Report

By Team 12: Sahil Janjua, Daniel Dai, Matthew Maxwell, Antonio Benzan

When we started development on our game Thief, we began by meeting together to discuss on how we would tackle this large project. We started by dividing the packages between all four members of the group. Antonio was given the responsibility to complete all the code needed in the game package, Matthew was given all the responsibilities in the map package which contained the code to generate the maze as he had previous experience with map generation, Daniel handled all the user interface components, and Sahil handled the character package and this report. Matthew, who was the most experienced programmer in the group, took a leadership role, and helped dictate how a project of this magnitude should be completed. While we all worked on our separate packages, we would meet two to three times a week to discuss and review the code each member had written, and by using the screen share feature on zoom to allow one person to write a class while the others gave instructions and help on how to implement the classes. As we continued to develop the project, we noticed that there was a lot of collaborations from all the members of the group in every class, including the ones that were assigned to be done separately, this led to an increase of productivity within the group.

While we all had experience with coding and individually developing our own projects, none of us had worked together in a group project. So, when we met, we wanted to approach the project by efficiently and seamlessly creating the smaller classes individually that would make up our game and slowly building our game's base and logic, initially we believed that by using this method, we would be able to avoid any major issues or changes to our code, later on we realized we were wrong. As we continued making the game, we noticed that without any sort of collaboration between the members, there were always issues on how to properly implement certain features so that we could properly follow the concept of encapsulation and maintaining

the concept of having a project that is loosely coupled and highly cohesive. After a week and a half of noticing these problems, we started making sure we had more meetings and that we were using our time in these meetings properly. We started making sure that we had code reviews and we made sure that we broke down each class to the most minor of details so that we all knew how classes should be coded and what changes were made to the code that could affect other people's work. When changes were made that could have potentially affected other member's code, the person that made the change would usually inform the group in the discord, so that everyone would be aware if they didn't thoroughly look at the commit history. With our approach becoming more collaborative, we found that not only did we increase our efficiency, but that our code continued to become more polished as the weekly meetings continued.

When we crafted the initial drafts and concepts for our game, there was a level of overconfidence and maybe even arrogance that the group had. The original plans were that we would be able to get the logic of the game done within the earlier weeks, leaving a good chunk of the time to not only refine the user interface, but to add elements that would separate our game from all the other group's games. In our use cases, there is talks of the inclusion of things such as the manipulation of sound in the game, the choice of including or excluding additional guards and toggling music among other such settings that would allow the user to customize the game more. While we see the value in these features and allowing the user more chance to customize their game experience, we wanted to spend our limited time making sure that all the elements of the game were working at a specific standard of quality. To make sure that the quality we wanted was achieved, we adopted a method that used preset options so we could focus our time on making sure the game functioned to the best possible standard. Another adjustment we made was moving from standard difficulty levels that we indicated we wanted to use in the use cases, to

levels where we scaled the difficulties so that it would become more difficult as the player chose a higher level. This minor change was made because we felt we had more opportunities by going down the level route, an example of this being the random maps option where we could play with the concept of different maps. When it came down to adding new methods that weren't on the UML diagram to classes, there was usually one of two reasons why we made modifications. It was rather to breakdown each function so that they would be easier to create and easier to implement in such a way that it wouldn't cause troubles with other group member's code, or, to aid in the process of creating the UI. Examples of some adjustments include the addition of the drawInitial, updateUIPOS, loadImages methods and the image variable which are prevalent in certain classes such as guard and thief, where they work in conjunction with update methods to get the characters of the game, the coins and the chest to visually move or disappear when the player does certain actions. A small alteration we made was changing the character class's name to entity to avoid confusion between the class and the character package. Another alteration we made to our work, was moving the move class which existed in the entity class, the thief class and the guard class to only existing in the entity class. This change was made to take advantage of the inheritance relationship that existed between these classes where thief and guard inherited from entity as this was a major focus in the class, and this ultimately made it easier to make the game. When we started the design portion of the game in phase one, a difficulty we had was deciding how to create the maze. We eventually decided that we would wait until we started coding to decide on our implementation, so a lot of new methods also came from choosing a way to create the maze as we needed the methods to support the frameworks of maze generator. A final batch of modifications we made were the several additions in the thief class such as the

direction, score and kill method. These were added so we wouldn't clutter the game and play methods, and so we could practice what we learned in class about encapsulation.

At the start of the development phase, there was a lot of discussion of on what would be the best way to go about implementing the graphics. Since Daniel was in charge of handling the user interface, a large part of the responsibility of deciding what library to use fell on him. Through a suggestion from Sahil, and experiences Daniel had with JavaFX where he realized he could use its components in our project, JavaFX was the library we ended up using. There were discussions about exploring different libraries to accomplish our task, but, JavaFX seemed the least complex and suitable for our needs and after seeing the end results, it ended up being perfect for the project.

The biggest challenges we faced during this project was the previously mentioned early lack of communication between the group members, and also a difficulty of lowering our expectations to a realistic level. There was always a desire for us to create a game of a certain standard that we may have been able to reach if we had more time, and at certain points that desire turned to expectations on how this project should look. A lot of our doubts of not being able to match these expectations were amplified with the daunting task of creating a game of this magnitude, and the slow progression of the game development process. As we started to see progression of the game, and we started to get more comfortable working together as a group, we realized that making the game wasn't as bad as we expected. Due to certain measures we took during our development of the game, we reached a version of the project that was close to what we initially envisioned when we were in our planning phase. These measures included meeting on discord two to three times a week and doing code reviews and coding sessions, and keeping a few weeks at the end of our development to play the game several times and iron out any bug no

matter how big and small. We did this until we were confident that the player would be impressed by the fluidity and depth to our game.