Phase 2 Report - Documentation of Implementation of Java 2D videogame “Squirrel Hunters”

Authors : Jin Hong, Jeff Li, Benjamin Schoening, Zekai Li

Describe your overall approach to implementing the game :

The approach first involved coming up with a better outline that build off the first phase. Our first phase outline was vague, and needed a lot more detail on how to manage the game and classes. Once agreeing on certain conditions and logic of how we were going to implement the game to make it better, we broke the game into small chunks and divided out the tasks evenly among all group members. Each chunk that was assigned to a group member was based on either their strength’s or interest’s which enhanced productivity and overall quality of work. The chunks were portrayed visually to each group member, with detailed descriptions of what they should’ve done, and what they shouldn’t done.

The chunks were also grouped together based on similarity in implementation to allow a faster development time to meet upcoming deadlines efficiently. For example, one of our group members was responsible for the development of the characters class which led to them being more efficient (in the code and timewise) creating the abstraction and two character classes. Overall this allowed skills learned to be better implemented in other related chunks. We also incorporated AGILE principles by having regular team sessions where we communicated problems and solutions and changing work as needed in a coordinated fashion. In this team meetings we held each other accountable for what work we did, and the work that came up. One of the most important parts about holding each other accountable was getting work done before deadlines so we could make sure we weren’t rushing to get our project done.

In conclusion the overall approach of our implementation was teamwork to create a realistic outline, division of chunks, individual work to implement quality code, team sessions to hold each other accountable, and reliability on one-another to make sure the project gets finished in time.

Explain the management process of this phase and the division of roles and responsibilities :

Central to our management approach was the belief that open and frequent communication should be prioritized. Since this was a group project, it was essential that collaboration should be done in a way where everyone’s input was respected and heard. Another approach we took was doing individual work, but then coming back to collectively review each other’s code. This helped to make sure the overall, and scoped implementation of the game was right and of good quality.

The way we approached the division of responsibilities is by having a clear numbered list of methods of similar implementation where we split the work amongst ourselves in

a clear and mannered fashion. This ensured that no one member was overwhelmed and everyone had a chance to contribute to the project equally. The division of work in our game complemented each other's work by having divisions relate to each other. Each division that was assigned was given based off members abilities and also how efficient it was to implement (as seem above). This lead to a well working process where when we came together made it easy to discuss, review, and collaborate.

State and justify the adjustments and modifications to the initial design of the project
(shown in class diagrams and use cases from Phase 1) :

We had to make a lot of adjustments to our initial plan. We had initially overestimated our ability to implement all the requirements within the short time frame. We re-engineered our requirements to adjust for the deadline and removed some features, such as multiple levels and scoreboards. We realized that these features would take a lot of time to develop and were not required by the requirements given to us in Phase 1. Thus, we decided to put more time on other work, such as enhancing the quality of code and implementing game logic. Enhancing the game logic became important the more we moved forward, due to our underestimation of how complex it could get. In phase 1 we underestimated how hard it would be to implement the game with the functionality and usability of our logic, so we stepped-back and made sure our code was divided into classes that would make the classes work with each other, enhancing the quality of our code.

List external libraries you used, for instance for the GUI and briefly justify the reason(s) for choosing the libraries :

The only external libraries we ended up using were AWT/SWING which was a basic GUI library that provided all the functionalities we needed. We also explored other libraries such as LWJGL and LibGDX, but they were meant more for graphic intensive games.

describe the measures you took to enhance the quality of your code :

In the initial process of our project, we made it clear our group would all communicate collectively to ensure any issues we were having we could work on together. We then went on to divide work into sections based off certain aspects of the game. With this process we first worked as a team for how the functionality of the game would work, and how classes would interact with each other. From there, once we started working individually, we wanted to make sure that all of us were on the same page. We made sure of this by doing bi-weekly meetings to check up with how each one of our parts were doing. While in the meetings we discussed our progress, problems, and solutions first individually, and then as a group. Next we would discuss and highlight how and why each class works the way it does, our give individual goals along with a collective goal of what we needed to finish the code. We realized quickly the process of working

individually can cause ambiguity with the way that we view a problem and the overall goal, so we made it clear by being detailed when in our outline for phase 2.

All of this planning and organizing as a team was to make sure that the code could be of high quality. But in order to know the quality of our code, we needed to outline what quality was. Quality for code was having the code reliable, organized, maintainable, readable, efficient, reusable, and of good performance. Seeing several examples of this in lectures, as a group we all had an expectation to implement quality code, but needed the management, communication, and collaboration as discussed above. Individual measures were taken as a honour system, with each member working to the best of their ability to maintain quality code. From there, in group meetings we would hold one another reliable for the quality of our code and work as a team to give tips and fix solutions on each other's code.

In conclusion our quality code came from the ability to work and manage as a team. The phrase teamwork makes the dreamwork is only true if everyone is on the same page, doing the work that is required by them. Having weekly meetings where we could discuss, display, collaborate, and agree/ disagree on ideas/ classes and code we got rid of inefficiency and ambiguity from the outline and worked collectively to finish the code. Another measurement that greatly helped was reviewing each other's code to ensure the quality of our code was uplifted together, not just individually.

And discuss the biggest challenges you faced during this phase :

We faced many challenges throughout this phase. Firstly, as we delved into the latest phase 2, the integration of Maven and Makefile presented unique challenges, particularly in areas of building script, path and environment variables. Aligning the two tolls required a delicate balance to avoid conflicts in naming conventions, directory structures and dependencies.

Moreover, we found ourselves investing time in ensuring a synchronized configuration, striving for consistency in the build process across different platforms. In tackling these challenges, our focus on learning the intricacies of Maven's POM and understanding the syntax of Makefile became instrumental. Debugging issues within this integrated environment demanded a systematic approach, often involving a deep dive into both Maven configurations and Makefile scripts.

A challenge that was encountered during this group project was the integration of different coding styles. As we divided out the methods to be implemented, we realized that our different coding styles would clash leading to confusion in the readability of the code. To solve these issues we emphasized the importance of clear and frequent communication. By the end of the phase 2 of the project this no longer became a concern as we understood each other's quirks in a better way.