# SEM Lab Journey

Group: 8008

## The team

The group creation went pretty smoothly, we are a group of friends which found each other easily when it was said that you had to create a group yourself. We had never worked on a project like this together, but we thought it would go pretty well.

One problem in the team was that the programming skills were pretty different. Some of the people were very good in programming and had a lot of experience and some were just average at programming and learned most of it through the study Computer Science. It would of course be best to have the same level of programming for everyone in the group, because it is sometimes hard for the average programmers to keep up with the better ones. For the better programmers it could also be frustrating, because every new feature takes a lot longer for the average programmers. For this project everyone was pretty tolerant and accepted the level of programming of the other team members.

## The game

With our lab assignment we made Doodle Jump. This game was not on the list of games, but it was accepted to create this game as it is a straightforward jumping game, and at our level of programming. We think that it is important to more stress the fact that the students can choose another game if they would like to, and to keep the 5 games more as ‘examples’. When students have a game they chose themselves they become more motivated to make something cool out of it. That is how we felt!

## The SCRUM process

Every Monday we created a sprint backlog, and every Friday we made a sprint retrospective. Both of these we had our doubts whether they were effective. For example creating user stories for every task was something that was not helping us understand the task better. This would probably work better for a team that does not speak a lot with each other. In our situation we have so much contact, that when something is unclear about what has to be implemented or what user story that belongs to we just ask each other about the idea behind it.

The part of the sprint retrospective is something that we would also use in other projects, but not the way we did it in this project. Now, we had to document the hours worked and the problems encountered, but how it went was that one person from the group would document this and send it. A better method would be to have a stand up meeting with the team, here we would discuss problems and make commitments for the next week. Then everything is central and everyone knows where problems are and what has to be done to fix them.

## Testing

At the start of the project we got a sort of challenge, it was to make a good game in 12 days. For this you had to work hard and produce code fast to create a working and fun game. We started these 12 days working very hard and creating a lot of lines of code. This initial game didn’t have to have any tests because it just had to be a working game that could be played. But what was really annoying was that the second week you were graded already on the amount of tests you had, which was not announced clearly. We had already a pretty big project and to get even close to 80% test coverage we had to test that week full time.

We were, over the whole project, a bit too ambitious with adding new features. But because we wanted so much to add new cool features, we didn’t set any time apart to do any testing.

## The workflow

What is great about this course is that you learn to work in a certain workflow. The workflow of adding a new feature consists of creating a new branch, then writing the code, writing tests, and then opening a pull-request. Especially the act of doing code reviews is something that really adds value. This way the overall code quality and the understanding of each other’s code improves. We will surely use this workflow in the future.

## The assignments

The assignments are of course the part of the course where you learn the essentials of what the course wants you to learn. We think it's good to have these assignments where you for example learn how to implement the design patterns you learn about in class. We implemented several design patterns and refactored the code a lot. This is a good part to learn about because writing software for the long term requires maintainability and clear code.

What we did wrong when making the assignments is not focusing enough on doing them, and really focus on the parts where you would get points for. While doing the assignments we were also adding new features to the game, but even though these features were cool and made the game more fun, we would not get any points for them. Also we got lower points for the assignments because we got less time scheduled for them.

What we would have liked to see is maybe another system for the points you can get, with for example every week points for new features you could receive. Or maybe something with bonus points if you implemented something cool.

## Last thoughts

The deadline every week on Friday afternoon at 17:55 was a bit unpractical. When students have the most time to work on projects like these is in the weekends or evenings. We struggled to have everything working and pushed to the repository at this time. This was mainly because we just had lectures that day. Some people, also of other groups, had to skip lectures to reach the deadline. We think it would be better to have the deadline later in the evening on Friday, or just on Sunday.

Lastly, like every project, you learn a lot from it. How to work together in a team, how to write better code and how to get better and faster in programming. We also learned a lot from this project and it was also much fun creating this game and having the freedom to create it however we liked.