CMPT 276: Introduction to Software Engineering
Project Phase 4 - Report

Group 13: Utsav Anantbhat, Swara Desai, Maggie Tan, and Hongrui Qu

---

The Game:

Our game is called “Resource Run” and it is set in a post-apocalyptic/dystopian urban environment. The player controls the gatherer whose goal is to gather all regular resources and bring them back to camp. They mainly gather food baskets (regular reward; +5 points) and gas cans(bonus reward; +25 points) if they are lucky enough to come across them and pick them up before they disappear. While carrying out their objective, the player must avoid the pit traps (static enemy; -10 points) and the scavengers (dynamic enemies). The player loses the game if they run into a scavenger or if their total points go below zero. The player wins the game by collecting all the food baskets, then going to the ending point.

Our original plan was more of a guideline of how we would go about the project. In other words, we knew that whatever we had planned in the planning stage was not set in stone and will be changed along the way. The user case scenario helps us to develop a deeper understanding of the requirements and guides us to understand what features our game needs to implement. Our first UML diagram was created so that we could wrap our heads around what are the main classes and the kinds of methods we would need for this project. The way we had organised it originally was not the best as we definitely could have broken the big classes into smaller ones and some methods were not needed. All this was changed as we implemented the game. For example, we made it so that the similar classes such as the different resources (regular and bonus) and trap classes inherited from a superclass contained the features an object would have. We broke things down into smaller items to reduce the amount of coupling going on in the code. By the end, the code organisation was a lot better compared to what was planned in phase 1. In the beginning, we had a small list of features that we were considering implementing but due to time constraints and our busy individual schedules, we were unable to get to this list. In particular, if we had more time, we would have

liked to implement a random map generation so that every time the player starts a new game, it would be a different layout each time.

For the biggest challenges, we had little experience and had a difficult time with Maven at first. It is also challenging for us to continuously improve the quality of code to be more reliable and maintainable. However, we also learned the lesson that teamwork is very important. We help each other to overcome any technical difficulties. Git version control also provides more flexibility to perform group work in terms of helping us track changes in files and establishing the coordination required between different people to work on them. Through working on this project, it was made clear that planning things out, in the beginning, helped make the progression of the project a lot smoother because we had an idea of what we were doing and in what direction we should go. It was also important that we held group meetings frequently to share our progress and make sure everyone was on the same page because it allowed us to steadily progress through the project without a hitch.

UML Diagram:

Our team uploaded an updated version of the UML diagram as a standalone file in the document repository. The UML diagram helps to visualise the architecture of the final design of our game.

Tutorial:

The following link is a video showcasing the features of the game and how it is played:

https://youtu.be/jF5Epvs29X0