Team:

- Viktor Jarl Palmason
- Ole Kristian Lund Lysø
- Genti Dautaj
- Oscar Vagle

Discussions:

Strengths and weaknesses of Lua and C#

Lua has a simple syntax so it was fast to learn and easy to use. C# has similarities to the other C languages so getting used to it was not that hard, and there was plenty of documentation.

We did not delve much into Lua since our Lua integration was only working with variables and functions, but the Lua integration is intended to work with all the features Lua has to offer beside I/O and OS libraries.

Controlling the process and the communication system

During the project, communication was done through Discord, Microsoft Teams, Zoom and emails.
The team communication was all done through discord, we had a discord server where we discussed tasks and ideas, had meetings and shared resources.

With our supervisor we used Teams and emails. Team was mostly used for meetings while questions, short discussions and handing in drafts of the reports were done through emails.

With our Product owner we initially only used Zoom for meetings and email for everything else, but then in April a discord server was created for us to have better communication. We still used Zoom for most meetings, rarely did we use discord for meetings, but we started using the discord server as the primary communication channel.

The team had frequent meetings during most weeks to have good control of our project. We had meetings after our meeting with our supervisor or the product owner to discuss the feedback we had gotten from them. We had a weekly meeting every Friday to manage tasks and discuss the progression of the project, and sometimes we would have meetings on other days if we needed to discuss more.

Version control

Our repository was hosted on bitbucket and initially we were not that structured with our version control. We did use branches to manage our work and during meetings we discussed merging and creating new branches but there was no clear structure to them. Some created branches for every little task and testing they were doing while others only had one or two to host all their work, and this did sometimes lead to merging conflicts. Later in our development we learned about Git flow and we started using that as our branching model and that did lead us to have better management of our repository. That said trunk-based development would have worked better for us. The reason being that all our tasks were connected to different degrees and if we were having features existing on individual branches it would be hard for team members implementing their task so it would be compatible. We did communicate through discord how we were implementing our task but that was not enough to hinder conflicts between the branches. So a main branch that everyone was frequently merging to would have made things more easy and productive for us.

We did not use issue tagging or versioning because we were not costumed to using it and we did not issue tracking that exists on bitbucket because we did not have access to the feature so we used trello instead to manage our tasks and toggl to track our time.

Programming Style and Commenting

We decided to use the C# coding guidelines provided by Unity:

https://wiki.unity3d.com/index.php/Csharp_Coding_Guidelines

Even though keeping a uniform programming style throughout is difficult, seeing as we all have had the same previous classes and experiences - our programming style stayed uniform

naturally. Of course, to some degree, we also took this into account whenever we did peer-reviews.

For Lua we could have decided to use guidelines, but the Moonsharp C# - Lua Interpreter only supported a very specific structure and style anyways - So we decided to use their own tutorials as a ‘template’:

https://www.moonsharp.org/getting_started.html

Integration of libraries, modules, and other ways for modularisation

In our project we mainly used the **Turn Based Strategy-Framework**, which is an asset pack retrieved from the Unity Asset Store. Unity took care of the file structure and any other requirements for importing this into our own project.
The TBS framework came with a rather thorough documentation, so implementing this into our own project wasn’t any concern.
A requirement in our project was that any persistent data should be saved in json-files.
We ended up using both the Newtonsoft JSON library AND the SimpleJSON library in different parts of the code. SimpleJSON has a more python-like approach to handling json, and could therefore be used in instances where we wanted some randomization in what data we were retrieving.
Of course, using two different libraries that essentially do the same because we had different experiences using JSON is a mistake. Not only does this make some of the code wildly inconsistent, but it can also be argued that we’re clogging up our project with one excess library.

Professionalism

As a team we were very vocal in our discussion on how to implement features and help solve problems, and we did question each other on our work to give reasons on why work was done in a certain way and establish better understanding. We did not establish any guidelines on how we should write code or comment but we did always keep in mind to create code that was readable and add comments when further explanation was needed.

We created rules for the team to follow to establish the roles and relationship each has by being part of the team. We had rules on how the tasks would be managed, we had rules that defined the proper workload each member would take on, and we had “rules” that emphasized that communication was important. For example, one should ask for help if needed and help should be given when possible.

From the start of the development we scheduled a weekly social gathering for us to hang out and do something enjoyable, like watching a movie or playing games. This was to help the group grow closer and become friendlier with each other.

We used 3-party assets in our project that none of us had any prior experience with and to efficiently have the group learn how to use the assets a team member would take some time to learn about one asset and later explain how to use it. That person consequently would be the leading individual on the asset but would not be the sole person responsible that it was integrated to our project. This also was done when a new feature was implemented or tools were added. Everything new was to be explained to the rest of the team so everyone had the same understanding of the code and project.

Code Reviews

We performed code reviews regularly throughout the project. If a feature branch was ready to be pushed to the development branch, we would get on Discord and peer review the code together as a group. Having multiple sets of eyes on our code like this definitely helped us spot inconsistencies and code that could otherwise be considered ‘bad’.
Of course, no system is perfect and occasionally a bug could slip through. However, as we did the reviews on feature-branches, finding the location where the bug stemmed from usually turned out to be quite easy.

Individual Report: Viktor

Good Code:

https://github.com/ViktorPalmason/Professional-Programming-Report/blob/d625ea1f004c3cac2acf097acf950e4b59466f9c/SetupGameUI.cs#L90

My example of a good code is a function that I wrote during the bachelor project. The function reads a JSON file with names of factions that players can play as and displays the names on lists that exist in the UI in the game. Then a player is supposed to then select a faction it wants to play as by clicking a button that exists on each entry and the faction it has chosen will have its name printed next to the player title.

I am happy with this code because I managed to neatly create and add the entries onto the lists on the UI by using prefabs on lines 102&103. I have the faction names be displayed on its entry in the list on line 108, and then have the functions that registers the players selection listen to the buttons that are on each entry on the list on lines 110&111. Doing this I could avoid making a class representing the entries on the lists that would have a reference to its button and have functions that would register a new listener.

Bad Code:

https://github.com/ViktorPalmason/Professional-Programming-Report/blob/8df0dd9bb53cf5e708ffe76375b94701bb820982/CreateMod.cs#L121

My example of a bad code is a function that I wrote during the bachelor project. The function generates a folder structure for a game mod. Why I dislike this function is that I see it as very messy, it contains a lot of code that is too much to read over. I feel that sections of code could be put into separate functions to have it shorter and the functions names would be descriptive as to what is going on. For example having a folder generation function for each folder type, having the would also make it easier to modify or add things that should be generated inside the folders (this would affect line 149 to 198). The bottom section (line 200 to 216) that registers the newly generated mod could also be separated into a different function.

Refactoring Code:

Before refactoring:

https://github.com/ViktorPalmason/Professional-Programming-Report/blob/859d56598991f498418cbeb1eb85df93296fba72/RandomUnitGenerator.cs#L25

After refactoring:

https://github.com/ViktorPalmason/Professional-Programming-Report/blob/7e2df71868fd21d5a47887d88d3626025f45b443/RandomUnitGeneratorRefactored.cs#L25

The code that I am using as a refactoring example is a refactoring of a function that belongs to one of the unity assets that was used in my bachelor project. The function spawns playable units randomly on the game map where the game map is oriented on either the XY(2D) or XZ(3D) plane and while testing the generator I found two bugs. The first one is that the generator forgets to rotate the unit objects if the game map is on the XZ plane. The second is if you do some position changing in the units Initialize function then it does not change relative to the units parent object, the object that stores all the units in the game. This is because the initialize function is called before the unit parent object is assigned. So to fix the bugs the unit object's parent must be assigned and the unit object's rotation must be assigned to the rotation of the parent object's before the unit initialize function is called.

Professionalism in programming

- *A personal reflection about professionalism in programming*

To me professionalism is something you learn and develop over time practising any profession. So the guidelines and standards that exist are not set criteria that one must strictly follow to be professional, but rather something to keep in mind when you are working. And one should take into account that the environment you are working in can fluctuate the value of one criteria from others.

I don't have a grasp on being professional. I am still too inexperienced and even though I have had multiple lectures and heard often from professors what it means to be professional, I still don't know how to be professional. At best I keep their teachings in my mind and do my best when working to fulfill them.

It is personally hard for me to take the advice and teachings from an experienced programmer about professionalism because it is hard for me to relate to them and their experience and it is hard for me to really feel the benefits without having felt the lack of it first, call it the lack of programmers empathy.

I have a few years of experience in programming and this includes experience in different languages and in different kinds of development. But during my time programming I was mostly just following what I was taught and did not really question it or ask what I could do to improve it. As mentioned before, professionalism is just a set of principles that I have in my mind when working but I do not know how to have. A characteristic can be “Make high quality code” but what does it mean exactly? My only reference to what code looks like is code written by me, from fellow students, professors and what little I have seen online and never did I sense a huge difference in quality. I am aware that code from more experienced programmers should be of higher quality but I can’t tell what exactly makes their code of higher quality than mine, and this I believe stems from my lack of experience.

The only characteristics of a quality code that I can confidently sense is readability which I think is the one the first quality a programmer will acknowledge and understand. That is because reading, literally or figuratively, is a skill we have done and practiced since we were children. When I will acknowledge and understand the other qualities, for example maintainability, extensibility or robustness, I have no idea. I don’t have any real life experience that I can take inspiration from but I know that continuing programming and gaining experience that I will eventually start sensing them.

Communication skills are important to a programmer and are part of being a professional programmer. How you communicate to your teammates, bosses, employees, clients, stakeholders etc. greatly affect the projects you are working on. But communication is not taught at this programming degree, just mentioned to students that the skill is part of being professional and but in my opinion great communication skills are not easily obtained. The difference between a great communicator and a good communicator is huge, just as it is with good and great programmers. The difference between them is experience and education, there are jobs and I believe degrees that solely focus on communication. This has to be understood by programming professors and universities. They need to take teaching communication more seriously and establish courses to teach students communication and not just mention it in several lectures to then overshadow it by focusing on how to write “Quality Code”.