager. In this example, player wants to see the instructions help page which contains the related information about the playing style and techniques of the game. Help Documents is an important lifeline just as GUI Manager is. The lifeline Help Documents sends the requested documents to GUI Manager, then the player is able to read the help documents about the game instructions of Bomberman.
Sequence Diagram 2:

Eren opens the game, chooses playing game option, starts a new game, moves Bomberman, gets a bonus by moving Bomberman onto the power-up points (bonus points) and state of Bomberman is changed at a particular time.

When Bomberman is playing in the Game Map, he may find power-ups at particular locations. These power-ups have specific types and bonus contributions to Bomberman. Power-ups are valid during a period which is defined as duration of Power-up parameter here. In this duration, power-up is active and after destroying the Timer object that makes power-up valid, Bomberman returns to his initial state.
In this sequence diagram, player executes the game and starts a new Bomberman game by sending requests to GUI Manager which will open the Game Engine where all the important actions of Bomberman game happening in. Game engine is requested to start a new game for the player. Then Game Map is shown in the Game window. (GUI Manager and Game Map are in interaction.)

Player tells Bomberman to move on the Game Map. Here example parameters are assigned to moveBomberman command. Player is trying to move Bomberman to east direction by 8 units on the map. Bomberman has a specified location on Game Map which is defined by analytical integer coordinates. Bomberman also holds this integer within its own data, meaning that we can access the location parameters of a Bomberman object by asking it. There is an interaction between Game Engine and Game Map. Game Engine asks Game Map whether Bomberman can move to the specified location on the map. Game Map responses it is possible to move that specified location defined by direction and integer units. In the sequence diagram above you see one integer and type of direction is given to canMove() method. Type of direction may get 4 possible values. These are north, south, east and west. After getting info about moving to specified location, Game Engine asks Game Map whether there is a power-up or bonus on that direction. A positive respond comes from Game Map that affirms there is a power-up at the respective location (13,8). Then Game Engine requests Game Map in order to get power-up and give it to Bomberman.

Game Map interacts with the Power-up lifeline as it is seen above. Game Map collects information about the type, specifications and duration of the power-up. This values are returned to Game Map in order to make Bomberman take the bonus as specified with these parameters. After taking power-up, Bomberman requests to create a Timer object. This Timer object will determine the time for the changed state of Bomberman. Power-up changes the state of Bomberman. After the message of Timer lifeline to Bomberman that tells the power-up has expired, Bomberman returns to its previous state. Then the Timer object is destroyed upon the request coming from Bomberman which tells the duration of power-up has finished.
**Sequence Diagram 3:**

Turgut opens the game, starts a loaded game, there is only one monster remaining, kills the last monster, then finds the exit door by bombing the last wall in the Game Map, then he proceeds to next round (or level).
In this sequence diagram, player requests move the hero, Bomberman, in the Game Map. Game Engine contains and controls the Game Map. Game Map contains Bomberman, monsters, walls, transitions and power-ups. We may assume that Game Engine is the topmost component in the game hierarchy. Player sends a request that he wants to move Bomberman. Game Engine takes this request and after getting respond from Game Map the player is allowed to move the Bomberman. Game Engine asks Game Map if there is any power-up for Bomberman in the direction that Bomberman moves on. Game Map tells there is no more power-ups. Then Game Engine sends another request whether all monsters are killed or not. Game Map tells that there exists one more monster. Then player moves Bomberman accordingly in order to kill the last monster. Player strategically points a location to set a bomb and kill the last monster. After getting the signal showing the last monster is killed, Bomberman searches for exit door.

Game Engine asks Game Map whether the exit door is found which was hidden under the walls. Game Map tells that the exit door is not still found. Then Game Engine sends instructions to control the walls. Game Map responds as there exists the last wall. Bomberman moves and explodes the last wall. Bomberman asks Game Map whether he can exit from the map. Now he can exit since there is no more bonuses, no more monsters and also the door is found. Game Engine tells Game Map to finish this level. And after creating the next level, Bomberman is able to proceed to next level. Actually this next level is another paradigm of Game Map class.
Sequence Diagram 4:

William starts the game, moves Bomberman. Bomberman is caught by a monster and killed off. There is no more lives of Bomberman. So game is over and a screen displays to inform the player.
Player starts the game. He/she requests to move Bomberman on the transitions or free ways in the Game Map. Game Engine asks Game Map whether it is possible to move Bomberman in such a direction. Game Map answers in a positive manner that will make Bomberman move on it. Then Game Engine checks that if there is power-ups remaining in the Game Map. Then there is no more power-ups. After that Game Engine asks Game Map if there is any monsters at the specified location as a parameter. Game Map tells Game Engine that there exists monster(s) in the specified location. Game Engine controls and moves the monsters. Bomberman and one monster collides on the Game Map. Monster kills Bomberman.

The news of Bomberman has been killed is posted to Game Engine. Game Engine commands to Game Map to cancel one live of Bomberman. Game Map decreases one live of Bomberman out of all lives of it. Then Game Engine asks whether Bomberman has any lives. The answer is that Bomberman has zero lives by that time. Game Engine tells Game Map to destroy itself. After receiving the info that Game Map has been destroyed, Game Engine sends a message to GUI Manager (responsible for GUI based events) that game has finished unsuccessfully. GUI Manager requests Game Engine to destroy itself at that time. GUI Manager requests to open a new window of Game Over Screen. Then GUI Manager makes Game Over window visible. And a kind of Game Over message is shown in this recent window.
Sequence Diagram 5:

Uğur starts Bomberman game; he has passed all levels and completed game successfully. A “Congratulations!” message is displayed in the main window. He records his player name and his score of that session.
In this scenario, the player is assumed to pass all the levels and finish the game successfully and also with a high score. Game Engine confirms that the player has finished and passed all levels in the game. Game Engine tells GUI Manager that game has been finished. GUI Manager asks for the game score of Bomberman to Game Engine. Game Engine requests for the score of Bomberman to Game Map. Game Map gets the integer score value from Bomberman object. Bomberman’s score of that game is returned to Game Map, Game Engine and finally GUI Manager. Then GUI Manager creates a congratulation screen. Uğur sets his player name to High Scores table. Congratulations Window makes a request for adding the new high score and its owner to Hall of Fame section. Then the updated Hall of Fame screen is returned to the user or player. Uğur sees the high scores window and his position in the table and then closes this window.
Sequence Diagram 6:

Fakih enters the game, decreases system sound 2 units. He starts the game. Bomberman kills one monster. However, the game period expires and Game Map sends many monsters onto Bomberman. Then, no lives remains, Game is over.
In this sequence diagram, a complex sequence of events is shown. Actually, these are some of the most frequented actions in Bomberman game. The player wants to change some settings before starting the game. In the main window, player chooses the Change Options button, then a new screen comes. Basically, this happens like that: GUI Manager opens a “Change Options” screen and displays it visually. Then player decreases the system or the game volume by 2 units. Change options screen informs as the game sound is lowered.

Player starts the game. Just as the level starts, Game Engine creates a timer object for measuring the period of that level. And the countdown starts for Bomberman. User tries to move Bomberman in the transitions of the Game Map. Game Engine asks whether Bomberman is able to move to different directions on the Game Map. The positive answer is received by Game Engine. Then, Game Engine sends another request in order to learn about the monsters. There exists monsters. Game Engine takes the state of Bomberman which will affect the situation that Bomberman kills a monster.

Game Engine tells Game Map to explode bombs at a specified location according to the position of the target monster. Game Engine tells Bomberman to set a bomb to the previously specified location by the user. Bomberman kills the monster and the info about the bomb explosion is delivered to Game Engine. Since Game Map contains Monster objects,

Game Map sends a command telling to decrement the lives of that attacked monster. Then, the monster is destroyed. Game Engine orders Game Map to increase the points of Bomberman accordingly. Game Map increases the current score of Bomberman.

Game Engine asks Timer component to learn whether there is enough time. However, the time for that level has just ended and Timer object is destroyed. After that, Game Engine creates many Monster objects to make them attack on Bomberman. Bomberman is killed after this “slaying” operation. Game Engine tells GUI Manager that game has finished since Bomberman is no more living. (Actually Bomberman has lost his last life here.) GUI Manager makes an invocation to destroy the Game Engine. Game Engine requests to open a game over window in the main frame. Game over message is displayed.
4.4. State Diagrams

4.4.1. Game State Diagram

Game is mainly in its default state, which provides playable environment to the player. Essentially player will only be able to interact with the map in this stage. If player manages to kill one of the monsters, game map goes into “Monster Killed” state. In this time, current number of monsters on the level checked if monsters are all cleared or not. When player eventually clear all the monsters current game level ends. On the other hand if character dies or could not be able to clear the stage in given amount of time period, game level again ends.

![Game State Diagram](Visual Paradigm for UML Standard Edition(Bilkent Univ.)

**Figure 3:** Game State Diagram
4.4.2. Character State Diagram

Player’s most general state is “move”. In this state player goes anywhere in the map and be subject to several activities which are take a power up, killed by a monster, killed by a bomb’s explosion or deploy a bomb. After deploying a bomb or getting a power up, player returns to the “move” state. But prior to returning some of the game map’s and player’s properties gets updated accordingly. If player exposes itself to an explosion or gets in contact with monsters, he’s number of lives checked in the “Lives” state. If player runs out of lives, game ends.

![Character State Diagram](Image)

**Figure 4:** Character State Diagram
4.4.3. Monster State Diagram

There are two main difficulty levels for monsters. In “easy” monsters are essentially non intelligent. They just move randomly or move on a pre defined pattern continuously. For beginner players this mode is appropriate. On the other side “hard” monsters have some artificial intelligence as they tend to move towards to the character as long as their environment lets them to. For every monster, activities stars as difficulty checking state. After that according to their difficulty they start looping in either “Move randomly” state or “Move towards the character” state. If one of the monsters go into a effect area of a bomb explosion it goes into “Explosion” state. Game checks for monsters lives and decides if monster has been killed or not. In latter case monsters returns back its difficulty driven loop, otherwise game deletes it from current stage.

Figure 5: Monster State Diagram
4.5. Activity Diagrams

4.5.1. In game activity diagram

In this diagram all the dynamics of a game depicted as an activity diagram. Mainly system is in “game loop” which continuously updates the map and saves the activities on it and listens for the players inputs. Player can either press one of the directional buttons for moving around or presses a bomb deploy button to setup a bomb to the current coordinate on the map.

![In game activity diagram](image)

**Figure 6:** In game activity diagram
There are three parallel processes in this game which are player movements, bomb’s explosions and monster movements. Last process is not in this diagram due to complexity concerns. Monster’s activities are explained in the next diagram. For the bombs when they explode, game checks their explosion range for walls, monsters or players. If there exist a wall, it gets destroyed, if a monster is on the way, its lives decremented or dies, for the last case if player happens to be in the explosion range again he’s treated as monsters, game checks player’s lives and decides if he should continue or drop dead. Additionally for praising the player’s good game play, if player manages to kill three monsters in a row without losing his lives, he is assumed to be in a killing spree which multiplies the score gained afterwards.

On the other hand, movement dynamics goes like this. If player deploys a bomb, its timer starts and waits for its explosion interrupt. Or if player moves around with directional buttons if there is no bomb or wall on the way, “game” lets the player to move that place and series of checks stars. First, game checks the newly arrived location if there is a power up, than if there is a monster on the way. For prior case player yields the power up’s specialty and for the latter case game decrements the player’s lives. All of those processes synchronize before the next iteration of “game loop”.
4.5.2. Monster Activity Diagram

Monsters are one of the parallel processes of the game activity. As mentioned before, they determine their next movement according to their difficulty level. For the first activity, monsters’ difficulty is checked, after that, the next movement is calculated with the corresponding algorithm. Then, monsters proceed to the new location coordinate. Lastly, their activity returns their main movement loop which continues until they are killed by the player.

Figure 7: Monster Activity Diagram
5. DETAILED DESCRIPTION OF THE GAME FEATURES

5.1. Types of Bonuses

There are several kinds of bonuses, and these bonuses will help our hero to finish the level. If he picks up those power ups, he can kill the enemies and pass to the other level easier.

Extra Speed bonus: This bonus will make the hero move faster so that he can escape from the monsters and the effects of the bombs.

Bomb Range bonus: This bonus will make the bombs that the hero has planted explode in larger dimensions. For example, if the bombs explode in 2 squares, after the hero picks up the bomb range bonus power up, the bombs will explode in 3 squares. This bonus may be combined with other bomb range bonuses so that the bombs are effective in a larger area according to the number of bomb range bonuses.

Extra Bomb bonus: This bonus allows the hero to plant more than one bomb on the map at the same time although normally he can plant only one bomb at a certain time. This bonus may be combined with other extra bomb bonuses so that the hero can plant more bombs in a certain time according to the number of extra bomb bonuses.

Remote Controller bonus: Normally, the bombs which the hero plants explode when a predefined time passes. However, thanks to this bonus, the hero can make the bombs explode whenever the player wants.

Bomb Power bonus: This bonus powers the bombs that the hero plants up and these bombs are able to break iron walls when the hero has this power up.

Holding Gloves bonus: This bonus allows the hero to hold the bombs which have been planted before and throw them to somewhere else.

Ghost bonus: This bonus allows the hero to pass through the walls and freezes all monsters for a certain time.

Gun bonus: This bonus allows the hero to shoot only one bullet whenever the player wants. This bonus may be combined with other gun bonuses so that the hero can shoot more than one bullet according to the number of unused gun bonuses.

Extra Life bonus: This bonus gives the hero an extra life and he gets one more chance to finish the level up.

5.2. Aspects of Creatures

There are mainly two types of creatures in the game. They look as the same and this will make the game a bit complicated and hard. Both kinds of creatures die after they are affected by the effects of the bombs.

Normal creatures: These creatures move randomly on the map and their artificial intelligence is in a very low level.
**Smart creatures**: These creatures move towards the hero and they try to escape from the bombs. Their intelligence is in a higher level.

**5.3. Maps**

The tile and iron walls, the some items which are non-moving but effective in a certain area, the power ups and the key for the door are put in different places in each map.

**5.4. Levels and Level Finishing**

There are 5 levels in the game. The levels get more complicated and harder. The tile and iron walls are in different places in every level. The positions and the number of the monsters change also. It is harder to find the key and the door in the next levels.

In the Key-Door Mode, the main aim is to find the key for the door without dying and before the time limit expires. While escaping from the monsters, the hero can kill them with his bombs and pick up different power ups to make the level easier. When he finds the key for the door, he needs to find the door as well. He needs to go to the door and unlock it with the key. Then the level finishes up.

In the Killing Spree Mode, the main goal is to kill all monsters in a specified time limit. If he can succeed to kill them all, the level finishes up.

In the Bomb Limited Mode, there are constant amount of bombs and he needs to find the key and the door. If he cannot find the door or the key and he has no bombs left, the game will be over.

**5.5. Game Over Conditions**

If the hero is killed by a monster, non-moving objects or his own bombs, his lives decrements by one. If he dies in his last life or the time limit expires before he unlock the door and escape, the game will be over and the user needs to start from the beginning or from a level he has saved before. If he succeeds a better score than anyone in the ‘High Scores’ list, he is asked for a name that will take place in the ‘High Scores’ table.

**5.6. GUI Examples**

**Main Menu:**

In the beginning of the game, main menu will be opened. This screen includes the choices to play the game, see the high scores and adjust the settings and exit.
Playing the game:

If the player chooses to start game button, the game starts. The following screen is for level-1. In the other levels, the items except for the map will remain the same.
Level Passed Screen:
This screen appears when the player passes one level to the other.

The Settings:
In the settings screen, the user can change the followings:

- Game Mode, (Key-Door, Killing Spree, Bomb Limited)
- Adjust the background music,
- Adjust the sound effects,
- Selecting the Bomberman character
- Game Difficulty
Controller Settings:

When the player clicks to the button in the settings screen, this screen appears and in this screen, the player can see and change control keys on keyboard.
High Scores:

In this screen, the players who made the top scores in the game are listed.

<table>
<thead>
<tr>
<th>RANK</th>
<th>NAME</th>
<th>DIFFICULTY</th>
<th>SCORE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Susan D.</td>
<td>HARD</td>
<td>75810</td>
</tr>
<tr>
<td>2</td>
<td>Tommy L.</td>
<td>EASY</td>
<td>56820</td>
</tr>
<tr>
<td>3</td>
<td>Isabel R.</td>
<td>EASY</td>
<td>41610</td>
</tr>
<tr>
<td>4</td>
<td>Mike Strike</td>
<td>EASY</td>
<td>37230</td>
</tr>
<tr>
<td>5</td>
<td>Gina M.</td>
<td>EASY</td>
<td>32850</td>
</tr>
<tr>
<td>6</td>
<td>Antonio T.</td>
<td>EASY</td>
<td>26280</td>
</tr>
<tr>
<td>7</td>
<td>Alain C.</td>
<td>EASY</td>
<td>19710</td>
</tr>
<tr>
<td>8</td>
<td>Linda M.</td>
<td>EASY</td>
<td>13140</td>
</tr>
<tr>
<td>9</td>
<td>Santina Z.</td>
<td>EASY</td>
<td>6570</td>
</tr>
<tr>
<td>10</td>
<td>Richard BB</td>
<td>EASY</td>
<td>2190</td>
</tr>
</tbody>
</table>

*Figure 13: High Score Table Screen*
Death and Game Over Message Screen:
If the player looses all his lives or the timer expires, he loses the game and dies. He can continue from a level he saved before or he can start to a new game.

Figure 14: Game Over Screen
6. CONCLUSION

In the analysis report, we have focused on what the game does, what functions it has, etc. However we have not considered how it does them, how it proceeds the events. Game features and given requirements were our main helper before we started to consider use cases and their scenarios.

The analysis report was a useful report such that it helped us to think about the design process. The scenarios and use cases in this report were led us to pass to the design process easier and helped us about the contents of the design report. Generally, the main aim of us was to make people understand the analysis of our games and make a better image of our game in people’s minds. To achieve that aim, we used case tools such as Visual Paradigm and using a modeling language, Unified Modeling Language (UML). Since UML is a unique programming language, it really helped us to communicate with other people easily. We used Visual Paradigm to create our UML diagrams such as Use Case diagrams, Sequence diagrams, Domain Analysis Diagram, State and Activity Diagrams etc.

Finally, in the Analysis Report, we have added new features to the classic Bomberman Game such as guns, ability to throw the bombs and pass through walls. We wrote our ideas into papers so there is almost a complete and beneficial information in front of us while we are designing and implementing our project and we can concentrate on the use case scenarios. To sum up, we have tried to understand and detail our game’s features so we will be able to design and implement our game in the design and implementation processes better.