Bilkent University

Department of Computer Engineering

CS319 PROJECT – GROUP #2

Project Final Report

CS 319 Project: Bombalamasyon

Oğuz Demir – 21201712

Anıl Sert – 21201526

Kaya Yıldırım – 21002071

Kaan Kale – 21000912

Course Instructor: Uğur DOĞRUSÖZ

Design Report

Apr 30, 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

List of Figures

1 Changes in the Design

The implementation is done with respect to MVC style for architectural style and Facade Pattern for low level architecture as it is stated in the design report and there is no major changes in design. The changes are the following:

- Communication between Model and View subsystem is removed and this communication is established through Controller subsystem with the feedback of instructor.
- For behavior of computer opponents, a class named AiEngine is implemented. More logical behavior is given to these opponents with the help of that class.
- In addition to existing 2D object array, another 2D integer array is held in GameEngine to draw whole map quickly. Otherwise, iteration through all object is needed.
- The grid structure of the game is changed. In each cell, there may be a bomb, a powerup or a wall. So all these objects are held in 2D object array which is mentioned above. So Wall and PowerUp lists are removed. Bomb array is still used in order to iterate and countdown all of the bombs in each game iteration. Also, since the bombers can between the grids as smoothly moving objects, and they can go over a bomb or a powerup, the bombers are held in separate array. By that way, they can be easily controlled and drawn.
- Since the drawing problem is solved with 2D integer array, the subclasses of Wall class became redundant. Instead of subclasses, an integer property is used for holding type of the Wall.
- beExploded method of Explodeable interface is changed to pass the owner of the bomb which caused that explosion in order to update the scores. It should be known that by whom a wall or a bomber is exploded.
- MainFrame class is changed to be a singleton class in order to be accessible from inner panels of that frame without holding a reference. Because, panels need to pass information that comes from user to main frame.
- Some panels are added for showing game related info on game screen for better gaming experience.

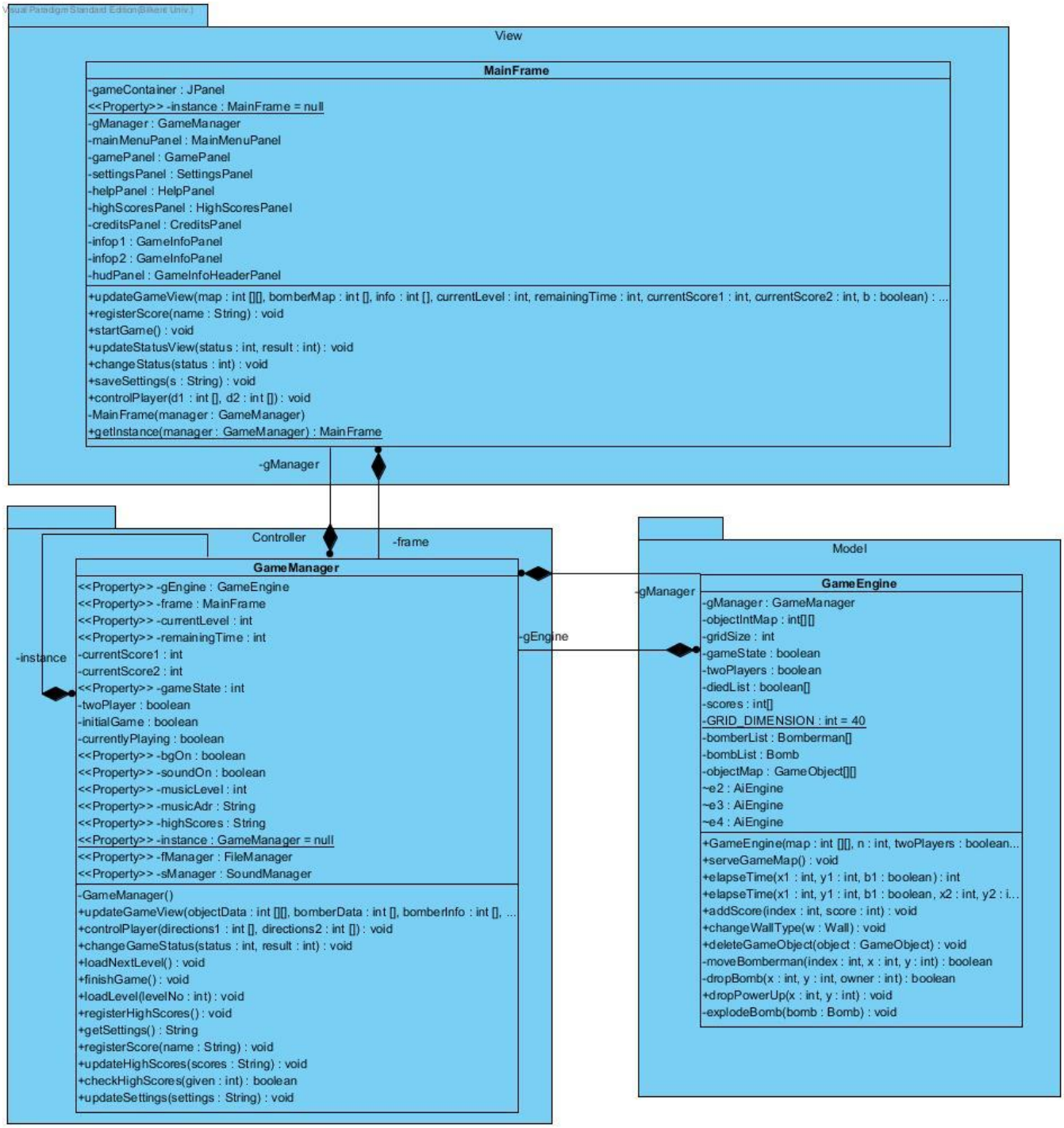


Figure 1 The 3 Subsystems (MVC), 3 Facade Classes and their Associations

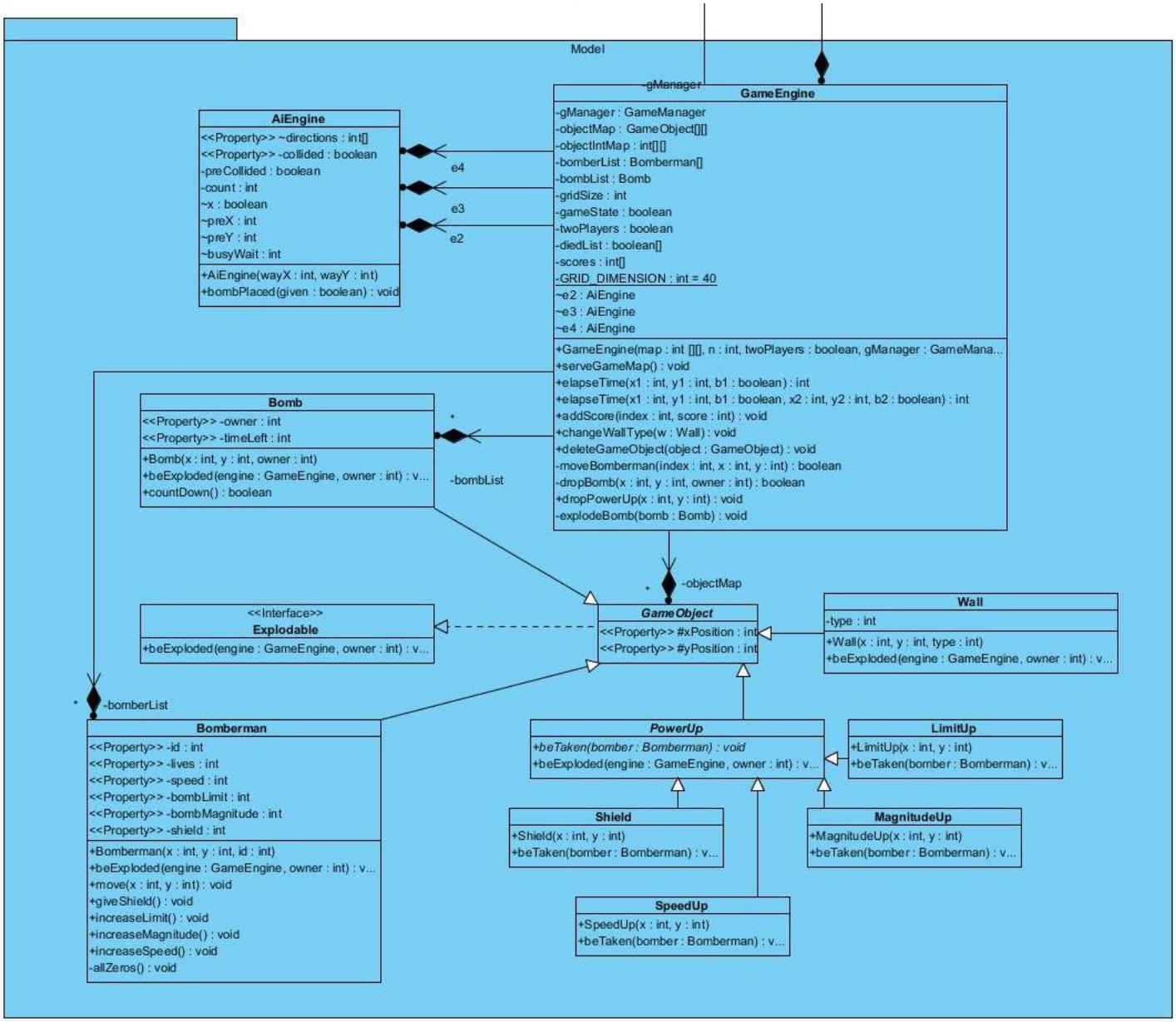


Figure 2 Detailed Model Subsystem

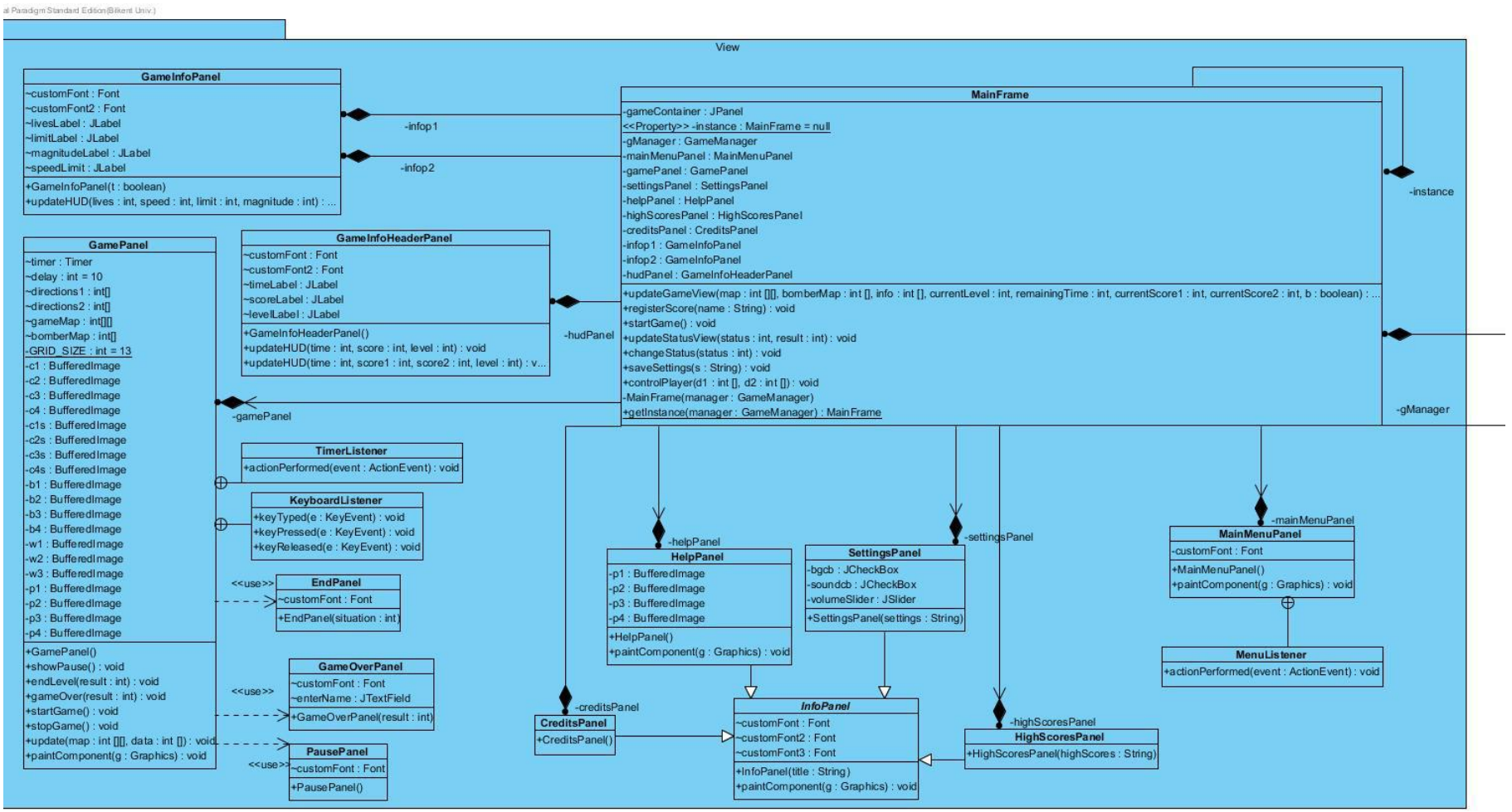


Figure 3 Detailed View Subsystem

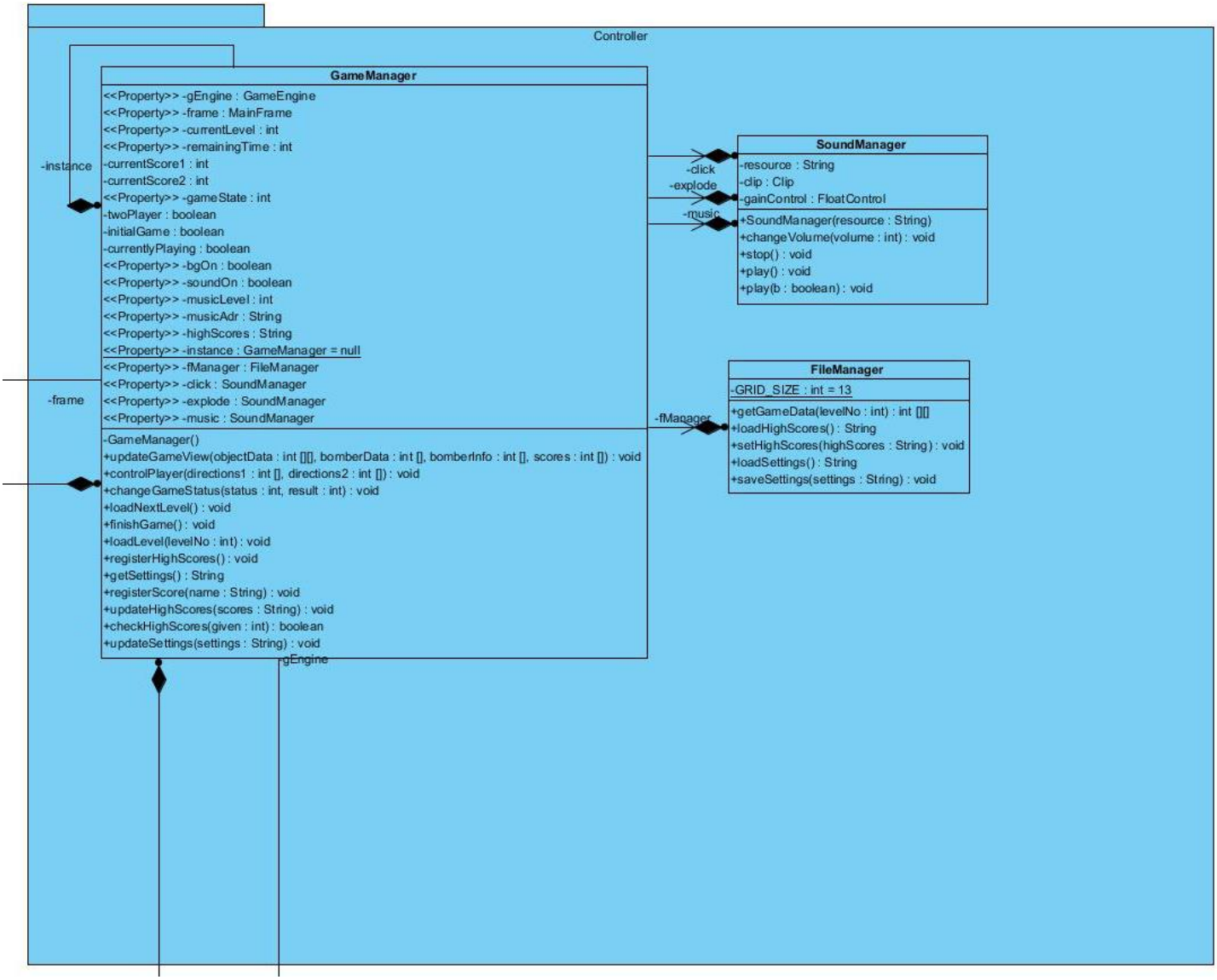


Figure 4 Detailed Controller Subsystem

2 Complications during the Implementation

The first challenge we encountered was move the Bomberman to between two walls. Since the size of the gap is equal to size of width of the bomber, it was not easy to move bomberman that gap due to collision. This problem is solved by with rounding the coordinates if the collision occurs at quarter of a grip. In other words, the corners of the walls are rounded logically to let bombers pass through.

Also, the movements of the computer opponents are planned to be random. Since for each game time, the bombers are moved 1 unit, ((x,y) = (1,0),(-1,0),(0,1),(0,-1)), the computer opponents constantly shake if there is randomization. So, a basic artificial intelligence class that serves commands for computer opponents are developed for basic cases, such as if a bomb is dropped, go and wait on second corner etc. This improvement bring more fun on game for single player.

Another complication we encountered that it was hard to inform player about different versions of situations because info on the panels should be changed according to game data, and there is so many cases (I.e., Level end, the both players may die, player1 may be the last man or player2 may be the last man). So, instead of adding extra states to game, the game related info sent by GameEngine is sent to the MainFrame to reduce complexity on GameManager. Also, 3 helper panels are used by GamePanel to show specific info on specific cases.

Implementation of volume adjustable sound effects and background music was problematic. We could not find suitable library for this purpose. Since background music is required to be played in a loop to be continuous, it was not possible to stop it with the user choice. Also, for our foundations about implementing sounds, volume adjustment was not also possible. This problem can be solved with an external library but it needs more research on that.

3 User's Guide

3.1 Introduction

Bombalamasyon is an arcade-style and maze-based game. In the single player mode there is only one player and three opponent bombers, in the multi player mode there is two player controls two bombers and two AI controlled which all opponents. Players tries to explode other bombers with their bombs. Meanwhile, bombers are also tries to find some power ups that are hidden under the destructible walls. The game map consists of three different walls. After one bomber is last man standing in the level the level changes and after 4 levels game is over.

3.2 System Requirements and Installation Info

3.2.1 System Requirements

The Java Runtime Environment (JRE) is required to play the game and it can be downloaded from the website below:

http://www.oracle.com/technetwork/java/javase/downloads/index.html

Minimum system requirements:

- Windows 7/8/10
- Screen Resolution: 1024x768
- Graphics Card: Not required.

3.2.2 Installation Info

Download the project from the Bombalamasyon project website:

https://github.com/oguzdemir/Bomberman

1) Run the bombalamasyon.jar file to execute and play the game or

2) If you are able to use a Java IDE or command line, compile the .java files and run the main method to play the game.

3.3 Inside the Bombalamasyon

3.3.1 Game Overview

In the game, players controls a Bomber and they gets into the action in a predefined map. Every bomber in the game tries to explode the other bombers with their only tools bombs while trying to survive others' bombs. Players can put their bombs to available locations in the game map.

There are 4 different levels in the game Bombalamasyon. Every level has different from each other and the later levels have less point and power up opportunities for the players.

There are essentially two structures in the game map. The first one is the bombers that are controlled by the players or the AI. The second one is the walls because players are seperated from the walls and powerups dropped by the exploded walls. The walls distributed differently for every level but the bombers' locations are the same for every level.

The other important thing is the time constraints for the level. Players' time start as soon as the started to play the game and they try to get as many as points before the timer ends or there is only one bomber remaining. If the timer ends, the level ends and skip to the next level.

There is a high score list in our game. This list shows the best 10 high scores that are accomplished in the game history by the players.

3.3.2 Game Objects

Bomberman is the main character. Players of the game and AI are controlling this with different colors (blue, red, yellow and green). This object can drop bombs and destroy walls and other bombers.

Shielded bomberman is the shielded version of the bomberman character. If the players or the AI takes the shielded powerup the bomberman turns into the shielded bomberman.

Bombs are the tools for the bombers. Bombers can destroy the walls and the other bombers with dropping bombs to the game map. All bombers have bombs to their color (blue, red, yellow and green).

Brick walls are the easiest walls to break. They are destroyed by the bombs with only one explosion.

Strong brick walls are stronger that the brick walls and harder to destroy. If a bomb exploded near a strong brick wall, it turns into a brick wall so players need to explode 2 bombs for destroying the strong brick walls.

Steel walls are the indestructable walls in the game. There is no way to destroy a steel wall and bombers have to walk around them.

Shield power up is the power up that protect the bombers from one explosion damage.

Speed up power up is the power up that increase the bomber speed.

Limit up power up is the power up that increase the bomb limit a bomber can drop by 1.

Magnitude up power up is the power up that increase the explosion area of the bombs dropped.

3.3.3 Scores, Time and Levels

Player scores are shown at the top of the screen. If the player destroy walls, bombers or dies their score changes. To make the players play the game more rapid there is a timer at top left corner of the screen and inform players the remaining time. Also players can see the current level they are playing from the top right corner of the screen. Players have to finish the level until time is up.

3.3.4 Powerup Observer

Players can see their bombers' attributes from the left of the screen for first player and right of the screen for the second player. These information can inform tha players what their bombers are capable of doing.

3.3.5 Controls

For the first player:

(UP) Moves bomberman towards up.

(DOWN) Moves bomberman towards down.

(RIGHT) Moves bomberman towards right.

(LEFT) Moves bomberman towards left.

(NUMLOCK-0) Make bomber to drop bombs.

For the second player:

(W) Moves bomberman towards up.

(S) Moves bomberman towards down.

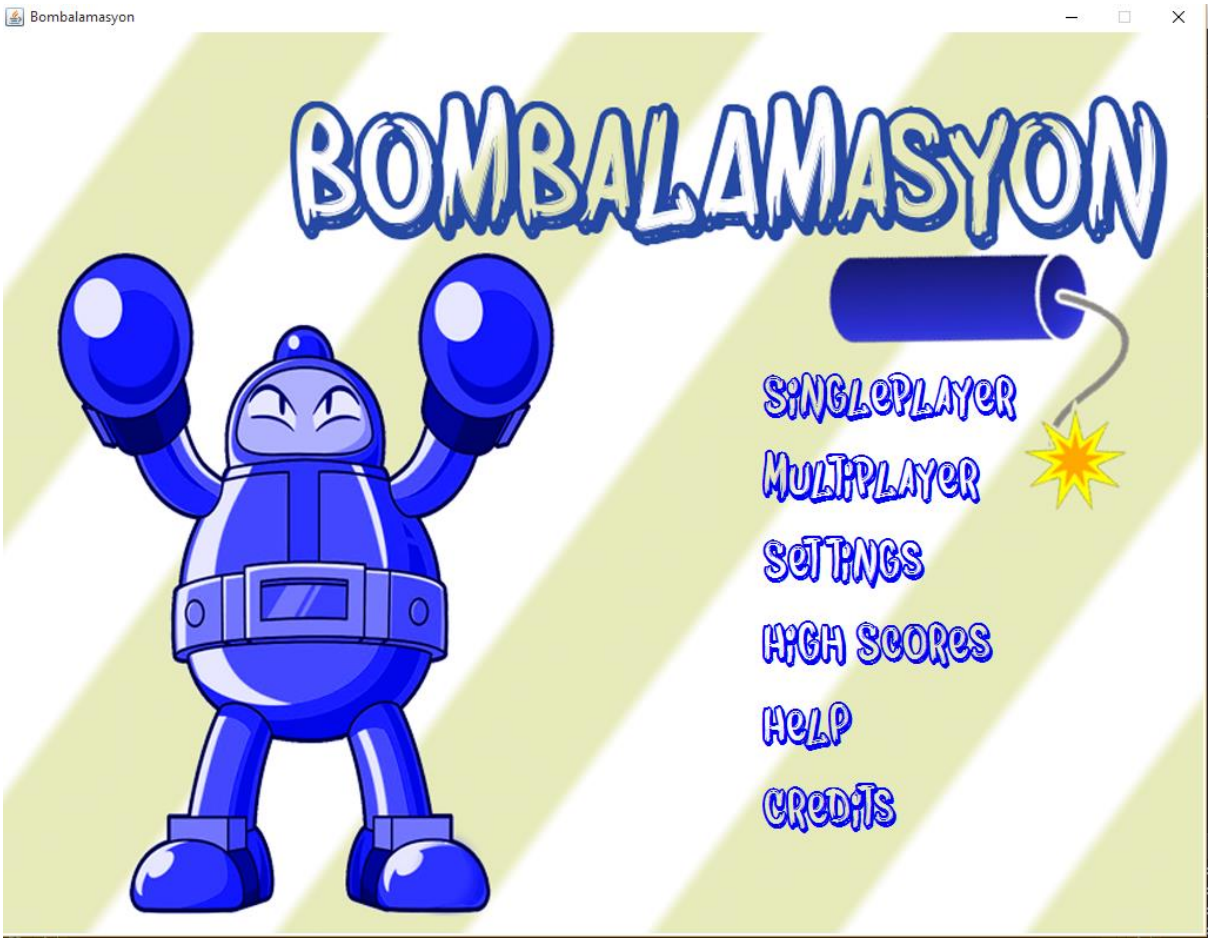
(D) Moves bomberman towards right.

(A) Moves bomberman towards left.

(SPACE) Make bomber to drop bombs.

3.4 Game Screens

Players can navigate between menus with buttons located in main menu, pause menu and with back buttons inside the panels. User can start singleplayer and multiplayer games from main menu and quit from them by pausing the game with pause button on left column. Also, user can see high scores, help and credits by navigating from main menu. The help also be seen from pause menu. Game settings are configurable from the settings menu which can be accessed from the main menu.



4 Current Status of System

The system is implemented in a way that it satisfies all functional and non-functional requirements which aimed in analysis stage. Also, all of the use cases are covered in current system. The game is able to be played without error or bugs and game related info is shown in game screen which is added after analysis and design.

However, the speed attribute of bombers cannot be successfully implemented. Since the bombers have constant speed to cover pixel of game area, they cannot cover smaller distances than their speed. This issue caused bombers to be stuck at one line of the map so the bombers speed are not changed with powerups. However, players see the game info as if their bombers speed is increased when they take a power up, so there is a placebo effect.

Background music and changeable volume also cannot be implemented. In the library we used, clips that are started for playing in a loop cannot be stopped so we removed the background music. Also, the clips volume cannot be changed in that library. This problem can be solved by using another library but we did not have enough time for it.