Year 3 - 2021/2022

Computer Science Project

Final Report

"Bomberman's Revelry"

Word count: 9886 words

Student Name : Xuanwei Xu

Student ID : 2089260

Supervisor name : Mirco Giacobbe

Programme name: Computer Science

Contents

1 Introduction

When it comes to games, many parents think they are meaningless to their children and there is no gain after playing games. Although games designed specifically for entertainment, they also have educational value to promote learning in the field of computer science. Many people have been inspired by games to understand and explore the field of game development due to the enjoyment of game, which undoubtedly leads to a great impact on the educational significance of computer science. As one of the most popular game series, *Bomberman*, has become fond memories of many people's childhood and brought joy to them. In this thesis I will use libGDX framework to recreate a *bomberman* game with more creative gameplay, which contains multiple optional difficulties and modes.

1.1 Bomberman Series of Game

Bomberman, also briefly known as *Dynablaster*, is a strategic maze-based video game franchise originally developed by Hudson Soft[1] in 1985. There are some screenshots of this version in figure 1 below. This game has attracted many fans and followers since it was released. After that, more and more new derivative works of *Bomberman* series have been released on different gaming platforms and both have been a huge commercial success with over 10 million units of games sold.[2] In addition, *Bomberman* has their own online game community called "*Bomberman wiki*" where many loyal players and hobbyist self-trained programmers express their opinions or suggestions and share their game experience, which shows the popularity of *Bomberman* series.

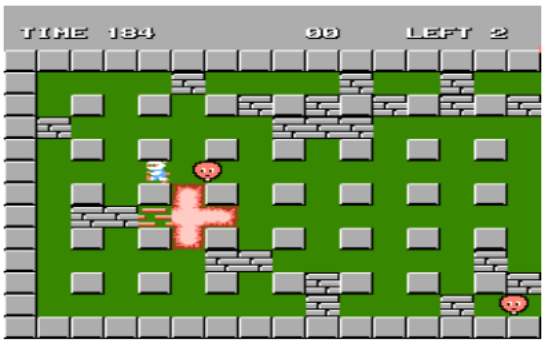


Figure 1: Screenshots from NES version of *bomberman*, taken from old-games website[3]

Usually the *bomberman* game series is played with the intent that players can win by planting bombs strategically to kill all enemies(AI or other players) in maze-based grid world. The bomb will explode and spread in four directions in the shape of a "cross" after it was placed, any obstacles or characters within the blast radius of planted bombs will be destroyed except for the indestructible obstacles. In the game scene, all of pathways on the map are blocked by many different types of obstacles to form a huge maze and some types of obstacles can be destroyed by planting bombs nearby them so that players can pass through.

One of the *Bomberman* series, "*Bomber Man World",* which was listed to be forty-sixth most popular arcade game at the time by *Play Meter.*[4] It introduced four different power-ups to the basic *bomberman* gameplay, and they can be gain after blowing up destructible obstacles, these are :

1. Speed: Increase movement speed .
2. Flame length: Increase explosions ranges of placed bombs.
3. Bombs: Place more than a single bomb.
4. Invincibility: Player cannot be killed by enemies or explosions.[5]

In addition, this series also has two game modes:

1. 'Normal Game': Single-player mode, which requires a player to defeat all AI enemies.
2. 'Super Game': Muti-player mode, which is the battle of players, the aim is to defeat all the other players.[6]

Both the introduction of power-ups and the design of various game modes in "*Bomber Man World"* adds the gameplay and enhances the gaming experience of players. This series exerted a tremendous fascination on a great number of people and brought them joy, including me, with its gameplay and precept.

1.2 Motivation and Project Goals

However, the important limitation of the series "*Bomber Man World"* is that this arcade game does not have cross-platform compatibility, which means that players cannot experience the joy of this game on other platforms, such as personal computers

or mobiles devices. Cross-platform *Bomberman series* are becoming increasingly rare because game companies want to increase their console sales, no one will buy exclusive game consoles from game companies if their developed games can be played on any platforms. For years, many players have dreamed of being able to play this game with others across different platforms so that they do not have to go to video arcades with their friends specially or cost a considerable sum of money to buy game consoles from game companies. Therefore, it inspired me to develop a cross-platform game so that I can publish it on different platforms after completing the game development on the desktop platform, which ensured my developed game can reach more people and let them experience the joy of this game.
Developing a cross-platform game is complicated and has big technical challenges because different devices work in different ways from one another, including hardware, data formats, so it is hard to ensure the code can be cross-compile on different platforms. My solution is to consider and use a suitable cross-platform game framework and a programming language, it can deal with these differences between different devices and help the code adapt for multiple systems.

In addition, there is still some room for improvement in series "*Bomber Man World"*.
Firstly, the number and variety of game props are relatively small, more weapon props with novelty can be designed and introduced to enrich the gameplay and heighten playing pleasure, like rockets launcher, grenades, landmines and so on. Secondly, players are not allowed to choose game difficulties, which means that players must start with the easiest levels and gradually play more and more difficult levels. Thirdly, this game interface is not pretty and has monotonous styles because it is an arcade game, it can be improved by adding gorgeous graphics. Therefore, I wanted to improve this game to recreate a brand new bomberman game from those aspects to make it better and more enjoyable.

Finally, my goal is to recreate an improved version of bomberman game written in java with more innovative gameplay based on the gameplay of *"Bomber Man World"* by using the cross-platform game development framework libGDX. This game not only included many game props but also provided multiple optional difficulties and modes for players. In offline mode, players are allowed to experience different difficulties and they need to play against AI enemies alone. The intelligence of the AI largely determines the fun of the game, so I make AI will become more intelligent with increasing difficulties, they could pick up power-ups to strengthen themselves and use weapons to attack players. In online mode, two players need to kill each other to win. That is what I plan to achieve with this game, make the game more enjoyable with more innovative and interesting gameplay and let more people can get access to my developed games on different platforms and experience the joy of the game.

2 Background

In this chapter I will give a overview of some fundamental concepts and principles required for getting started with game development.

2.1 Game-development Framework:

To start with, I will present the concepts of the game framework and talk about what is it used for. A game framework is a working environment optimized for creating games software, which consists of a set of libraries that should facilitate the development of games. The underlying library, in turn, is a set of data (programs, subprograms, objects, functions) that a programmer needs in their work.[7] A game framework defined many essential APIs to deal with graphics, sound, user input, data files,etc.[8] Therefore, developers do not need take much time on defining these functionalities to develop a game, it can help developers accomplish many common development tasks, such as rendering, handle of a sound effect, creating user interfaces, drawing text, linear algebra and so on, which greatly pushes forward the progress of game development and improved the development efficiency. Therefore, it is a wise choice to develop a game by using a suitable game-development framework.

LibGDX is an open-source cross-platform game-development framework written in java programming language[9]. Therefore, it facilitates to debug code while creating a game using java. Java is a programming language which has platform independence compared others, it can compile and run on multiple platforms or devices very easily with its portability. Except for including libraries that many general game-development frameworks would have, the greatest feature of the LibGDX is its good compatibility, it provides a single and unified API for many platforms so that I can access multiple host platforms to write or debug code on these platforms. I can also publish the code on many different platforms without writing the code for each specific platforms after finishing the development of a game on desktop. Using libGDX and programming language java in game development only meet the cross-platform feature requirements of developing this game exactly but also improve my development efficiency significantly.

2.2 2D Animation:

2D Animation is an indispensable technique applied in game development, which is used to create the illusion of movement using static images. An animation consists of multiple frames which are shown in a sequence at set intervals. [10] The movement and jumping of game characters in the game scene is essentially an animation in which multiple static images are switched and displayed in sequence over a very short period of time in a loop. The character in following image was I used in my developed game, this image contains every frame of animation of the character walks respectively in four directions (left, right, up, down), an animation of walking is created by displaying the series of frames sequentially over a period of time.

Frame rate refers to the displayed number of frames per second. For example, a character completed all actions of moving left, this will be regard as a cycle and each cycle has 3 frames of animation in above image, which means that these 3 frames should be displayed per second if the character has to complete a cycle in one second, the frame rate is 3 FPS at this time so the amount of time to display every frame is approximately 0.3.

2.3 Collision Detection Problems:

Collision problem is one of the inevitable problems to consider in games development. Collision detection is the computational problem of detecting the intersection of two or more objects and applied in multiple computing fields.[11]
Collisions are very common in games, for example, there may be collisions between

characters and obstacles, collisions between characters and bullets, collisions between bullets and obstacle, etc. These collisions problems among these elements in a game need to be detected and handled in real time by writing separate code. For instance, there will be an explosion if the collision between a bullet and an obstacle happens, it is necessary to write the code of logical judgement to check whether the fired bullet collided with an obstacle at real time, if yes, it will play an animation of explosion.

Hit-box is a type of invisible bounding box which is used to detect collisions between elements in games in real-time[12], it is usually a rectangle attached around the objects or elements in 2D game, such as characters, bullets, obstacles. There will be the simplest collision detection can be implemented by checking whether these two rectangles overlap with each other if the two axis aligned rectangles attached around two objects are not rotated or scaled before they colliding.

I will describe more details of game design and requirements setting after reviewing the background material about my game development.

3 Software Design

In this chapter I will present the specifics of the game design according to project goals, including game features, functional requirements and non-functional requirements, the interface designs.

3.1 Key Features

To start with, I will mention the features of this game in terms of game playable and creativity.
The first feature of the game is the improvement of gameplay and rules, it introduced new destructible obstacles “treasure chests” and more game props with novelty while keeping the basic bomberman precept unchanged, including powerful weapons and general power-ups that enhance characters’ attributes. Each game prop can only be obtained by blowing up “treasure chests” with equal probability. There are 9 game props in total:

1.  Medical kits: Restores 1 health point.
2.  Running Shoes: Increase movement speed, which can be accumulated.
3.  Nitroglycerine solution: Increase the explosion ranges of bombs placed,

which can be accumulated.

4.  Landmines: Placing a landmine which is invisible to all characters.

5.  Bazooka: Getting a bazooka allows to launch two rockets.

6.  Hand grenades: Throwing a hand grenade behind an obstacle.

7.  Bomb package: Place more than one single bomb, the number of placed bombs can be accumulated.

8.  Riot shield: Withstand explosive damages from bombs, landmines, bazooka, hand grenades at once.

9.  Gloves: Push bombs out before 3 seconds of explosion.

For game rules, players need to kill all enemies to win by placing bombs strategically or using weapons, the game failed if players did not kill all the enemies within the time limit or are killed by them. The improvement of gameplay enriched the content of game and raise interest.

The second feature is providing multiple optional difficulties for single-player mode so that players are allowed to choose any level with different difficulties. AI will become more intelligent and could use game props in a rational way with increasing the difficulties, they will constantly look for "treasure chests" and place bombs to destroy them in order to collect more game props, they will also use weapons to attack players, which makes AI harder to be kill for players. There are 3 optional difficulties in total:

1. Easy level: AI can not pick up and use any of dropped props.
2. Medium level: AI can only use non-weapon props that can enhance their attributes (including medical kits, running shoes, Nitroglycerine solution).
3. Nightmare level: AI can use any of dropped props to enhance their attributes and use weapons to attack players.

More Intelligent AI increased the difficulty of the game, making the game more challenging and fun.

The third feature is adding the function of chatting room for two players in online mode, it allows one of players plays one-on-one with another by creating or joining the room to establish connection in a local network area, players can send messages to

each other during the game, which increased the interactivity and competition between players.

It is necessary to carry out requirements analysis to determine what specific functions the game needs to implement according to these game features, so I will described the functional requirements and non-functional requirements.

3.2 Functional requirements

It is important to clarify and define functional requirements of a product, which can help me to check whether this game product can achieve these goals or identify missing requirements.
This software system can be divided to 7 sub-components , I listed the specific functionalities that each of the sub-components needs to perform in order to check whether the captured behaviors of them meet requirements that mentioned in goals.

3.2.1 Obstacles

- Iron blocks can not be destroyed by bombs or weapons.
- Brick blocks can be destroyed by bombs or weapons.
- Treasure chests can be destroyed by bombs or weapons.
- Game props dropped after destroying treasure chests.

3.2.2 Bombs

- Placed bombs will detonate after three seconds.
- Bombs can kill players and AI.
- Players and AI can not pass through bombs.
- Explosion range of bombs can be increased and accumulated.
- The number of placed bombs in three seconds can be increased and accumulated.
- The spread range of bombs should be blocked by obstacles.
- The bombs can be pushed by AI or players by using non-weapon props gloves.
- The bombs being pushed will explode if they hit obstacles.
- The bombs can not be pushed to outside of map boundaries.

3.2.3 Players

- Players can manipulate bomberman to move around in four directions(up, down, left, right) by using arrow keys.
- Players can plant bombs by pressing the space bar.
- Players can pick up any dropped game props.
- Players can use 4 different weapons by pressing numeric keys 1, 2, 3, and 4.
- Players can be killed by bombs or weapons.
- Players should not pass through obstacles and bombs.
- Players cannot go outside the boundaries of the map.
- Players can pause or resume game by clicking corresponding buttons at any time during game play.
- Players can check game rules or help by clicking corresponding buttons at any time during game play.
- Players can exit and return to main game menu at any time during game play.
- Players can send messages to each other by clicking “send” button at any time in duel mode.
- Players can receive messages at any time in duel mode.
- Players should lose health points if hurt from bombs or weapons.
- Players will win if all AI enemies are eliminated within the time limit in single-player mode.
- Players will lose if all AI enemies are not eliminated within the time limit within single-player mode.
- Players will lose if they are eliminated by AI enemies.
- This should bring up a popup window to show game fails if players lose.
- This should bring up a popup window to show game succeed if players win.
- Players can choose to restart the game or return to the main screen in popup game failure window.
- Players can only choose to return to the main screen in popup game victory window.
- The game should be a tie if two players still survive within the time limit in duel mode.
- Players can choose game difficulties, game modes by clicking corresponding buttons in main game menu.
- Players can check game rules or help by clicking corresponding buttons in main game menu.
- Players can slide left and right to adjust volume of background music or gameplay sound effects in settings screen.
- Players can turn on and off gameplay sound effects or background music in settings screen.
- Players can create a room or join a room by clicking corresponding buttons in screen of duel mode.

3.2.4 Game Props

3.2.4.1 Non-weapons props

- Getting a medical kit can restore one Health point.
- Getting a pair of running shoes can increase movement speed.
- Getting a glass of Nitroglycerine solution can increase explosion range of placed bombs by one grid in four directions respectively(up,down,left,right).
- Getting a riot shield can withstand one blast damage from bombs, bazooka, hand grenades, landmines.

3.2.4.2 Weapons

- Getting a bazooka allows characters to launch two rockets.
- Pressing numeric key 1 can launch a rocket by using a bazooka.
- Launched rockets can not fly beyond the boundaries of the map.
- Pressing numeric key 3 can plant a landmine which is invisible after three seconds.
- Landmines can only be placed on empty grids.
- Grenades can only be thrown when there is an obstacle on the grid in front of the character.
- Pressing numeric key 4 can throw a hand grenade behind an obstacle and causes a explosion of 3 by 3 grids on the third grid ahead.
- Getting a bomb package can add an additional placed bomb in three seconds.
- Pressing numeric key 2 to push the bomb out in three seconds by using gloves.

3.2.5 Game panel

- Game panel should count and display the health points of the players and AI in real-time.
- Game panel should count and display the acquired game props of players in real-time.
- Game panel should record and display the countdown of game in real time.
- Game panel can display the chat transcript between two players in duel mode.

3.2.6 AI

- AI must navigate around obstacles to move.
- AI can place bombs.
- AI should evade the blast radius of placed bombs.
- AI should not pass through obstacles.
- AI can find paths to treasure chests.
- AI can look for dropped game props.
- AI cannot pick up any game props in easy difficulty.
- AI can pick up non-weapon props in medium difficulty.
- AI can use all game props in the nightmare difficulty.
- AI should use weapons without hurting themselves in the nightmare difficulty.
- AI can be killed by bombs or weapons.
- AI should lose health points if hurts from bombs or weapons.
- AI can win or lose this game.

3.2.7 Networking

- Building a server by creating a room.
- The client can search for servers by typing IP address in the local area network.
- The server should accept the requests from the client.
- Establish a connection between the server and client by joining the room.
- Game data must be transmitted between the server and client normally.
- Game progress between the client and server must be synchronized and updated in real time.
- Messages sent and received between the client and server must be synchronized at all times.
- This should bring up a pop-up window which shows offline state if the server or client leaves the game.

3.3 Non-functional requirements

We have already covered functional requirements of software from different sub-components, now I will present the non-functional requirements that describe the specification that how this game behaves or operates instead of defining specific functionalities, which is totally different from functional requirements.

- **Compatibility: This game must be compatible and run on multiple platforms which have different environments, including Windows, macOS, Android, Linux, iOS, WebGL.** Developing this game by using LibGDX which is a cross-platform game framework that has the characteristic of strong compatibility, it provides a single and unified API for many platforms, it means that I only need to carry out some deployment configuration and define launchers for these different platforms so that I can publish the code on these platforms after compiling on desktop without writing codes for specific platforms.

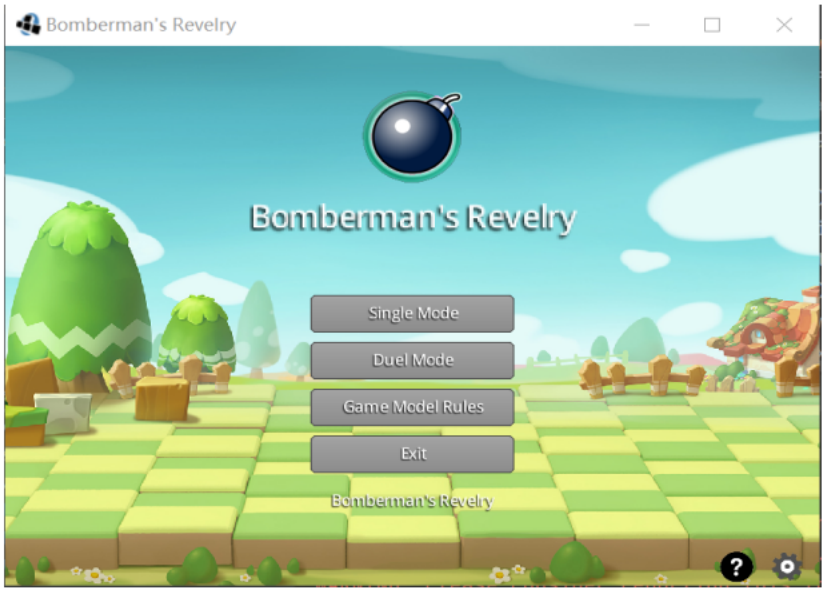
- **Usability:**
  1. The user interface should be intuitive and easy for operation. Users can take less time to perform this series of operations successfully by clicking corresponding buttons in the game interface according to their own ideas: check the game help, adjust the volume of the game, select the game mode and difficulty, pause and continue the game, create or join the room, send messages. User interface could be easy to learn and allow users to perform certain specific operations while they are browsing the user interface without having to go through a complex learning processes such as training or tutorials, which ensures users satisfied with the game.
  2. Game interface should be loaded quickly. The game resources should be loaded rapidly once the players start playing the game, including images, game elements, the game panel. Players will not lose patience and interest while waiting for the game starts.

- **Performance: The connection time between the server and client should not exceed 4 seconds in total when the network is normal in online mode.** The time for creating a server should not exceed one second, the response time for a client to search for a server in the local area network should not exceed 1.5 seconds, and the time for a server to accept the request from a client should not exceed 1.5 seconds. Waiting for a connection should not be too long to affect the experience of both players in online mode.

3.4 User Interface Design

The interaction between players and computers is very important, which determines whether users can use the game system properly.
Now I will demonstrate the design of the user interfaces of the game. After launching

the game, the main screen of the game will be shown first and can be seen below:



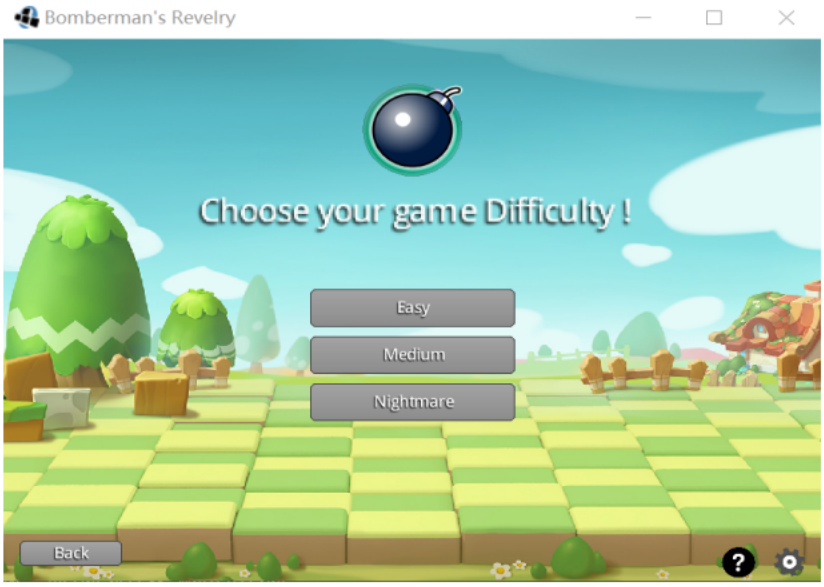
Screenshot 1 : Main game menu

The main game menu includes 3 types of basic operations:

1. Selection of different game modes to start playing game
    - Single mode
    - Duel mode
2. View game help and rules
    - Game controls
    - Game props description
    - Game rules
3. Audio management
    - Turn on or off the background music and gameplay sound effects
    - Adjust the volume of background music and gameplay sound effects

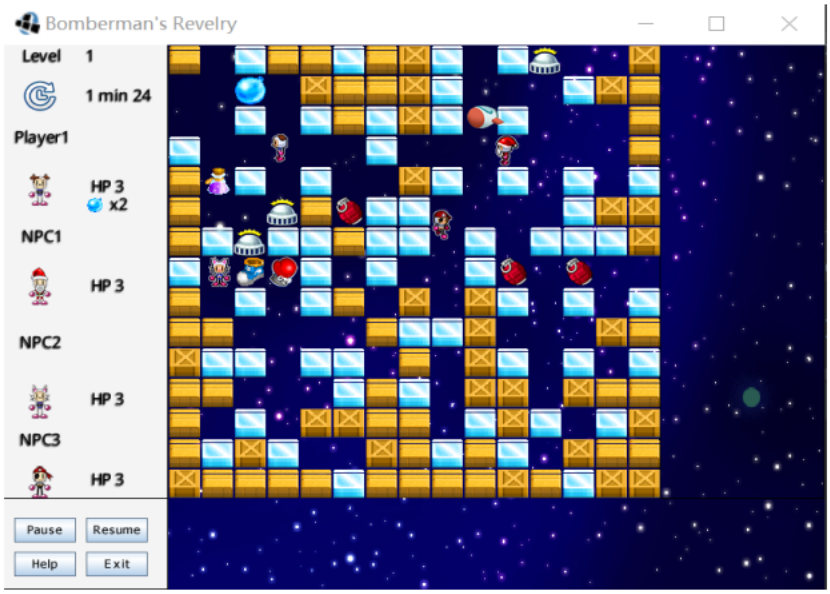
I will present and describe game interfaces of these three operations to you in turn, start from the first operation which is the selection of game modes, players are allowed to choose single-player mode and duel mode(two -player mode):

- Choose the single-player mode: there are three optional game difficulties for players to choose in single-player mode, they are easy, medium and nightmare. The difficulty selection screen is shown below:



Screenshot 2: Difficulty Selection Screen

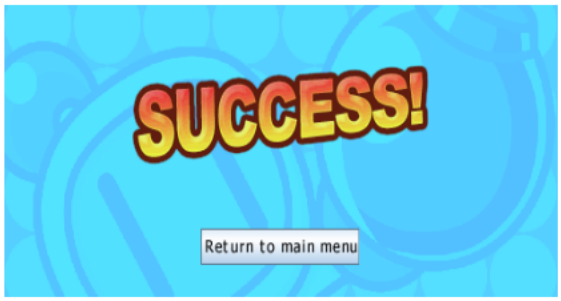
The game screen will be loaded and displayed after selecting one of the game's difficulties, as we can see the screenshot 3 below:



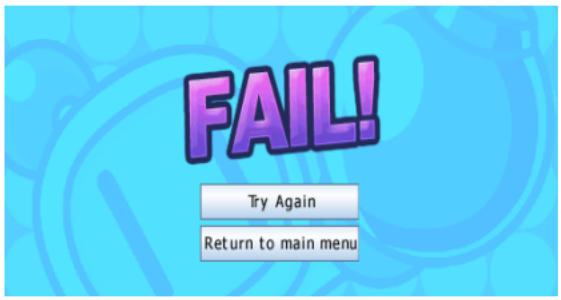
Screenshot 3: Game screen in singe-player mode

From the screenshot 3, the top of the left game panel counts the remaining time that the game ends, followed by the remaining health points of the player and other three enemies AI. In addition, the game panel is added a extra functionality of recording what game props the player has picked up and its remaining number so that players can know which props have not used used and allow them to use the props strategically. The bottom of panel allows players to perform some conventional operations during playing game, players can pause or resume the game, view game help and exit the current game at any time.

In terms of the screens of game finale, a victory or defeat window will pop up to prompt the player wins or loses this game when the victory or defeat condition is met, as we can see the two screenshots below:



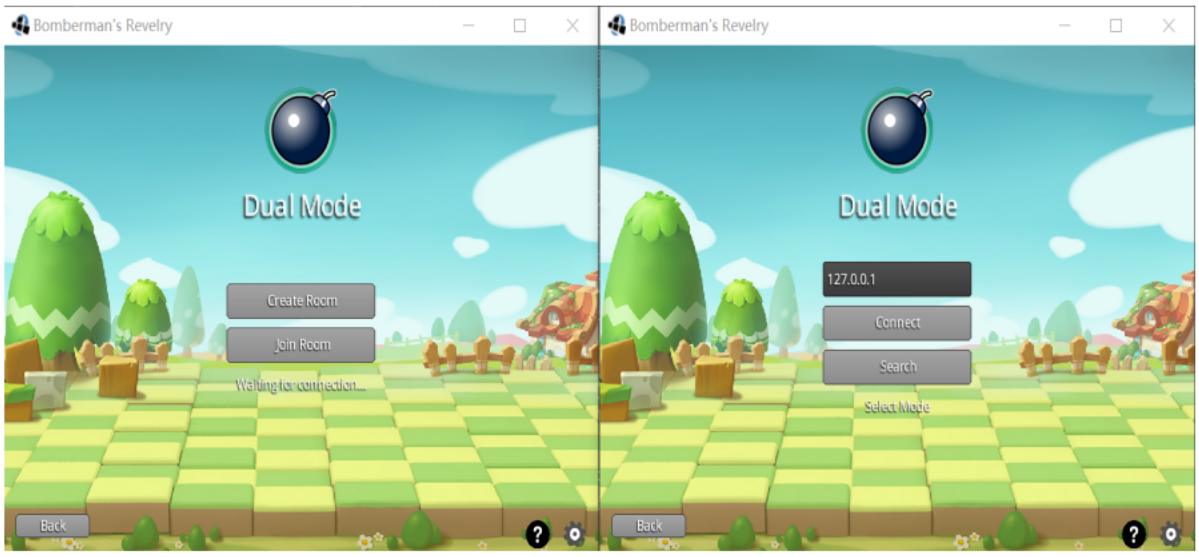
Screenshot 4: The victory window



Screenshot 5: The defeat window

The game ends and players will be directed back to the main screen when they win this game, players can choose to start the game again directly by clicking the corresponding button if they lost this game so that players do not need to return to main menu and then select the same difficulty of the same mode as in the previous round, which will save time and effort for players.

- Choose the duel mode: One of two players can choose to create or join a room



Screenshot 6: The screen of connection in duel mode

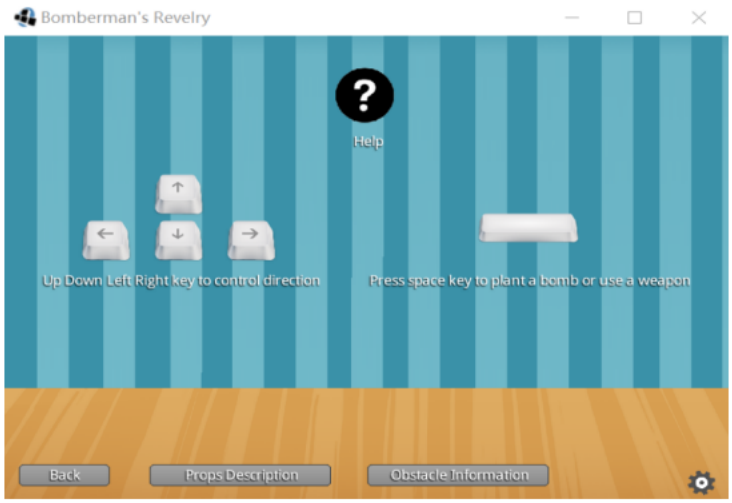
As we can see from screenshot 6, the player who creates a room will represent a server and need to wait for another player who represents a client joining the room to establish a connection in duel mode. In this process, the player who represents the client needs to type the IP address of the host server which is usually 127.0.0.1 by default. It also provided the functionality of searching the IP address of the adjacent servers in the same local area network.

The online game interface will appear when two players have successfully connected:

Screenshot 7: The game screen of duel mode

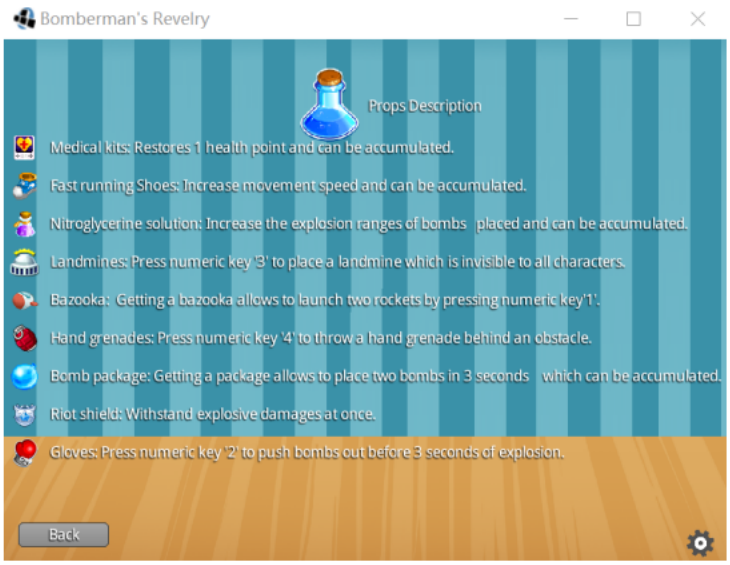
The right game panel recorded the remaining time in the game, the opponent's health points, and information about props usage. There is an additional functionality that allows two players to communicate with each other by sending messages, the left panel records and displays the chat transcripts between two players, which increases the interactivity significantly.

The second operation allows players to view and learn about the game controls, props description and game rules to ensure them know how to play this game.



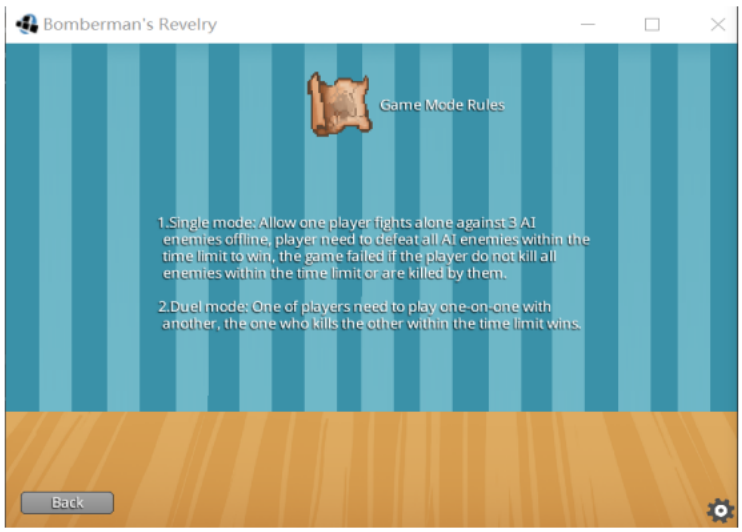
Screenshot 8: The screen of Game control specification

From screenshot 8, we can see the instructions for manipulating the bomberman to move around in four directions(up,down,left,right) and place bombs.



Screenshot 9: The screen of game control specification

The screenshot 9 describes the information about introduction of game props and usage in details, each prop has its own features and can bring special effects to the players, which allows players to use props strategically.

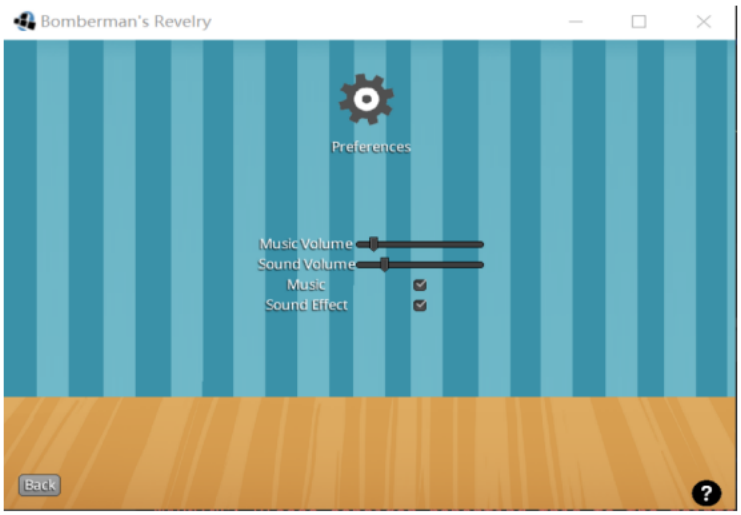


Screenshot 10: The screen of game rules

The screenshot 10 describes the game rules of two different modes, players need to

learn about these rules to win this game.

The last operation is audio management, as we can see from the screenshot 11 below, players are allowed adjust volume of background music or gameplay sound effects by sliding left and right in the preference screen, they can also turn on and off gameplay sound effects or background music in the preference screen.



Screenshot 11: The screen of game audio settings

Finally, I will model the game system to make it visualize and formulate its structure in the next chapter after describing the game features, functions and the interface design of the game.

4 Problem Analysis and solution design

4.1 Software Architecture

Software architecture provides a framework for developing a software to deal with its complexity and huge size, it could define a structured solution for my developed game to meet all functional, non-functional, technical and operational requirements.[13]
The organization of the code will be chaotic and messy and really hard to manage code in my game without using any architectures.Therefore, choosing a good and suitable architectural pattern is very important for me, it represented my design

decision of the overall structure of this game, which will determine whether it is a successful game significantly. Therefore, in this chapter I will illustrate the architectural pattern used in my game system and describe the logical organization of every sub-component, how they work with each other within the whole system.

An architectural pattern is a general, reusable solution to a commonly occurring problem in the software architecture within a given context.[14] I applied MVC in my overall game architecture, it is one of the architectural patterns, which divided the game system into three parts, including model, view and controller. These modules performed different tasks respectively, I will present the structure and describe how they work together via these two diagrams below:

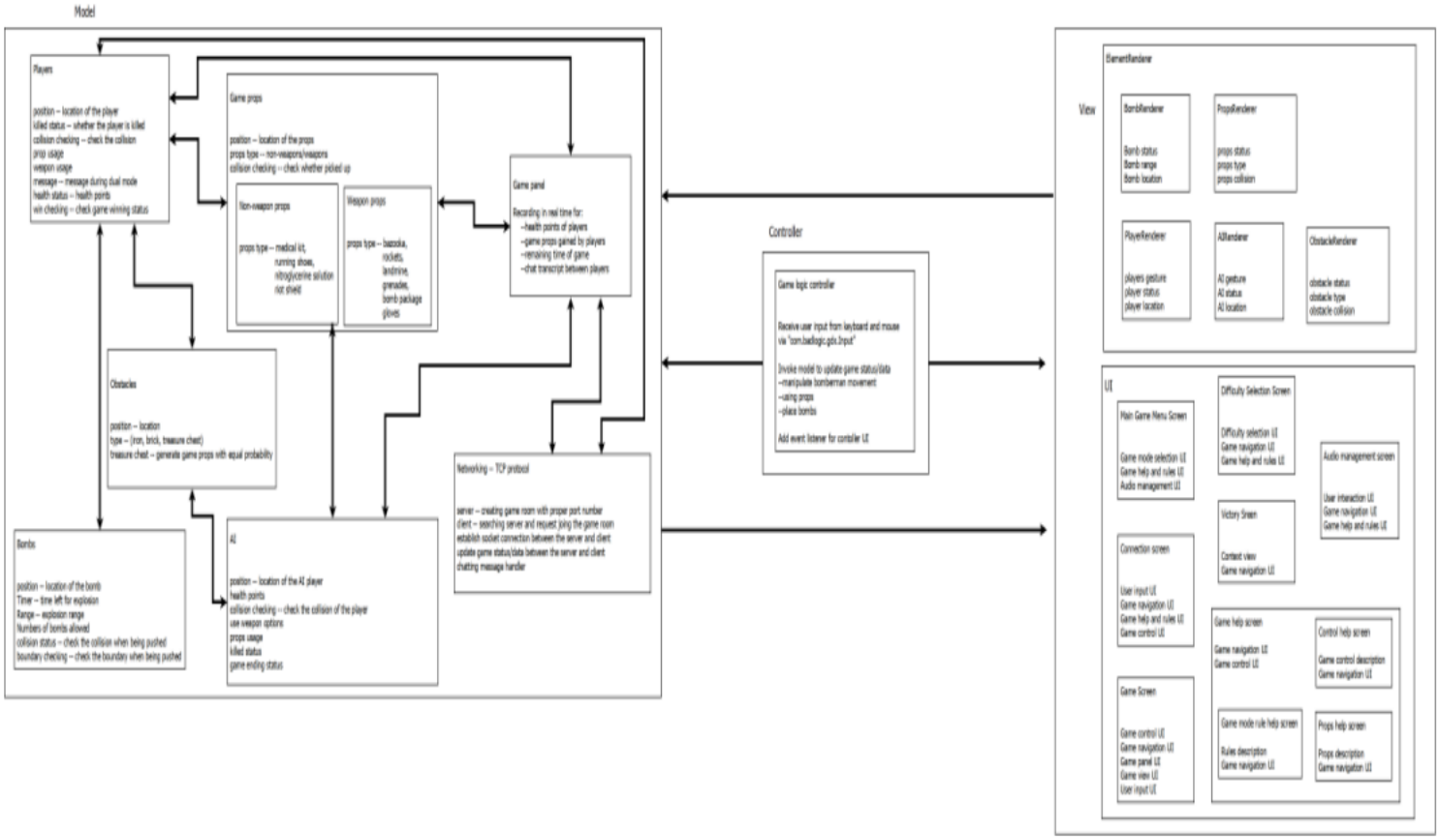


Figure 1: MVC architectural pattern in my game

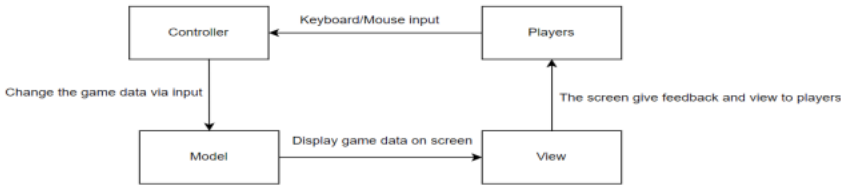


Figure 2: Work flow of MVC

The controller used to handle all user inputs from keyboard and mouse in real time by adding events listener, then translate them and change game data in model, followed by displaying on screen via the view module, for example, players use the mouse to click the buttons on user interfaces and manipulate characters to move around, use props or place bombs on the screen by pressing corresponding keys.
Model is used to manage logic and data of my game, it will transmit the game data to view module and display on screen when it was changed. It not only defined all game elements and their attributes, including props, characters, bombs, obstacles, but also illustrated how they related or affected to each other. For example, the brick is destroyed and players lost health points due to the explosion of bombs. In addition, it defined all functionalities that covered in game, I give some examples: 1. Game panel used to record information about health points, game props and remaining time. 2. The network module used to realize the online battle of two players and the chatting room.
The view can be used to display the game on screen, which refers to the user interfaces essentially. It is the part users to interact with, it presented the updated content on the screen when model's data is changed. For example, the game screen will freeze rapidly if players click the pause buttons during playing game.

MVC provided many benefits for me develop this game. Firstly, it separated my code into three modules and it will be structured and modular, which means that it will be very easy to manage and revise. I can easily find the corresponding code in specific module in order to revise if there are some bugs. For example, there must be a problem with the players' attributes defined in the model module if the player moves too fast. Secondly, my code will have good extensibility and convenient to add new features. Each module just needs to perform specific tasks with no interference each other, for example, the view just need to handle the display of the game so that I just only to create a game help screen in this module if I wanted to add a new user interface to introduce game rules, it will not affect the overall structure. Thirdly, it makes the code become readable and understandable so that it can facilitate game development and improve efficiency significantly.

Libgdx framework integrates several common modules to provide services for each step of building game architectures. Below I will describe the core modules and class

libraries that I used in my game development provided by libGDX framework.

4.2 Core Sub-module

4.2.1 Graphics

The graphic module in libGDX provided a set of libraries for graphic development, OpenGL API allows to draw images on screen. Texture class can be used to decode an image which has PNG format and loading it into GPU memory by importing the library "com.badlogic.gdx.graphics.Texture", which can be regarded as an instantiated Texture object to represent an image. After using the texture, it is necessary to use Texture.dispose() method to release the memory, otherwise it will lead to the leakage of memories. AssetManager class is responsible for loading multiple types of resources, including texture, music, sound effects, fonts and so on. I put all image resources, including background, characters, obstacles, game props,explosion into the assets folder so that I can use the method assetManager.load(fileName, type) provided by AssetManager class to load my all assets given the type of assets and file names. It is very convenient to use AssetManager API to manage and load my all types of resources uniformly, which saves me much time and effort.

```
private static <T> void loadResource(String name, String fileName, Class<T> type) {
    assetManager.load(fileName, type);
    assetNames.put(name, fileName);
}
```

```
loadResource( name: "door",  fileName: "bricks/door.png", Texture.class);
loadResource( name: "brick",  fileName: "bricks/brick.png", Texture.class);
loadResource( name: "broken_brick_animation",  fileName: "bricks/broken_brick_animation.png", Texture.class);
loadResource( name: "chest",  fileName: "bricks/chest.png", Texture.class);
loadResource( name: "iron",  fileName: "bricks/iron.png", Texture.class);
```

For example, loading a picture "brick" which has PNG format with as a texture asset by calling the method assetManager.load("player/bomb_idle.png", Texture.class).
These instances of background pictures can be got by creating texture objects after loading these picture resources by calling assetManager.get(asset name, Texture.class). In order to facilitate the development of the game interfaces, libGDX framework provided many common widgets which can be used by importing the library "com.badlogic.gdx.scenes.scene2d.ui", they extended the class of actor and have all attributes and functions so that I do not have to take time to define Myactor. Image class provided scale operations of images after getting the texture objects, for example,

I use setSize() method to adjust the size of background picture so that I can add it to stage to display on the game screen.

```
public static <T> T get (String assetName, Class<T> type) {
    return assetManager.get(assetNames.get(assetName), type);
}
```

```
private void createUI() {
    background = ResourceManager.get( assetName: "tutorial_bg1", Texture.class);
    background = new Texture(Gdx.files.internal("background/white.png"));

    Image backgroundImage = new Image(background);
    backgroundImage.setSize(Constants.SCREEN_WIDTH, Constants.SCREEN_HEIGHT);
    stage.addActor(backgroundImage);
```

The libGDX also provided animation class to show the movement of characters, unleash of weapon props, explosion effects. In background I have already described the basic concepts of animation and how it works, so I will talk about how to create an animation to display on the screen by using this class. Animation is created by defining Animation <T>, where T represented generic type of frames in animation and I use TextureRegion as objects of animation frames, it need to be provided two parameters to create an animation instance: 1. The list of images with the specific type, which also called key frames 2. Time interval between playing each frame. Here are the operations to form a 2D animation of the bomberman is moving right.
Firstly, loading a sprite sheet which has the class of texture to show each frame in this animation. Secondly, split this sprite sheet into small unit cells with specified width and height, then store them in a 2D array, which has the type of TextureRegion. Thirdly, put all unit cells stored in a 2D array into a 1D array as key frames to be played for creating an animation. At last, create the animation instance by creating the constructor with these key frames and the time for playing each frame 0.15 seconds so that this animation will be displayed on the screen. The animations of unleashing weapons or explosion effects are created by using the same technique.

```
Texture spriteSheet = ResourceManager.get( assetName: fileNamePrefix + "_right_animation", Texture.class);
TextureRegion[][] frames = TextureRegion.split(spriteSheet,  tileWidth: spriteSheet.getWidth() / FRAMES, spriteSheet.getHeight());
TextureRegion[] rightFrames = new TextureRegion[FRAMES];
for (int i = 0; i < FRAMES; i++) {
    rightFrames[i] = frames[0][i];
}
rightAnimation = new Animation<TextureRegion>( frameDuration: 0.15f, rightFrames);
```

4.2.2 Audio

The audio module in libGDX provided the management and handling of audio resources, including music and sound effects, which supports three different formats mp3, ogg, wav. Music class is responsible for dealing with longer pieces of background music and the sound class usually handles the shorter pieces of a variety of sound effects(explosion, place bombs, use game props) in game, including playing,

pausing and replaying. The process of loading audio resources is the same as load pictures resources, firstly I put all audio files into assets folder, assetManager class can also be used to read and load these audio resources by creating corresponding instances:

```
loadResource( name: "placeBomb", fileName: "sounds/placeBomb.wav", Sound.class);
loadResource( name: "explore", fileName: "sounds/explore.wav", Sound.class);
loadResource( name: "placeMine", fileName: "sounds/placeMine.mp3", Sound.class);

loadResource( name: "bg_main_001", fileName: "musics/bg_main_001.wav", Music.class);
loadResource( name: "bg_ad_001", fileName: "musics/bg_ad_001.wav", Music.class);
loadResource( name: "victory", fileName: "musics/victory.wav", Music.class);
loadResource( name: "lose", fileName: "musics/lose.wav", Music.class);
```

These instances can be gain by creating music and sound objects after loading audio resources by calling assetManager.get(asset name, type).

```
public static <T> T get (String assetName, Class<T> type) {
    return assetManager.get(assetNames.get(assetName), type);
}
```

```
mainMenuTheme = ResourceManager.get( assetName: "bg_main_001", Music.class);
gameTheme = ResourceManager.get( assetName: "bg_ad_001", Music.class);

victory = ResourceManager.get( assetName: "victory", Music.class);
lose = ResourceManager.get( assetName: "lose", Music.class);

placeBomb = ResourceManager.get( assetName: "placeBomb", Sound.class);
placeMine = ResourceManager.get( assetName: "placeMine", Sound.class);
explore = ResourceManager.get( assetName: "explore", Sound.class);
```

All resources managed by the assetManager need to suspend the loading of the resource and save the loading status through the assetManager.update() method:

```
public static boolean update() { return assetManager.update(); }
```

After loading all audio resources, the music.play() and music.stop() method used to play specific background music and stop playing when they were called manually, it allows to restart or resume the music if a music ended or was paused. The music.setVolume(float volume) allows to set the range of background music, music.setLooping(Boolean value) is used to play background music on a loop.

```
public void playMusic(String name, boolean looping) {
    if (musicEnabled) {
        Music music = musics.get(name);

        if (music != null && !music.isPlaying()) {
            music.setVolume(musicVolume);
            music.setLooping(looping);
            music.play();
        }
    }
}

public void stopMusic(String name) {
    Music music = musics.get(name);

    if (music != null) {
        music.stop();
    }
}
```

4.2.3 Input

In my desktop game development, libGDX provided input module to handle the inputs of keyboard and mouse. There are several methods provided by GDX.input to give responses from user inputs, which judges whether or not the user clicks by using a mouse or presses a key from a keyboard. For example, the method isKeyPressed(keys.SPACE) returns a boolean value to represent whether players press the space key, the character manipulated by the player will place a bomb if the value is true.

```
if (Gdx.input.isKeyPressed(Keys.SPACE)) {
    spriteManager.player.placeBomb(Bomberman.NetworkType.LOCAL);

    spriteManager.computePowerupRate();
```

In this case, I use addListener()add a event listener to the setting button by importing "com.badlogic.gdx.scenes.scene2d.Actor", it converted the inputs to another different events after the listener receiving the specific event. The ClickListener() and touchUP() method are called while players clicking the setting button by clicking the left mouse button, then the game screen will be switched to the preference screen so that users are allowed to adjust the volume of background music and sound effects without exiting the current game.

```
settingsButton.setPosition( x: Constants.SCREEN_WIDTH - 50.0f,  y: 0.0f);
settingsButton.setSize( width: 40.0f,  height: 40.0f);

settingsButton.addListener(new ClickListener() {
    public void touchUp(InputEvent event, float x, float y, int pointer, int button) {
        dispose();
        game.changeScreen(Bomberman.PREFERENCES);
    }
});
```

4.3 Core Sub-systems

4.3.1 AI system

AI is one of the most critical parts in my game, a clever enemy can be created by applying data structure algorithms, which makes the player feel like it is manipulated by an another player. The intelligent AI agents will play a important role in enhancing game experiences of players, they will constantly look for treasure boxes in order to collect more game props to enhance their attributes and use weapons to attack players. In this process, they can also avoid hurting from the explosion of placed bombs and use weapons props in a rational way, players need to battle against with them and kill all enemies to win this game in offline mode, which makes the game become more challenging and fun.

4.3.1.1 Path-finding : Breadth-first search algorithm

AI could often perform specific commands through a variety of search algorithms in game. In my game, AI enemies carried out the tasks of searching for game props or treasure boxes by applying Breadth-first search algorithm, it starts from an initial node and then search for all adjacent nodes at the same depth before extending to the next depth, this algorithm will keep searching nodes until the target node is found and return a path solution for AI to move towards to the target node, it ends if there are no another nodes to extend. This path-finding problem can be formulated as four parts:

Initial state : the current coordinates of AI.
Search space: all coordinates of null grids.
Target node : The coordinates of destructible obstacles or game props.
Path solution: array of coordinates.

AI starts with locating the current coordinates as an initial node, then exploring all the neighbourhood coordinates nodes nearby the initial node in order of right, down, left, up. After that, extend all these neighbourhood nodes and repeat this step until an indestructible obstacle is explored, then stop extending the current node and start to extend the neighbourhood nodes in other three directions, and so on until the a destructible obstacle or game prop is found. At last, returns an array of all coordinates that can reach the target node as a path solution for AI so that they can move towards to the target position to place a bomb or pick up game props. In my design of game

map, the placement of each obstacle in this maze is very reasonable, players or AI will not surround by four indestructible obstacles in four directions and get stuck in specific area. Therefore, it would never happen that no more nodes can be extended and the algorithm can not find a path solution unless there are no destructible obstacles or game props anymore.The reason why I use BFS algorithm is that this algorithm is suitable for finding the shortest path from the origin to the target node because it considers all adjacent nodes nearby the previous node, which ensures AI can take less time or costs to find the shortest path to destructible obstacles from current position and collect game props rapidly to strengthen their attributes or prepare to use weapons to attack players. It will be really hard to defeat AI when they collect the enough game props.

As we can see the pseudo-code below, BFS applied the data structure of queue to store and explore coordinates information, the new coordinates nodes will be inserted from the end of queue, the nodes that have been extended will be removed from the start of the queue, there is an array recorded whether the current coordinates node has been visited before, it will not be inserted into the queue to explore if the node was visited. This algorithm will end and return a path solution which is a path solution for this path-finding problem if the destructible obstacles or game props were found.

Pseudo-code for BFS algorithms

```
1. input: Given the map spriteManager.map, current coordinate(x0, y0), target coordinate(x1, y1)
2. output: path <- Array<Vector2>
3. Procedure: BFS(x0, y0, x1, y1):
4. Define queue <- Node array
5. Add the current node into the tail of the queue:
     queue[tail] := (x0, y0, parent, step)
6.   visited[x0][y0] = True
7.   tail := tail + 1
8.   let flag := 0 label for stop criteria
9.   While queue is not empty:    tail > head
10.       Loop: search four directions(up, right, down, left)
11.             if out of the map then continue; end if
12.             if the next step can be reached and is not visited then
13.               Set visited[tx][ty] := 1 where (tx, ty) denotes the step coordinate
14.                Add the the step to the tail of the queue:
                       queue[tail] := (tx, ty, parent, step + 1)
15.               Update the tail: tail = tail + 1
                End if
16.       if find the target (x1, y1) then set flag := 1;break;end if
17.       End Loop.
18.       if flag is equal to 1 then break; end if
19.       Dequeue: set head := head + 1
20.   End
```

4.3.1.2 Overall Work flow

AI also needs to perform other tasks after finding the path to the destructible obstacles or game props, they need to place a bomb nearby the destructible obstacles to destroy them in order to get game props dropped from treasure boxes. In this process, AI should avoid the explosion of placed bombs, which means that they have to find safe area that will not hurt from explosion and wait until the bombs exploded before keeping searching new paths to get game props. In this process, it would never be the situation that AI cannot find the safe area because the bomb exploded and spread in four directions in the shape of "cross" after it was placed, which means that no matter how large the explosion range of the placed bomb, AI just needs to move to the diagonal where the bomb is placed to bypass the explosion area. Even though the AI can only place bombs in the same direction in one path, I set the maximum blast radius of a bomb can be spread in one direction is 4 grids so that AI could have enough time to move forward more than 4 grids to keep away from the explosion area after they placing a bomb. To sum up, the state of AI will start from finding paths and change to placing bombs, then change to finding the safe area, followed by the waiting state. As we can see the state machine diagram below, it illustrates all states of AI system:

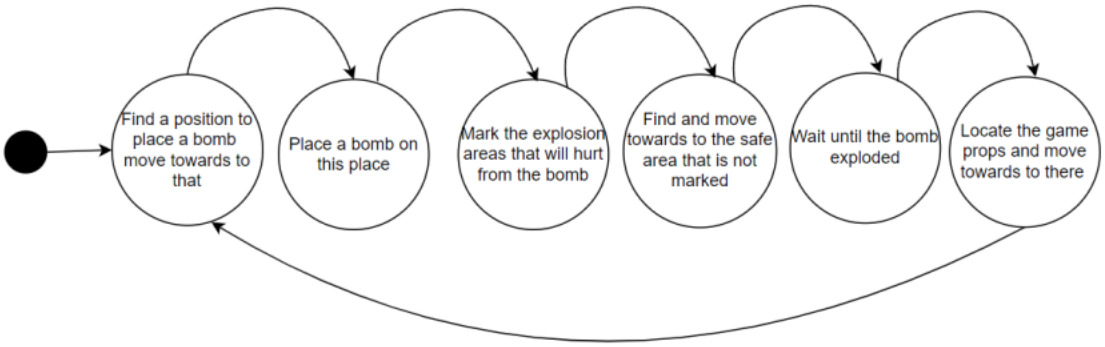


Figure 1 : State Machine Diagram

The activity diagram is a special case of state machines, it emphasized the sequence and concurrency of AI system's behaviour. AI will lose health points or die if they hurt from bombs or weapons props at any time while performing these tasks in order, this process is performed concurrently. As we can see the activity diagram below:

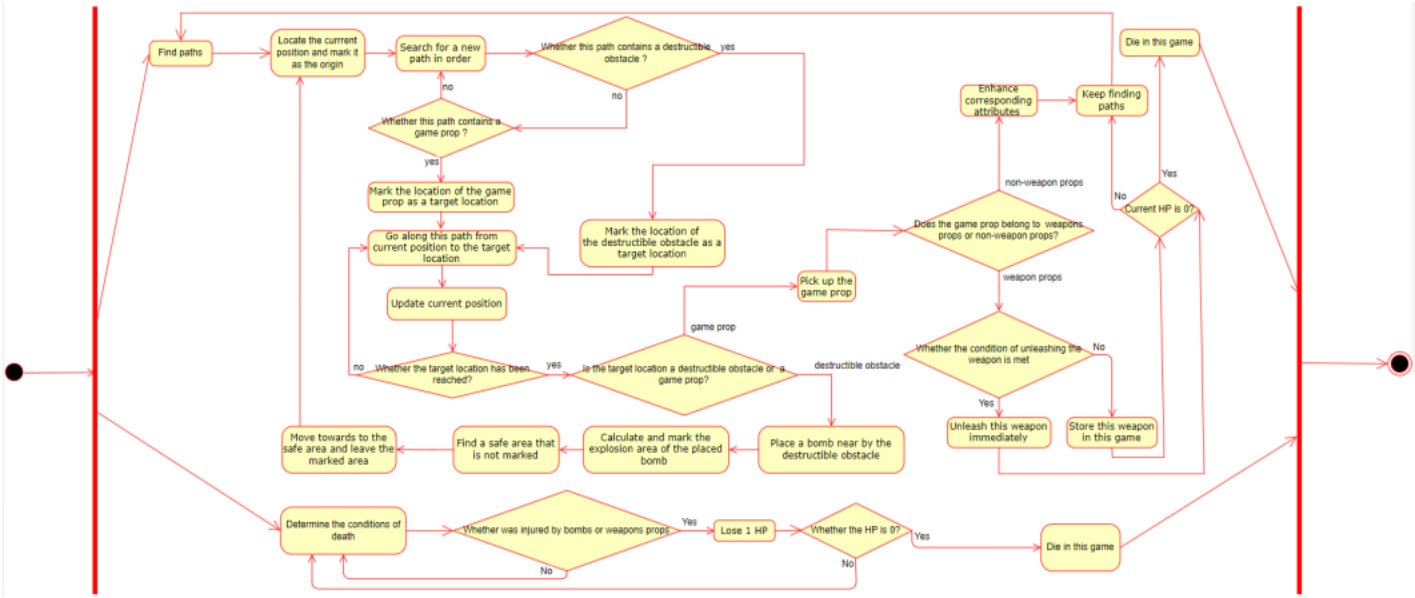


Figure 2: Activity diagram

This diagram presented the overall work flows of AI system, described their decision processes. Considering the problem that AI may hurt themselves while they using weapons props, I added the unleash condition of each prop for AI so that they can use all weapons in a rational way.

1. Bazooka: The explosion range of a rocket fired by the bazooka is 1 grid, AI can only fire a rocket when there are no obstacles on first two grids in front of them according to the movement speed of fired rockets and AI.
2. Landmines: AI could record the coordinates of the landmine after it was placed and bypass this area although it is invisible, they will move to the opposite direction of the placed landmines if there is only one path.
3. Bomb package: AI can only place multiple bombs nearby destructible obstacles, they will move to the diagonal of the bombs after they placing more than one single bombs in one direction to prevent from hurting from explosion.
4. Gloves: AI can only push the bombs when there are no obstacles on the first 4 grids in front of them because the maximum blast radius of a bomb can be expanded in one direction is 4 grids.
5. Hand grenades: The explosion range of a hand grenade is 9 grids centered on its landing point which is 3 by 3, so AI must throw a grenade on the second grid so that they will not hurt themselves. Moreover, they can only throw the grenade when there are no obstacles on the second grid.

AI will only unleash these weapons when specific conditions are met, which makes AI harder to deal with. This game will become more challenging and interesting.

4.3.2 Networking system

4.3.2.1 Network Architecture : Peer to Peer architecture

I applied the peer to peer architecture in network parts of my game, each of the peers or nodes have the exactly same responsibilities and can both act as a server to provided services or client to send requests. The player who created a room will represent a server to listen and accept incoming requests, so the another player will act as a client to send the request of joining the room to the server, the game will start when the request was accepted. The mechanism is very flexible that the player does not have to play specific roles before every game starts.
The main reason why I use this architecture is its high reliability, the client will not be affected and can still work normally if the server crashes in online mode, which ensured players have a good user experience when they overcome unstable network and disconnect from the internet. As we can see the architecture diagram:

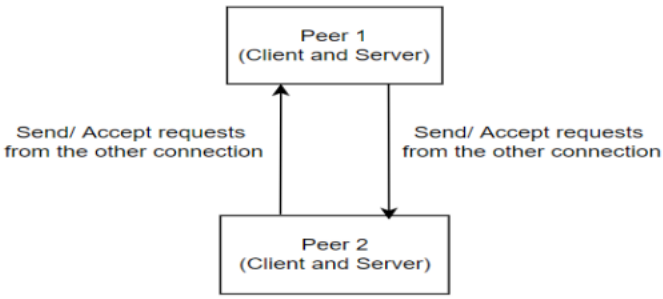


Figure 4: Peer-to-Peer architecture diagram

4.3.2.2 Network Communication Framework: The Battle of Two Players

The two-players mode in my game support two players battle with each other on the same local area network. I used the network communication framework kryoNet provided by java libraries, it based on the TCP and UDP connection protocols and is very suitable for network parts of game which applied client/server and peer-to-peer architectures because it allows to serialize object graphs and convert to format of strings in order to store in memory and then transmit game data between the server and client rapidly, which ensured the game state or status between two players can be synchronized. Let me talk about how to implement this online mode in my game.

Firstly, if one of players would like to create a room in two-player mode, it will establish a server given the TCP port number of 8888 and UDP port number of 7777

by importing "com.esotericsoftware.kryonet.Server" and creating a server object, there will be an I/O exception if it was not created.

```
game.server = new Server();

try {
    game.server.bind( tcpPort: 8888,  udpPort: 7777);
} catch (IOException e) {
    e.printStackTrace();
}

Kryo kryo = game.server.getKryo();
kryo.register(MessageProtocol.class);
kryo.register(EnterProtocol.class);
kryo.register(ConnectProtocol.class);
kryo.register(MoveProtocol.class);
kryo.register(PlaceBombProtocol.class);
kryo.register(LaunchRocketProtocol.class);
kryo.register(ThrowGrenadeProtocol.class);
kryo.register(PlaceMineProtocol.class);
kryo.register(PauseGameProtocol.class);
kryo.register(ExitGameProtocol.class);
kryo.register(Vector2.class);

game.server.start();
```

Secondly, the another player should join this room via creating a client by importing "com.esotericsoftware.kryonet.Client" and creating a client object.
It is necessary to got the Kryo instances of the server and the client respectively, then registered these protocols classes used for transmitting data to each other before starting the server and client. It should be noted that protocol classes in client and the server must registered in the same order.

```
game.client = new Client();

Kryo kryo = game.client.getKryo();
kryo.register(MessageProtocol.class);
kryo.register(EnterProtocol.class);
kryo.register(ConnectProtocol.class);
kryo.register(MoveProtocol.class);
kryo.register(PlaceBombProtocol.class);
kryo.register(LaunchRocketProtocol.class);
kryo.register(ThrowGrenadeProtocol.class);
kryo.register(PlaceMineProtocol.class);
kryo.register(PauseGameProtocol.class);
kryo.register(ExitGameProtocol.class);
kryo.register(Vector2.class);

game.client.start();
```

Thirdly, making the server can receive and deal with the requests sent from the client by adding an event listener to it.

```
game.server.addListener(new Listener() {
    public void received(Connection connection, Object object) {
        if (object instanceof ConnectProtocol) {
            gameReady = true;
            game.clientConnectionId = connection.getID();
            System.out.println(connection.getEndPoint());
            game.server.removeListener(this);
        }
    }
});
```

Fourthly, Establishing the connection between the client the server via the port numbers bond by the server after typing the IP address of the host server. I set the maximum time out for the connection is 1000 milliseconds, there will be an I/O exception to prompt the connection failed if connection between them was time out.

```
connect.addListener(ChangeListener) (event, actor) -> {

            try {
                game.client.connect( timeout: 1000, ipAddress.getText(), tcpPort: 8888, udpPort: 7777);
                game.networkType = Bomberman.NetworkType.REMOTE;
                gameReady = true;
            } catch (IOException e) {
                e.printStackTrace();
            }
});
```

Finally, the client will send the connection protocol objects to the server via a TCP when they were ready to start the game by using the sendTCP() method. The game screen will be shown when the server accepted the requests.

```
Timer.schedule(() -> {
        if (gameReady) {
            ConnectProtocol connectProtocol = new ConnectProtocol();
            game.client.sendTCP(connectProtocol);
            Timer.schedule(() -> {
                    game.changeScreen(Bomberman.GAME);
                    cancel();
            }, delaySeconds: 1.0f);
            cancel();
        }
}, delaySeconds: 0.0f, intervalSeconds: 1.0f);
```

4.3.2.3 Network techniques : The functionality of chatting room

In terms of the implementation of chatting room in two-players mode, I used the network module provided by libgdx and muti-thread technique to deal with messages. These are networking operations to implement this functionality:

1. Creating a server socket used to receive and accept all requests sent from a client given the specific port number via TCP.
2. Creating a client socket used to establish the connection with the server given the IP address of the host server and the specific port number via TCP.
3. Create the input stream and the output stream to read data and write data via the sockets, which is the process of sending or receiving messages.

It is necessary to create two threads to process real-time messages receiving and sending between the server and client, because every thread can represent a sequential work flow and perform their specific tasks concurrently so that it will improve response time and efficiency for receiving messages from the other connection, which ensures that messages can be displayed synchronously and rapidly on their screen when they send messages to each other. In addition, each thread will run independently and not interfere with each other, players can chat with each other at any time without affecting the update and synchronization of game state, which ensured they can interact with each other smoothly. That is why I use multi-thread technique to implement this process.

I draw a sequence diagram to clearly describe the whole process of network connection in two-players mode, which emphasized the sequence in which objects interacts on a network, it is shown below:

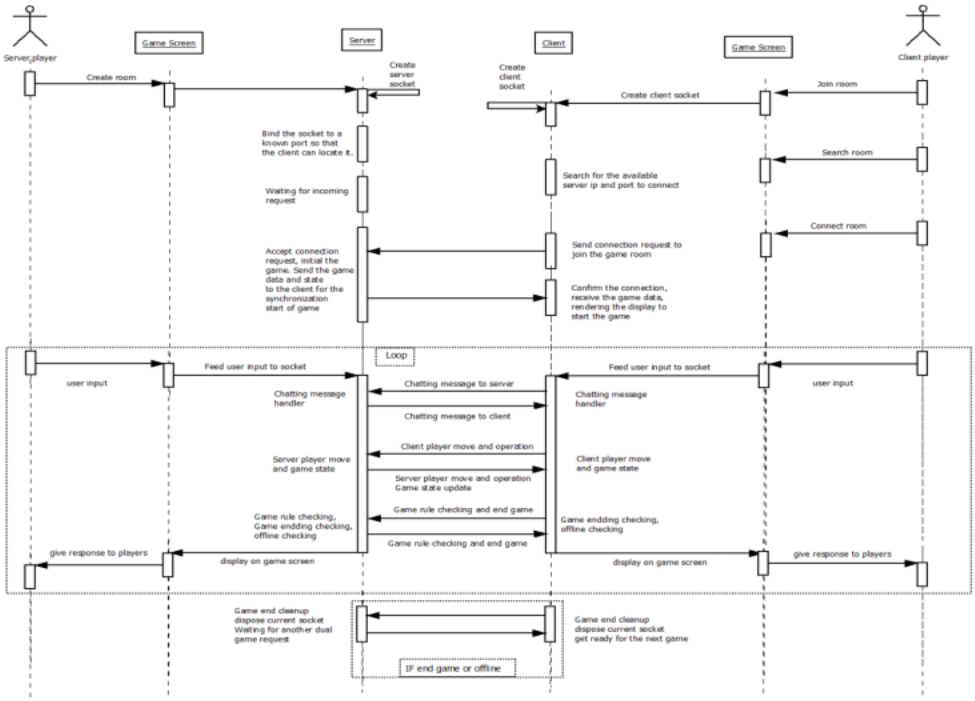


Figure 5: Sequence diagram

The reason why I applied TCP in network is that it is a suitable protocol to implement the small-scale online game, especially for the battle of two-players and functionality of chatting room in my game. It can segment large data into appropriate data blocks or packets so that they can be delivered in network, there is a very small probability that packets will be lost and leads to the network disruption in online mode, which ensured the game data and messages can be transmitted stably and completely between the server and client. Therefore, it is very reliable and provides a good game experience for players.

5 Verification and Evaluation

Testing can verify my game can work as expect and meet my requirements, whether all functionalities were implemented, it helped me to identify and find the potential problems or bugs that I need to revise in game, which guaranteed game quality and reliability. Evaluation helped me review of what was good and needed to be improved during the game development, some deviations from the initial plans, which accumulated work experience for the career future. I will describe test strategies used in game and how it works below, followed by the evaluation of my project.

5.1 Software Testing

5.1.1 Black-box testing

Game system is regard as a black box that can not be opened while the test executing, I applied black-box testing to examine user interfaces and functionalities of the game, this test can help me check whether this game product can achieve these goals defined in the project proposal and identify the missing or wrong functionalities. The test is carried out from the player's point of view to check whether the user inputs from the keyboard and mouse matches the game system output so that the testers do not need to consider how the game's internal systems work, these test results output by black-box are very intuitive and it is easy for me to identify the potential problems existed in my game via the differences between inputs and outputs. For example, the game screen showed that the rocket launched by the player flies out of the boundary of the game map, there must be a lack of logical judgment of outside of the boundaries in the “rockets” class within the model module.

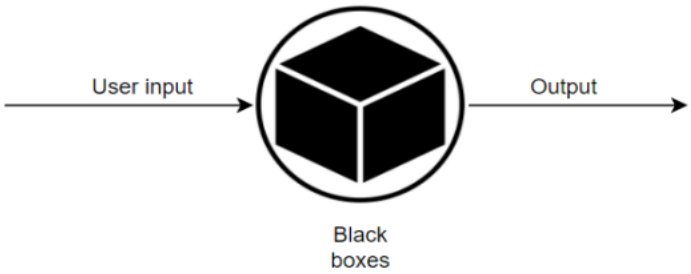


Figure 6: Black box testing

I defined test cases according to functional requirements that mentioned in design parts before I starting testing, then check whether each of them was passed during the test. Fortunately, all the test cases were passed after countless runs of testing and revisions of code by debugging, which ensured that this game was as good as I had expected. I will put the results for each test case in the appendix.

5.1.2 Usability Testing

Usability plays an indispensable role in game development, which largely determines the user experiences of this game, a glitch or problem in the game may bring poor user experiences and caused the players will lose interest or patience in this game, they may end up giving up playing this game no matter how attractive its gameplay and precept. The method of evaluating usability of a game is the same as other applications, which applied usability testing to evaluate how easy for real users to use this software product by observing and recording users' behaviors while they performing specific tasks allocated by researchers, it helped me to identify the problems or design defects existed in my game by analyzing these intuitive data feedback from users. It is also carried out from player's perspective so this method is the most effective and representative. Therefore, that is why I applied the usability testing in my game. I was both a game developer and an researcher for this test, so I will describe the specific design decisions made by me.

1. Find the specific testing participants

I want my game can still bring a good user experience for a each segment of audiences. Therefore, I invited 6 local friends who came from different age groups to participate in this test to play the prototype game specially, they will represent different target groups, 10-17, 18-25, 26-40, 41-54, 55-64, 65 or more respectively.

2. Designed and define the test tasks for participants to collect problems

I designed different tasks for them according to the functionalities of the game and they are expected to attempt to complete these tasks alone without asking help for me. I explained to them the rules of my game and how to play it before carrying out tests, which gives them an intuition impression about this game. I had to keep an eye on the their on-screen actions while the test was going on, I will ask why did it like this when they started performing tasks that go against expectations in order to learn about their thoughts and identify possible problems that need to be improved. I summarized a potential problem in total they faced during the test and putted it in appendix B.
The tasks are defined for different parts in my game below:

Tasks designed for testers

- **Audio:**

(1) Open the settings screen
(2) Turn off and on the background music
(3) Turn up and down the background music
(4) Turn off and on the sound effects in game
(5) Turn up and down the sound effects in game

- **Game help and instructions**

(1) Open the game help screen
(2) Open the props description screen
(3) Open the game rule screen
(4) Open the obstacle information screen

- **Game logic:**

(1) Choose easy level, medium level and nightmare level one-by-one in single-player mode.
(2) Manipulate the bomberman to move up, down, left, right
(3) Manipulate the bomberman to place a bomb near an obstacle
(4) Manipulate the bomberman to pick up props
(5) Manipulate the bomberman to launch a rocket by using a bazooka
(6) Manipulate the bomberman to plant a landmine
(7) Manipulate the bomberman to push a bomb by using gloves
(8) Manipulate the bomberman to place two bombs after getting a bomb package
(9) Manipulate the bomberman to throw a hand grenade behind an obstacle
(10) Pause and resume the game
(11) Check game help during game play
(12) Exit current game to return to main menu during game play
(13) Open the screen of duel mode
(14) Create a room to wait for connection in duel mode
(15) Search for IP address of the neighbour host servers in duel mode
(16) Type the IP address for searching in text field
(17) Join a room to start duel mode
(18) Type messages in text field and send them in duel mode

3. Evaluation of usability

After finishing the test, the system usability scale used to measure the usability of the my game product from 3 aspects, effectiveness, efficiency and satisfaction of users. It is a SUS questionnaire which included 10 questions for them to complete in order to get a direct and quick feedback. [15]*The evaluation method of system usability scale presented here is based on Andrew (2020).* They have no more than 2 minutes to

complete and the scores are calculated via 5-point Likert Scale according to their questionnaire results, each response for every question represented different scores. As we can see my template of questionnaire below:

Usability Scale questions

1. I think that I would like to play this game frequently.

Strongly agree	Agree	Neutral	Disgree	Strongly disgree
☐	☐	☐	☐	☐

2. I found the game unnecessarily complex.

Strongly agree	Agree	Neutral	Disgree	Strongly disgree
☐	☐	☐	☐	☐

3. I thought the game software was easy to use.

Strongly agree	Agree	Neutral	Disgree	Strongly disgree
☐	☐	☐	☐	☐

4. I think that I would need the support of a technical person to be able to use this game product.

Strongly agree	Agree	Neutral	Disgree	Strongly disgree
☐	☐	☐	☐	☐

5. I found the various functions in this game were well integrated.

Strongly agree	Agree	Neutral	Disgree	Strongly disgree
☐	☐	☐	☐	☐

6. I thought there was too much inconsistency in this game product.

Strongly agree	Agree	Neutral	Disgree	Strongly disgree
☐	☐	☐	☐	☐

7. I imagine that most people would learn to use this game product very quickly.

Strongly agree	Agree	Neutral	Disgree	Strongly disgree
☐	☐	☐	☐	☐

8. I found this game product very cumbersome to use.

Strongly agree	Agree	Neutral	Disgree	Strongly disgree
☐	☐	☐	☐	☐

9. I felt very confident using this game product.

Strongly agree	Agree	Neutral	Disgree	Strongly disgree
☐	☐	☐	☐	☐

10. I needed to learn a lot of things before I could get going with this game product.

Strongly agree	Agree	Neutral	Disgree	Strongly disgree
☐	☐	☐	☐	☐

Response results	Usability Scores
Strongly agree	**5 points**
Agree	**4 points**
Neutral	**3 points**
Disagree	**2 points**
Strongly disagree	**1 points**

Figure 7: The template of usability scale questions

I collected all questionnaire results of all six of them and calculated the corresponding SUS scores by using these criteria[15]:

1. Calculate X: Add up all the odd-numbered question scores and subtract 5 from total to get X.
2. Calculate Y: Add up all the even-numbered question scores, using 25 subtract this total value.
3. SUS scores: Calculate X + Y and multiply the sum by 2.5 to get SUS scores.[15]

Testers/Scores	Q1/pt	Q2/pt	Q3/pt	Q4/pt	Q5/pt	Q6/pt	Q7/pt	Q8/pt	Q9/pt	Q10/pt	X/pt	Y/pt	SUS Scores	Average SUS Scores
10-17	5	1	5	1	4	1	5	1	5	1	19	20	97.5	
18-25	5	2	5	1	5	2	4	1	5	1	19	18	92.5	
26-40	4	1	5	2	5	1	5	1	4	2	18	18	90.0	(97.5+92.5+90.0+87.5+77.5+70.0)/6 ≈ 85.8
41-54	4	2	4	2	4	1	5	1	5	1	17	18	87.5	
55-64	3	2	5	1	4	2	3	1	4	2	14	17	77.5	
More than 65	3	1	4	2	5	2	4	2	3	4	14	14	70.0	

Table 8: The scores results of Usability Scale Question

We can see a phenomenon from the table that the older the audiences, the lower the SUS scores they got, which means that they cannot use this game product very easily because of different thinking modes between different age groups. The average SUS score of these six participants who represented these six different age groups is 85.8 points, which belonged to the excellent category in the mark schema of SUS scores[15] and it means that the majority of user groups are relatively satisfied and recognized my game product, there are no major flaws in the my product's functionality or user interface design. This method is effective to use and showed the usability of my game product through intuitive scores data.

5.2 Evaluation

In general, my project basically coincides with my initial expectations and almost all the functionalities defined in the plan have been implemented, there are no serious problems or glitch after carrying the black-box tests so I am very happy with my current game product. The only minor flaw is that AI cannot detect and avoid hurting from the explosion of the bombs placed by the players at real time, which means that AI are not as difficult to be killed as you might thought. Unfortunately, I could not add this functionality to AI because it will affect the change of its states because they must perform tasks in sequence constantly, they will not be able to keep performing other tasks if AI could dodge bombs, so it taught me it is better to make sure your codes have extensibility and adaptability while writing programs.

However, the development process was not going well, especially for design of AI, which takes me a long time of debugging and revising. In addition to applying data structure algorithms to find paths in AI, they also needs to calculate and mark the blast area of placed bombs in order to avoid hurting from its explosion they. In this process, they are also allowed to use all weapon props without hurting themselves so you have to add unleash condition of each weapon for AI. Moreover, you need to write logic for each of game props and put them together to test to ensure they can be used normally, props also imposed impacts to each other because they have different functionalities so it is really hard to deal with the logic between them. The part of network communication in this game needed high requirements on real time so it is difficult to deal with the latency due to the unstable network.

6 Summary

My goal is to recreate a improved version of bomberman game with more innovative gameplay, it seems like I have already achieved this objective. In terms of game content, my game has both unique game rules and rich gameplay, including a variety of game props, intelligent AI enemies, multiple game difficulties and modes, which increased playing pleasure in my game significantly. The vast majority of user groups are satisfied with my game product and can easily use it without any teaching after finishing usability testing, they both think it is a enjoyable and interesting game.
I would continue to develop other versions of this game for other platforms if I have more time due to the game framework which has the characteristic of cross-platform.

Bibliography

[1] McFerran, Damien (2009). *Hudson Profile – Part 2 (RG)*. Issue 67. Retro Gamer Magazine. pp. 44–49. Retrieved January 19, 2011

[2] Fandom (2011). *Bomberman Wiki*. [online] Available at: https://bomberman.fandom.com/wiki/Main_Page [Accessed 26 Dec. 2021].

[3] Hudson Soft Company Ltd (2020). *Bomberman download*. [online] Available at: https://www.oldgames.sk/en/game/bomberman/translations/ [Accessed 26 Dec. 2021].

[4] Play Meter (1994). *Equipment Poll - Video & Pinball Combined*. Vol. 20, no. 3. Skybird Publishing. p. 8.

[5][6] Gaming-History (1992). *Bomber Man World*. [online] Available at: https://www.arcade-history.com/?n=bomber-man-world&page=detail&id=309 [Accessed 27 Dec. 2021].

[7] Martin Eden (2020). *The Essentials of a Common Game Development Framework*. [online] Available at: https://meliorgames.com/game-development/the-essentials-of-a-common-game-development-framework/ [Accessed 1 Jan. 2022].

[8] Wikipedia (2022). *Game framework*. [online] Available at: https://wiki.freepascal.org/Game_framework#See_also [Accessed 1 Jan. 2022].

[9] libGDX (2022). *Goals and Features*. [online] Available at: https://libgdx.com/ [Accessed 2 Jan 2022].

[10] libGDX (2021). *2D Animation*. [online] Available at: https://www.bookstack.cn/read/libgdx-1.10-en/c8601e303e116d6d.md [Accessed 5 Jan 2022].

[11] Teschner, M.; Kimmerle, S.; Heidelberger, B.; Zachmann, G.; Raghupathi, L.; Fuhrmann, A.; Cani, M.-P.; Faure, F.; Magnenat-Thalmann, N.; Strasser, W.; Volino, P. (2005). *Collision Detection for Deformable Objects*. *Computer Graphics Forum*. 24: 61–81. doi:10.1111/j.1467-8659.2005.00829. x. S2CID 1359430.

[12] Valve Developer Community (2012). *Hitbox*. [online] Available at: https://developer.valvesoftware.com/wiki/Hitbox [Accessed 10 Jan 2022].

[13] Tutorialspoint (2013). *Software Architecture & Design Introduction*. [online] Available at: https://www.tutorialspoint.com/software_architecture_design/introduction.htm [Accessed 2 Feb 2022].

[14] N. Medvidovic and R. N. Taylor (2010). S*oftware architecture: foundations, theory, and practice*. 2010 ACM/IEEE 32nd International Conference on Software Engineering, pp. 471-472, doi: 10.1145/1810295.1810435.
[15] S. Andrew (2020). *The System Usability Scale & How It's Used in UX*. [online] Available at: https://xd.adobe.com/ideas/process/user-testing/sus-system-usability-scale-ux/ [Accessed 20 Mar 2022].

Appendix A

The results of test cases:

1. Test cases of Obstacles

- Iron blocks can not be destroyed by bombs or weapons. Pass
- Brick blocks can be destroyed by bombs or weapons. Pass
- Treasure chests can be destroyed by bombs or weapons. Pass
- Game props dropped after destroying treasure chests. Pass

2. Test cases of Bombs

- Placed bombs will detonate after three seconds. Pass
- Bombs can kill players and AI. Pass
- Players and AI can not pass through bombs. Pass
- Explosion range of bombs can be increased and accumulated. Pass
- The number of placed bombs in three seconds can be increased and accumulated. Pass
- The spread range of bombs should be blocked by obstacles. Pass
- The bombs can be pushed by AI or players by using non-weapon props gloves. Pass
- The bombs being pushed will explode if they hits obstacles. Pass
- The bombs can not be pushed to outside of map boundaries. Pass

3. Test cases of Players

- Players can manipulate bomberman to move around in four directions(up, down, left, right) by using arrow keys. Pass
- Players can plant bombs by pressing space bar. Pass
- Players can pick up any dropped game props. Pass
- Players can use 4 different weapons by pressing numeric keys 1, 2, 3, and 4. Pass
- Players can be killed by bombs or weapons. Pass
- Players should not pass through obstacles and bombs. Pass
- Players cannot go outside the boundaries of the map. Pass
- Players can pause or resume game by clicking corresponding buttons at any time during game play. Pass
- Players can check game rules or help by clicking corresponding buttons at any time during game play. Pass
- Players can exit and return to main game menu at any time during game play. Pass
- Players can send messages to each other by clicking “send” button at any time in duel mode. Pass
- Players can receive messages at any time in duel mode. Pass
- Players should lose health points if hurts from bombs or weapons. Pass
- Players will win if all AI enemies are eliminated within the time limit in single-player mode. Pass
- Players will lose if all AI enemies are not eliminated within the time limit within single-player mode. Pass
- Players will lose if they are eliminated by AI enemies. Pass
- This should bring up a popup window to show game fails if players lose. Pass
- This should bring up a popup window to show game succeed if players win. Pass
- Players can choose to restart the game or return to the main screen in popup game failure window. Pass
- Players can only choose to return to the main screen in popup game victory window. Pass
- The game should be a tie if two players still survive within time limit in duel mode. Pass
- Players can choose game difficulties, game modes by clicking corresponding buttons in main game menu. Pass
- Players can check game rules or help by clicking corresponding buttons in main game menu. Pass
- Players can slide left and right to adjust volume of background music or gameplay sound effects in settings screen. Pass
- Players can turn on and off gameplay sound effects or background music in settings screen. Pass

- Players can create a room or join a room by clicking corresponding buttons in screen of duel mode. Pass

4. Test cases of Game Props

4.1 Non-weapons props

- Getting a medical kit can restore one Health point. Pass
- Getting a pair of running shoes can increase movement speed. Pass
- Getting a glass of Nitroglycerine solution can increase explosion range of placed bombs by one grid in four directions respectively(up,down,left,right). Pass
- Getting a riot shield can withstand one blast damage from bombs, bazooka, Hand grenades, landmines. Pass

4.2 Weapons

- Getting a bazooka allows characters to launch two rockets. Pass
- Pressing numeric key 1 can launch a rocket by using a bazooka. Pass
- Launched rockets can not fly beyond the boundaries of the map. Pass
- Pressing numeric key 3 can plant a landmine which is invisible after three seconds. Pass
- Landmines can only be placed on empty grids. Pass
- Grenades can only be thrown when there is an obstacle on the grid in front of the character. Pass
- Pressing numeric key 4 can throw a hand grenade behind an obstacle and causes a explosion of 3 by 3 grids on the third grid ahead. Pass
- Getting a bomb package can add an additional placed bomb in three seconds. Pass
- Pressing numeric key 2 to push the bomb out in three seconds by using gloves. Pass

5. Test cases of Game panel

- Game panel should count and display the health points of the players and AI in real-time. Pass
- Game panel should count and display the acquired game props of players in real-time. Pass
- Game panel should record and display the countdown of game in real time. Pass
- Game panel can display chat transcript between two players in duel mode. Pass

6. Test cases of AI

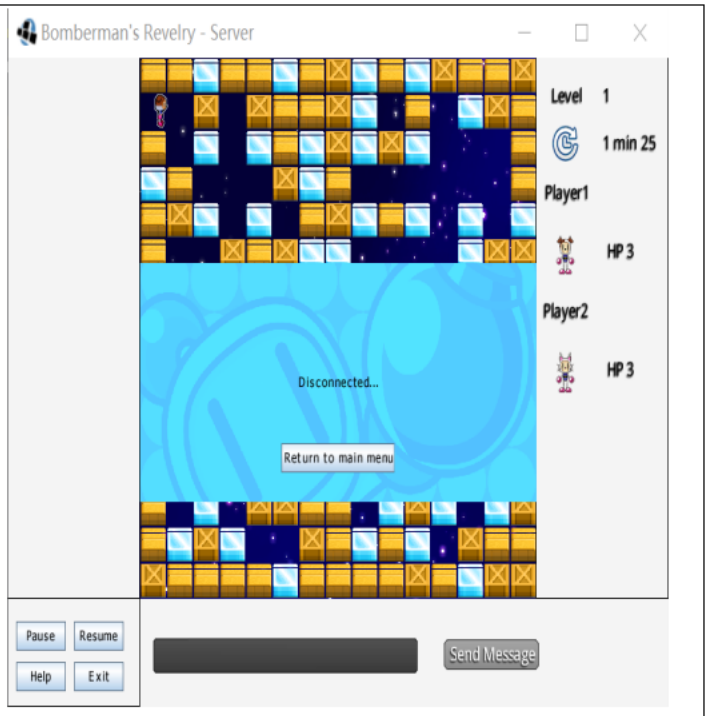
- AI must navigate around obstacles to move. Pass
- AI can place bombs. Pass
- AI should evade the blast radius of placed bombs. Pass
- AI should not pass through obstacles. Pass
- AI can find paths to treasure chests. Pass
- AI can look for dropped game props. Pass
- AI can not pick up any game props in easy difficulty. Pass
- AI can pick up non-weapon props in medium difficulty. Pass
- AI can use all game props in nightmare difficulty. Pass
- AI should use weapons without hurting themselves in nightmare difficulty.
- AI can be killed by bombs or weapons. Pass
- AI should lose health points if hurts from bombs or weapons. Pass
- AI can win or lose this game. Pass

7. Test cases of Networking

- Building a server by creating a room. Pass
- The client can search for servers by typing IP address in the local area network. Pass
- The server should accept the requests from the client. Pass
- Establish a connection between the server and client by joining the room. Pass
- Game data must be transmitted between the server and client normally. Pass
- Game progress between the client and server must be synchronized and updated in real time. Pass
- Messages sent and received between the client and server must be synchronized at all times. Pass
- This should bring up a pop-up window which shows offline state if the server or client leaves the game. Pass

Appendix B

The collection of problem during usability testing, there is only one problem existed:

Problem Users did not know if their opponent was still online in two-player mode	
Description If one player quits the game for network reasons or on his own, the other player doesn't know and waits blindly for a connection	
Solution Designed and Added a window to prompt disconnection, it will pop up and allow users to return to the main menu when one of the players disconnects for some reasons.	