Team Plan

Group 5

1 Team information

- Malin Jakobsen (xun007) - Team leader. Responsibility: Maintain house rules and general management of team. Experience: Have taken part in project management activities before. Have technical experience through INF100, INF101, INF102.
- Olav Gjerde (ogj005) - Git repo leader. Responsibility: Maintaining the git repo structure, and assisting the other teammates with the use of git and its features. Experience: Have technical expertise through INF100, INF101 and INF102. Will be taking the role of git repo lead and have
- Keerthan Kumaranayagam (kku007) - role Responsibility: later Experience: INF100, INF101, INF102 and basics within Photoshop.
- Ole Kristian Solheim Gjerlw (ogj003) - role Responsibility: later Experience: INF100, INF101, INF102. Experience with Illustrator/Photoshop if we want to makeour own assets/textures. Some experience with JavaFX.
- Renate Nikolajeva (rni006) - role Responsibility: later Experience: Had INF100, INF101, INF102. Have some experience in Photoshop and Vegas Pro.
- Rune Vatne (rva013) - role Responsibility: later Experience: INF100, INF101, INF102.
- Simen Gad Hasvi (qax007) - role Responsibility: Class diagram and project structure. Experience: Had INF100, INF101 and INF102. No experience with GUI.

More roles will be assigned as the project progresses and as we establish a clear view of what it is that must be done.

2 House rules

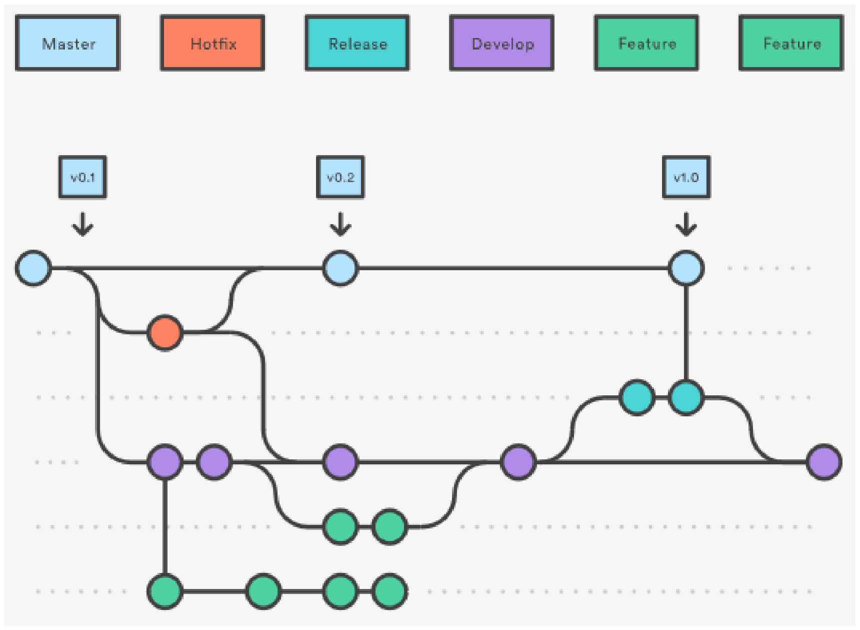
- Every individual implementation is best effort, meaning that if an individual encounters major problems or notices that they have stagnated they are requested to seek help with the others in the group at the earliest convenience.
- Meeting with the group at least once per week outside of the obligatory lab is the goal, adding additional meetings may be considered when enough work has been accumulated.
- Most communication will take place through Slack and Discord, on both of the platforms there will be dedicated channels to programming, tools, guides, documentation and non-project related activities.
- All code submitted to the repository shall be commented and be properly formatted for easy understanding within the team.
- If features that are being implemented changes the product in such a way that the documentation needs altering, then this should be written down and notified to the rest of the team.

3 Team repository structure

We will use the git-flow idea of a git workflow to structure our repository. This means that instead of just working at the master branch, we will have additional branches such as develop, feature, release and in special cases hotfix. Though we may omit some of these at certains parts of the project if it causes to much confusion for the participants.

The master branch will store the release history of the project, and the develop branch will serve as an integration branch which we will be able to merge finished feature branches into. Each major feature should have its own branch which then can be pushed to the remote repository for further collaboration. The feature branches are branched off of the latest develop branch.
Release branches are forked off of the develop branch after the develop branch has acquired enough features. This will be the product that will be shipped when the deadline date hits. Only bug fixes, additional documentation and such may be added to the release branch after is has been established. When it is ready the release branch will be merged into the master branch, representing a new version of the product.

Hotfix branches are used for fixing urgent issues with the released product in the master branch, this means that it is forked off of the master branch. When a given fix is complete this should be merged in master and develop. Initial documentation will be produced with google docs for fast collaboration before being translated into latex and pdf formats. After this changes to the documentation happens through the git repository.



"1. A develop branch is created from master
2. A release branch is created from develop
3. Feature branches are created from develop
4. When a feature is complete it is merged into the develop branch
5. When the release branch is done it is merged into develop and master
6. If an issue in master is detected a hotfix branch is created from master
7. Once the hotfix is complete it is merged to both develop and master"

4 General risk analysis

This is a list containing the risks that we found most pressing for this project. The list is subject to change, and may also grow as the project progresses.

Risk	Description and Probability	Strategy
Staff Illness	If staff falls ill in a critical period of development, and is unable to contribute (- Moderate)	Get an overview of what the individual was working on, if critical, we assign the task to other team members, if not, put on hold.
Requirements changes	If additional requirements are delivered to the team that requires altering of the product (- Moderate)	Assess the impact the changes will have on the current state of the product and generate a list of the most pressing issues. Allocate workforce by looking at the list.
Underestimated development time	If the team underestimated the work a feature requires in regards to the deadline (- High)	Cut back on features that does contribute directly to the main specification of the product, and reallocate team members to the feature features in question.