Bilkent University

Department of Computer Engineering

CS319 PROJECT – GROUP #2

System Design Report

CS 319 Project: Bombalamasyon

Oğuz Demir – 21201712

Anıl Sert – 21201526

Kaya Yıldırım – 21002071

Kaan Kale – 21000912

Course Instructor: Uğur DOĞRUSÖZ

Design Report

Mar 27, 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

1 Introduction

1.1 Purpose of the system

Bombalamasyon is a system that aims to provide users with modified version of bomberman game with multiplayer and singleplayer features. The main purpose is serving a simple but challenging game for small breaks of computer users. The designed game is poor in terms of today's game standards (such as complex AI, 3D smooth graphics, fluent movements etc.) so the gameplay of the game is changed to maximize the pleasure of achievement. The "Multiplayer" game mode allows two users to play at the same time from the same computer and to challenge each other. The game is designed to be played from most of computer platforms with different standards and without internet connection. The system also aims to provide a plain interface to make users learn the game easily and improve gaming experience.

1.2 Design goals

In order to compose the system we should clarify the design goals we focused on. These design goals provided in analysis stage from non-functional requirements that we did before design. Here are described design goals:

1.2.1 Ease of Use

Easiness in the usage is one of the most important design goals because it will determine whether the users continue to play the game or not. The game is designed to be played in small time intervals to enjoy and it would not be successful if the users have trouble while playing this game. Similarly, learning the game should not take much time not to waste the limited game time with learning.

1.2.2 Reliability

The system is aimed to be bug-free in order to prevent from crashes or errors while gaming. The unexpected terminations at the middle of the games would be annoying for users. Also, any error that can cause loss of game statistic information of game statistics cannot be welcomed by users with high scores.

1.2.3 Extendibility

The game is planned to be completed in limited time, so it is limited in terms of number of different features and levels. It is important to add new features to games in order to make

them more attractive. Also, changes in game, will bring back the users who completed the game and abandon it. So, the system is aimed to be designed in a way that it can be easily extended for new features and levels.

1.2.4 Responsiveness

The game is interactive game, the players use their in-game characters to complete the game objective. So, the users should immediately see their commands' effect on the screen. In order to satisfy enough responsiveness for the users, the game view is designed to be refreshed every 0.1 second, in other words, the game shows 10 frames per second.

1.2.5 Portability

In order to serve the game for the users from different platforms, the system should be portable, platform independent. The system aimed to be developed in Java so that it could be run every computer which has Java Virtual Machine.

2 Software Architecture

2.1 Subsystem decomposition

For the system Model-View-Controller (MVC) style is chosen to split system into parts for sharing the complexity of the system among the components. Also, since the components are designed to be independent as much as possible, it increases the readability of the code and extendibility of the system.

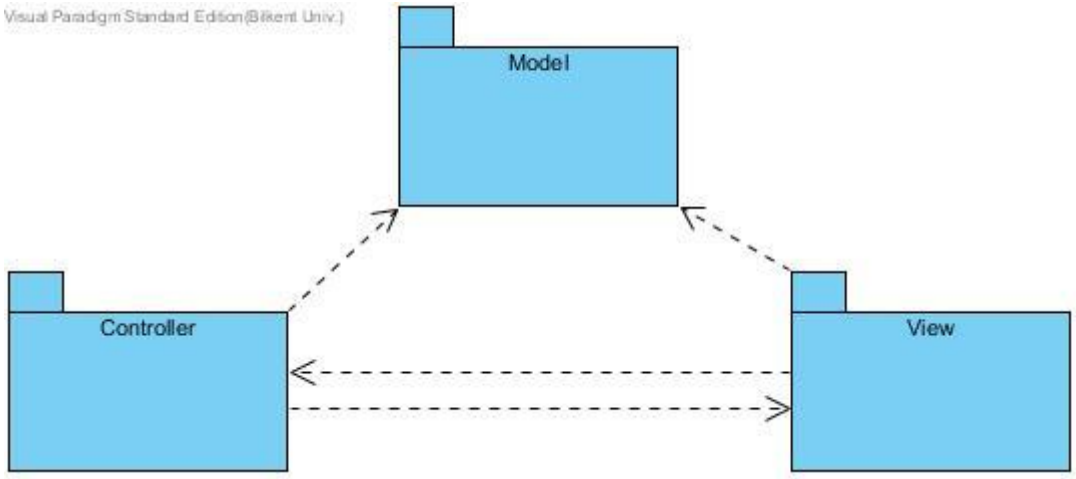

Figure 1 Subsystem Decomposition Diagram

In the Model subsystem we will hold all the game's model objects which are interacted with each other during the play time and these models will be updated with the advance of the game. The physical events such as movements, collisions, map creations, object creations, ai behaviour of bombers are responsibilites of the Model subsystem.

Controller subsystem is the brain of our system. It includes controlers for Game, File Management, Sound Management. Responsibility of the Controller Subsystem is to manage the flow of the game, take neccessary information from files and pass it to other subsystems and to play appropriate sounds according to game state.

View subsystem has all of the view components of the system and the responsibilty of the View subsystem is to reflect the correct window with needed information on to the screen. The view is updated and/or current window is changed according to changes in the game state.

2.2 Hardware/software mapping

Bombalamasyon requires Java Runtime Environment (JRE) to be played because it is developed by the using Java programming language. Game can be executed with a single executable Java file.

For the I/O requirements computer needs a keyboard, mouse and a monitor to let player interact with the game. For multiplayer gaming, the keyboard should not have ghosting (blocking some of the keys that are pressed at the same time) when 6 keys are pressed (6 keys: 3 for each player; 2 for direction, 1 for dropping bomb). It requires very little system requirements to be played. Graphical Processing Unit (GPU) is not required to play the game.

File system will be used for .png, .wav, and .txt files in order to take the game data and game images and play sound effects and background music.

2.3 Persistent data management

Files are stored in the hard disk drive. The game keeps names and top ten scores in plain text file in order to display to the player in "High Scores" section. To provide better gaming experience to player, some image and sound files are also used at some parts of the game. When they are needed, these files are read from the disk with their specified directions as

parameters. In addition, level data is stored in hard disk drive. There are different game maps for each level in hard drive.

2.4 Access control and security

Bombalamasyon does not implement any user authentication system therefore we do not have any database that stores user credentials. Also, as mentioned earlier (in Hardware / Software Mapping), our game does not require network connection. Therefore, player who has no network connection is able to play the game. So that, there is no restriction or control for access the game. In addition, the game has no user profile, only player names and scores. Therefore, there is not security issues in Bombalamasyon.

2.5 Boundary conditions

Initialization

When player execute the .jar file, the game initializes. Player does not have to install the game.

Secondly, the game tries to load every file that can be needed during execution. If a critical file such as level map is missing, the initialization will fail.

Termination

In order to terminate the game, player can click the “Quit Game” in the main menu. When player is playing the game, he/she wants to exit, firstly the player is need to go to “Pause menu” and then click the “Quit Game”.

Game will return to the main menu if all the levels are done. In case of finishing, high scores are updated if score is higher than 10^{th} best score and the game returns to the main menu.

Game also be closed by the “X” on the top right corner of the window. Unsaved data will be loss.

Error

If any file (game resources) could not be loaded such as images or sounds, the game starts without these files. If the game does not respond because of other issues such as problem at hardware, software or operating system, player lose his/her current data.

3 Subsystem Services

The system is decomposed into 3 parts as model, view, controller and there are 4 main services between these components. The flow is the following: when user give the input, the View takes the input as the boundary component, and it passes the related input to the Controller with Controllers' service. Controller change the game status in itself and/or the properties of the Model with Model's service. After that, the Controller ask for an update on the View via View's service and before updating the current view, the View component can take the game data from Model with the service of the model. At the end, view is updated and changes with the user's input is reflected on screen.

3.1 Services of the Controller:

takeUserInput: This service of the controller is used by view component in order to pass the related user input to change the program status (main menu, paused game, in game etc.) and to control the game(move bomberman or drop bomb). For example, if the user pauses the game while playing, the view component who has the action listeners for the keys, pass the corresponding input through takeUserInput service of controller for changing state and controller change the game status which is stored in the controller itself to "pausedGame".

3.2 Services of the View:

updateView: This service of the view is used by the controller to change the program display between menus or reflect the changes in the game map to screen. The status of the program such as main menu, in-game etc. is passed to the viewer and if it is in-game, the game data is taken from model component with the help of getGameMapData service of the model component.

3.3 Services of the Model:

getGameMapData: This service of the model is used by the view component in order to get the game map data, in other words, positions of the game objects with their types.

updateGameObjects: This service of the model is used by controller to manage in-game data with the desire of the user within a time interval and to process CPU controlled objects in that interval. The score that is earned during that interval is returned.