Stephen Dao, Danyaal Patel, Peter Salnikov, Brendan Saw

# CMPT 276 Phase 4 Group 20: Escape From Corona

## Description

The game, 'Escape From Corona', follows a pandemic theme. It includes menus that allow for users to choose to start the game, restart the game, view instructions, and exit the game. The game also includes sound effects and music. Through a tick based system, the user can move the main character, Honnie Benry, once every tick. Enemies, represented by the corona viruses, can move every other tick. The user aims to collect all vaccines (checkpoint rewards) to open and reach the exit. Masks and sanitizer bottles (bonus rewards) have a chance to generate on every tick and provide an increase to the user's protection level (score), but only last for a limited amount of ticks. Geese (punishments) are stationary and can decrease the user's protection level when stepped on. Trees (walls) surround the map and create the level by preventing the movement of entities.

## Comparing the Final Product to the Original Design

We thoroughly deliberated the design of our code to minimize the number of changes we need to make during implementation. We built our UML diagram in a collaborative program and shared ideas among ourselves, which allowed us to catch design faults more effectively before we wrote code. This reduced our potential for technical debt that we would have needed to pay later in the semester, and the diagram served as a solid guide to build our code upon.

We also followed our modular UML and skeleton diagrams, identifying TileAction, BoardDesign, Characters, and Game packages and various classes within. The Game package was changed to "Core" and adopted some other classes. Outside of that, every class and package made it to the final version of the game. Sticking to this plan helped us tend towards low coupling, high cohesion, and the avoidance of large functions, for clarity and brevity.

Not surprisingly, new methods were added as needed, including Getters and Setters. In Phase 2, we changed some of our initial design choices, such as changing MainCharacter to a singleton class. Other changes involved creating boolean variables to check whether certain things were true or false, to help with game logic.

When it comes to actual game features, we did not change much from our actual initial design. There is still an entry tile, exit tile, and the main character is confined to within walls. Scores(protection), and punishments/bonuses all function the same as initially planned. Collecting all the checkpoints to complete the level and open the exit door, as well as having stationary secondary enemies and non-stationary primary enemies are all features that stayed the same.

The Menus and UI on the other hand, were completely revamped and are unrecognizable from the initial plan and design. This could be due to not yet having a sense of identity or a theme at the time of Phase 1, and focusing more on the actual game functions. The actual button functionalities are similar, with a vertical column of 3-4 rectangle buttons for each screen, ranging from “start game”, “instructions”, and “exit”. However, extra zing was added through the form of button indents upon hover, glows upon select, and opacity and colour changes on hovering and unwavering over the buttons. This all added a great feel and overall look to the game, and custom created backgrounds for the menu screens topped it all off.

Sound effects, and the theme audio for the game were never discussed in the initial plan, however were added to the game in the end to create immersion and add to the full experience.

**Lessons**

Throughout this project, we learned many things that aided in the development of our program. As stated in the first report for this project, we decided to use the evolutionary model software development model to ensure that we build upon our previous codebase, improving our work through multiple stages in the development lifecycle.

A result of the way that we decided to develop our project was that we learned that better/more planning and communication between group members was necessary for optimal implementation of project components. We found that our group meetings were somewhat infrequent (about once a week, on weekends) which often caused some form of code conflicts when we merged large amounts of different code together. Perhaps, taking a more intense approach to development and merging more often would allow for less time spent on resolving merge conflicts, resulting in a faster development speed.

We also learned an important lesson about modularity with lower coupling. While we tried to stick to our UML diagram which was relatively modular on its own, when we devised our UML diagram, we didn’t consider all possible concepts and dependencies. This lack of greater foresight would provide a greater challenge as we tried to fit new code and methods into these existing foundations. Due to this, we found that we had to refactor a large amount of code to lower dependencies and separate functionalities in order to make testing easier.

**Tutorial**

We decided to create a video tutorial for the project, which can be found at the following link: https://www.youtube.com/watch?v=l2fZTEQB7kQ

**Already Created Artifacts**

Included artifacts can be found at **...\Project\Documents\'Phase 4 JAR and Javadocs'**. Though instructions to create the artifacts independently are located in the README.md.