Post-mortem: The Legend of Zelda Clone + Trapdoor Levels / Torch Custom Mechanics

Keaton Bonds (keatonb) and Nigel Charleston (ndcharle)

To begin development, we first got an overview of the project by attending the lecture on 9/11. During this lecture, we had the opportunity to play our game (the Legend of Zelda) and took note of several mechanics. The grid-based movement system, in particular, was a feature we spent time figuring out with other students in the class. After the lecture, we set up meetings through Google Hangouts to discuss how we were going to break up our work. We also set up our Jira board for project management and created our git repository for version control. After completing the tutorial, we began to work on the features listed in the Zelda task sheet.

Throughout the 3 weeks, we met often to keep each other up to date with our progress, work together, and discuss approaches to implement some of the trickier features in the project (such as the locked door mechanic). We used Jira to prioritize our tasks by priorities given in the p1_zelda spreadsheet, where items had high, normal, or low priority. Along with the tasks, we made separate sprints for p1_milestone, p1_alpha, and p1_gold. With the tasks organized within the sprints, we focused on implementing high-priority features first, such as stalfos movement and health for Link. Then, normal- and low-priority assignments came after. In this way, we believe that we maximized our potential score for the products of each sprint.

We did not have the opportunity to playtest our games with other students, which we feel was one of our greatest factors for the level of polish our game was submitted with. We would frequently playtest the game ourselves, using the play button in Unity to check out new mechanics each of us worked on, but we did not get the game to a state where we got external feedback. As a result, our playtest feedback was mostly technical. After testing our features, we would often talk about how we could better implement a feature so that our project was more maintainable and extensible. One such feature that changed was how we implemented

damage. We designed a “damage dealer” component that could be applied to any object that caused damage: weapons, traps, Link, and enemies. Appropriately applying layers to each part of the level controlled what was allowed to damage what. With this component, it became much easier for us to implement all the varying types of combat within the dungeon.

For the custom mechanic idea we chose, Keaton initially discussed the idea with his roommate, drawing inspiration from the mystery dungeon franchise. In this game, the player navigates through dungeons with randomly generated enemies and decals. Some of the “tiles” the player walks on causse random status effects to the player. Each of these can either be beneficial, or negative. For example, some tiles make it easier to evade enemy attacks, gain monetary rewards, or damage your player with negative status effects. After discussing the concept together and iterating on some concepts, we thought it could turn into a good custom mechanic for our dungeon.

We wanted to add an element of surprise, challenge, and reward to the Legend of Zelda. We also wanted to have a custom mechanic that was easy to implement, given the tight deadline for p1 gold. Using the tile mechanics in mystery dungeon as inspiration, we ended up creating our “trap tile” mechanic. We felt this mechanic allowed for us to make a deep custom mechanic while minimizing our development time; we could reuse the assets given to us to build the custom dungeons and develop a teleportation mechanic to warp the player to the new dungeons.

We hoped that the player would not only be challenged but also be given more decisions to make when progressing through the original dungeon. With our mechanic, the player could either risk death to obtain strong rewards to progress through the normal level or play it safe to keep their progress. All the while, we wanted to subconsciously train the player to handle the mechanic. The first trap dungeon serves as a warning to the player. The later ones are much

more difficult, but hold the potential for better rewards. Completing the dungeon as a whole requires understanding the torch mechanic to discover the final level and obtain the triforce.

We believed that we worked well together. Both of us frequently communicated about what we were working on, and what future tasks we would be responsible for. This was especially important for managing our git repository. By communicating with each other, and making sure we did not work on the same component simultaneously, we minimized the number of merge conflicts we had when pushing code to our master branch. We were also very flexible with each other's schedules; we figured out times where we could meet either through google hangout or in person.

We believe that our project may have been overengineered for the scope of this assignment. Throughout our development process, we focused on making our code maintainable and extensible, at the cost of additional development time. Several of the components we made, such as the way health is handled and our system for dealing damage, are structured in a way to make future revisions easier. However, the strict deadline for each segment of the project (especially p1_gold) made this approach impractical. With the short deadline, quicker and "hackier" implementations would have been favorable.

We also had time management issues with the project. During the p1_alpha, we both attended a professional development conference for 3 days. During that time, we did very little work on the project, resulting in us having to crunch for the deadline the following week. Moreover, all of the tasks that we did not implement in p1_alpha carried over to p1_gold, resulting in us being under greater stress.

In the future, we will coordinate with our partner to schedule a time to work on our project every day. Even if we cannot meet in person, we can message each other to stay up to date with our progress remotely. Also, we will be more conscious of our design decisions and scope.

We must understand the time deadlines given to us and engineer our features such that they are structured well, but not over-engineered. With the deadlines given in this course, we must be careful not to do too much work to accomplish each task.