CS 319 - Object-Oriented Software Engineering

Final Report

LOST

GROUP 2-D

Eren Bilaloğlu - 21401603

Gunduz Huseynli - 21402821

Onur Sönmez - 21300749

Yasin İlkağan Tepeli - 21301707

1. Implementation Process

Implementation process of LOST started with distributing the tasks among the group members. Since the software architecture was determined as three tier architecture, task division has been done accordingly. Two of the group members focused on application logic, one of the group members focused on data management and other group member focused on user interface layer. It was easy to divide the task like this because we used Façade Design Pattern. Altough we divide the tasks, it does not mean we didn't interfere another tasks. In order to have best efficiency in the least time, we all give support to other tasks.

If we think about the timeline, we first decide on the parts, who is better in which part and which parts are hard. After that we divide the tasks and give everyone a task. Throughout the process we constantly shared our opinion and develop the program. As first step of implementation, we started from Application Layer and inside of Application Game Objects are our first aim to complete since they are the main core of the program. After we test all objects we started to do the gameEngine and simultaneously User Interface and Data Management. We had some improvements while doing the UI and Data Management because as we implement the codes since we use libraries and learn them better while coding.

At latest, we test the layers and then try to integrate them. After we first integrate them succesfully, we started to test the complete program. We spent more than 2 days for finding bugs and debugging so that program does not include bugs. As we do it, we had to remove some options from gui because it had a chance to give error sometimes and because of limited time it was hard to follow them.

As a last step we put sounds, images and background to make it more user friendly.

2. Changes in Design

In our design report, we declared that two interfaces would be employed. Updatable and Interactable interfaces which were in design diagram have not been implemented. The reason behind not implementing Updatable interface is, this interfaces' methods must have different signatures in the classes that would use them. To clarify, Updatable interface would be used by GameEngine and UpdateManager classes. However, in the implementation process we realized that these two classes cannot realize same methods. Since number of parameters differed in these methods for two classes, we decided not to implement Updatable interface. Another design change, removing Interactable interface is caused by differences in needs of the classes that would implement it. Item and Character classes would use this interface to override getInteractionsList(). In the game, items can have more than one interactions, so they need to have an interaction list. However, characters can have only one interaction, which is fighting. Therefore, they do not need to hold a list to store just one interaction. During the design process, we thought that characters can have more than one interactions too but while implementing our game, we limited character interactions just with fight since our scenario does not need any other interaction with characters.

In design report, Food class was extending the BoostingItem class, which causes to Food class having attributes such as attackPoint and defensePoint. That

was a logical mistake for our game scenario, since food would not give attackPoint and defensePoint. Therefore, we changed that inheritance relationship. Food class has been defined as subclass of Item class, as having same hierarchical status with BoostingItem class.

Our last design change, which is relatively minor comparing to other changes, was making the Event class as abstract class. In design report, Event class was not defined as abstract class. That was a mistake because there would be no Event objects instantiated in the game. Just the subclasses of Event class would have their objects. Therefore, we changed Event class to an abstract class.

Other than these changes, we have made minor changes in class methods and attributes. We mostly kept loyal to our design report.

3. Status of the Implementation

Most of the project's functional requirements have been fulfilled. Almost all of the gameplay functions have been implemented (character and item interactions, navigation between areas, entering story events, implementation of inventories, image processing on map, storing records, main menu operations except load game, etc.). Due to time constraints, some minor bugs could not be fixed, because of this, game might crash time to time. Besides, due to lack of technical expertise, online data management could not be handled completely.

In LOST, we are using Maven for project object model. In persistency layer, GSON library for JSON-POJO mapping are used. In this case, initial data retrieved from the json files and converted into root objects. Because in game environment,

there are lots of object and in order to avoid poor design and syntax, we are using json files as a source for characters and items. Also Cloud Datastore Api was used, but there are problems while integrating it. In loading screen,the data does not preserve itself in some cases and unfortunately we did not fix it.

Considering our design goals, we have fulfilled user-friendliness and maintainability goals. Our code is easy to maintain and our user interface is self-explanatory. Performance of the game needs to be improved and with a better optimization, our code can be more robust.