Group Report

Coursework Property Tycoon (Team 22)

From the beginning of the project, we agreed on having a team leader (Vivien) who was responsible for managing the project, documenting the sprints and the current project status, leading the meetings, writing the meeting minutes, doing the requirement analysis and keeping an eye on the project timeline as well as the deliverables. The four other team members (Foret, Hallton, Hayden and Tang) were responsible for the implementation of the project including the code base, its documentation and the testing. All of us were working equally on the coursework and divided the tasks we have scheduled throughout the semester. Since we worked with an agile approach, the development of the application Property Tycoon was organised in sprints. In the beginning we planned 2-week sprints but had to adjust it because of time clashes and the experience that our planned tasks could be done in less than 14 days. We started with setting up the project in the first teaching week of the spring semester, but did not make much progress in the first weeks. However, when we found our working flow and shortened the sprints to a length of only a week or even less to enhance the communication and having more internal deadlines, we were able to work continuously on our game.

Due to the covid-19 pandemie, we worked mainly remotely on the project. This came with some difficulties, e.g. being in different time zones, not seeing each other, and especially in the beginning, having less direct communication than planned. It also took more self-discipline to do the uni work from home since it was easier to get distracted. Additionally, we had to refine the broad tasks we had defined in the beginning of the project and we spent more time on communicating and coordinating the tasks than under regular conditions with real life meetings.
Nevertheless, we were fortunate that we all had access to a working internet connection so we could continue to work on the project and did our best to handle the uncertain situation. We managed to find time slots which worked for everyone and had several meetings where we just talked about the current project status and clarified tasks instead of finishing and reflecting the last sprint. Although it took more time to coordinate the project, this helped us to achieve a common understanding of the project work and the specification as well as helped us to assign detailed tasks to team members. At the end of some sprints, we had to push some tasks into the next sprint, because we did not manage to finish the features in time. But by using the agile development process we were able to adjust our initial project plan, schedule tasks in a more flexible way according to our timetables and divide work into several smaller tasks or even improving some features in the following sprint.

The following describes additional lessons we have learned during the development of the game Property Tycoon:

- The agile development process helped us to adjust to external factors that we could not control (the outbreak of covid-19) but influenced our coursework. By adjusting the project plan, switching to exclusively online meetings, increasing the communication by having shorter sprints, we were able to continue working on our project and finish it on time despite the unforeseeable circumstances.
- The communication via our discord channel worked well for issues that had to be fixed quickly, but did not work for status updates, where we preferred (online) meetings.
- The more calls we had, the more active we worked on the project. Having smaller tasks and shorter sprint circles helped us to stick to our tasks and work more efficiently.
- Planning the next sprint in advance (before the sprint planning meeting) helped to have a rough plan as a basis for the discussion which can be then adjusted according to the current

status of the tasks and the time we have in the following week. The same applies for the requirement analysis: Before implementing the feature, the task card and the requirements have to be written and clear to the one who implements the feature.

- When assigning the tasks, it was useful to look at the previous tasks and who implemented which feature. We tried to have one responsible person for each class that implemented all features related to that class and hence, save time and development effort.
- The importance of regularly updating on the newest code version is something we have learned the hard way. Several times, we had recurring bugs that were already fixed by someone. This resulted in merging conflicts and extra effort to solve these issues. Hence, it is extremely important to update the branch with the newest code version, merge into the master when the implementation of a feature is finished and without any issues.
- Github is a useful tool to manage a project, especially for a team working together and despite some merging conflicts it worked well for us.
- Command line is more powerful to manipulate git than gui.
- If you notice an issue in the code, it is more efficient to notify the team so the team can solve it together rather than trying to fix it by yourself

GUI:

An issue that became apparent at the start of development was that our team members had very little GUI and 3D game development experience. Due to this, we decided it would be too much work for one individual to learn how to, and create a reliable GUI with no previous experience but, assigning more group members to help would have a too great impact on development of the code base. Thus, we decided a very basic GUI consisting of buttons and text boxes should be implemented to demonstrate the functionality of the game and the beginnings of a more advanced GUI could be implemented later in development. This would allow us to focus on implementing a robust code base so that GUI / game developers could continue development of our project and easily implement an advanced GUI / 3D game engine to control our game. Nearing the end of development, we decided that development of a more advanced GUI was not the most effective use of the remaining time in the project, and focused on improving the separation between the code base and the existing GUI we had implemented. This would reduce the amount of work for an external team to create an advanced GUI or implement a 3D engine as they would have to create less code to have the game work under a new GUI. To this, we decided to implement a command line interface for the game, to help increase the separation between code base and GUI further and make the game easier to understand from a developers perspective.

Testing:

Despite the usual practice of continuous testing in an agile development process, we did not manage to always test and document the features right after implementing them (in the same sprint). The initial goal was to test all implemented features after each sprint, using the defined requirements as a basis for the test cases and add specific tests if necessary. Everyone was responsible for testing its features. Reflecting our development process, there were some sprints in which we tested the finished task cards based on the requirements, and in other sprints we did not test at all. This is something we should have improved and put more focus on.

Open Issues:

- Sometimes the dice read 0 or null”

- Double!" message will become stuck and not disappear with the next player's turn
- When potluck or opportunity knocks card is actioned, the player cannot know what the card reads or its actions
- (Suspected) moving over (but not landing on) a potluck or opportunity knocks place may sometimes activate it
- Agents can be added to the game but they do not pass their turn to the next player
- No leaving the game checks (currently the last player can leave the game, causing it to crash)
- Some users encounter JavaFX errors, this could be down to their specific platform and version of Java
- On some Windows machines, the command line input does not appear until you press enter. The reasons are unknown and may vary across PCs.
- First roll may trick roll again.
- Code design issues:
    - For some procedures the GUI has to include logic to allow the game to function, this can be moved to the code base to improve the separation between code base and GUI
    - When potluck or opportunity knocks card is actioned, the player cannot know what the card reads or its actions

Please find all records of any group meeting in the folder "Meeting Minutes". Our sprint documentation includes the summary of actions and documents our development progress in general. Additionally, a detailed project plan can be found in the folder "Project Plan". The detailed guidance about how to run the game is described in the Game Manual.

Link to GitHub: https://github.com/jht412ntu/PropertyTycoon