Team Plan

Recap

As mentioned in the previous assignment, the workflow and communication we had until now worked really well. We didn't get to use Trello as much due to the nature of the assignment, so this was something we really wanted to use actively for the next and all upcoming assigments. We kept the distributed roles as it was, and currently didn't see any need to assign other roles. This doesn't mean that the team members with no rules would do nothing, as they would still contribute as much through coding, planning, discussion and all other activities and tasks we would encounter. There was also a mention of planning poker, and this is something we also wanted to as it seemed like a good concept.

Roles

We have kept the distributed roles as is, and currently don't see any need to assign other roles either. This doesn't mean that the team members with no roles will do nothing, but contribute as much through coding, planning, discussion and all other activities and tasks we would encounter.

The majority of the theam had more or less experience in terms of programming, both in relation to relevant subjects we had taken, but also relation to programming in teams. Jan was the member using git the most, so it was only natural to give him the main responsibility of the reposity and everything related to git. Mikael has participated in a programming boot camp earlier, meaning he was the most experienced programmer, so he voluntarily took the role to coordinate the code structure. In terms of general documentation, we decided to have everything written into LaTeX, and since Viet has been using it for almost a year now, everyone on the team can ask him for help regarding LaTeX.

Team leader

Name: Viet
Team leader has general an overview of the group and project

Support

Name: Tom
More or less left-hand man for team leader. Steps in for team leader whenever necessary.

Git-responsible

Name: Jan
Git-responsible keeps track of the git-repository, ensures it works as intended and solves any problems that occurs.

Programmering Manager

Name: Mikael
Programming manager has complete overview of the source code and directs the code design.

Graphical designer

Name: Magnus
Graphical designer is responsible for everything related to the user interface. This includes the game window, chess board, chess piece sprites, icons etc.

LaTeX-responsible

Name: Viet
Helps the team with everything related to LaTeX when needed.

Code tester

Name: Jan Writes tests to assure quaility and functionality of code.

Git-repository

To avoid complications as much as possible, we decided for a conventions as following; always pull from the reposity before doing any work, and also do it often. This ensures that local reposity is always up to date with the main repository. To make the repository organized and clean, we divided the repository into two main folders; documentation and programming. The folders will contain what the names suggest; documentation of the project, plans, product specifications, manuals etc, source code and respectively.

Midway the 4th sprint, our team figured that we had to start branching out, since after doing planning poker, more work was being done simultaneously than previously. Master branch would be our hand-in branch as before. We made a

new branch called wip (work in progress) which would be the branch containing all new working features. When a team member would start working on a new feature, said member would then have to branch out the wip branch. When it worked as intended without bugs, a merge would then be requested.

Communication & Meetings

To ensure that the teamwork is as best as it can possible be, we decided to set up one extra meeting where we could discuss and work on the project together. We also created a Discord chat and Facebook group to communicate outside of the meetings. The Facebook group was however only necessary for initial online connection and we do not use it at all. Discord work wonders. Meet ups:

- Tuesdays 1000-1200 at VilVite (Gruppetime)
- Thursdays 1000-1200 at Høyteknologisenteret

Task board

To keep track of and orgranize our work and tasks, we used an online website called Trello. Trello is an online notice board which let us create cards where we can specify what has to be done. It has a token feature which let's each member tag their name on it, and we can also set priority level and deadlines on the task cards. The Trello board is divided into several sections, "Programming in progress" and "Known Bugs" being the most important sections. The team members wil tag their name on the tasks they have been assigned and want to do which tells the other members that this feature or function of the application is being worked. This ensures no duplicate work and avoids unecessary merge conflicts. And as soon tasks are completed, they will be moved over to the section "Programming done". If there are any unccertainty around the board, the members are always available on discord.

Javadoc

As our source code grew and got more complicated, walls of code with little to none and bad commenting convention made code not written by yourself quite hard to navigate through. To fight this, we agreed on using Javadoc style commenting after postponing it's use for far too long. Javadoc is easy and simple to use, but still give enough description for team members to make sense of the code.

Training

One of our biggest concern regarding git was merge conflicts. In order to reduce the amount of confusion around these, Jan made a presentation for the rest of the team regarding branching including most important commands and explanation about what actually happens when we do all these commands. We also had the members who attended the lecture about TDD to have a brief about that since not everybody attended that lecture, as we thought this might be a good practice in the long run.

Risks

As with all kind of group projects, there are always some risks that can influence the end result of the project. The most fatal, but perhaps the least likely to happens, is that one of the members drops out. The consequence of this is that we have to redistribute tasks and roles amongst the remaining members. This can in turn be too much for the remaining members, and the team may have to drop some functionality. This takes us to another risk which is less fatal, but more likely to happen, and that is overplanning. When the team realize that the planned functionality turns out to be too much, the team has to remove the features which is least critical for the project to work and still meet the deadline. Another relevant risk is loss of data. Git is a very safe storage for the project, but to be on the safe side, each member takes a separate backup of their local repository periodically, and keeps it on a separate drive.