Retrospective - fourth iteration

Syntax Highlighters

May 11, 2018

What worked well?

Communication channel revolution: After a lot of back-and-forth regarding communication, we settled on Messenger as the de-facto standard. Although this wasn't ideal in one sense due to the fact that Discord would be a more versatile option, it's better overall because it seems to be easier to get in touch with people on Messenger, which is also why it eventually re-established itself as the default communication channel. Communication on Discord was mostly relegated to the purpose of reaching certain individuals who would be more likely to check their messages there, as well as the #help channel. While it would have been better to have the communication sorted from the start, it's good that the mode of communication has finally settled itself, and it's uplifting to know that we can experiment with different tools and find the one that works the best for our purposes.

Improved workload distribution: Due to increasing general competence within the group, it became easier to involve team members more evenly in the process. The fact that we had several unrelated and relatively large features to work on also worked well with our previously established role entrenchment and the resulting lack of flexibility - because the features weren't heavily dependent on each other, we could form subgroups which only limitedly intercommunicated, and develop the features independently of each other, without having to constantly synchronize with the others, and without having to worry about changes in one branch (read: one version of the develop branch) conflicting with stuff other people were working on. This also helped with the workload distribution, in the sense that we weren't dependent on *when* someone else would complete their work, and thus it was easier for everyone to work at their own pace, and the earliest bird wouldn't have to get frustrated from waiting and dig up all the worms on their own.

Team programming: We greatly benefited from team programming for the implementations of sjadam and Fire Chess. While it frequently was one person who wrote most of the code, it was extremely helpful to have two people who together could discuss design decisions and bounce ideas off each other, as well as making the process more social and enjoyable in general.

Administrative efficiency: Apart from a couple of hiccups, the administrative part of the meetings was usually efficient. We benefited from a strong leadership who took the initiative to go through the generic agenda at the start of each meeting, which we believe helped keeping everyone on track and on the same page throughout this iteration.

Diligence: We maintain that our team members are very responsible, and treat the group meetings as a serious commitment. The attendance is generally good, and we usually receive notifications of any absence ahead of time.

What didn't work so well?

Structural extension challenges: We noticed very early on that the way we had implemented the Game class did not lend itself well to extension of the core game mechanics. We also noticed that what we had initially thought would be a good idea for extension, namely pushing the complexity down to the edges of the class hierarchy, did not lend itself that well to minor behavioral modifications, since there would be a lot of subclasses which would all re-implement very similar behavior. Instead, we ended up pushing some complexity up to the Game level of the hierarchy, refactoring common functionality into the new AbstractGame class, and allowed subclasses to determine the precise set of moves that specific mode would allow at any given time. This did, however, eat a little away at our time in the very beginning of the process, because most of the features we were going to implement were dependent on this refactoring before it could begin - with a system where this design decision was made from the start, this would have gone smoother.

"Nice to have" feature requests denied: While we had a lot of ideas for features we would have "liked to have", we had to decide to significantly cut back on the number of smaller features to add, as well as limiting the scope of the bigger features, because of the limited time we had to work on the project. For instance, we didn't implement a settings screen overlay as we would have liked, and multiplayer only supports regular chess game - as for Fire Chess, we're not even certain synchronizing the two games to a satisfying extent would be feasible, and as for sjadam, by the time we felt like it started to be achievable, we were already running out of time.

Generality Is Hard™: Related to the above point, we found that it was more difficult to make the features we wanted to implement as abstract as we would have liked, from a UI and networking perspective, which played a large role in the downscoping decisions we made towards the end of the project - making things work in the general case would, contrary to what we would have liked, actually require some additional effort, and time, both of which were in scarce supply and needed to be focused on the most essential aspects of this iteration.

Race to the finish line: Again, we experienced a rush towards the end, with a number of uncompleted issues - this may be in part because we were single-mindedly advancing the larger features without paying enough attention to the future, and maybe also because of the primarily programming-focused meetings. While we benefited from the team programming at the time, it might also have served us well to keep an eye on what would come after, which is something that could have been better this iteration.

Lack of progress transparency: Due to the larger features staying disconnected from the common branch for long stretches of time, it was difficult to get a handle on how other parts of the project were doing, and to which extent the different pieces would end up working together and how much conflict they would cause. This caused a mild anxiety in some of the more

susceptible members, but it ended up working out all right in the end, as everyone knew around how hard they would have to push in order to get things to a working level, and thus we managed to complete the main features we set out to implement in this iteration.

Control Freak Syndrome: (AKA "Design Conflicts 2: Resistance Is Futile".) In a similar manner to the design conflicts detailed in the second retrospective document, we were aware that certain individuals (read: me) on the team were making cooperation difficult by pushing their own design decisions to the point of overriding other people collaborating with them. However, while this caused a bit of friction in the earlier iteration, this time it was met with quiet resignation, which in its own way was even more disconcerting. On the behalf of any such individuals, I would like to apologize to anyone who might have felt like their efforts didn't matter, and also thank them for still providing their valuable input on the challenges we faced during this iteration of the project.

The future

There is no future for this project. On behalf of the team, I would like to thank you for your time and attention throughout these past months. May you fare well in all of your future endeavors.

Goodbye.