Personal Reflection

KF5012 Software Engineering Practice

W20021023

Wordcount: 3542 words

# Part A: Reflection on Development

A1: Description 2

A1.1: My Work/Missions – 2D Art and Interface and Sound Design 2

A1.2: My Work/Missions – Project Management and Game Design 3

A2: Analysis 5

A3: Lessons to be Learned 7

## A1: Description

Since I am on the “games stream” path (studying Computer Science with Games Development), the task our team has undertaken as the main part of this module was to create a fully-fledged (while small-scale) video game.

The game we have made was titled “Temple Run”; it is a 3D 3rd person death-run style game, where the player attempts to reach the end as fast as possible, while navigating around traps or different obstacles, made to slow the player down or completely end the run. It was inspired by different games in the same genre or in the genre of 3D platformers as a whole. The game is themed to Mesoamerican (Aztec, Inca, Mayan) motifs and takes place in an accordingly designed temple.

As required by the module, the game was made in Unreal Engine 4, however our team also used other applications (such as Discord or GitHub) for communication and collaboration.

I have undertaken the missions “Project Management” and “2D Art and Interface” and the team missions “Game Design” and “Sound Design”. My responsibilities also included other tasks, mainly maintaining, and taking care of our team’s GitHub repository.

Overall, the development is presumably to be considered as success, as we have achieved the major objectives - that is to create a working video game with all the fundamental features a video game should have, such as gameplay with a clear goal and failure, functional controls and menus, graphics, and sound. However, the process has definitely not went without any complications or issues. These could be divided into two main categories – game development related, and team coordination related. And then further into game features wanted/planned versus game features actually made, the quality and polish of such features, troubles related to software used in the development process, difficulties with teamwork and cooperation and finally problems with meeting deadlines and time constraints.

### A1.1: My Work/Missions – 2D Art and Interface and Sound Design

When it comes to my missions associated with creating the game itself (2D Art and Interface and Sound Design), I am quite satisfied with my work, the features I wanted to implement (and implemented) and their quality.

Due to the nature of our game, which had very little information that needed to be displayed to the player in-game (i.e., no inventory, no health, no minimap, etc) and thus a substantial HUD (heads-up display) did not need to be built, I chose to focus on a more robust menu system, which was the second major part of the 2D Art and Interface mission. Besides the visuals and the functionality of the menus, I have also implemented the game’s settings system.

I believe that accessibility, or the effort to make video games playable for everyone (including for example persons with disabilities), is very important, and as a consequence of that, I had decided to design a special set of “accessible” user interfaces catered to this use case. Every title screen or in-game (i.e., pause, game over, victory) menu and the score counter have an accessible version with a clear white background, large black font, standard buttons and other widgets, and the option for menu narration (screen reading).

The game interfaces consist of several main (title screen) menus, a couple of in-game menus and the score display. The Main menu contains the traditional Play and Quit buttons and buttons that bring up other menus – Score menu, Options menu, and Credits menu. The Score menu displays the best scores achieved by the player in the game, the Options menu allows the player to modify the game’s settings (such as graphics quality, volume, etc) and the Credits menu presents the authors of the game and the technologies used in its development. The in-game menus are comprised of Pause menu, Game Over menu and Victory menu, which are brought up either by the player or by a specific event in the game. The in-game score display shows the current score.

The game can be altered in several ways through settings (in the Options menu), these include for example maximum frame rate, fullscreen mode, or volume. The effects take place immediately and are locally saved and then loaded when the game is launched, making the player’s options choices persistent over sessions. The code driving these settings is loosely based on the singleton design pattern.

One part of the Options menu that had been planned was ultimately not created – the controls settings, which would allow the player to modify the input controls (movement keyboard keys, mouse sensitivity, etc). The functionality of this segment, that is applying the chosen keys to the character controller in the game and their subsequent saving and loading, was to be made by another team member, however this has not come to pass.

Initially, the visuals for the (classic/not accessible) menus were supposed to be made with 2D assets, however no adequate 2D assets that would match the game’s theme and were at the same time suitable for a game menu were found. Instead, the scene was built from 3D assets from several different free asset packs.

Our team chose Sound Design as one of our team missions. At one of our team meetings, we had collectively decided to tackle it (the game’s sound) separately in relation to everyone’s personal missions (i.e., a person with the mission Environment Creation would create sounds for the game’s environments, for example the background ambience and the sounds of static objects). Thus, I had been tasked with making sounds for the game’s user interfaces. I had decided to create my own background music (alongside the button sounds and ambience) for the classic title screen menus and also do a narration of on-screen text for the accessible menus, toggleable in options.

Later in the project I also made a custom themed cursor for the classic menus. Unfortunately, Unreal Engine 4 does not have switching between multiple different cursors properly implemented, which meant that some already finished parts of the menu system had to be reworked, in particular the accessible menus, which were created early in development.

### A1.2: My Work/Missions – Project Management and Game Design

Evaluating the other two missions I have undertaken is more difficult, since the work done cannot be as easily quantified, however the game meets all the requirements of a game and was finished in time with no major disputers or obstacles.

Our game was based on an established video game genre, and we had a clear vision for its parts and features since the very beginning of the development process, and thus there were no large obstacles that would need to be overcome in terms of the game’s design process. The team agreed pretty much on all components of the gameplay, on the game’s theme and on the direction of its design throughout the development.

When it comes to project management, I am not fully sure how to accurately judge the development process. The project was successfully developed and finished, however the team cooperation was far from perfect, which has had impact on the project’s progression and caused a few substantial setbacks.

The biggest problems overall were associated with adopting the use of git (and GitHub) and as a consequence with properly sharing work. No team member had any previous experience/knowledge of git or version control system technologies as a whole; the team still agreed that utilizing a GitHub repository would be the best approach to game development collaboration. I have taken the repository management upon myself (which eventually led to me adopting the Project Management mission) – setting up the repository for the team’s needs, creating some useful files and resources and handling errors. While I have familiarized myself quite well with the technology over the beginning weeks, other team members were very reluctant to do so, and whenever they would encounter any complications, they would refuse to continue working on the assignment until those complications were resolved (remotely by me), which led to (already made) parts of the project being unavailable for other team members, sometimes for several days, and caused delays in the progress of the project. I had tried to provide the team with many resources related to version control systems and git/GitHub and even offered my help multiple times, but no effort was made to use it.

One major issue we have found is that git cannot correctly combine some Unreal Engine files, this led to our work being erased/overwritten when two team members worked on the same (particular) file at the same time. Because of this we had to make workarounds, where when one person was working on a file known to cause such issues, others would have to wait until the work on that file was finished and only then start to make changes to it. Even then it still happened on multiple occasions, which meant we had to rebuild parts of the game that had already been made (and tested) before.

Unfortunately, one of our team members was unable to attend in person. This was obviously respected, and the team decided to collaborate fully online to not leave any members behind, but that definitely led to less focused work and less cooperation. I had proposed to at least carry out a few work together sessions with people who could meet in person, and to then inform the remaining team member individually about the progress and decisions that were made there, however this was refused by the rest of the team. Even though I had suggested to do some work collectively, in particular on the parts of the game that required more direct collaboration, other team members preferred to work alone, which caused multiple problems – for example the game’s environment not being built to scale to the character controller.

While we had managed to keep the game in the projected boundaries with no “feature creep”, some team members had slightly overestimated their abilities/underestimated the time commitment needed for some of their tasks. This had been somewhat accounted for in the proposed development plan, where the three weeks of holidays were left out as buffer time for completing unfinished work or getting ahead on some future tasks, but that still (particularly in combination with other troubles, such as git deleting our work) did not avoid “development crunch” in the last few days of the assignment.

I had set up a task calendar for the team to keep track of what is finished, what is being worked on and what is planned, but this tool was unfortunately seldomly used by other team members and even though I had planned my personal work quite well, I (and the team as a whole) had still been impacted, for example the controls menu I had planned was never created (because the systems for changing the input in the character controller were not made).

## A2: Analysis

Overall, developing the project (and the module as a whole) was a valuable experience to be gained. Working in a team and participating on a software creation throughout its whole development cycle from its inception to release are both very beneficial for personal improvement.

The biggest issues stemmed from lack of knowledge and unfamiliarity with the software that was being used for the development of the game. Me (or any of our team members) had no previous experience with git and Unreal Engine. While most of us used Unity (a different program for game development) before and were thus not completely new to game engines, Unreal Engine, and particularly its Blueprint scripting system is quite different and takes some time to get used to. The same can be said about git, there are not as many commands to be remembered (for GitHub there is even an application with visual user interface for those who do not want to use the command line; or there are version control system tools built into Unreal Engine itself), but the functionality is rather unintuitive at first, especially in the case where a person has not used it before.

Unreal Engine is not as beginner friendly compared to Unity. The widget editor, where I have spent the most amount of time because of my mission (2D Art and Interface), is very similar to other user interface editors I have seen before and I had little problems with it, however other team members have reported some difficulties and annoyances with other parts of the software. No one on the team liked the way Unreal Engine connects assets through meta references, which sometimes made folders and files unable to be deleted, renamed, moved, or properly copied in the editor; some team members also felt like that the engine does not offer a lot of freedom and working in it often feels like searching for the only “correct” way of doing something, instead of creating it to one’s liking. Perhaps a different technology (and certainly a better familiarity with it) would make the team better at meeting the project’s objectives.

This could be also tied to problems with the team’s collaboration, which suffered mainly from team members realizing how much time a task would take them too late, thus running behind on agreed schedule and blocking other team members from working on the assignment. This could be mitigated with allocating more time for the project (which was obviously impossible because of the assignment’s deadline), using a different software for the project’s development or being more experienced with it as suggested above, or with better communication, however an imperfect teamwork was to be expected with a team of four strangers, who have not collaborated together on anything before. Some things related to that could also not be avoided, such as a team member being unable to meet in person. Looking back at the process, an argument could be made for better distributing the given time between tasks, however since for most team members this project was their first major software engineering project, it was difficult to correctly guess how much time and labour a portion of the game would require.

The team was also slightly unsure about how much is expected from them in terms of the assignment, that is if too little or too much was being done. While the assignment briefs (and videos) are very clear, and we had even asked the tutors about this, it did not fully clarify the extent of the work expected. Perhaps an example project could be shown to make it more apparent, on the other hand a better communication with tutors would definitely be beneficial to remove any ambiguity.

While the teamwork was not flawless, all team members acted professionally throughout the whole duration of the project, all ideas and opinions were respected, and all suggestions discussed and considered (the weekly team meetings helped immensely with shaping the vision of the game and the future of the development). Everyone attempted to make and finish their work to the best of their abilities, and no one tried to undermine the project or somebody else’s efforts. There were no disputes that would need to be resolved and the team even mostly agreed on the game’s parts and the jobs that needed to be undertaken.

## A3: Lessons to be Learned

The whole development process could be considered a big lesson, both in getting to engage with industry standard software and in teamwork and cooperation with others on a larger scale project.

It is always difficult to become fully acquainted with a new technology, especially one as complex as a game engine, but each subsequent use will make working with it easier and easier. It is definitely better to slightly underestimate one’s abilities, particularly on tasks one is not as (or not at all) familiar with, and to not expect perfection or a flawless product. As an example, I had chosen to write my own music for the menus, despite me never composing any (particularly not Mesoamerican) music before; while I believe it turned out “alright” and was a fun experience and I would do it again, considering the time spent on such a small improvement, it is hard to justify in hindsight. The same could be said about the custom cursor, which, because of not being very well supported in Unreal Engine, caused me to rework a significant portion of the accessible menus, that is to recreate something that had already been made (and tested to be functioning), again, for a minor enhancement.

Being forced to fix issues with both Unreal Engine and git/GitHub had taught me a lot and since both are widely adopted in the software engineering community, it is thus an invaluable experience, and even though resolving the (sometimes quite aggravating) problems with them was often very time consuming and an annoyance to be had, it made me more confident in my abilities and can be undeniably not viewed as “lost time”.

I believe I had done a decent job at leading the team. I think (and hope) that none of the team members felt like their ideas were not being listened to, or that they were forced into doing more work than expected or work they did not want to do. That being said, the teamwork, while not being bad, was certainly not perfect. Being a “Project Manager” while having no power is a more difficult job that it may seem at first, especially in a team of complete strangers, where keeping good attitude is (in my opinion) more valuable than forcing others to strictly meeting deadlines or punishing those who do not finish their work in agreed time. Unfortunately, this had caused me to mostly doing propositions or making instructions, instead of giving orders, and a few times downright passively-aggressively begging for something to be finished as soon as possible, because other parts of the project were being stalled as a result of that not being done yet. Being unable to demand something (with authority) undeniably impacted the progress, which crystalized as flaws in the final product; for example, the controls menu that was never made, the lacking in-game HUD, or the un-avoided development crunch towards the end of the assignment (for instance I had waited weeks for a couple of short voice recordings from other team members for the accessible credits menu, despite asking for them multiple times, something that could have been quickly finished much sooner).

I had definitely learned a lot about managing a team and being in charge of a larger software engineering project, in particular the importance of proper planning, task allocation, communication, and how to approach troubles and solving difficulties. While the exact tools and ways in which a team operates depend strongly on the team itself, knowing which approaches have (and have not) worked in the past is very beneficial, and can only help the success of a project, from any position, whether managerial or not.

# Part B: Employability Skill Plan

B1: Target Position 9

B2: Job Adverts Used 10

B3: Skills List 11

B4: Personal Skills Audit 12

B5: List of Final Year Modules 13

B6: Identification of Which Skills (from B4) Will Be Improved by Chosen Final Year Modules (from B5) 14

B7: Skills Shortfall 15

B8: Action Plan 16

## B1: Target Position

I have not yet decided on the exact position I would want to apply for. I would prefer to work in software development (and rather in frontend than backend), but I would not mind a more hardware related job. If it was possible to get a job in game development specifically (including for example game engine design), I would favour such posts to others, however other factors would have to be taken into consideration, such as location, workload, and financial compensation.

## B2: Job Adverts Used

Qualifications/skills requirements from several job adverts for software engineering positions for games development are in the appendices.

## B3: Skills List

Often required/wanted skills from the job adverts include:

* Proficiency with a programming language, preferably C++ or C#.
* Experience with a game engine, such as Unreal Engine or Unity
* Understanding of game design concepts
* Knowledge of programming principles, object-oriented programming, design patterns
* Excellent communication and the ability to work in a team
* Good problem solving skills
* Portfolio, previous professional experience

## B4: Personal Skills Audit

I am quite confident in my programming abilities, specifically C#, where I have written the most amount of code; I have even programmed in C++ before, however this was a rather long time ago and I have not used it since. While not very extensive, I have experience with both Unity and Unreal Engine. I believe I have a good grasp of software (and even some hardware) engineering concepts, including game design and objective-oriented programming. I feel as though I possess good communication and team-working skills. However, I have very little when it comes to a personal portfolio and no professional software development experience.

## B5: List of Final Year Modules

* KF6015 Games Design
* KF6017 Software Architecture for Games
* KF6018 Computer Graphics and Animation
* KV6002 Team Project and Professionalism
* KV6003 Individual Computing Project

## B6: Identification of Which Skills (from B4) Will Be Improved by Chosen Final Year Modules (from B5)

The final year modules will improve all of the necessary skills, mainly games programming, software design and team working abilities. The “KV6003 Individual Computing Project” module will also provide something for a personal portfolio.

## B7: Skills Shortfall

The final years modules will mostly cover all of the necessary skills but may fall short on providing as many large-scale projects for a personal portfolio.

## B8: Action Plan

I will work on some personal projects, which could be presented in a portfolio, such as participating in game jams.

# Appendices

Appendix A 18

Appendix B 19

Appendix C 20

Appendix D 21

Appendix E 22

## Appendix A

**QUALIFICATIONS**

* A Computer Science or Game Design degree with a programming focus is considered an advantage. However, relevant experience is taken into consideration.

**SKILLS**

* Strong C# programming skills.
* A strong understanding of top-down design, object-oriented programming, design patterns, and game design principles.
* An online portfolio with examples of your work, including code examples.
* Excellent communication, technical & problem-solving skills.
* Highly organized with proven ability to deliver on deadlines.
* A game design-oriented mindset, and a passion for creating games.
* Candidates must be highly motivated, positive and personable, with the ability to work as part of a team.
* Adaptable and ability to multi-task and prioritize work based on changing schedules and feedback/critiques with a detail-obsessed eye for quality.
* Ability to architect, engineer, and implement clean and concise APIs.

**PLUSES**

* Please note that these are desirable skills and are not required to apply for the position.
* Prior experience working in medium to large sized development teams.
* A shipped title, preferably Xbox/PlayStation.
* C++ experience.
* Experience of game design/scripting in a professional environment.

## Appendix B

**Requirements**

**Who we are looking for:**

**The successful candidate will be able to demonstrate the following:**

* Experience of professional game development with Unity
* Experience and knowledge of mobile development
* Ability to write gameplay functionality
* A good understanding of game architecture and programming practices
* Strong Unity and C# skills
* Highly mathematical
* Ability to work well independently

**Desired technical experience/knowledge:**

* Knowledge of Dependency Injection frameworks (Zenject/Extenject)
* Knowledge of C++ and other languages
* Experience with AWS or Azure
* Experience with BaaS (PlayFab, Gamesparks, Chilliconnect)
* Experience with platform specific frameworks (Gradle, Cocoapods, Xcode)
* Experience developing multiplayer games

## Appendix C

**Core Skills**

* Deep understanding of C++
* Clean and robust programming
* Experience with Git and other version control systems
* Creative attitude to problem-solving
* Ability to work as part of a team and take an initiative
* Exceptional verbal and written communications skills

## Appendix D

**Key requirements for this remote role:**

* Strong C# and gameplay programming skills
* Knowledge of Unity and experience with game optimisation for all platforms
* Experience with game optimisation
* Knowledge of programming design patterns for modular and reusable code
* Knowledge of debugging tools and build pipelines
* Experience with DevOps/CI tools
* Experience with Version Control software

## Appendix E

**We are looking to hear from candidates who have the following skills/experience:**

* Proficient and competent with at least one programming language that is widely used in the games industry (C++, java, C#).
* Familiar with at least one major game engine (Unity preferred) and scripting within the engine.
* Technical abilities in game dev software used in the production pipeline (e.g. 2D, 3D modelling, animation, sfx).
* Programming skills for gameplay, physics, UI, AI, audio, and porting (including mobile platforms).
* Familiar with at least one major game engine (Unity preferred).
* Proven development experience in delivering programming for games
* Proven project management and planning skills.
* Understanding of co-design principles, rapid prototyping and sprint methodologies.