Simon Fraser University
Faculty of Applied Science
CMPT 276 - Introduction to Software Engineering

Term Project - Phase 2

Implementation Phase Report

Group 5

Overall Approach in Implementation

Our group's overall approach still mainly follows the approach that we had initially discussed in Phase 1: Design Phase. During the first few days, we laid out all the features our game should have and discussed the importance of these features towards our game. In addition to that, we subcategorized all those features into functional and non-functional features. After laying out all those features, we turn those to-be-implemented features into individual modules in our To-do list with functional features having a higher priority.

In our initial cycle of the implementation phase, our group members all focused on the implementation of game logic which we believe as the core of the game. We did this in this way as it will provide the basic foundation and platform for us to branch off implementing separate parts of the game. Throughout the implementation phase, our project has two main branches which are User Interface (UI) and Gamelogic working concurrently. Each branch has at least a member working on it and we reconvene periodically to maintain the consistency of the code in the master branch.

Modification of the Initial Design Pattern

One of the main changes we did was the decision of splitting our game into interface and gamelogic. Our antecedent UML Design did not really account much for user interface design of our game as a separate package. Furthermore, we realized that splitting interface and gamelogic into two separate packages is a better design as we encapsulated much of the gamelogic implementation from the interface design and achieved better modularity with this design. Both packages communicate mainly through the core class of gamelogic package which is Board class. Furthermore, we also splitted out the User Interface into multiple small classes rather than bunching it up into one big class which was stated in the preliminary UML diagram. We categorized them by the screen that it will display.

We also made multiple adjustments towards our core class in the Gamelogic package which is the Board class. Firstly, we took out the method for generating the board and created a class for itself. We justified this modification because it brings more modularity towards the code and much effective information encapsulation. Furthermore, it reduces the size of the class which also brings better readability and prevents our program taking in the monolithic software design. Moreover with the respect to adopting more modularity and data encapsulation into our program, we also decided to create a seperate class for the Exit in the game.

However, we removed Barrier class from our UML class diagram as we decided that Barrier does not have additional attributes that need modulation based on the requirements and our feature list. Instead, we can easily handle Barrier with a set of matrices in the LevelGenerator class. We also took in account of balancing the right amount of modulation and integration in our program.

Management Process, Roles and Responsibility

Our management process differs slightly from our initial planned methods in the design phase. Instead of using “Doodle” as our task scheduling tool, we decided to use the built-in functionality of Git itself. We use the issue tab as a todo list to list out tasks and functionality that still need to be implemented. Git’s ability of assigning people to certain tasks let us know who is in charge of implementing those tasks and ensures all group members are synchronous with the current status and development. Discord was still our primary communication source and any concerns or suggestions are all communicated through there. We update everyone in the group through Discord when we push any new code into git with a description of new commit.

During the initial period of the implementation phase, we agreed on meeting every Monday, Wednesday and Friday in the morning to catch up on each other's progress and discuss some concerns and suggestions we had. We realized that meeting up in person as a group has a lot of benefits in our progress as we can all discuss together on a specific issue with visual aids such as a whiteboard. It dramatically helps in understanding the explanation given by our other group members. Unfortunately, due to the outbreak of COVID-19 following the university shifting to online lectures and promoting social distancing, we limited our in-person meetings and shifted towards online video call and voice chat. Although it is not as effective as in-person meetings, it still provides us with the basic functionality of discussing a topic in real time.

Our group member, Jocelyn spearheaded the user interface design of our game. She designed all the sprites, game screens, the implementation of buttons and text boxes in the game. Alex mainly focused on the algorithm which the enemy uses to chase the player and the implementation of the rest of the enemy class along with other small features in the game. Kyle and Jason are mainly in charge of implementing the other features of the game such as Traps, Rewards and Score. The other features of the game and debugging are generally done by all group members collaboratively. Although all of us have specific roles throughout the implementation phase, we always slide in to help each other out when there are concerns in the implementation or software design pattern used.

External Libraries used in the Game

No external libraries were used in this game and we mainly use libraries that are already included into the Java Development Kit (JDK).

Measures Used to Enhance Quality of Code

Consistency and concurrency are the main criteria for good quality of code. We all agreed on a specific coding style beforehand and generally what tools we will be using such as the version of JDK during the implementation phase. We all agreed to use the same IDE for this program which IntelliJ. This provides us with a consistent platform to work on. It is much easier to communicate a problem and debug when we have the same platform. We agreed on certain coding styles such as putting a code of a method in the same line as its declaration if it only contains one line of code. This improves the readability, consistency and overall cleanliness of the code.

Modularity and Information Encapsulation are also key things in enhancing the quality of a code. Modularity provides an easy and efficient way of adding new features into the game. Furthermore, modularity and information encapsulation greatly increases the efficiency in debugging and reduces chances of errors. In order to enhance the quality of our code, we incorporated both designs along with other software design patterns mainly the factory methods to take full advantage of the benefits brought through object-oriented programming.

Biggest Challenges Faced during this Program

Throughout the implementation phase, we undergo few challenges along the way. However, there are 2 biggest challenges that we encountered.

Firstly, we faced a dilemma of splitting our Interface from our game logic into a separate package. We would have to redesign major parts of the UML diagram to accommodate for this approach. This sparks concerns in our group whether we would have enough time to implement the program as we would have to spend at least two days redesigning our new approach in our already short time duration. However, in the end, we decided the benefits of this approach outweighs the risk involved and we decided to go for it. Throughout this period we were glad that we took the new approach as it greatly streamlined our design of our classes and implementation.

Lastly, as mentioned before, is the outbreak of COVID-19 around the world. Due to this outbreak, the university cancelled all in person lectures and shifted to online classes and promotes social distancing. It caused some chaos into our planned schedule as we would not be able to hold our regular in person meeting and coding session together

along with the uncertainty brought by the transition to online classes. This did cause a slight delay in our implementation progress. However, we quickly adapted and adopted online meetings instead.