Introduction

Emoji Strike is a 2-D arcade game. The game is designed so that a group of

friends can have good a time playing the game on the same device competitively.

The game firstly introduces the user with a brief tutorial to teach the user how the

game is played. As the user(s) decide that they have learned this gameplay and

controls, they move on to the game building stage which includes user’s picking

their desired character and map of choice. When this game building stage is done

users can enjoy the game they built for themselves until one of the them becomes

the victor. Therefore, our main purpose is to design a dynamic and a competitive

game.

4

Changes to the design

As we began implementing our system, we realized that some aspects of our design were poorly considered, or could be improved with a different pattern. Additionally, some features promised to be part of the final system were difficult to accomplish. On the whole, our system is mostly similar to our initial plans, but new developments in our design and our decision to use Unity resulted in a few changes to the final product, and drastic changes to our planned implementation process.

We have abandoned projectile motion in our guns, and have instead provided only facilities for linear shooting. The destructions which resulted from projectile shots looked very odd and did not make for a good user experience, so we decided to choose a slightly lower level of functionality for a more attractive user design. Since there is no projectile shooting, we allowed direct front and back shooting and to do this we added an extra command whereby pressing “F” shoots in the direction opposite the user’s character.

The changes in our design were not that huge. The UI created some more classes for the

listeners and some private methods for code reuse which were scheduled to run at specific

times. The private functions were designed to be called frequently so where as efficient as

possible. We also didn’t consider the scheduled caller objects in our design so in the

implementation those were also part of it. Also we forgot some minor view components in the

view classes like a text field in LobbyCreatorView. We also forgot a few functions in database

communication but overall there were no major changes from the design. The layering remained

the same. Also the architecture of database layer stayed same but we just had to add some

extra methods to get some data we didn’t think about before. Mainly, the middle layer had some

improvements to make the player-system interactions more safe and fast and thus most of the

methods that are used in the middle layer classes are integrated with the server.

UI layer

The UI is the same as we had planned it originally. Changes in the functionality, such as no projectile shooting, resulted in changes in the UI since we no longer have projectile arrows indicating where the weapon is aimed. These minor changes are almost indistinguishable form our original imagining of the game.

Middle layer

The middle layer, which includes the main logic behind the game and interactions between

database side has kept the general logic same as in the design process. But, there are some

extra methods added, and some old methods are changed in a way that their parameters are

made to be compatible with server side.

Since the implementation process is more complex and detailed due to unknown system and

database errors, most of the methods in java classes are fully dependent to the third layer,

database side. The reason for this is to prevent user-system interaction collisions and keep

users’ information safe. This means, when a method creates a new object by the request of a

user, firstly the database checks if this action is applicable and does not collide with other users’

action and then allows method to execute itself. For example, by this way we can block two

users entering a lobby which has only one empty seats left. Thus, only the first user that clicked

the enter button can enter the lobby, and since the quota for the empty seats are updated in the

server side, the second user is not allowed to enter.

Also, another significant change that we did is in the methods that return objects and take

objects as parameters, we changed them to take object ID’s which are stored in the database.

This way, we do not change objects directly, but with using their ID’s the database change them

for us.

Database layer

We were able to protect main shape of the layer. However we had to add extra functions to use

them in extra parts we didn’t consider at design step. Like setting the state of a lobby or getting

the character informations of a story. Also we had to added some private functions to convert

the result set we got from database to objects. Also since it was easier to handle the data

coming from database as array lists in other layer, we changed the type of multiple results to

lists from arrays. Although the return type of DBInterface stayed as arrays, the change was

made in other classes that directly used by middle layer. Finally we had to added extra static

table names to database classes for new implemented functions.

User’s guide

System Requirements

This is a very simple system, intended for all sorts of computer users, including those who are not very computer literate. The machine should have the Java 8 SDK so that it may run the .jar files with which the system runs. Internet, high amounts of RAM, memory, or processor power are not necessary for our game.

How to build

The master folder in the github has a build.xml file which takes care of all the building. By

running “ant test” command in the master directory the ANT will evaluate the build.xml file and

return successful. For that the README.md file contains the build status from

https://travis-ci.org/Albocoder/CS319-Group22 which already has passed and is working.

Running “ant build“ will build a jar file out of the project and then by assigning execute

permission to the newly created file a simple double-click will load the file in JVM and the game

will be up and running. The user will be able to view the login screen and prompted to enter

username and password.

How to install

There is no installation process. The compilation process will create a .jar file which, when run, will open the game.

How to play

When the game begins the user will have the option to either play the game, open a tutorial for the game, or load a saved game and play that. The tutorial screen is simple, and the user can return back to the original position with a press of the backspace key. To play a saved game, the user will be presented with a list of saved games from which they can select one to enter. They can start playing the game normally at that point. To start a new game, the user will be redirected to screens where they must choose the number of players playing the game, the name for each user, and the emojis which will be used to represent each individual player.

When in the actual game, the user will be able to play intuitively. W, A, S, and D keys will navigate the player’s character. The P key will pause the game, the R key will send the user back to the main host page, and the Shift+S keys when pressed together will send the user to a screen where they may save their current game for later playing. The Space Bar will shoot the weapon. There are no other complex commands or moves, and the user should have an entertaining time playing with the provided functionality. The aim of the game is to use all of the above commands to shoot and kill all other players, and be the last player remaining in the game.

The game will be displayed in the maximum windowed resolution. In the top-left corner

will be a menu with options to pause, quit, quit + save, and restart the game. Next to the

commands will be the letters which should be pressed to bring the user into the desired

state. The rest of the available space will be occupied by the game.

During each move, a small circular timer will display the amount of time remaining for

the player currently doing their move.

When beginning the game, the users will select the number of players. Following this,

they will select the Emoji and set the name for each player in sequence, and then finally

select the map they will play on, after which they game will start.

3

The users will select Emojis from a list, and type in the name for each player. They will

also select the number of players from a list of pre-approved numbers. A table of maps

with small previews will allow them to select the map.

If a player chooses to load the game, there will appear a list of files which the user

may load the game from. While saving, there will be a text input box where the user types

in the saved game’s name and press Enter to quit.