Individual Report

Summary

This project is an application of Sorry, which follows all rules of the real board game but users play with computers through an interactive interface. Users can decide pawn's color, the number of computers, and how smart and how nice they are. To play the game, users draw a card first, select an option, and then click the first pawn or skip if there's not any possible move. After the pawn moves to the destination, computer will start its turn automatically and pick an options and pawns according to their settings. Users can also save a game and resume it later. After someone wins, a summary of this game will show on the screen and users can go back to the main menu and check the statistics.

The implementation of this game was divided into four parts for all team members. One implemented board GUI, game logic, and pawn movement. Another was responsible to instructions, computer AI, and testing. Another developed main menu GUI, statistics, possible movement, and created images. The other finished database, deck, and save.

Development

a. A short description of what the system does

The system provides an interactive interface for users to play Sorry game. Once a new game starts, users type in their names, select a pawn color, set difficulties of all computer opponents. Then, selected color will be placed at the bottom of the board, and all pawns will be placed in their start area. A relaxed start button is provided for users to start the game easily, because they may not want to wait for card 1 and 2 in the beginning. Users can follow instructions shown on the screen and click on buttons or pawns to trigger next event. Users can save a game and resume it later, and all pawns and the deck of cards will be maintained in previous status. When a game ends, users can see their game summary. In addition, they can find overall statistics in the main menu.

b. How the simulation engine works

There is a for loop in the main program, where the program detects mouse events, processes the game, and draws objects on the screen. If a button is shown on the screen, users can click on the button. Also, based on the situation, they can click pawns of their own or of opponents. During a player's turn, the player draws a card first. If there is no any possible move, a skip button will be shown on the screen; otherwise, the button is still invisible. There may be one or two options, so the player needs to select an option even if there is only one option. Then, the player can click on a pawn of the player's color. In some situations, the player needs to click on the second pawn either of player's color or an opponent's color. After selected pawns

move to destination, computers will draw cards and move their pawns automatically based on their difficulty settings.

c. What the components of the application are and what part they played in the development of the system

All components are organized into four parts. The first parts includes board GUI, game logic, and pawn movement. The second includes instructions, computer AI, and testing. The third includes main menu GUI, statistics, possible movement, and images. And the last includes database, deck, and save/resume.

Based on the steps users will experience, they see the main menu GUI in the beginning. They can view the statistics and instructions here, or they can start a new game or resume a game. The color and difficulty settings are designed in the main menu GUI. Once a game starts, the board GUI is shown. The GUI maps a pawn's position to the real position and draws all pawns and buttons on the screen. The game follows the logic and players draw a card from the deck after their turn starts. Then, the program checks current player's pawns position and find possible moves. If there's more than one move, the player can select an option and click a pawn. Then, the pawn will move based on how the program processes this card and move one step at a time instead of jumping to the destination. Computers choose options and pawns based on their AI setting. The game status is stored when users click the save button, and the game can be resumed from the main menu.

d. What part the other members of the team played in the development

I implemented board GUI, game logic, and pawn movement. Gavin was responsible to instructions, computer AI, and testing. Joseph developed main menu

GUI, statistics, possible movement, and created images. Peter finished database, deck, and save/resume.

e. How the software was tested

The game was tested by all members but mainly by Gavin. We played the game continuously and tried all possible events. Saving and resuming were also executed several times. Once a bug was found, we explained the situation to others when that bug happened, and others tried to fix the bug. After a bug was fixed or even if it was temporarily ignored, we continued playing the game for testing.

f. How much time and effort was spent on the four activities of software development (specification, development, testing, and deployment).

We spent our second meeting on playing the game and discussing about the specification. Also, we studied for the instructions and wrote a document to list all requirements after that meeting. Therefore, we spent about 4 hours in the beginning on the specification. Sometimes, we discussed about the specification during our meetings again to clarify some questions and designed our program again, so we might have spent more time on it.

I usually scheduled 4 hours per week for this project, and more as the due day approaching. Therefore, I spent about 35 to 40 hours on the development.

For testing, before all weekly meetings, I tested my program and checked if all implemented features could run properly. Also, all members tested the project after finishing it and before our presentation. In conclusion, I personally spent about 8 hours on testing.

Finally, we presented this project during our presentation. That is the first time we showed the project to public together. I also showed this game to my friends, but I did not play through the whole game. So I only spent less than one hour on the deployment.

Analysis

a. Describe which activities were accomplished easily, and which were difficult and why

Requirement gathering was the most easy activity, because instructions of this game and other rules were provided on Blackboard. We only needed to read through all instructions and play the game to get familiar with the game.

Design and development were more difficult. In terms of design, it was hard to consider all included parts in the beginning. For example, we did not realize how hard it would be to process an option based on the card and the chosen pawn. We spent a lot of time on discussing how to implement some features during meetings. In terms of development, the main behavior of the program is just a loop for detecting event, processing game, and drawing objects. We move a pawn by changing the position and then drawing it on the screen. We were not familiar with this kind of class processing. Therefore, we spent unexpected time on learning this structure.

b. How members communicated with each other and whether this communication was effective

We had a meeting weekly in a library study room. We usually discussed our progress and problems. Sometimes when we had a quick question or we wanted to remind others some information, we left messages on Facebook. We also used Github to do the version control. These tools were enough for us to communicate with each others, because we discussed complicated problems face to face and simple problems via messages.

c. What errors were uncovered during testing

There were many errors found during testing, because we did not consider all possible situations when implementing our program. For example, there were a lot of flags to show the status of the game, so when Peter wanted to resume a game, he needed to set all flags. However, I declared some variables in functions instead of in the constructor, so he could not know the variables until there were error messages. Moreover, I did not know “is” and “==” have different meaning, so I used “is” a lot. This misuse caused some errors but I spent one night to figure out the error was caused by the if statement containing “is”. Furthermore, we highlighted pawns that were possible to move, but sometimes they were not unhighlighted after a turn.

d. Whether development actually proceeded as planned

I would say our development process followed our plan very well. Sometimes we could not finish assigned task for that week, but we usually spent more time on finishing it and did more for the next week.

e. What worked well and what didn't work well

Our meetings were really efficient and we also wrote weekly notes during our meetings. The note included what to do, what we had finished, and what we needed to prepare for the next meeting. We first talked about what we worked on and discussed problems and tried to fix them together. Then, we assigned tasks to ourselves or if there was something needed to be done, we asked someone to do it. I believe this procedure made our meetings more efficient than all of my experience.

We had some conflict issues on Github. We found the conflict when we encountered the same bugs and tried to fix the bugs. Finally, we found that we had fixed that but the code was recovered to the old and wrong code. Also, computer AI

was disappeared after some commits. Gavin needed to rewrite it just one day before our presentation. Fortunately, we could follow our commit history and rewrite the code easily.

f. What you would do differently if you had to do this again

I would try to build our project after all members are familiar with Git. Merge conflicts is an annoying problem. Therefore, if all members could know how to fix it, it would be much easier for us to prevent this issue.