
# CS319 - Object Oriented Software Engineering

## Final Report Second Iteration

Infinite Tale

Group 1-H

Erim Erdal
Halil Azak
Hüseyin Taşkesen
Can Özgürel

Supervisor: Uğur Doğrusöz

# 1- Implementation:

Initial steps of the game were taken before the second iteration of our game. We can consider it as a baby in womb at that time, however we were able to finish a playable prototype before the second iteration.

As second iteration started, as a group we immediately started to improve our lacking parts in our code as we continued producing reports. We used IntelliJ IDE for developing our game for several reasons, first of which is that it is perfectly compatible with GitHub, where you can pull and push in the IDE environment which was a big plus because of the group project requiring us to act as one. This enables writing code together, as one finished their part and pushed the other could pull and continue on their part with ease.

We liked to separate our work as a group because it was easier to handle the complexity that way. Since our design work of our game was very extensive and good, it was not a problem to completely divide the classes because we all knew initially how classes interact with each other and what input or outputs should be. Of course there were small problems with these but they were fixed quickly with the help of teamwork.

On a general look, Erim implemented the initial Model classes named as subsystem Map in our game, Halil implemented the initial Controller classes named as subsystem Game in our game, and View classes named UI were implemented with the help of everyone. This is of course a general look and does not represent full workload of each individual. We sometimes helped each other to fulfill the requirements, for example Erim helped in UI and Halil helped in UI and Controller too in several classes, also Hüseyin and Can helped UI, Model and Controller. We had group meetings and phone calls to achieve

continuity and working as a team. Detailed information of whom did which part extensively must be gathered by looking at GitHub contributions.

We worked on our lacking together and as we saw some failings in design report, we immediately started working on them and added it to our design.

Below are the things that we were able to finish as we promised, also the things were not promised but added for the sake of the game. As we have already talked about functional requirements in the first iteration, we will mostly be speaking about additional functional requirements:

### 1- UI Changes:

As we promised in design, UI changes were made:

1- In order to see both geographical and political view of the Map which enables user to act according to the geographical surface and their advantages, also checking the political view to create a plan to move on.


Figure -1,2- ( Geographic View )

Not promised but added:

2- Zooming in and out of the map added to get a better view of Tiles and to the places of the Map which is left out of the current screen.

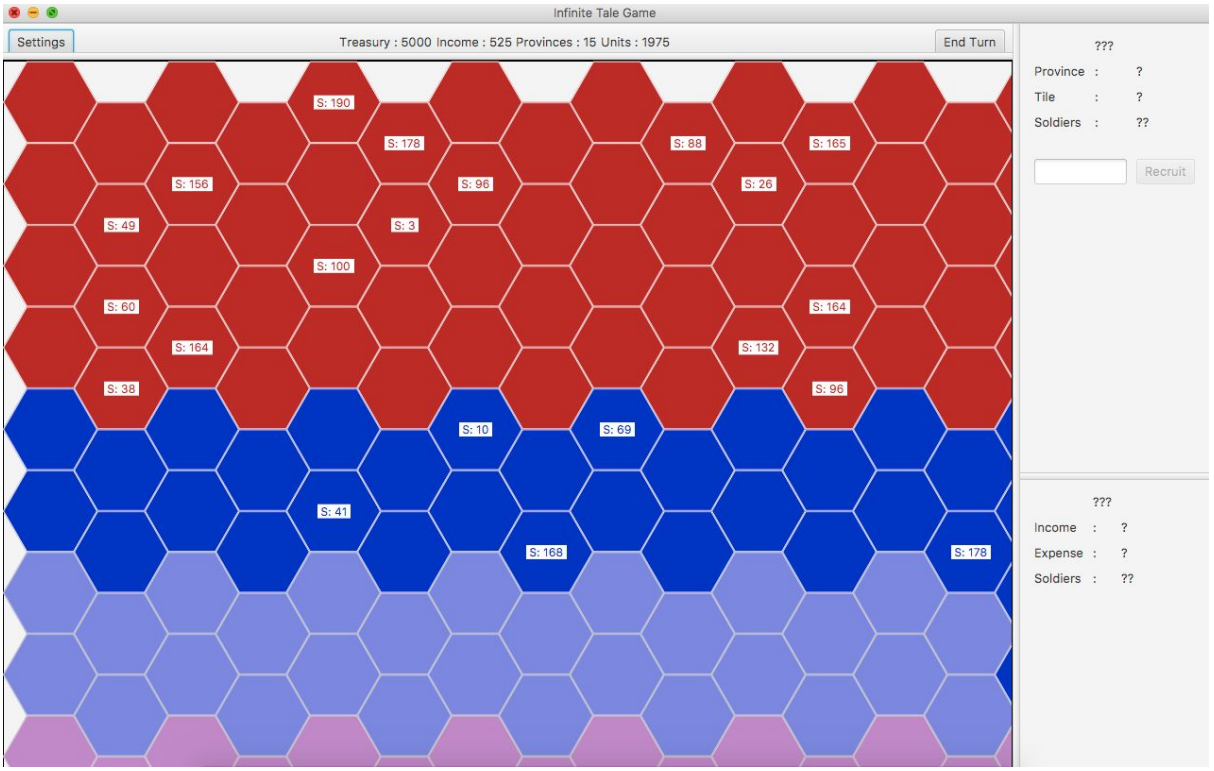


Figure -3- (Zoomed out)

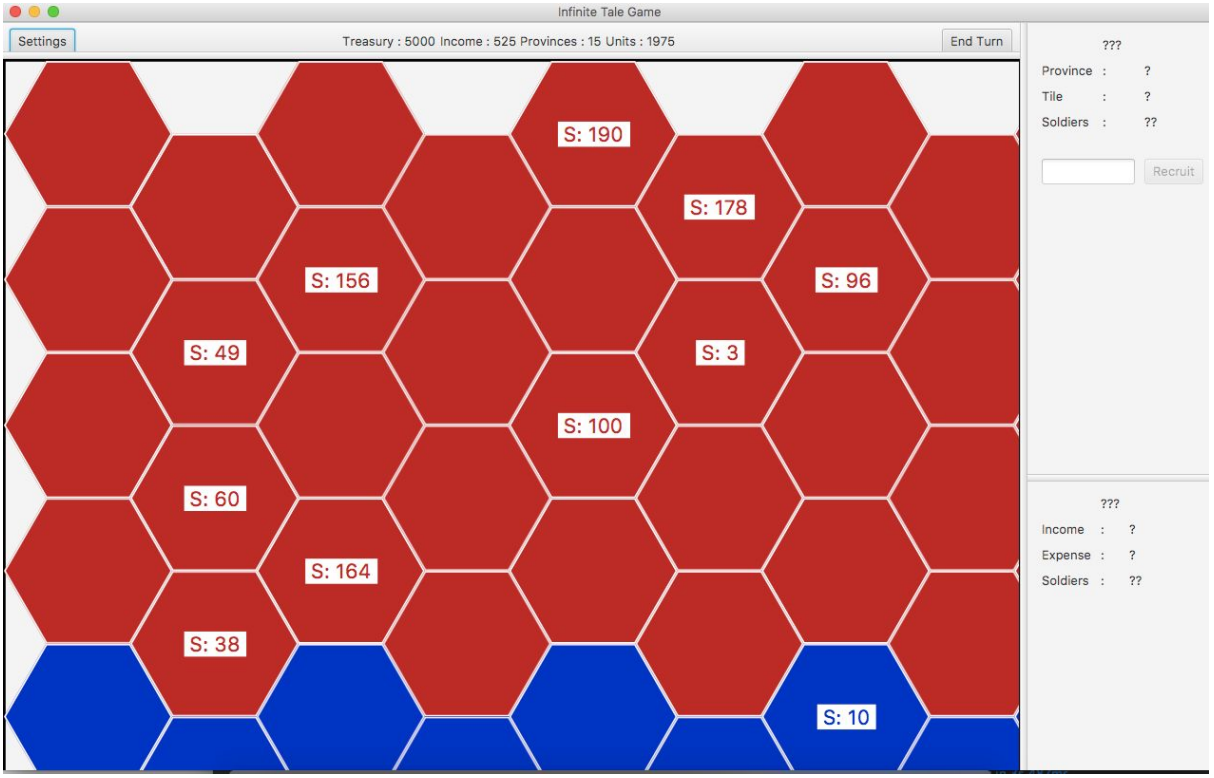


Figure -4- (Zoomed in)

3- Advancing Treasury and Buildings: We thought that it would be better to add some visuals to give sense of what Tiles look like and we added some Buildings and Treasury which change depending on your level of development.

Figures -5,6- ( Depending on your income, treasure progresses.)


Figures -7-8- ( Depending on number of troops and villagers, your buildings progress.)

## 2- Map Changes:

Promised:

1- We promised that conqueror would not start with a single tile anymore to increase complexity in game and we achieved it. Red tiles belong to conqueror and other colors belong to enemy. This increases complexity that player should consider and makes the game more challenging.

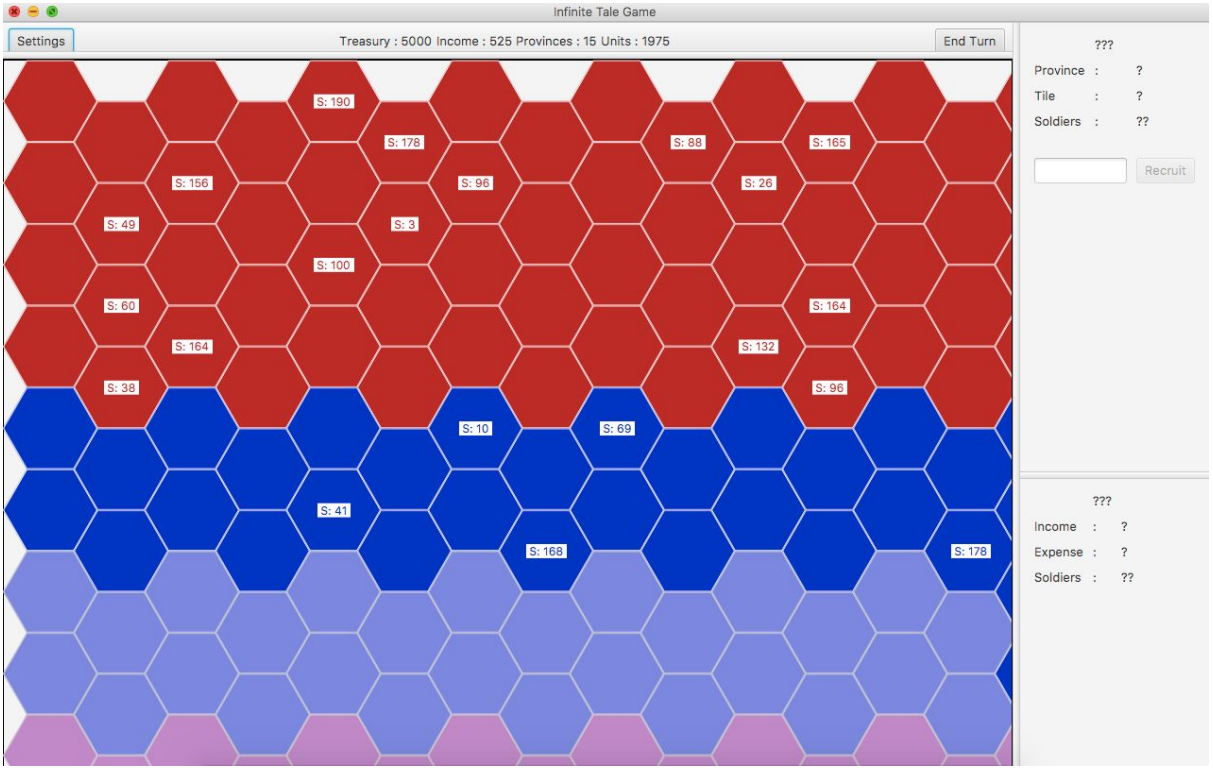


Figure -9- ( Tiles of Conqueror, Neutral lands removed. )

2- Also we promised that Neutral lands will be removed because it decreased the ways that player could take in order to win since they were easy to conquer and had the highest priority. We removed neutral tiles and now all other tiles belong to the enemy.

## 3- Read Lore:

1- We promised that we would write a lore of the game for the ones interested in it so they could understand how the name “Infinite Tale” was founded and could gain an insight of what is the purpose of this war.

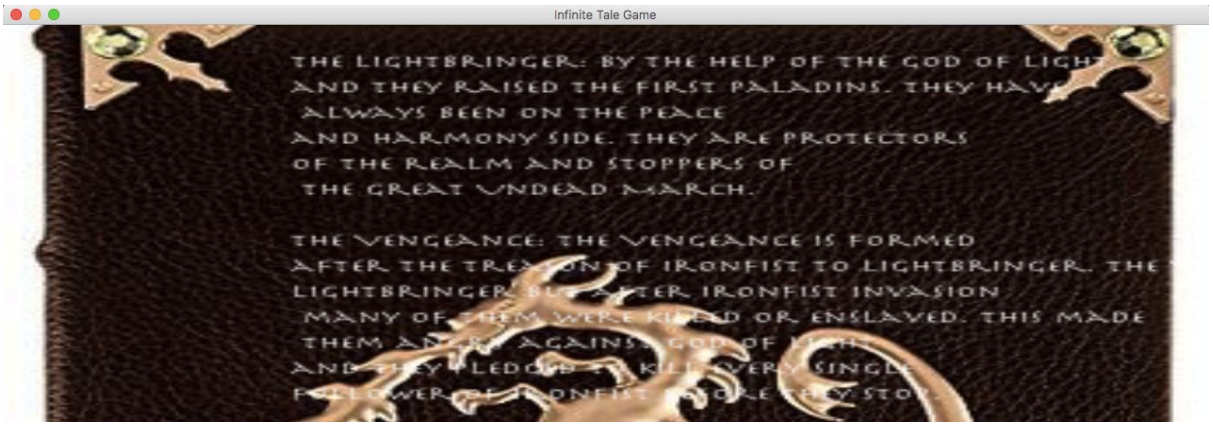


Figure -10- ( Read Lore)

## 4- Credits:

1- Credits can be read and the developers of the game can be seen.


Figure -11- ( Credits )

## 5- Battle Mechanics Improved:

1- As we promised, battle mechanics were improved drastically. In initial form of first iteration, battle mechanics were very simple, if attacker had bigger number of units they would win. In second iteration we changed battle mechanics to consider weather type and terrain type of the tile where battle occurs, they give certain bonuses to attacker or defender depending on the types. These attributes are randomly assigned to each province.

The battle works on rolling a dice. A dice for both teams is rolled and bonuses are added or decreased from the cap of the dice. For example attacker has 30 percent bonus regarding to terrain and weather type. Then attacker can roll between 30-130. Depending on number of army the attackers hit points are calculated and decreased from the number of defenders.

Hence we have added bonuses from weather and terrain, also added luck chance to our game which brings RNG factor.

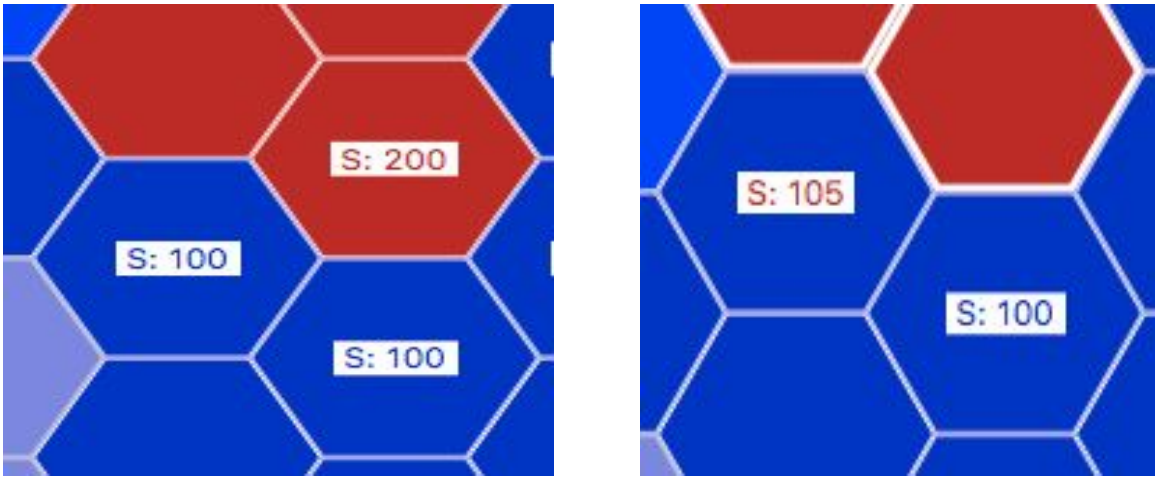


Figure -12,13- ( Battle Mechanics )

As you can see results of the battle are not as predictable as before.

## 6 - Better Visualization and New Musics

Better visualization achieved through making a smoother code for visuals. We added many UI improvements which we talked about changed the visualization and made

it better. New musics are also added to give a different taste to player if game takes long.

## 7 - AI

We lacked any kind of AI which will represent enemy players and will respond to player's movements. We added a fairly complex AI which will recruit soldiers on borders with player, attack and conquer the places they see rather empty and fit, also reassigns their units by moving them to "hot" tiles.

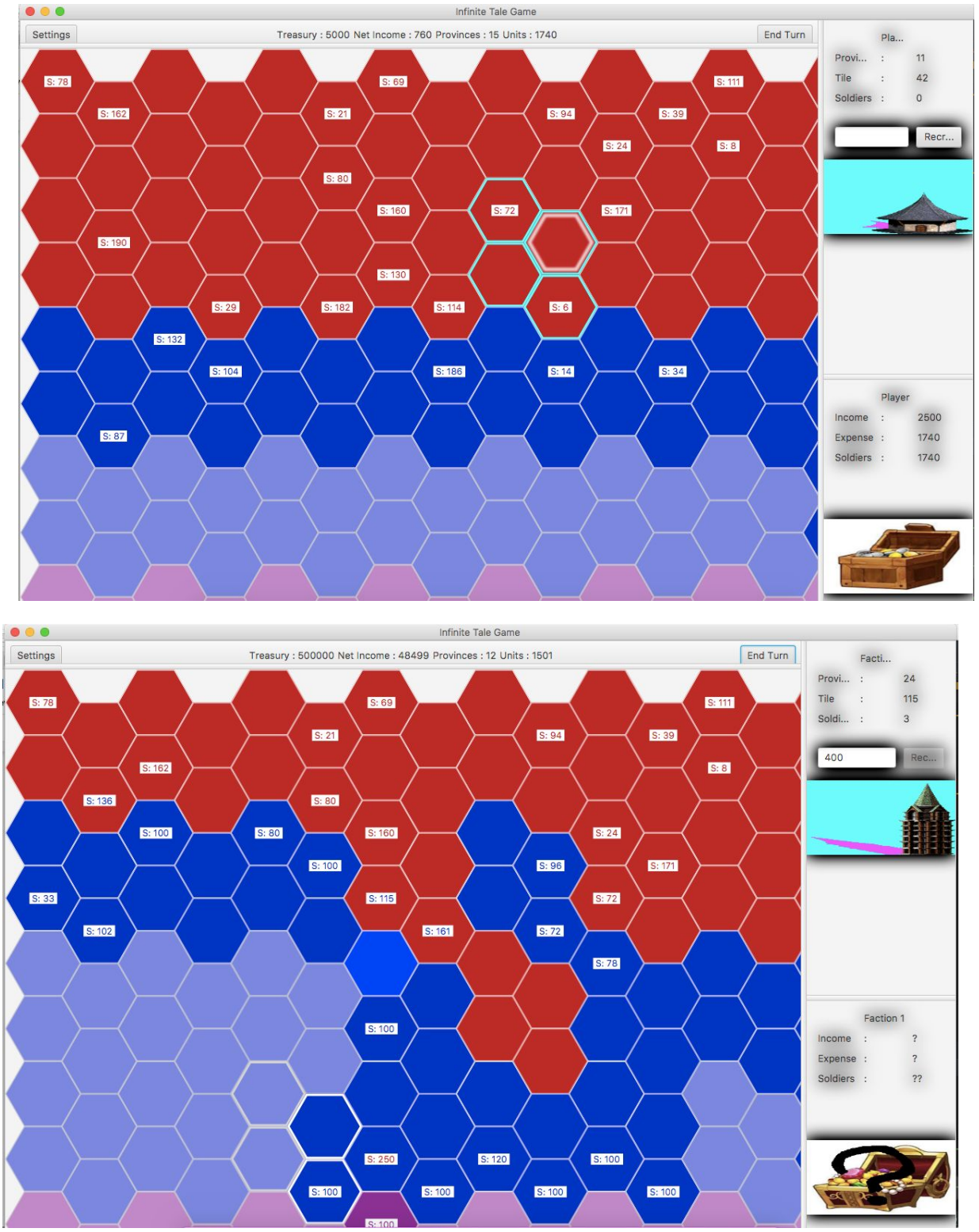


Figure -14,15- ( AI )

As you can see, AI can make decisions in order to conquer player's tiles.

### What we could not achieve in Functional Requirements ?

1- Save and Load of the Game:

We tried it but we could not achieve saving and loading of the game. It was hard to make our code compatible with Serializable.

2- Different Types of Units:

We did not implement different kinds of units, there is a single soldier unit.

# About Design

As we know, our object of this game is not to only fulfill functional and nonfunctional requirements but also the design patterns. We will look over what we promised and what we achieved in the design, also what we could not achieve if we have any.

### Façade Design Pattern:

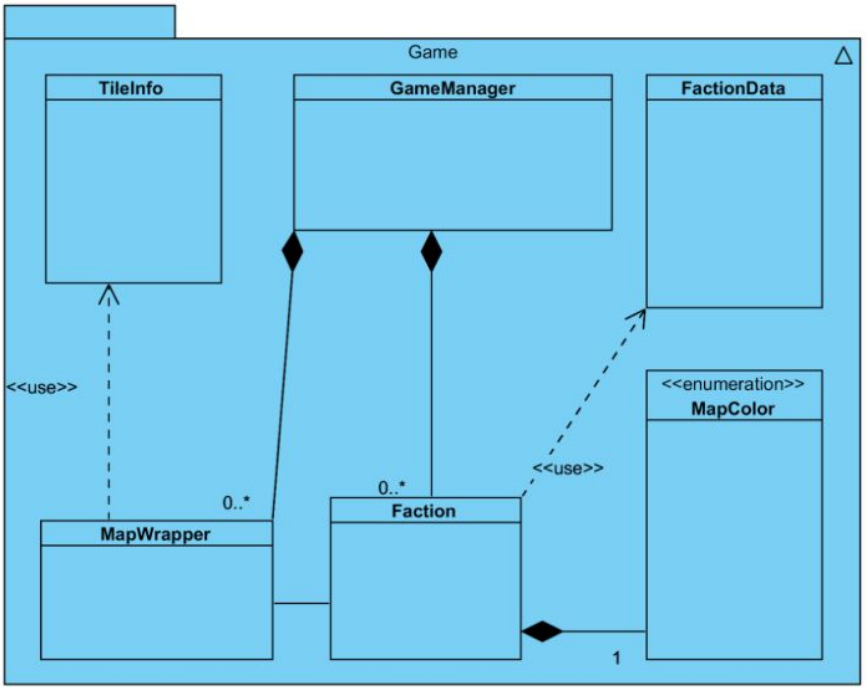


Figure -16- ( Façade )

As you can see here, we achieved Façade Design Pattern in all of our subsystems. In this picture the Façade Class is GameManager.

## Ball-and-Socket Approach ( Lollipop ):

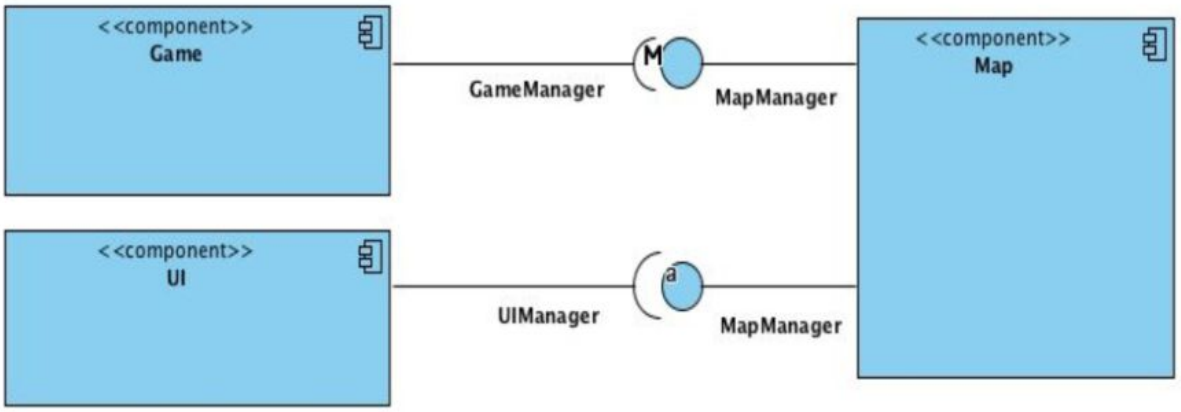


Figure -17- ( Lollipop )

We also managed to achieve lollipop approach to show what is going on in our program in a better way.

## Singleton Low Level Design Pattern:

We also made use of singleton low level design patterns on our manager classes. Singleton helps to initialize only one instance of every class during a particular execution. GameManager, UIManager, MapManager and all the other managers (ex: MusicManager) are designed with singleton because we need them to have only one instances.

## MVC Architectural Style:

We also managed to obey Model-View-Controller rules as we promised. Model classes are in subsystem Map, Controller classes are in subsystem Game and View classes are in subsystem UI.

There is no need to put all of these classes as a picture. You can check it yourself from design report.

## What we could not achieve in Design?

Actually, there is nothing in design that we could not achieve that we promised.

## Conclusion

We think we did a very good job in determining design patterns and obeying them which helped us hugely because it was easier to code and understand each other's work when talked in the language of UML and after making use of design patterns.

Also we were able to achieve many of our promised functional and nonfunctional requirements. We even added more functionality than we promised. There were some functional requirements we could not fulfill but happily those were not among the important ones considering the state of the game.

As a conclusion, we were able to fulfill the desire of this course very well and learnt a lot throughout this project.