Phase 4 Report

Group Members:

Evan Sarkozi
Kevin Park
Harry Nguyen
Daniel Wang

Abstract

Our game is Super Fun Game Ultra Edition. The premise is simple: you start at the bottom of a twenty by twenty walled-off chamber filled with barriers and traps and must navigate around your environment in search of yellow star-balls. Once you have collected five of them, a gate on the north side of the map opens up and you must pass through it to win. While still inside the chamber, you can also collect bonus rewards, which increase your score -- it may, however, be wise to leave as soon as possible, since numerous enemies are chasing you and countless traps are strewn about the map which could cause you to lose the game!

Showcase video: https://www.youtube.com/watch?v=k-FhH6qVhJ0

Adherence/Changes to Design

Due to numerous factors, including technical feasibility, miscommunication, insights from playtesting, and rapidly approaching deadlines, numerous aspects of our initial design were ultimately changed or scrapped.

The following is a list of notable things that were adhered to from the design phase:

User Interface

The game's interface looks almost exactly like the UI mockup which we designed for phase one, except for an added restart button. Additionally, most of the visuals used in the mockup were used as the basis for the in-game pixel art graphics.

Mechanics & Specification

We made sure to follow all the requirements in the specification. This meant that the core rules of the game -- how the enemies, rewards, and obstacles worked, how the ticks were handled, etcetera -- stayed 100% consistent throughout the entire process. The only changes we made were additions to the base requirements like new enemy types, and the only thing missing in the final product from our specification was choosable map sizes.

Phase 4 Report

Group Members:

Evan Sarkozi
Kevin Park
Harry Nguyen
Daniel Wang

The following is a list of notable things changed/added/removed from the design phase:

Use Cases

The way that several of our use cases played out (mostly the ones involving navigating menus) was drastically altered. Initially, we planned for numerous pop-up menus with complex, branching behavior (e.g., having a pop-up asking whether you want to record your high score, then being directed to a keyboard input prompt for the high score name). In the end, however, we opted for a much simpler way of doing things, having only one menu tab on the left-hand side, and three menus with just one option (start, win, lose). This saved a lot of time on something that didn't need that level of complexity.

UML / Code Structure

There were a lot of things that required changing from our original UML design document. While it gave us an excellent look at what the hierarchy of inheritance should be, and the essential properties each of the classes should contain, many aspects didn't translate well into the implementation phase (e.g., DashBoard's methods didn't make sense with how our rendering library worked), and countless fields and functions had to be added to most of the classes. This was to be expected, as the UML was written before we chose any libraries, or wrote a single line of code -- obviously, there were going to be some things that didn't mesh right in practice.

Overall Conclusion:
We believe that our end product is very consistent with what was required of it, and what we set out to do during phase one. The only real differences between our initial plan/design and result are *how* we went about doing certain things, with extensive refactoring being an important part of the process.

Phase 4 Report

Group Members:

Evan Sarkozi
Kevin Park
Harry Nguyen
Daniel Wang

Lessons Learned

There were many lessons learned during this whole process, outlined below:

Choice of Library

All of us were quite new to Java when we began this assignment, so we were initially quite unsure of what libraries we would use to implement our game. After some brief research, we decided to use LibGDX. This was kind of a mistake. While our result *technically* works, it often feels more despite the library, as opposed to because of it. This is mainly because it was designed to build projects with Gradle, its hooks into low-level rendering are hard-coded, and many of the library's functions are kind of nonsensical. If we were to do it all again, we probably would have researched libraries more thoroughly.

Designing for Testing

When designing and implementing the game, we did not pause to think about how we would test our functionality nearly enough. While our final testing suite feels thorough and effective, we did have to make some sweeping changes to the codebase to write it (though, these changes were overall a net positive, even outside of testing). Additionally, our primary game loop was encapsulated by a class that could *only* be used at runtime (due to problems with the library), so that made non-unit-tests particularly difficult. If we were to do it all again, we would have designed our systems to be testable from the start.

Communication

Fortunately, we were not plagued with merge conflicts, arguments, and the like, as many projects/teams inevitably face. We did, however, find some trouble in the division of work. While this problem was not particularly detrimental to the final product, we likely could have been far more productive from the get-go if everybody knew what they were supposed to be doing. If we were to do it all again, we would meet more often to discuss progress and delegate things on the to-do list.