Project Phase 2 Report

Game Implementation Approach

Our approach to implementing our game was with a "step-by-step mentality". We ended up dividing the stages of the game development process into four distinct phases, each with its own theme. The idea behind this approach was to get as many of our ideas onto metaphorical paper as possible so it would be possible to begin development and section ideas off to different members of the team as easily as possible.

In each phase (section), we set it up so that each section would look something like:
<u>Phase 1: Art & User Interfaces</u>

1. Main Menu
    a. Settings
        i. Volume Slider
    b. Campaign
        i. Difficulty Selection (Easy, Medium, Hard)
    c. Practice Mode
        i. Difficulty Selection (Easy, Medium, Hard)
        ii. Level Picker (Level 1, 2, 3, …)
    d. Leaderboard(s)
    e. "Return To Main Menu" Buttons (for each of the different subsections)
    f. Quit Button
    g. Title background, image(s), font(s), etc.
2. Art, assets, etc.
    a. Floor tiles
        i. e.g. grass tiles, etc.
        ii. "Finish"/"End" tile
    b. Wall/barrier sprites
    c. Moving Entity Sprites
        i. Players, enemies
        ii. Walking and idle animations, accounting for all four movement directions
    d. Stationary Sprites
        i. Secondary enemies (traps)
        ii. Primary and secondary objectives

Phase 2: Technical and Gameplay Implementation

1. Movement
    a. Player movement
    b. Enemy movement
2. Trap functionality
3. Collision
4. Objectives
    a. Primary and optional
5. Lives
6. Score calculation

Phase 3: Level Design

1. Level Design
    a. Tutorial Level
        i. Aim: Teach players, either explicitly or implicitly, the mechanics of the game, such as player and enemy movement (e.g. how enemy movement is reactionary to player's), traps, objectives (primary and optional), etc.
    b. 3 Levels/stages
        i. Technically this ends up being 9 levels in case we end up doing modifications for different difficulties.
2. Level Playtesting

We as a team felt that level design deserved its own section as while it may just be one part of the entire development process, we felt that it would take a significant amount of time to get the feel of the stages right and accounting for how to change them for each difficulty, etc.

Phase 4: Bug-fixing, Full-game Playtesting, Final Modifications

1. Playtest Full Game
    a. Report any glitches and bugs encountered during playtesting
2. Implement recommended move count for levels
    a. Each rising difficulty should have a stricter move count recommendation
    b. Move count will also affect the final score
3. Fix any glitches and bugs
4. Ship game!

Division of Roles and Responsibilities

Regarding the management of the game and division of roles for the team, we settled with two of the four people in the team, Dae and Darren, primarily working on the code and any design documents for the game, and the other two, Andrew and Andrew, primarily working on anything art-related, such as sprites, level design, etc., and acting as back-up for any coding if necessary. Everyone on the team is also responsible for contributing to any general submission requirements, such as write-ups, like this one.

Challenges and Hurdles

There were a few challenges that came with the art portion of the game development process, with many of them being trying to get over the initial learning "bump" for many of the skills necessary. For example, this would be our first time doing real level design where the creations would be incorporated into an actual game. This meant that there were more things to take into consideration in regards to the game, such as if the program or platform used to create the levels could be easily integrated into our game, or if we'd have to create an in-house solution. Another challenge that we faced was finding or creating appropriate assets that fit the theme of the game. Planning a game to look like a certain idea is one thing, however, we learned that actually turning that idea into reality is a completely different challenge. This became prevalent when we had to create our own sprites, or when looking for appropriate sound effects and fonts to use for the game that fit our criteria.

Regarding the code for the game, this was Darren's first time working with Java Swing, a GUI toolkit for Java. Much like the challenge faced with level design, there was a steep learning curve involved in regards to how to utilize Java Swing effectively and in a way where it would be possible to implement the game in the way we have envisioned it. Another code-related challenge we faced were bugs. While encountering bugs is a given in any programming environment, finding them in a scenario where a lot of the game was already planned out brought forth some road bumps as it required a lot of backwards thinking to find out where in the process something went wrong. This also meant that usually, we had to go back and change some things. It meant there was a gap in our original planning.

As for just general, non-specific challenges or hurdles we faced, there were still a few. Getting into the flow of creating and assigning issues and tickets on GitHub was challenging at first, as this was the first time many of us have gone through the process. One other challenge related to GitHub that was faced was timing pull requests. Progress was sometimes halted due to differing schedules to everyone in the team, as going from step A to step B sometimes required a pull request to be approved first.

External Libraries and Code Enhancement Methods

For our graphical user interface (GUI), we primarily utilize Java Swing. Java Swing checked a few boxes for us, but primarily due to its lightweightedness and customizability. We value the lightweight features of Java Swing as it means that everything related to our GUI is implemented in Java, giving us flexibility in comparison to the heavyweight Java Abstract Windows Toolkit (AWT).

As for measures we took to enhance the quality of our code, there were a few steps that were taken. The primary step we took to ensure quality code was being written was to have more than one set of eyes on the code at any given time. Every time a change was made or a new feature was pushed, code was checked for any potential issues and ways that it could be written better. We expanded on this using branches and pull requests to ensure multiple eyes were on the code before it was fully committed.