CS 319 - Object-Oriented Software Engineering

Implementation Report

Nightmare Dungeon

Group 3-J

Mehmet Oğuz Göçmen

Berk Mandıracıoğlu

Hakan Sarp Aydemir

Hüseyin Emre Başar

Implementation

After the design process, we began thinking about the GUI aspects of the system and after some research on the internet, we found suitable libraries which are slick2d and lwjgl that suited our system. These libraries made the translations between menu elements such as settings screen and high scores screen easier to implement because of the state based system that slick2d uses. The use of slick2d and lwjgl also made it easier to implement the rendering of the game such as getting an image file from a resources folder but the use of these libraries didn't affect the logic and structure of our game, it just made it easier to implement certain GUI operations. Using Slick2D library made some minor changes in design but still we managed to stick with MVC design in most parts. During implementation process, we all got together and looked to the ways we can implement this product. Overall code is the collaboration of our whole group.

Changes in Design

Model Subsystem:

- **Design Additions:** In the Model Subsystem We have used Facade Design Pattern in order to hide the complexities of the game logic from the Controller and View subsystems. Our Facade class was *Map* which helps to initialize game and perform game logic(collision, movement and etc.). Controller subsystem has access to *Map* class by has-a relation and uses this class as an interface to Model subsystem.

- **Class Based Changes:** We removed the *Sprite* class in order to prevent confusions and made Entity class the base class of inheritance. We added *Asset* class and stored the directories of images in this class as static strings so that the *GameManager* class can use them without creating an *Asset* object. We used this *Asset* class instead of using a spritesheet and getting individual sprites out of it by using mathematical expressions on pixels.
- Moreover, In our we have moved the Projectiles from the *Room* class to *Character* class. This change in *Projectile* logic enabled us to code the attack and damage logic more efficiently in time.
- We are now using ArrayList as opposed to HashMaps instead of keeping Rooms and Maps in the memory. We figured out in terms of time efficiency it would not make to much difference to have ArrayLists over HashMaps since we won't have thousands of rooms and maps. It was more feasible to code with ArrayLists so we decided to use them.

View Subsystem:

- **Design Additions:** *Menu* class is the major class of this subsystem, it draws menu, settings , game frame and etc. *Menu* uses State Based Game design which implements the idea of finite state machines. For example *Menu* is the initial state in our design and it changes states of the game (Play state, Settings State, etc.) according to user input.

Controller Subsystem:

- **User Input Changes:** By using slick2d and lwjgl, we could get input from the user with slick2d's *Keyboard* class. There is a value for each key on the keyboard so it is easy to detect which key the user is pressing by using this class.

Progress So Far:

- Transitions between Main Menu-Play, Main Menu-Settings, Main Menu-Highscores, Main Menu-Help, Main Menu-Credits are done. Transition between In-Game Pause - Gameplay is to be made.
- Examples of player, items and monsters are drawn in map manually, random distribution is to be made. Obstacles, wall boundaries, doors etc. are to be drawn, borders of room is to be established.
- Collision checks between different characters are established. Player can collect items and upgrade accordingly. Moreover, Monsters bounces back when they collide with each other and with the player as well.
- Some crucial methods for the game logic are written: playerAttack(), playerMove(), monsterMove() and etc. Monsters can follow the player and player can kill them by shooting projectiles.
- Transitions between rooms are to be made.
- In-game score, health and life bars will be drawn later.

User Manual

A user needs to first download the slick2D and lwjgl libraries that include necessary .jar files to use their methods. The user then should add these jar files to his/her workspace and add the correct native folder(according to user's operating system) to native library of lwjgl.jar file that comes with lwjgl. Then the user can run the program.