Software Design – Group Project Report

Niamh Kirsty Ferns

Knowledge & Experience Gained

I feel that the knowledge and experience I gained from this project can be broken down into main areas: teamwork and code.

By far the most I got out of this assignment experience wise is learning just how hard and how much work it takes to bring together a larger project that's split among multiple people. I feel that I am now better at both communicating what I want to put into code as well as understanding what others are trying to communicating with me.

I feel that working on this project code-wise has given me more experience working with modularized code. That being working with packages, having access controls being vital and also just how important it is to build things in such a way that not only are they useful now, but are useful in the future. I feel like it has slowly reinforced a lot of what I've learned in a practical way.

Major Challenges

I'd say the three most major challenges faced were how to handle levels, how to actually handle the UI and the gameplay viewport, and how to handle the recorder. I ended up implementing a level as a record that has the various properties associated with any arbitrary level. Adding a new level just requires a path and a name. From here we can just activate that level via the path in domain and instantiate a renderer to draw it.

This brings me on to handling the gameplay viewport. Java swing is, to me at least, not a very intuitive framework and as such was fairly tricky to get working even just to draw. To make this viewport work with the rest of the game, I spoke with Kitt and we designed Renderer() to essentially just be the JPanel that is being drawn. This gives him the ability to work in any way he sees fit and I can slot a new renderer instance into a level when the level is instantiated. No need to pass in the level or anything as renderer can get all that from domain.

Finally, recorder was the last major hurdle. Santino and I had to sit down and work out together how to approach this as we wanted it to just cleanly slot into the already existing domain and renderer management system. To do this, I provided access to the Observer class so that we could call recorder to load a level as we would normally but register the replayer for that recorded level to the GameClock to receive updates. This replayer (and fuzz) have access to an InputGenerator that lets them call inputs on App. I also some methods that Santino could call in Recorder to pause and unpause the GameClock for step by step playback. This meant we could store a recording as a stack of key presses to replay as the game is entirely deterministic around key input.

Teams Methods & Approach

As a team we decided very early on to go as simple as possible. In art there is the idea that limitations inspire creativity and I personally believe that it applies to code as well. We have a simple game and a simple set of tools. This is all we needed and this is what worked for us.

The one thing that didn’t work for us super well is how we were communicating over discord. It lead to some major slowdowns toward the beginning of the project and we only really picked up steam once we decided to embrace in-person meetings and coding sessions. Coding is a form of communication and communication is almost always better done face to face.

Design Pattern of Note

The most notable design pattern used in App would have to be the Singleton. A lot of the games updating and access to the GameHandler is done through singletons. We should only ever have one game and that game should only ever have one clock to update it.

Singletons do have a fair amount of cons but the only two that are significant here are: A) that it makes the code dangerous in the sense that anyone with access to the singleton has access to features that could potentially break the game. This is readily pointed out by Spotbugs as well. B) they massively hamper extensibility. Having a GameClock as a singleton makes it easy upfront but if for some reason I were to need 2 clocks (say I have a mini-game inside my game), I would need another work around or a completely different design.

The main advantage is that it’s dead simple to implement and use. Given we were making a small project that would not be a package or library in some greater program, the danger flaws of my implementation of a singleton I feel aren’t all that scary. The added simplicity it brings to accessing different core components of the game when needed, in my opinion, far outweigh the consequences in this case.

Approaching this Assignment Again

If this project had exactly the same requirements as before, I would have changed where I started. I started thinking from the GameClock out toward other aspects of App and that put me in a mindset that I feel brought up some issues that didn’t need to be there.

I would also focus far more on UI first and take the time to practice Swing before diving in on a big project. Much of my Swing code was trial and error at first and as a result it is by far the weakest part of App.

Thoughts on Improved Team Approach

I would have spend far more time planning and would have heavily pushed inperson coding and communication very early on. These were the two things that by far hurt us the most and I feel that if these were changed we could have completed a lot of what we managed in around half the time.

If I were to approach the project handling for this differently as well, I’d use a method like scrum or kanban to increment on development rather than splitting up modules. Not having people read and correct your code is devastating and it traps you in your own little bubble.

Team Contributions

Haosong Zhang	zhanghaos@ecs.vuw.ac.nz	3
Jamie Gordon	gordonjame@ecs.vuw.ac.nz	5
Kitt McEvoy	mcevoykitt@ecs.vuw.ac.nz	4
Micky Snadden	snaddemick@ecs.vuw.ac.nz	5
Niamh Ferns	fernsniam@ecs.vuw.ac.nz	4
Santino Gaeta	gaetasant@ecs.vuw.ac.nz	4

Top Three Commits

https://gitlab.ecs.vuw.ac.nz/course-work/swen225/2022/project1/t4/chips_challenge/-/merge_requests/1/diffs?commit_id=f6ad01c649806a38948a9b056e52c848a4788dc0	This was the most significant commit in a series of commits in a merge request that made the GameClock feature rich enough to be usable in the rest of the program.
https://gitlab.ecs.vuw.ac.nz/course-work/swen225/2022/project1/t4/chips_challenge/-/merge_requests/12/diffs?commit_id=7bc3e81a4eec9d9e0be49cc1f9345032e3250975	This was the commit that put in place the foundation for how level switching and input redirection is handled. That is, it is part of a series of commits that added the ability to call movement on domain and also switch levels in GameHandler then start that level in domain.
https://gitlab.ecs.vuw.ac.nz/course-work/swen225/2022/project1/t4/chips_challenge/-/merge_requests/18/diffs?commit_id=8feb58b03906e38d2c7c2a6c1e46f75331350bc7	Here I fixed a major major bug that caused the various different JPannels in my program to steal focus from the actual viewport thus blocking key input. This also added the calls to Recorder to record input during a level for later playback.