Mini Project Report

Justin W. Flory, Timothy Endersby

Approach

- How did you first approach this assignment?
    - Did you start coding, meet and discuss first, other?
- Did that approach change during the course of the project?
    - In what way(s)?

Before we began writing code, we decided to first meet and discuss the logistics and possibilities of the project. We believe it would be easier to map out the project in advance before diving into the code head-first. At our first meeting, we established all of the classes we thought we would need and created rough ideas for the different parts or sections that each class would contain. On Trello, we represented the different elements of each class as a "card", similar to the Kanban style of project management.

As we went along with the project, we found that certain ideas or parts of our initial plans were not the most ideal way of implementing the game logic or parts of the GUI. As a result, some of our beginning blueprints were modified along the way, but overall, we did do a fairly effective job of mapping out our project in the beginning. The only changes we made were to parts of the plan that were overly specific. The more general ideas were fairly accurate to how we actually implemented Othello.

Technical Problems

- What significant programming problems did you encounter?
- Were there planned items that didn't make it into the final program?
    - What were they? Why do you think they didn't make it?
- Describe what were the trouble spots with this project? Both Technical/Non-technical.

The largest problems we encountered were with the actual logic of how the game board and game pieces interacted with each other. After discussing how we wanted to go about doing the logic after we had the other elements of the game finished (such as the overall GUI and the scoreboard logic), we began coding the ways we would handle checking for valid moves, seeing how many pieces in a given direction were eligible to be changed, and changing the status of pieces.

The biggest problem we faced was an issue in our logic, where every direction worked for changing the pieces in a direction *except* the northeastern direction. It took a significant amount of time in the project to figure out the issue with our loops and then fix and test the problem. Additionally, we chose to use git for version control and collaborative programming for the project. At one point, both of us were working on the project and had an issue where our code conflicted with each other, so we had to do some figuring out about how to handle merges in git.

For the project, we anticipated the minimum qualifications being a challenge within itself, so we intentionally tried to focus on realistic planning from the start. Fortunately, this realistically-oriented planning paid off in the sense that everything we planned to implement actually was. We realized that there are definitely ways that the game could have been made prettier (like previewing a move by seeing what pieces would be changed) or perhaps made more efficient (reducing the number of times we scanned the board), but for our purposes, we were on target to our goals.

The main trouble spots we faced were in the final sprints working on the game logic. We had a few errors that took a couple of hours to debug and figure out within themselves, and they proved to be a huge time sink. However, because of the way we planned ahead, the time sinks proved to not get in the way of completing the project on time.

Group issues and Interaction

- What was the organizational structure of the team, and did it change over time?
  - Such as, did you have meetings? Were they scheduled meetings, or meet when/if you can.
- What was the working structure?
  - Was the work a distinct split of responsibilities, or ad-hoc?

Using Trello,we were able to assign “cards” (i.e. specific tasks) to each other to break up the work. Tim took on most of the general GUI construction while Justin aimed to handle more logic-oriented tasks, commenting, and documentation. We both worked on heavy pieces of the logic together. Outside of the loose responsibilities we assigned at the beginning of the project, there was not more of a hierarchy or order beyond that.

Because we chose to use git, a lot of our work happened at asynchronous times while we were away from each other. Both of us would commit small pieces of the project during the time it was assigned. In the final weekend before the deadline, we met up to finish the last pieces of logic, run unit tests on the game, and work on the report. We did meet occasionally for informal check-ups during the assigned time, but we never actively programmed together in the same room until the final weekend.

The working structure was informal but kept to the originally planned breakdown of tasks, as explained earlier. It would probably fall into a hybrid classification between distinct responsibilities and ad-hoc task completion. We had our originally assigned set of tasks but we often overlapped or had some input on what the other one was doing. So while it was structured, in a sense, it was a flexible structure that changed based on current circumstances and where we were in the project.

Plans for Next Time

- What would you recommend for next quarter’s students working on this mini-project?
- What would you recommend for the instructor should provide or do for next quarter?

- What would you do different on such a project like this, or a larger project?

Next semester, students should be aware of deadlines and aim to have specific, small tasks due at certain intervals for the project. Saving the work for the last weekend or even hard sprints interspersed during the project time frame isn't useful. When the project is broken up into smaller, more manageable deadlines, it allows more thought to go into the planning and logistics for how the game is created, and prevents any last-minute, unexpected problems from sinking too much time into the workflow. Being realistic and minimalist with personal deadlines is the best way to go into the mini project.

We discussed about this with each other and thought there was no feedback we had for how the professor could have made the project more manageable or doable. It seemed like a doable amount of work for the time frame granted to work on this.

For future planning, if we had more people, we could assign two people to the same set of general tasks to help assure mutual responsibility for those areas and to troubleshoot any major trouble areas or problematic areas of the task. It would also have helped if we were more familiar with tooling like git for reverting and merging our changes for the occasional instances where our code conflicted with each other. For the one time where this happened and it wasn't an easy solution, it took more time than it was probably worth to figure out and debug the code differences.