CS 319 - Object-Oriented Software Engineering

Implementation Report Iteration 2

Nightmare Dungeon

Group 3-J

Mehmet Oğuz Göçmen

Berk Mandıracıoğlu

Hakan Sarp Aydemir

Hüseyin Emre Başar

Introduction

In this project, we've tried to make a 2D dungeon crawler rogue-like game. Concept of our game is about a character Alice who got stuck in her dream and tries to get out from that dream by finding items which can give her several power ups and facing monsters who tries to kill her and going into different rooms and different layers of her dream. All rooms are locked in the beginning but Alice can unlock them by clearing out all the monsters inside that room. Each layer has a nightmare keeper(Boss). If Alice defeats that boss, they drop portals that enables Alice to teleport into deeper layers of her dream. Her dream gets harder by each layer that she goes deeper. When our system design is considered, we thought that it should be user friendly and it should have good performance in order to be enjoyable.

Implementation:

We are pretty much successful about achieving our goals before starting to code this game. Of course there are things left undone but this version is pretty much same with the reports we have written. For developing the game we've used IntelliJ IDE because it looked more comfortable with github.

Game Menu

We've improved game menu. Now settings work properly, game sounds or game music can be activated or deactivated. Help section is more helpful right now because we've divided it into 2 parts. First part gives information about how to play and second part gives information about

types of monsters items etc. Credits section remains the same. Added highscore section for keeping top 10 scores of players and sort them highest score to lowest score.

Score System

As it is mentioned above, while players are playing the game, there is a score system going on. Score starts 30k by default and players can get scores by taking items or lose points by taking damage etc. After the instance of the game ends by reaching the last portal, player is prompted to enter their username and if the players score is high enough, his/her score is placed in the appropriate place of HighScores.txt file. The scores of the players are taken from HighScores.txt file when the user wants to see them by pressing High Scores button on main menu.

Choose Character Option

Now players are allowed to choose their character before starting to game. There are 3 character options. Different characters start with different attributes for example Alice starts with 200 Health points but Lazarus starts with 250 health points.

In Game UI

At the top left corner of the game, there is health bar which shows the current health and maximum health of player. Maximum health is drawn with black and current health is drawn red on it. Bar grows or becomes smaller according to current health of player. There is another information on the screen which shows if you have active item or not.

Map

Currently, there are 3 maps in our game. Each map has 3 rooms which are connected. Each room has monsters, obstacles, passive items, active items. The last rooms have bosses in it. After beating boss a portal opens and if room is cleared, Alice can move on to another map, in other words another layer of her dream.

Doors

Doors are added into the game for linking rooms. At default doors are locked, this means that collision with doors doesn't let you go into another room. They are unlocked after Alice clears room, in other words, after defeating all monsters in the current room. In addition to that, when player enters a room from a door which is at the left of the screen, in other room, previous door is created at the reverse location of the previous door (left door), so Alice can return back to previous room with that door.

Room

After first iteration we only had one room, we increased this number to 3, to make it a short game but with small time consuming improvement, it can be added.

Random Logic

For making game entertainable, and different in every instance, as it is mentioned in our previous reports, we've made everything random, by meaning everything, I mean obstacle drops, item drops, monster spawns, monster types etc. Bosses are not random but the room which contains boss is random too. Room places are random too.

Bosses

As there are 3 maps there are 3 bosses in end of every map. Bosses have different looks and different attack mechanisms. For example one boss has really fast attack speed, the other boss fires 30 projectiles in the same time and the other boss moves really fast etc.

Portal

After beating bosses, a portal opens for linking maps. After last boss, if Alice goes into portal, game enters end game state.

Active Item

In rooms, if it is created, players can go and take active item which they can use it with space button. Alice can hold one active item so if you take another active item, she loses the active item before. Active items can be used once and after usage it disappears.

Monster's Attributes

In first iteration every monster had same attributes but in this version, we've changed it. Every monster type have different attack damage different health etc.

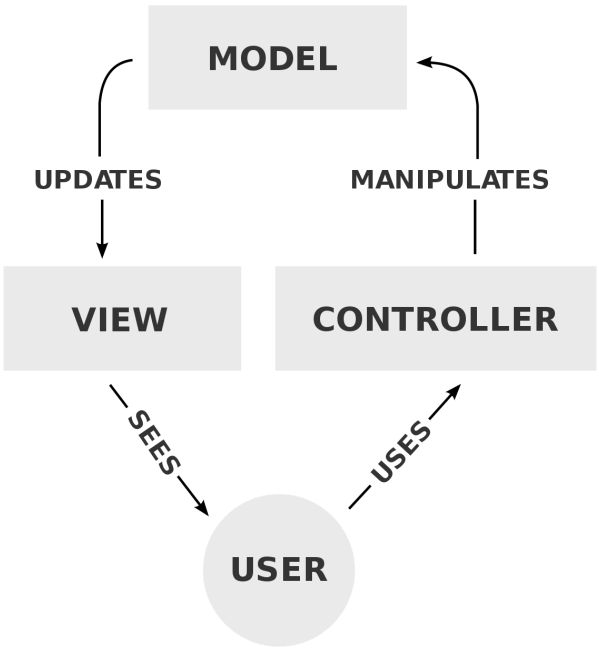
Sounds

We've added new sounds. In first iteration we only had one sound which was attack of our character but in this version, addition to first iteration, bosses attack, monster attack, changing rooms, changing maps, and death of Alice has different sounds. Also we've changed game music. It is now suitable for a nightmare concepted game.

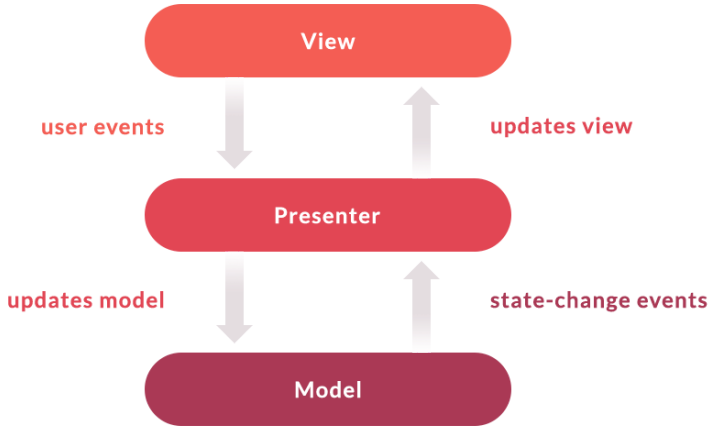
Complications and Changes:

The main complication that we were confronted with was that our code was growing different from our initial subsystem decomposition. We initially decided to use MVC(Model View Controller) architectural design but as we programmed our game we strayed from the MVC path since we used state based game because of slick2d and lwjgl. We then agreed to switch to MVP(Model View Presenter) and by doing this we combined our updater and renderer classes in one class for it to mediate between the View and Model. By doing this our Presenter has become able to both reach the data in Model and the gui elements in View instead of Controller just reaching Model and View Reaching Controller in MVP.

MVC:



MVP:



We've faced several difficulties while coding the game. According to us hardest part was using understanding and using github. As we used Slick2D library, after someone pushed his code into github, when the new code is pulled, we've needed to define project structure and add libraries again and again. Another difficulty about github was, as we were doing different parts of the project, merge option of intelliJ created some conflicts. We sometimes lost our working code. Another hard part was making a group work. Everyone's mind works different and there is nearly infinite way of writing code so, integrating our code sometimes did not end up well. Writing code according to a design was hard too. Sometimes easy things took so much of our time because we tried to code according to our design reports. Time to time we saw that some parts of our design failed, we immediately fixed our design according to it.

Installation and How To Use:

Download the whole src file from github project page. Download lwjgl(version lowee than 3.0) and slick2d. The links are provided at the end of the report. Add the jars from these files to your project structure. Add the appropriate native files for your system(linux, windows, mac, etc.) that are included in lwjgl and slick2d files to your project structure as well. The jar files are also in the lib/jars directory as a backup plan of or project but compatibility issues can sometimes occur so it is best to download them form their own sites and installing them locally. Now you can run the game by pressing the run button. Intellij is advised since we coded this project in intellij and it can solve some compatibility problems.

Things That are Left after the Deadline:

Some of the promised passive and active items are left to be done, we decided to show how the functionalities of these in game components actually occur, hence we implemented a few of them for demonstration purposes.

We never specified the amount of floors and maps we are going to implement but our aim was 5 floors in 5 different maps, i.e. 25 levels. However, this seemed too much to us so we implemented 3 floors in 3 different maps, i.e. 9 levels in total. We think this amount is more than enough to show the functionalities of our game.

We specified 20 items in the reports, but we couldn’t make that much. There are only 3 items. 2 items passive item and the other item is active item.

Conclusion

As a conclusion, although there are some functional and non-functional requirements which we promised but could not achieve, we nearly achieved our all goals and we ended up with a pretty much playable game. This project taught us much, maybe did not increase our coding skills much but increased our knowledge about design patterns, writing code according to a design pattern and acting as a team, writing code together etc. Plus we did a very good job with adapting to design patterns. It helped us with understanding each others code in a great way.

Lwjgl and slick2d download links:

http://legacy.lwjgl.org/

http://slick.ninjacave.com/