Bilkent University

Department of Computer Engineering

Spring 2020 CS319 Project

1D - Slay the Spire

Final Report

Özge Yaşayan

Gülnihal Koruk

Fatih Karahan

Mehmet Bora Kurucu

Batuhan Özçömlekçi

Table of Contents

1. Introduction **2**

2. Alterations in the Design of the Game **3**

3. Lessons Learned **6**

4. User’s Guide **7**

4.1. Main Menu 7

4.2. Game 7

4.2.1. Creatures 8

4.2.1.1. Monster 8

4.2.1.2. Character 9

4.2.1.3. Pet 9

4.2.2. Objects 10

4.2.2.1. Relics 10

4.2.2.2. Powers 10

4.2.2.3. Cards 11

4.2.3. Map 11

4.2.4. Rooms 12

4.2.4.1. Fight Room 12

4.2.4.2. Treasure Room 13

4.2.4.3. Merchant Room 14

4.2.4.4. Rest Room 15

4.2.4.5. Event Room 15

4.3. Compendium 15

4.4. Additional Features 17

5. Build Instructions and Setting Up The Game **17**

6. Work Allocation **19**

7. References **21**

1. Introduction

After the submission of the *Project Analysis Iteration 1* report, we started the implementation. IDE was chosen as IntelliJ, and the game was decided to be implemented in Java. GitHub was used as a version control system. The additional library that was used is JavaFX library, which was used in the implementation of GUI. Core game components are now completed, all of the rooms, cards, relics, monsters, etc work successfully.

Also, as an additional feature, a pet was added. Pet serves as a companion which helps the player in their combats and can be bought/upgraded from the merchant room. Also, map generation of the game was improved. The map is generated dynamically, so player is not bored from playing on the same sequence of rooms. Also, player is also able to choose their room, which gives also a more sense of adventure.

2. Alterations in the Design of the Game

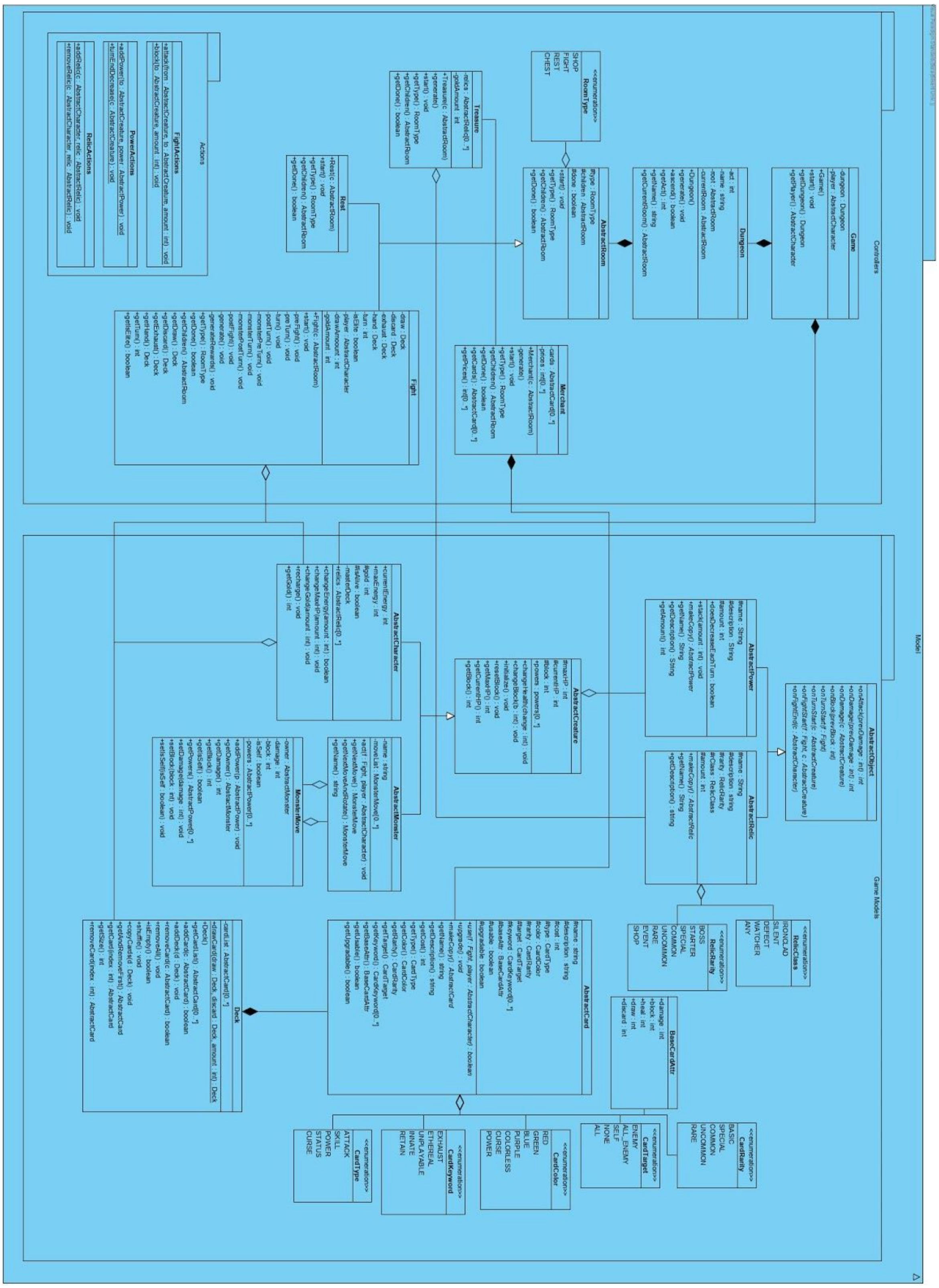


Figure 1: Initial Status of the Game Model and Controller Subsystems

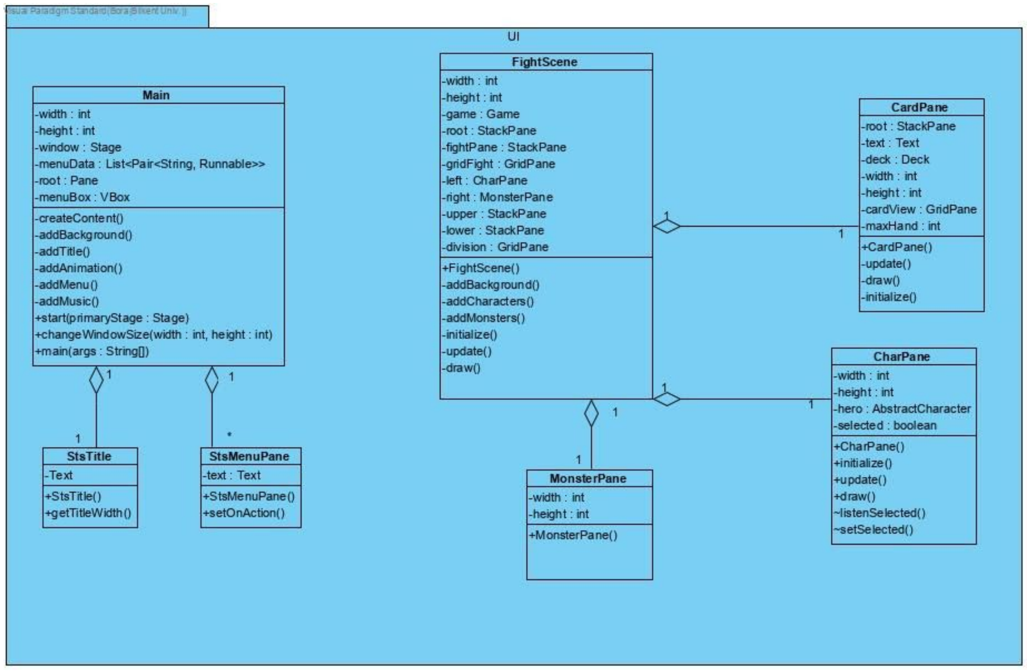


Figure 2: Initial Status of the UI Subsystem

The figures above (1 and 2) show the initial status of the subsystem decomposition of the game and their classes. Since the UI of the game haven't been completely constructed and connected to the controller and model subsystems of the game in the first iteration, the UI subsystem in its initial status is partly shown.

The preliminary decision on the design choice of the game classes was the MVC design pattern which requires us to classify our game classes according to their functionality and access relationship between game classes. In the implementation of UI classes, JavaFX libraries and their object classes such as Stage, Scene, Pane are utilized. We prefer constructing the UI classes by Java code directly instead of using automatic display construction tools such as SceneBuilder. Likewise, FXML files haven't been used for the display yet objects coming from JavaFX libraries are utilized, extended and embedded into the UI classes of the game. By doing this, our purpose of coding this game have been more parallel with the practices of the object-oriented software engineering. Moreover, it enabled the game to be more modifiable. For example, the display button constructed with a SceneBuilder doesn't explain the background mechanics of that button and enables a limited number of optional modifications for that button. However, in our design of that button, the modifiable button extends a JavaFX Rectangle (or Circle) which embodies a background image assigned to it accordingly and performs several actions on click. One can also easily change the graphical effect assigned to that button by simple modifications in the code.

Diagram itself can be found on this link.

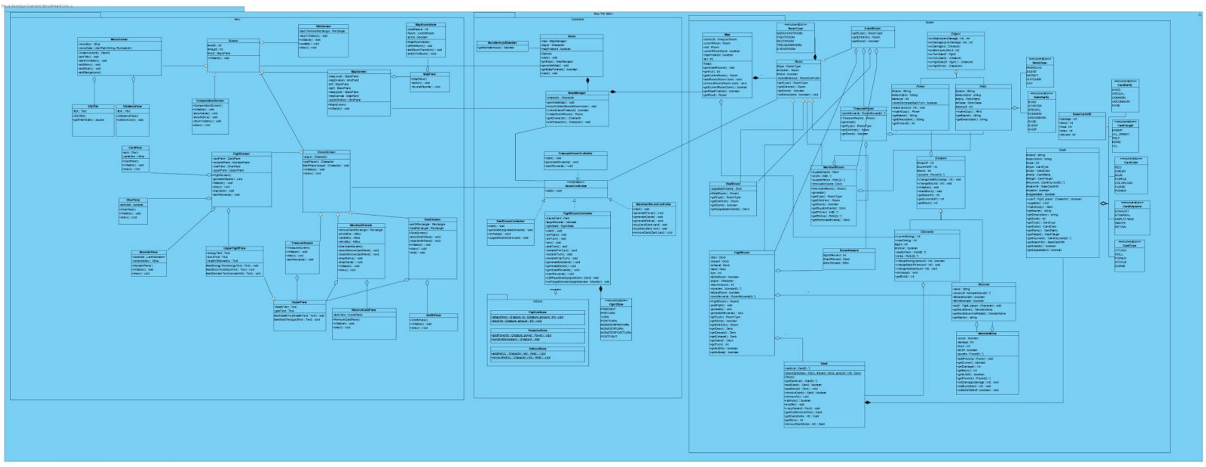


Figure 3: Final Subsystem Decomposition Diagram (Second Iteration)

The final subsystem decomposition diagram models the basic class structure in the game. In fact, there are still various classes under some basic title of a class yet such classes such as various monster classes, card classes or classes designed for some UI panes are too much detailed to mention in this diagram. Instead, one can observe the complete documentation of general decomposition of the game classes that are necessary for the game in this diagram. Basically, our group classified our game classes under View, Controller and Model packages.

In the final design, more design patterns introduced onto the MVC pattern. This process of introducing more design patterns have been parallel with the learnings in our class lectures. Sometimes, it was the case that a design pattern was already implemented (State Design Pattern) Some of the new design patterns are Façade design pattern which is utilized in Controller-connected UI Screens (“Scene”s) managing UI “Pane”s (i.e. FightScreen and its panes), State design pattern demonstrating the turn-based structure of the fights (i.e. FightRoomController), and Composite design pattern in the relationship of rooms with their children. Refer to the design report for more detailed information about the design.
Up until the demo of the first iteration, we had finished the basic (must) functionalities for the game and we had provided a fundamental graphical as well as textual view for the game. The second iteration includes the complete functionality of game mechanics with novel features as well as an interactable UI for these functionalities. However, there are some graphical bugs related to screening of the game via UI. Since the team tries to ensure that the game is playable in various sized monitors, we have enabled players to change their

game display size in the “Options” section yet it introduced the discussed screening bugs into the game. These bugs are mostly resolved but there could be edge cases that could be improved. Our innovative features such as companion pet are present in the final copy of the game. User friendly view and game mechanics was the focus in the final design of the game in order to improve the enjoyment of the player of the game.

3. Lessons Learned

Firstly, we have experienced how to work as a team in collaborate. Arranging meetings was not a big problem in our group. We set our meetings on Discord and WhatsApp. Through the semester, we met once in every week. During the last month, we have met twice a week. Since we all have different courses and schedules, setting meeting was difficult sometimes but we have never postponed our meetings. That discussing the progress and distributing the workload in every week was a good for the project progress. We saw where we were and what we should do next, which gave us an important advantage to finish the project before the deadline. Therefore, we learned working in a scheduled program is important for the progress.

One of the most effective factors to finish the project on time was the proper distribution of the workload. 2 members focused on the model classes and 3 members focused on the GUI, but still we all have contributions to almost every class. We debugged the code together, and in any kind of problem, we always helped each other. Before the submitting of the reports, at least 2 members always reviewed the work done. Making a project from the scratch brings along some troubles. To have an enough time for handling the problems, we started to our project early. Nevertheless, still we had small works to do 2 days before the deadline. Thus, we learned that starting the project early has a huge advantage.

The most important thing for the process of the project was the reports. Both design and analysis reports have helped us to understand the project better. Writing the analysis report before the coding, we saw which classes we need and possible sequence interactions between systems and users, and that helped us with seeing how to do the implementation. The design report also showed us the interactions between classes and objects in the system. This prevented many possible problems that may be encountered during the coding.

4. User’s Guide

4.1. Main Menu

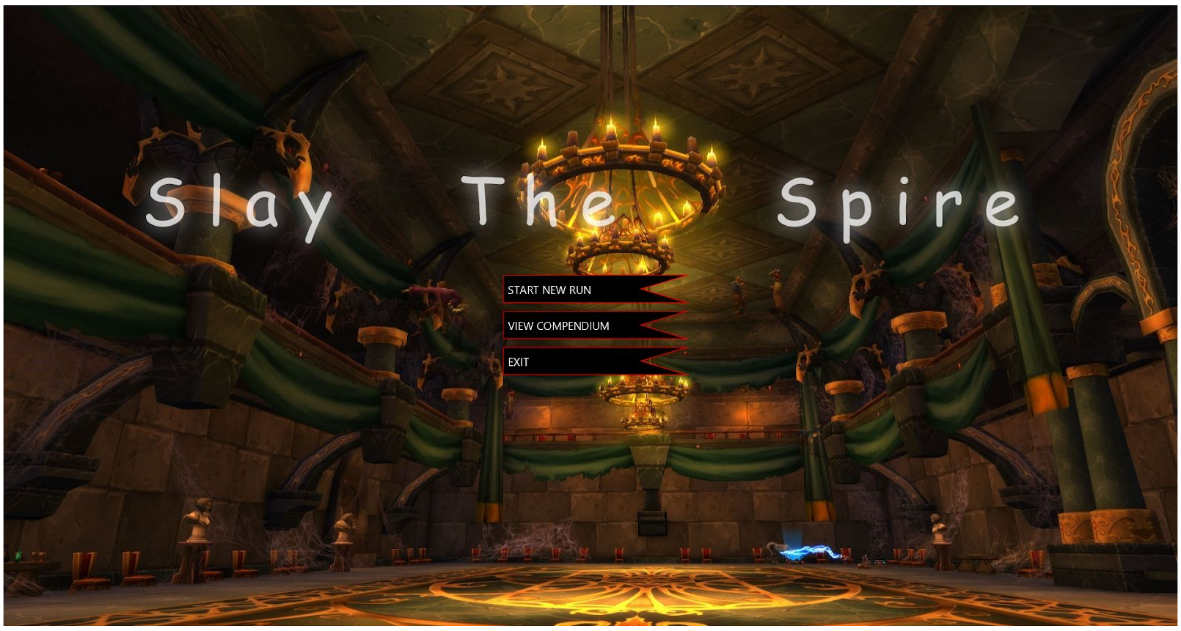


Figure 4: Main menu of the game

On the main menu, the user encounters three different options. They can either choose to start a new run, view the compendium or exit the game.
When they start a new run, they see a map which will help them navigate through the rooms in the game.
When they click on view the compendium, they can see a list of the relics and cards available in the game.
The users also hear the music throughout the game starting from the main menu.

4.2. Game

The game consists of various components that can be grouped as creatures, objects, the game map and the rooms on the map. This sections explains the components of the game and gives further information about the game mechanics.

4.2.1. Creatures

4.2.1.1. Monster

Figure 5: Jaw Worm, a monster from the game

Our game has a lot of monsters of varying difficulties. Some monsters are easier to beat - they are called “common” monsters (such as the Cultist, Green Louse, Red Louse, Jaw Worm) and the players encounter them in ordinary fight rooms. Winning against ordinary monsters do not give big bonuses.
Elite monsters are more difficult to beat, and our game has various elites such as Sentry and Gremlin Nob. Winning against them gives the player better rewards.
At the end of the map, the player encounters a boss monster. In our game, that boss is The Guardian. Winning against him results in a win overall, and after the game ends, the user sees a screen informing them that they won and they can opt to return to the main menu. Losing against him results in the player being defeated and the player sees a screen that tells them they lost, and they can opt to restart the game or return to the main menu.
Monsters have a list of moves that they are going to make, based on their characteristics. For example, Cultist has two possible moves: Incantation and Dark Strike. Note that we referred to the Slay the Spire Wiki for the moves of the monsters and their explanations.
The users are able to see the monster’s next move when they hover over the monster. This gives them a chance to adjust their strategy and play cards based on their opponent’s intentions.
The original game has more variety of monsters but because of time constraints we were able to implement a limited amount of them, albeit still maintaining some variety to allow randomness in the encounters.

4.2.1.2. Character

Figure 6: Ironclad, the starting character

The player plays as a character in the game, and every character has a different set of cards as well as relics. The game starts off with the Ironclad character which is symbolized by the color red, therefore all of the Ironclad specific cards are red.
There are more characters in the original game such as the Defect, Watcher, and Silent. However, because of time constraints we were only able to implement Ironclad and his cards.

4.2.1.3. Pet

Figure 7: Pet in the merchant room

As an extra feature, we added a “pet” as an addition to the character in combat. More information about the pet will be given in the New Features section.

4.2.2. Objects

Objects are the items the player uses to win fights or do certain stuff like heal themselves.

4.2.2.1. Relics

Figure 8: Bag of Marbles, a relic

Relics are permanent items that the player keeps all until the game is finished with the player either winning or losing. Relics can have very different effects like increasing the Max HP of the player by a certain amount, or more frequently with a trigger. For example, gain Strength at the start of each fight, or gain energy at the start of each turn.

4.2.2.2. Powers

Figure 9: Strength, a power

Powers are temporary modifiers that last through each fight, and then reset. They are divided into two categories: ones that are temporary even in the fight, and ones that stay all until the end of it. Temporary powers usually decrease by one in intensity at the end of each turn until they reach zero, where they are then deleted. Permanent powers are powers that don't decrease in intensity.

Powers, like relics, can have wildly different effects. For example, one power can increase the damage of your attacks, another can make you vulnerable and take more damage or make your blocks stronger.

4.2.2.3. Cards

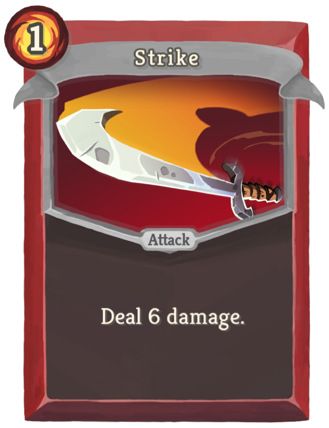


Figure 10: Strike, a card

Cards are the most important objects that are in the game. Each card represents an action the player can take, ranging from defense to offense, healing up to drawing cards. Each card has a certain colour and rarity attached to them, which affects the chance the player can encounter them in various places they can add cards to their deck.
Each card also has an energy amount which needs to be consumed for that to be used. The energy cost usually depends on what the card does, for example a card that hits more than average can have a higher cost than a card that hits less.
Cards can be upgraded through the smith in the rest rooms in order to show more effect during combats.

4.2.3. Map

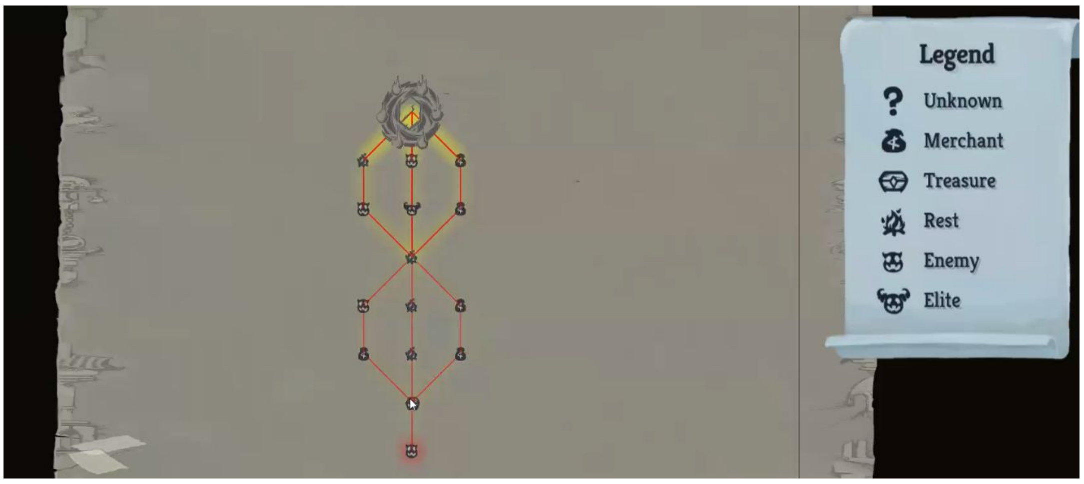


Figure 11: The map screen in the game

The map of the game is not exactly the same as the original game. The original game uses a procedural generation algorithm that was too complex for us to implement in the given time. Therefore, we made it a bit simpler but we still retained the randomness factor and we gave the user the choice to move in the direction they wanted. So, every time a user starts a new run, the generated map will be different.
The map always starts with a fight room that is followed by a treasure room. After that, there are multiple ways in which the user can choose to go to. The end of a map is always a boss fight. The users can adjust their strategies based on the rooms on the map and pick the path that is best for their interest.
The user’s current location on the map is indicated with a red circle around the room node.
The screen contains a legend for the players to be able to distinguish the different room types. These room types that can be visited through the map are going to be explained in the following section.

4.2.4. Rooms

Rooms are the scenes that the player sees most of the time. They are the scenes that the player moves through in order to progress. There are multiple types of rooms, the order which the player sees them is dependent on their selection on the map.

4.2.4.1. Fight Room

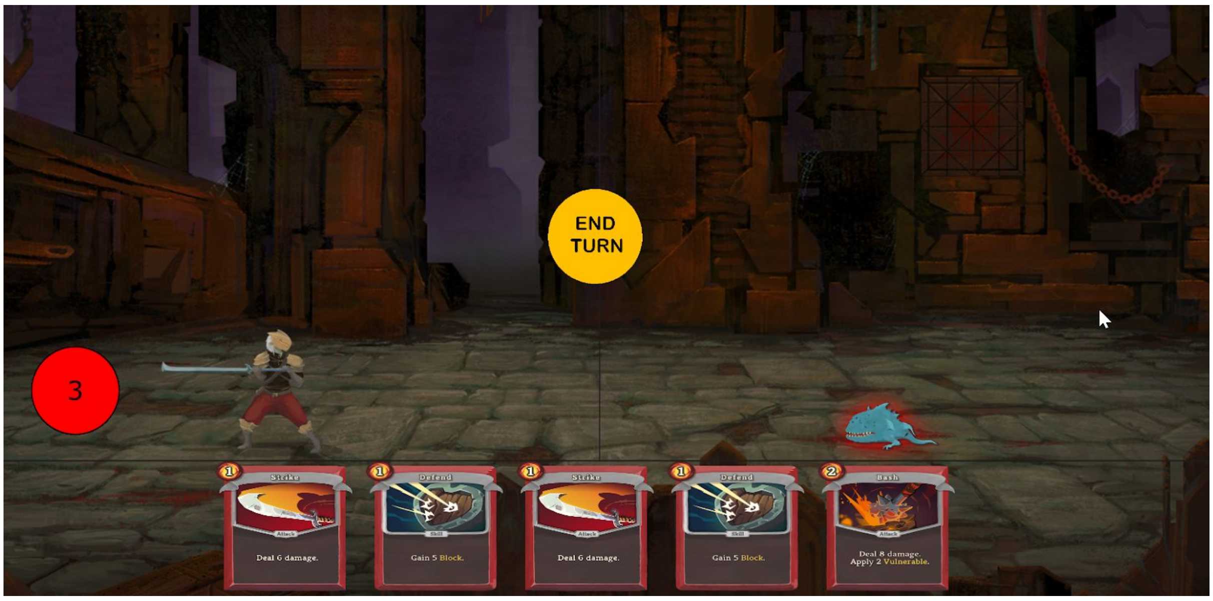


Figure 12: A snapshot of the combat

The fight room can be considered the most important room. Fight room is the room that the player encounters different types of monsters,each with their unique move and attack sets that challenges the player and tries to hinder their ability to continue ascending the dungeon. Fights consist of turns that switches between the player and the enemy. Each turn, the player draws a certain amount of cards from their draw pile. Then the player has a certain amount of energy they can use to play the cards they draw, which at the end of the turn moves to the discard pile, which is shuffled back into the draw pile if the draw pile is empty. At the end of each fight room if the player manages to win the fight without dying, the player is given a set of rewards that consists of cards, gold and relics. As the player wins more and more fights, their ability to fight increases due to the objects they've collected.

4.2.4.2. Treasure Room

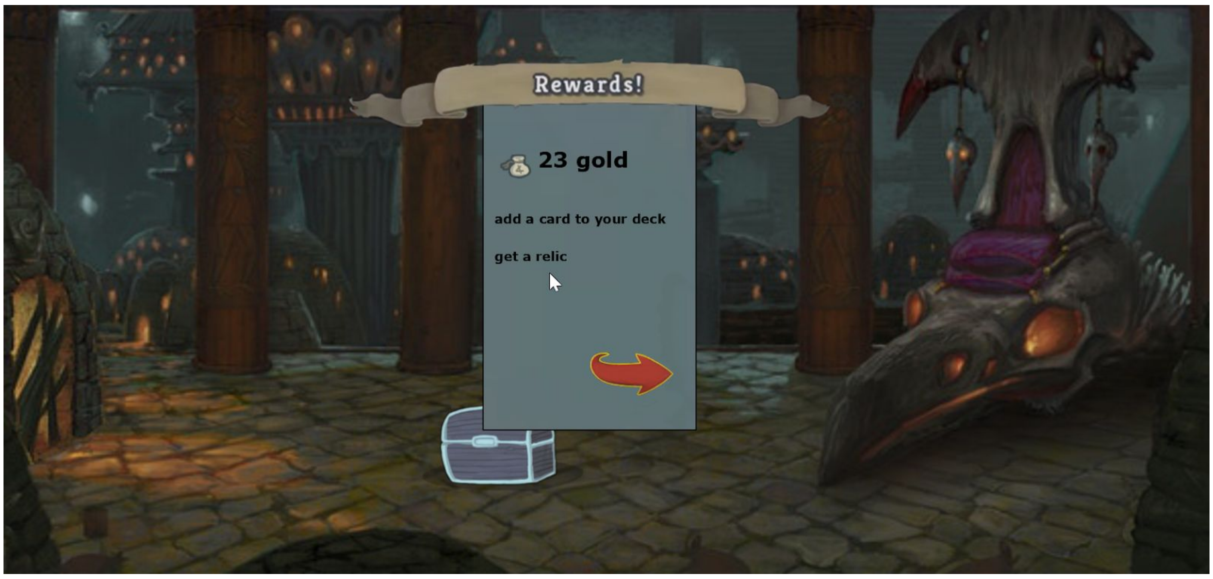


Figure 13: A snapshot of the treasure room

The treasure room is a very simple room. It only consists of a chest, and some rewards in it. There's no fighting, or anything else the player needs to do. If the player manages to get to a treasure room, it means the player can add some free gold and relics to themselves.

4.2.4.3. Merchant Room

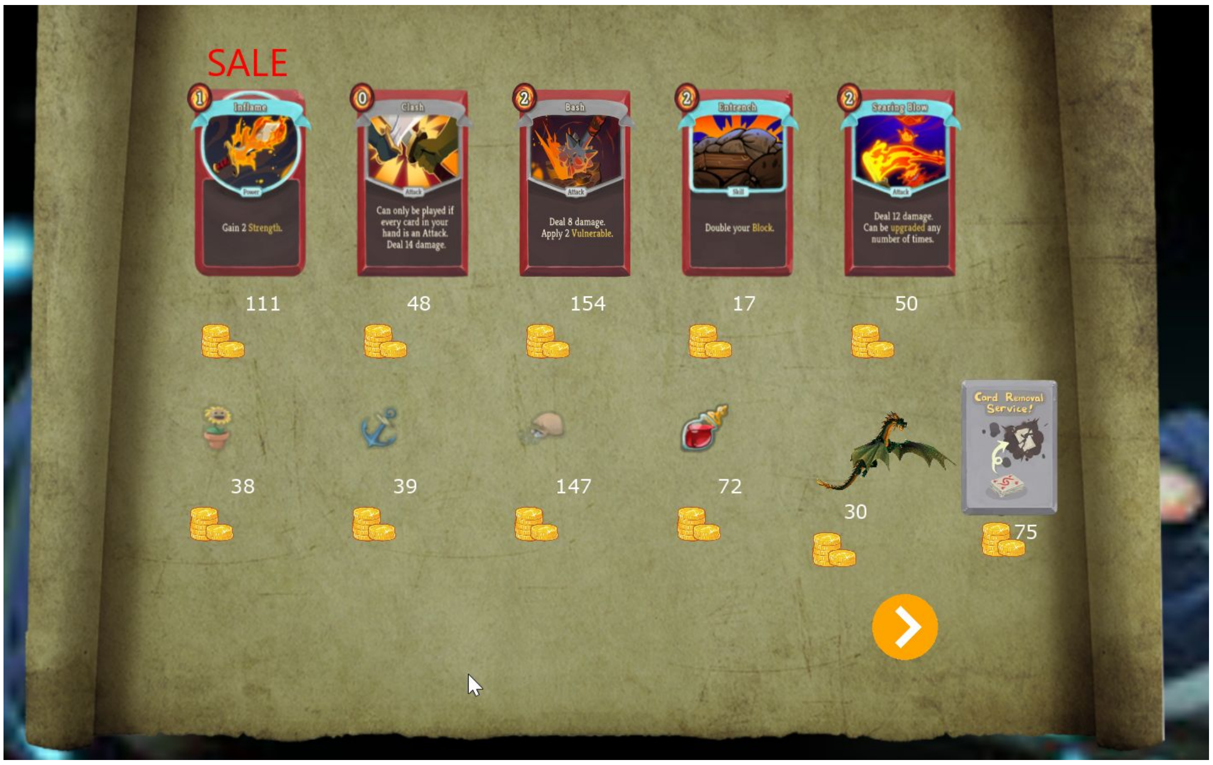

SALE
1 Inflame Power Gain 2 Strength.
0 Clash Attack Can only be played if every card in your hand is an Attack. Deal 14 damage.
2 Bash Attack Deal 8 damage. Apply 2 Vulnerable.
2 Entrench Skill Double your Block.
2 Searing Blow Attack Deal 12 damage. Can be upgraded any number of times.
111
48
154
17
50
38
39
147
72
30
Card Removal Service!
75

Figure 14: A snapshot of the merchant room

The merchant room is the room the player uses the gold they've collected through different means, like end of fight rewards or treasure rooms. Here, the player can buy cards and relics to increase their fighting power and survivability. There is also a pet that can be bought and then upgraded in another entrances of the room.

The player is also given the chance to remove a single card from their deck. It is so if a player wants to keep their deck smaller because they want to increase the chance of certain cards coming to them each round in the fights.

4.2.4.4. Rest Room

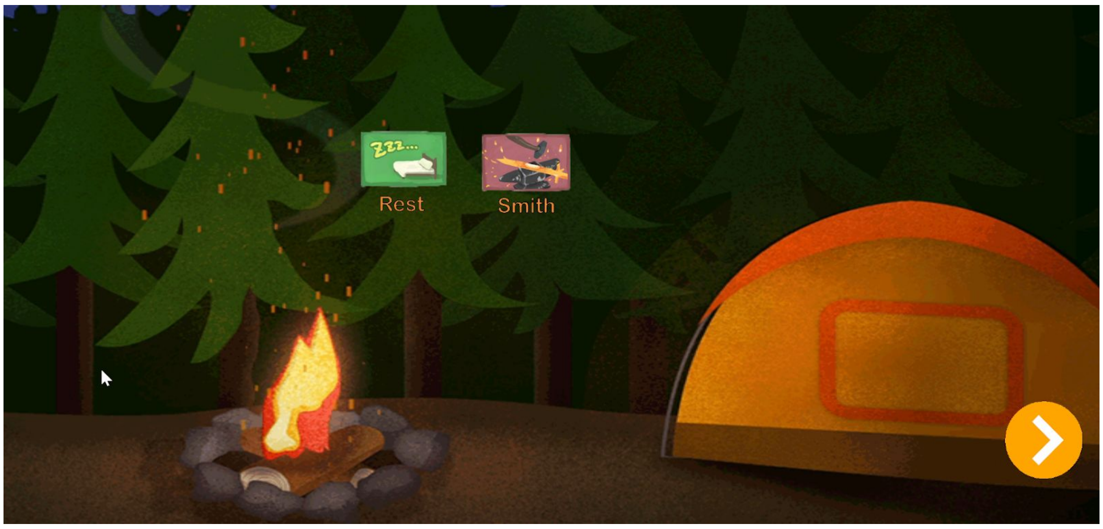


Figure 15: A snapshot of the rest room

The Rest Room is the room that the player can use to rest and heal up, or if they feel like they're doing okay on health front, upgrade one of their cards through the smith.

4.2.4.5. Event Room

An event room is a room that is randomly generated to be one of the other types of rooms, which the player doesn't know before entering.

4.3. Compendium

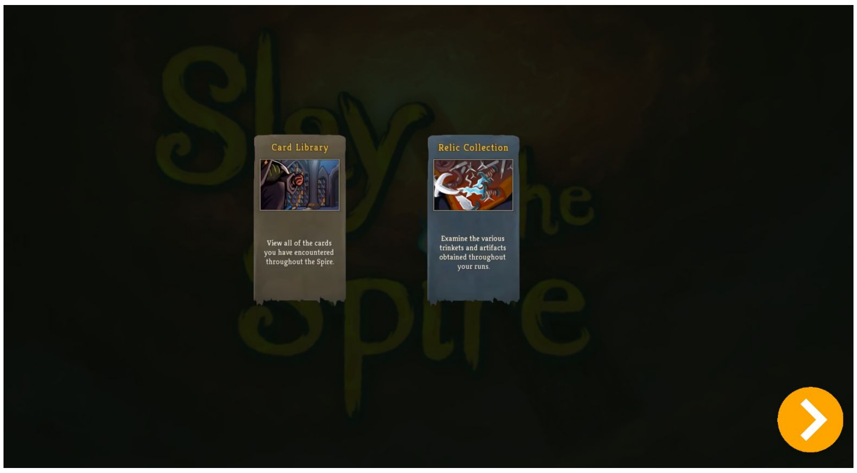
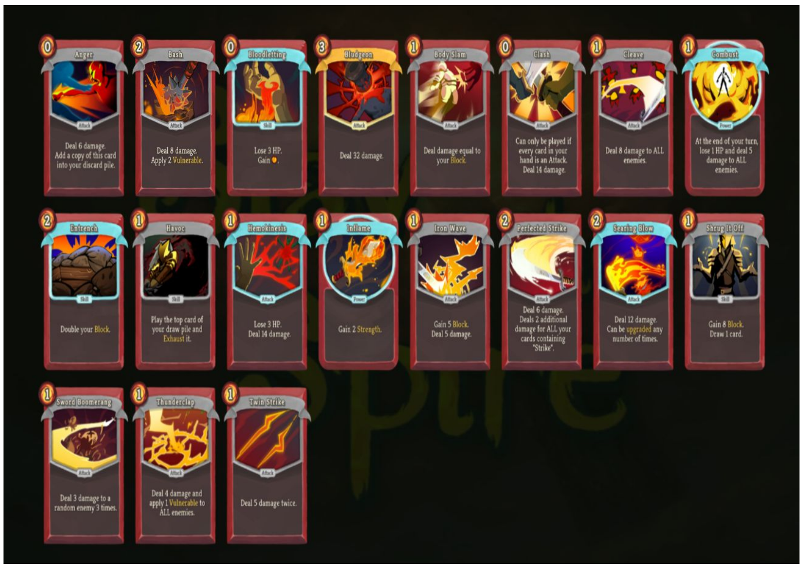

Slay the Spire
Card Library
View all of the cards you have encountered throughout the Spire.
Relic Collection
Examine the various trinkets and artifacts obtained throughout your runs.

Figure 16: Compendium screen


0 Anger
Attack
Deal 6 damage. Add a copy of this card into your discard pile.

2 Bash
Attack
Deal 8 damage. Apply 2 Vulnerable.

0 Bloodletting
Skill
Lose 3 HP. Gain ●.

3 Bludgeon
Attack
Deal 32 damage.

1 Body Slam
Attack
Deal damage equal to your Block.

0 Clash
Attack
Can only be played if every card in your hand is an Attack. Deal 14 damage.

1 Cleave
Attack
Deal 8 damage to ALL enemies.

1 Combust
Power
At the end of your turn, lose 1 HP and deal 5 damage to ALL enemies.

2 Entrench
Skill
Double your Block.

1 Havoc
Skill
Play the top card of your draw pile and Exhaust it.

1 Hemokinesis
Attack
Lose 3 HP. Deal 14 damage.

1 Inflame
Power
Gain 2 Strength.

1 Iron Wave
Attack
Gain 5 Block. Deal 5 damage.

2 Perfected Strike
Attack
Deal 6 damage. Deals 2 additional damage for ALL your cards containing "Strike".

2 Searing Blow
Attack
Deal 12 damage. Can be upgraded any number of times.

1 Shrug It Off
Skill
Gain 8 Block. Draw 1 card.

1 Sword Boomerang
Attack
Deal 3 damage to a random enemy 3 times.

1 Thunderclap
Attack
Deal 4 damage and apply 1 Vulnerable to ALL enemies.

1 Twin Strike
Attack
Deal 5 damage twice.

Figure 17: Card collection in the game

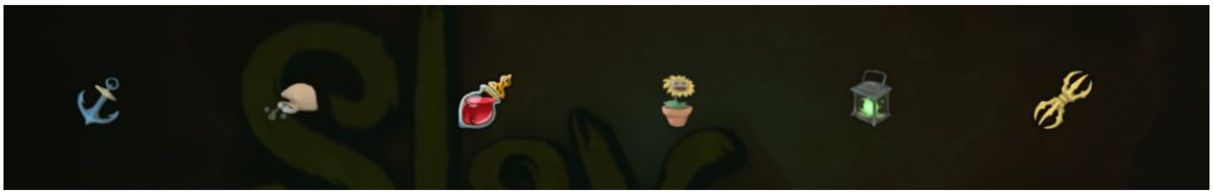


Figure 18: Relic collection in the game

When the user chooses to see their compendium on the main menu, they can either look at the card list or the relic list. The compendium contains the items that are available for the user to acquire throughout the game. It provides a list of those items so that the users can adjust their strategy according to the cards and relics that are available to them.

4.4. Additional Features

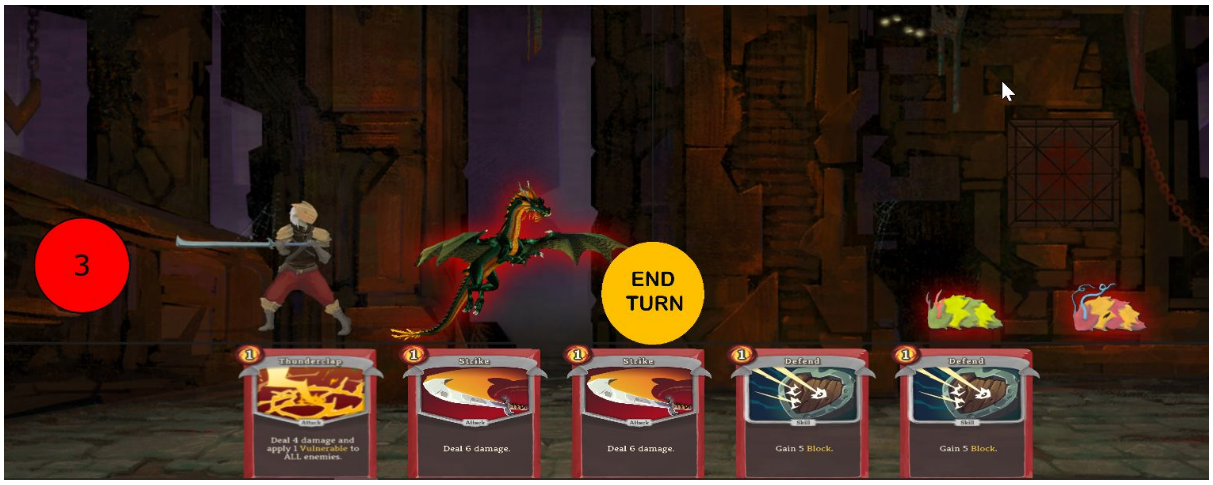


Figure 19: Pet next to character in combat

As an additional feature to the original version of the game, we made it possible for the user to purchase a pet from the merchant in exchange for gold. After the purchase of the pet, it starts appearing next to the player during combats.
After the player clicks on "end turn", the pet deals damage to the monsters by a certain amount. Initially this amount is 5. However, the user can choose to upgrade the pet in exchange for gold, which in return would increase the damage dealt by the pet to the monsters. With every upgrade, the pet deals 5 more damage.
Having a pet is advantageous for the player since it can do extra damage to all of the monsters in the room and it has no effect on the player's energy.

5. Build Instructions and Setting Up The Game

Although we couldn't test the game with a low performance computer, it is preferred to have 512MB RAM or higher, a decent fundamental GPU that will work with JRE (java Runtime Environment) and JDK 11. Since the game is coded on the computers with 1920x1080 and 1280x720 screen dimensions, it would be better use these settings. Refer to the system requirements for JRE and JDK for more detail.

Recommended System for the Game:

- **512MB** RAM or higher.
- Recommended monitor dimensions: **1920x1080** or **1280x720** pixels.
- Processor with **1.0 GHz** clock rate.

Installation:

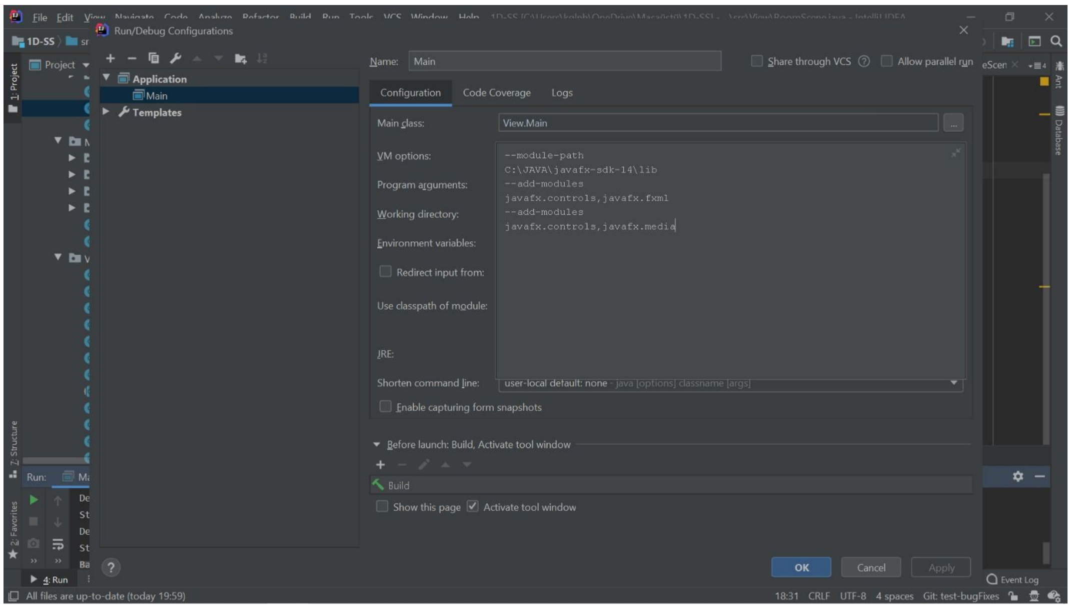
One can access the jar file of the game to directly play it (you can find it on the Github page of the project).

In case one wants to access the source code of the game, the game code is accessible in the Github page as well. First source code should be fetched via a Git tool. After completing the fetching, it is important to check that project media files (images, musics) are complete. JavaFX 14 and its libraries are required for the game to run. Therefore download JavaFX14 using the official page.

If you receive an error related to music files or ScrollPane class of JavaFX, type the following lines of command into your VM Options.

```
--module-path
Here type path to the JavaFX14 lib directory
--add-modules
javafx.controls,javafx.fxml
--add-modules
javafx.controls,javafx.media
```

After typing the above lines to your VM Options you should be able to run your code on your IDE. The team utilized IntelliJ as the IDE. You can access VM Options by opening "Run - Edit Configurations - Configuration" in the upper menu side of your IntelliJ IDE.



Run/Debug Configurations

Name: Main

☐ Share through VCS ☐ Allow parallel run

Configuration | Code Coverage | Logs

Main class: View.Main

VM options:

```
--module-path
C:\JAVA\javafx-sdk-14\lib
--add-modules
javafx.controls,javafx.fxml
--add-modules
javafx.controls,javafx.media
```

Shorten command line: user-local default: none - java [options] classname [args]

☐ Enable capturing form snapshots

Before launch: Build, Activate tool window

Build

☐ Show this page ☑ Activate tool window

Figure 20: A snapshot from the IntelliJ IDE settings

Finishing these steps should let you run the game by running the Main class of the View package.

6. Work Allocation

Gülnihal Koruk:

- Contributed to reports.
- Added pet GUI.
- Added music functionality.
- Coded some GUI classes with all its related classes; TreasureScene, Compendium, RestScene, MerchantScene.
- Contributed to all other GUI classes.
- Reviewed and helped for the format of reports.

Batuhan Özçömlekçi:

- Contributed to the reports and diagrams.
- Contributed to GUI classes (MenuScene, FightScene, OptionsScene, RoomScene, GameScene, MapScene, MapPane, MapRoomNode) and their general relationship.
- Contributed to map generation.

- Contributed to OptionsManager, MusicPlayer (Control) classes
- Contributed to the initialization of some of the entity classes.
- Contributed to debugging the game.

Özge Yaşayan:

- Contributed to reports.
- Adjusted the format of the reports.
- Added game entities such as individual card classes (e.g. Anger, Bash, Bloodletting etc.), relic classes (BagOfMarbles, BurningBlood, Anchor etc.) and some of the monsters (Sentry, TheGuardian, RedLouse, GreenLouse etc.).
- Added the pet functionality and related classes to pet.
- Contributed to the generation of the map.
- Helped with debugging the features.

Bora Kurucu:

- Worked as a product manager, distributed works in both code and reports,helped communication between members, arranged meetings.
- Coded CharPane, CardPane, UpperPane, MonsterPane, FightScene, TreasureScene, RestScene, MerchantScene, WinScene, MenuScene classes.
- Contributed to reports by writing them and drawing diagrams. Also reviewed other people's work and corrected them to make the reports consistent.
- Helped with debugging of all classes.

Fatih Karahan:

- Designed the general structure and the flow of the game.
- Designed the class structure and how they were going to be connected to each other, as well as why we needed them.
- Implemented the main backend classes such as Object, Card, Character, Monster, Room, Fight, Merchant, Rest, Treasure, Power etc.
- Added all the Power entities and some of the Monster entities and Card entities.
- Drew the Class and Object Diagrams.
- Contributed to the reports.

7. References

[1]“Slay the Spire Wiki,” Fandom. [Online]. Available:
https://slay-the-spire.fandom.com/wiki/Slay_the_Spire_Wiki. [Accessed: 18-May-2020].
[2]“JavaFX,” Gluon. [Online]. Available:
https://openjfx.io. [Accessed: 18-May-2020].