CMPT 276 Phase 4: Report

Group 8: Darren Jennedy, Setu Patel, Steve Lam, Samuel Wong

For our term project, we designed, implemented, and tested a 2D university-themed arcade game named “How to Not Fail University”.

Game Description:

The player plays as an unnamed university student whose goal is to graduate university without getting expelled. To graduate, the student needs to finish all 40 of their course requirements by collecting all 40 P grades, which indicate a passing grade on their courses. The P grades give the student 2 points each. Once the student has collected all 40 P grades, the previously locked exit doors will open and the student will be able to walk past the exit doors to graduate, winning the game. There are also F grades which are scattered throughout the map. If the student collects an F grade, the player loses 5 points, and if the point total of the player becomes negative, the student is expelled and the player loses the game. Additionally, A grades will appear and disappear randomly throughout the game, and collecting A grades will grant the player 5 points.

Throughout the game, there will be evil professors chasing the student. If the student is caught by one of the professors, the student is expelled and the player loses the game. The university is also full of construction zones which appear and disappear randomly, blocking passage for both the student and the professor.

Changes and Justification:

While there weren't any changes to the initial use cases of the game, the final product of the game did receive changes from our initial narrative, graphical interface design, and class structure design.

Initially, the main character which the player plays as was supposed to be a computer science student. However, we felt that limiting our main character into only one category can create a rift between the game and the player. We wanted the game to be relatable to all students, not only computer science majors. Therefore, we changed the main character to not have a specific major, to not let our players feel left out.

Following the narrative change, some corresponding user interfaces also have to change. In our original design, there was a list of computer science classes that the student was supposed to pass before graduating in the game interface, located beside the map of the game. Due to our previous narrative change, we had to change the classes to not be specific to computer science. Furthermore, we also realized that having a big list of classes covering a big portion of the screen reduces the screen real estate of the main game map itself. Because of that, we decided to remove the list altogether, increasing the size of the map instead.

The biggest change from our original design was our changes for the design of our class structure. While our initial design strived to achieve good separation of concerns, during implementation of the game we realized that we needed more classes to maintain that separation. For instance, adding the GameResult and GameEffect modules to communicate the game results (condition of the game when the game ends) and game effects (score updates, whether or not the player lost, etc.) to other modules in our class structure. We also realized that some of the previously designed classes in our original design were redundant, and we trimmed those classes in the final product. An example of one of these classes is the DirectionOrNone enumeration in the movement package in our original design.

In our original design, we also did not consider the class structure for the graphical user interface rendering, such as images or GUI menus. In our final product, we added modules responsible for creating those graphics, such as our GUI module which is responsible for creating the menu panels and the MainFrame class, to maintain the game frame. We also did not consider sound effects and background music in our original design, something that we added in our final product. Therefore, we added classes which maintained audio responsibilities, such as the SoundEffects class and the BackgroundMusic class.

Lessons We Learned:

There were a lot of lessons learned from this project, but the biggest lesson we learned as a group is the importance of teamwork and communication when working as a group. Throughout all stages of the project, the design, implementation, and testing of the game, our team has always had frequent meetings. We had meetings every 2-3 days, and for each meeting, we would discuss what we have accomplished since the last meeting and what we should do next. We also discussed any difficulties or problems we might be having in our tasks, and helped each other in solving those issues. Because of this system, even though most of our team had little to no experience in programming in java, game development,graphical interfaces, git, or maven, we helped each other to understand the task we had at hand and managed to catch up quickly. Without good communication, we wouldn’t be able to help each other to understand the topics we needed to complete the project.

We also learned how to implement good design principles in our code. Throughout the project, we took extensive measures to make sure that our classes had as high cohesion and low coupling as possible. For example, when we were trying to show the score and time during game runtime, we realised that we didn’t have a good method in communicating the game updates throughout different classes. We also realized that we had a monolithic world class, where the world class was not only responsible for generating and maintaining the game world, but also maintaining the game updates. Therefore, we added modules responsible for the state of the ending of the game and the runtime updates of the game. These modules were created to communicate information without introducing interdependence, and to separate these responsibilities away from the game classes or the world classes.

Video Demo:

We created a video demo for the game, you can watch it here: https://youtu.be/RxVfeI6rw6c