

BACHELOROPPGAVE:

**Project NORS
A Multiplayer Online Battle Arena Game
Implemented in Unreal Engine 4.**

FORFATTERE:

Jonathan Ness
Aleksander Olsen
Marius Rødland
Chris Andre Sand

DATO:

15.05.2015

Sammendrag av Bacheloroppgaven

Tittel:	Project NORS Et Flerspiller Nettbasert Kamp Arena spill Utviklet i Unreal Engine 4.
Dato:	15.05.2015
Deltakere:	Jonathan Ness Aleksander Olsen Marius Rødland Chris Andre Sand
Veiledere:	Simon McCallum
Oppdragsgiver:	Høgskolen i Gjøvik
Kontaktperson:	Project NORS, Project.NORS@gmail.com
Nøkkelord:	Norway, Norsk, Nors, Unreal Engine, Bachelor, MOBA, IMT
Antall sider:	204
Antall vedlegg:	
Tilgjengelighet:	Åpen

Sammendrag: NORS er et nytt MOBA spill med spennende egenskaper som det å kunne mestre naturens krefter. Med denne kraften kan spillern endre miljøet til sin fordel i denne episke kampen om Storbritania mellom de brutale Vikingene og Saxons. Denne oppgaven vil beskrive prosessen vi brukte når vi skulle lage NORS og hvordan det hele startet som en ide og endte opp som et fullverdig spill.

Summary of Graduate Project

Title:	Project NORS A Multiplayer Online Battle Arena Game Implemented in Unreal Engine 4.
Date:	15.05.2015
Participants:	Jonathan Ness Aleksander Olsen Marius Rødland Chris Andre Sand
Supervisor:	Simon McCallum
Employer:	Høgskolen i Gjøvik
Contact Person:	Project NORS, Project.NORS@gmail.com
Keywords:	Nors, Unreal Engine, Bachelor, MOBA, IMT
Pages:	204
Attachments:	
Availability:	Open

Abstract: NORS is a new MOBA game with exiting features such as, controlling the elements. With this ability, the player can control and change the environment to their benefit in the epic battle for Great Britannia between the brutal Vikings and the Saxons. This thesis will describe the process we used to make NORS and how it started as a mere idea to a complete game.

Preface

We would like to extend our thanks to Simon McCallum for being the project supervisor. He provided valuable guidance and encouragement throughout the project. A big thanks to Epic Games and the Unreal Engine community for valuable tutorials, plugins and help that were used throughout the project. Notable ones being the VaRest plugin by Vladimir Alyamkin and the Video Tutorials by Epic Games.

Contents

Preface	iii
Contents	iv
List of Figures	viii
1 Introduction	1
1.1 Project Description	1
1.1.1 Background	1
1.1.2 Goal	2
1.2 Project Audience	2
1.2.1 Product Audience	2
1.2.2 Thesis Audience	3
1.3 Academic Background	3
1.4 Roles	3
1.5 Development Framework	3
1.5.1 Software Development Methodology	3
1.5.2 Schedule	4
1.6 Document Structure	4
2 Requirements and Design	5
2.1 Design	5
2.1.1 Game Design Document	5
2.1.2 Initial Design	5
2.1.3 Psychology	7
2.1.4 Final Design	7
2.2 Requirements	8
3 Technical Design	9
3.1 Unreal Engine	9
3.2 Blueprint	9
3.2.1 Classes	10
3.2.2 Variable	10

3.2.3	Function	10
3.2.4	Event	11
3.2.5	Macro	11
3.2.6	Colapsed Code	12
3.2.7	Actor Component	12
3.2.8	Widget	12
3.2.9	Construction Script	12
3.3	Version Control	13
3.4	Tracking and documentation	13
3.5	Blender	13
4	Development Process	15
4.1	Working Hours	15
4.2	Development Tools	15
4.2.1	Unreal Engine 4	15
4.2.2	Blender	17
4.2.3	Version control	18
4.3	Project Workflow	19
4.3.1	Scrum	19
4.3.2	Schedule	20
4.4	Development Workflow	21
5	Implementation	23
5.1	Spells	23
5.2	Items	24
5.3	GUI	25
5.3.1	Inventory	26
5.3.2	Shop	26
5.3.3	Minimap	27
5.4	Blueprint Structure	27
5.4.1	Pawns and Character	27
5.4.2	Champion and Proxy Pawn	28
5.4.3	BaseMinions	28
5.4.4	Minion_FootSoldier	29

5.5	AI	30
5.5.1	BaseMinions	31
5.5.2	Minion_FootSoldier	32
5.5.3	Jungle Creeps	32
5.5.4	Pathfinding	32
5.5.5	Behavior Tree	34
5.5.6	Behavior Tree in Unreal Engine 4	34
5.5.7	AI controller	37
5.5.8	Minion AI Controller	37
5.5.9	Tower Targeting	38
5.5.10	Conclusion	38
5.6	Physics	38
5.6.1	Collision	39
5.7	User Data	40
5.7.1	Security	41
5.7.2	Blueprint Solution	41
5.8	Ingame Networking	42
5.8.1	Replicated Variables	43
5.8.2	Replicated Functions	43
5.8.3	Problems	44
5.9	Buildings	44
5.9.1	Tower	44
5.9.2	Main Hall	44
5.9.3	Barracks	44
5.10	Animation	45
6	Deployment	47
6.1	Building	47
6.1.1	Different Machines	47
6.1.2	Errors	47
7	Testing and User Feedback	49
7.1	Testing	49
7.1.1	Testing with kids	49

7.1.2	Testing with Game Programming students	50
7.2	User Feedback	50
7.3	Balancing	50
8	Discussion	53
8.1	Different approaches	53
8.1.1	Testing and User Feedback	53
8.1.2	Class Structure	53
8.1.3	Code Limitation	53
8.2	Future Development	53
8.2.1	Features	53
8.2.2	Ingame Content	54
8.2.3	Future Bachelor Project	54
9	Conclusion	55
	Bibliography	56
A	Project NORS: Terminology	57
B	Project NORS: Project Plan	59
C	Project NORS: Design Document	71
D	Project NORS: Gantt chart	78
E	Project NORS: Confluence Wiki	81
F	Project NORS: Business Model	94
G	Project NORS: Grouprules	97
H	Project NORS: Daily Scrum Log	100
I	Project NORS: Meeting Notes	150
J	Project NORS: Toggle Time Tracking	169

List of Figures

1	The different classes	6
2	The different elements	7
3	Variables in Blueprint	10
4	Function in Blueprint	11
5	Widget Designer	13
6	Bad Blueprint Code	15
7	Better Blueprint Code	16
8	Good Blueprint Code	16
9	Blender: Blender Start Viking	17
10	Blender: Armature of Viking	18
11	Blender: Weight-Paint Viking	19
12	Blender: Viking UV	20
13	Blender: Viking Animation	21
14	Blender: Viking Apply	21
15	Blender: Viking Fbx Export	22
16	The spells data table	23
17	The flow of the spells	24
18	Table example	25
19	GUI Explained	26
20	Shop example	27
21	UML Diagram of BaseMinion	29
22	Creation of subclasses	30
23	Minion	31
24	Unreal Engine Components for the Minions	31
25	Navmesh Polygon graph	33
26	BehaviorTree: minionBehaviorTree	36
27	UML AI Controller	37
28	Minion AI Controller UML	38

29	Creating of new Trace Channel	40
30	Simple Collision box	41
31	Collision options for objects	42
32	Animation Structure	45
33	BlendSpace	46
34	ChampionStartAnimation	46
35	Feedback diagram	51

1 Introduction

Introduction to the bachelor project called Project: NORS. For clarification of terms being used in the thesis, refer to Appendix A Terminology.

1.1 Project Description

The task is to make a MOBA game for our bachelor thesis. We decided to work in Unreal Engine 4 (4.7.x) with its benefits and limitations. MOBA is a type of game where two teams of players battle against each other on a map. Each user control their own champion and fight the other teams champions. The objective of the game is usually to destroy the enemy's main base building through various means. Our intention is to have some unique tweaks to the game, to prevent it from being just another MOBA clone. As such we have the following major differences we would like to implement to make the game stand out.

- Parts of the environment is going to be changed by some elemental powers (fire, water, air and nature.) that the characters possess. For example: The Ability to freeze water so that you will lose some control over movement in a specific area, potentially even tripping.¹
- The game will implement a currency system like DotA2² and LoL, two of the bigger MOBA games. Here you slowly accumulate income over time, as well as earn it from killing enemy units. We will also incorporate control points that will increase the income over time when controlled by your own team.
- MOBA games have minions³ that constantly push against the enemy defenses, and work as the main source of income from killing them. We want to give them a smarter AI, to avoid having them run to their death immediately. They will also try to utilize the environment to their advantage, as well as trying to gain xp and grow stronger themselves.
- In most of the MOBA games out there you pick a predefined hero/champion⁴ with a specific set of skills. We will rather focus on having the user “build” up their Hero/Champion by deciding on their fighting style (Melee, Ranged or Magic) and what kind of element (Fire, Water, Air or Nature) they're going to use.

1.1.1 Background

We are all gamers, and everyone in our group are interested in MOBA games. Our Bachelor's degree in Game Programming lead up to us working on a big project, as such we decided on making a MOBA game.

¹http://divinity.wikia.com/wiki/Divinity:_Original_Sin_Environmental_Effects - Example of environment effects

²<http://dota2.gamepedia.com/Gold> - Currency in DotA2

³<http://leagueoflegends.wikia.com/wiki/Minion> - Minions in LoL

⁴<http://www.heroesofnewerth.com/heroes> - HoN

Previously in our studies we had to work with creating our own game engine. This proved to demand large amounts of resources and yield very little game in return. While there could be some benefits to making your own engine, we did not have enough resources to develop one. As such, we decided to work with an existing engine. The choice was mainly between Unreal Engine 4 or Unity 4. The deciding factor was that it was an engine neither of us had previously used.

Among the benefits of choosing Unreal Engine over Unity, is that the source code of the engine is available. Meaning we could change the source code if we found it lacking. Unreal Engine also have the possibility to write everything in C++, but we would mainly focus on the editor's visual scripting language called Blueprint.

1.1.2 Goal

Summary of Effects and Results we would hope to achieve by the project.

Effect

- We want to make 60% of the people who try the game, become regular returning players
- Our goal is to have 20% of the users become paying users to some degree.
- We want 5% of the userbase to be generating content for the game. (Skins, Hats, Voiceovers, etc) We will use the same business model as Vale does with DotA2⁵. This is described in further detail in Appendix F - Project NORS: Business Model.

Results

- We will make our 3D models openly available to make it easier for the users to generate custom skins and textures, as well as eventually edit the 3D models as well.
- We want to give the regular players a free bonus of some sort to encourage them to come back and play more.
- By having some regular sales (weekly or monthly) to encourage the users to become paying users to benefit from a sale.

1.2 Project Audience

Description and reasoning as to who the audience is for our Bachelor Project.

1.2.1 Product Audience

Since we are making a MOBA game, our targeting audience will be people that like and/or want to try a new MOBA game. We will try to target males in between the age of 15-30, but we estimate that the females (in the same age span) will be interested as well. Since we are only building for the windows platform, only windows users will be targeted. Some knowledge of MOBA games should be expected.

⁵<http://www.dota2.com/workshop>

1.2.2 Thesis Audience

The thesis is written in English to target a wider audience. The content of the thesis is targeted towards other game developers, other students with interest in Game Development, or just people in general with a slight interest. Some basic programming or IT knowledge is expected, we will describe more project specific elements about the Blueprint language in chapter 3.

1.3 Academic Background

We have the same academic background in the Game Programming degree. The degree is heavy on programming, so we have broad experience with languages. We have done most in C++ and Java, but have touched other languages as C#, JavaScript, Python and C as well. Using an already made Game engine as the Unreal or Unity is something we have done through the Rapid Prototyping course⁶. This will come in handy when we are working with Unreal Engine 4.

When it comes to different Software Development Methodologies, all we have is theoretical knowledge. Bigger projects such as the bachelor project will be a good chance to get practical knowledge and experience regarding Software Development Methodologies.

1.4 Roles

The fact that one person has something as their responsibility does not mean that others can't work on that field. Having the responsibility simply means that this person will be the one making the decisions on issues in that field. It's also safe to assume that the person with a particular responsibility will focus their attention on that field. The responsibilities are divided up as such, but is in no way final, and are subject to change on a "I want to work with..." basis.

- Aleksander has responsibility for Design.
- Chris will be responsible for AI.
- Marius work and focus on Networking.
- Jonathan on the webpage and environment.

1.5 Development Framework

The different frameworks and methodologies we will be using during development.

1.5.1 Software Development Methodology

Games are usually more agile processes where features get added, removed and improved between each iteration. As such we decided on using Scrum as our development process. Scrum is flexible to changes in the project because one work in sprints of a given length, and changes can be discussed and decided on the Sprint Planning Meetings.

⁶http://english.hig.no/course_catalogue/student_handbook/2015_2016/courses/avdeling_for_informatikk_og_medieteknikk/imt2581_rapid_prototyping_and_innovation

We decided on having sprints that last a week. As decided in the rules, we will have a meeting on Mondays. This will consist of the Sprint Review, Sprint Retrospective and Sprint Planning meeting. As well as change of Project Manager, which goes on rotation as per the rules. We will discuss this more in subsection [4.3.1](#).

1.5.2 Schedule

In the start of the project we made a Gantt chart for the progress of the project. This can be found in the appendix [B](#). Each month are divided in to its own milestone, where next milestone iterates on the previous. Some features rely on another, and as such they need to be completed in a given order. For instance: the AI can't run around on a map, if there is no map.

The plan was made before we had any experience with Unreal Engine 4, so some of the thing we planned happens at a different time or not at all. Quite early on we found we had overscoped, partly because of problems and bad time estimation. As such the milestones weren't as successful as we had hoped. We will discuss this more in subsection [4.3.2](#).

There will also be an independent meeting with our internal supervisor, Simon McCallum. This will happen weekly on Monday at 9 AM. Later in the project this might change to biweekly meetings. In the event that we need additional meetings, it's the Project Manager's job to arrange, with supervisor if necessary, as well as inform the rest of the group.

1.6 Document Structure

Summary of the Chapters in the Thesis.

1. [Introduction](#) A shorter introduction to the project.
2. [Requirements and Design](#) Describes the idea for the game, game design and what we did or didn't do.
3. [Technical Design](#) Describes the technology behind the game.
4. [Development Process](#) Describes how we worked, what tools we used and the development methodology.
5. [Implementation](#) Explains in more detail how different parts of the game was implemented.
6. [Deployment](#) Describes how Unreal Engine 4 packages the game and how the game turned out.
7. [Testing and User Feedback](#) How we tested the game, and got user feedback.
8. [Discussion](#) Discussion on the results, how we worked, and what could have been done different, added or removed.
9. [Conclusion](#) Evaluation on the whole project.

2 Requirements and Design

Chapter with basic information about the requirements and design of the game.

2.1 Design

Section containing information about our design of the game.

2.1.1 Game Design Document

According to Jesse Schell in *The Art of Game Design* [1], there are four basic elements of a game. Mechanic, Story, Aesthetics and Technology. Based on those aspects we built our design as following.

- **Mechanic** - As most MOBA games, the controls for the game is based on point and click with a mouse, and abilities being triggered by specific key presses. The map is dynamic as opposed to grid based, and utilize Unreal Engine's built in Navigation Mesh for pathfinding.
- **Story** - Story is mainly in lore and background story for the game. It could provide some story for the gameplay, but it is not the focus when playing the game. It would also appear quite repetitive unless we set up a complex dynamic story generation based on the progress of the battle.
- **Aesthetics** - As most other MOBA, the game is set in top down view. The background is semi realistic, while characters and enemies are stylized. They blend together due to their color scheme, and as such the game have a complete feel to the aesthetics. The UI is currently plain Unreal Engine 4 UI elements, but would be changed to fit the scheme of the other components later.
- **Technology** - The technology is laptop and desktop computers. Preferably with an external mouse on the laptop compared to the touchpad. A screen and keyboard is also required together with the mouse. The recommended hardware is found in section 2.2.

2.1.2 Initial Design

In the early phases of the design we came up with several aspects that would be important to the game in order to differ from the many other games in the genre. Due to time limitations and issues with development, we didn't have time to implement all of those features.

Champion Selection Aside from how creating a large champion pool of unique champions would be time consuming and difficult, we wanted to let the player build their experience a bit more by themselves. Similar to an RPG game, we would let the player chose a class and an element. These choices would determine where the player would fit best in game, but the players would be free to play wherever they choose.

The classes were intended to work like rock-paper-scissor in that one always have another class they are stronger or weaker against. As seen in Figure 1. This provide balance so that there is no one dominant strategy in class selection.

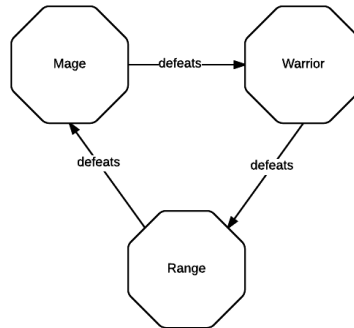


Figure 1: The different classes and how they are stronger than each other.

- **Mage** - The Mage character would specialize in ranged area of effect attacks.
- **Ranger** - The Ranger character is suited for ambush and finishing off fleeing enemies.
- **Warrior** - The Warrior character would be an up close and personal fighter.

The different classes have different abilities at their disposal, but we tried being consistent with the abilities for simplicity. The idea being that you could try out different classes and still be familiar with most of the class' abilities.

- **Q** - Would be a simple damage ability. Stronger than the basic automatic attacks.
- **W** - Would be related to movement, escaping or engaging in combat.
- **E** - Would be a unique ability for the class, changing depending on the element.
- **S** - Would be an ability meant to deal large amounts of damage.

We have four elements: Fire, Water, Nature and Air. As seen in Figure 2, they are balanced similar to rock paper scissors, with the two opposing elements simply not providing an advantage for either side.

Environmental Changes The environment is meant to have certain spots where one could apply one's chosen elemental power and change the environment. These effects would loop around depending on the elements used. Using a simple example of a patch of soil.

1. **Water** turn the soil into mud, which slow movement speed.
2. **Nature** create bushes that suck up the water in the mud, and the bushes provide cover and protection.
3. **Fire** would then burn down the bushes and leave scorched ground with glowing embers.
4. **Wind** then simply blow the embers away and return the ground to the base state of plain soil.

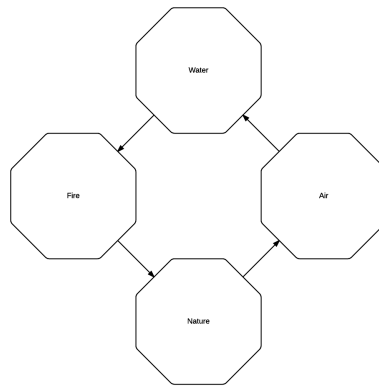


Figure 2: The different elements and how they are stronger than each other.

This would mean the players could drastically change the gameplay depending on which elements are available to play. The fact that you require some elements to counter others also encourage teamwork and communication. You need someone on the team to have wind to blow away the embers on the ground, otherwise the rest of the team take damage simply by advancing on the enemy team.

2.1.3 Psychology

There are some aspects of psychology that have determined some factors of the game. They will briefly be described here and how we've dealt with them.

1. **Motivation** - We would motivate the player to keep playing by matching them with other players by the same level of skill, as well as having rewards and progress outside of the actual game that they can strive for and achieve.
2. **Risk vs Reward** - The game will have aspects where the player can take greater risk for greater rewards. This outcome would largely be determined by the other players in the game and how they respond to the action. Some could be panicking because it's an unexpected surprise attack, and as such, the reward is great.
3. **Challenge** - As a counterpart to motivation, we would try to keep each battle balanced, so that you will feel a challenge whenever you play. Sometimes you lose, sometimes you win. This would prove that you can always improve your skills.
4. **Multiplayer vs Single player** - The MOBA genre is generally a multiplayer genre. You could play against bots, but it's more fun to play together with friends. It is reasonable that one have single player battles against the AI in order to learn the basics of the game, and improve if you don't feel you are good enough, as well as test cool tricks and abilities.

2.1.4 Final Design

Our final design deviate little from the initial design. There are some features that are yet to be implemented, but they are still in the design. As well as a few new features that we planned to add. Notable features that came later on are:

- **Game Progress by Environment Foliage** - We currently have two different colored flowers, which correspond to the different team colors. The plan was to create them as Blueprints that would monitor the state of towers, and change accordingly. This would allow a player to easily recognize if they are within safe area by seeing the flowers on the ground matching their team color. The edge where the flowers change color would move according to what towers are still standing, providing a front line. Thus, even late in the game, if both sides have destroyed equally many towers, the front line will remain even.
- **Minion Commander** - The idea of a Minion Commander came later in the project. The Minion Commander would retain experience and level after they respawn, and as such would grow stronger throughout the game. They are spawned from the Barracks, and destroying the Barracks would therefore prevent them from respawning.
- **Roaming Jungle Creeps** - We would have some smarter jungle monsters that would roam around a predefined path, instead of just standing still by themselves until attacked. Knowing where the monsters are could provide for strategies where you would lure enemy Champions into the jungle, only to have the monsters attack them.

2.2 Requirements

The project is built for Windows 32 bit, as such it is required that one have compatible OS. Unreal Engine have support for building to other OS', but this was never a priority and would require further testing on different machines.

Based on specification by Epic Games¹, hardware recommendation is as follows:

- **CPU** - Quad-core Intel or AMD, 2.5GHz or faster
- **RAM** - 8 GB of RAM
- **GPU** - Nvidia GeForce GTX 470 or AMD Radeon 6870 HD series or higher.

Software requirements are as follows:

- **OS** - Windows 7 or 8, at 64-bit.
- **DirectX** - DirectX End-User Runtimes (June 2010)
- **.NET** - .Net 4.0 (which should already be installed via Windows Update).

Any other software requirements should be installed automatically when the program is installed.

¹<https://docs.unrealengine.com/latest/INT/GettingStarted/RecommendedSpecifications/> - Engine Requirements

3 Technical Design

This chapter will provide information about the technologies we used throughout the development, and what function they have in the game.

3.1 Unreal Engine

The game is made in Unreal Engine 4.7.x, and we decided on using the Visual Scripting language that come with the engine. This scripting language is called Blueprints. This would provide a learning goal as we hadn't used Blueprints before starting the project. After struggling in the start as we were learning the language, it got better and we're able to program fast and efficiently. The language itself is easily translated from the scripting language into C++ code. It is powerful and have advanced functionality that counterparts C++ functionality. Notable examples being Event dispatchers and latent functions, working similarly to function pointers and running a function on a spawned thread.

In the event that we needed something that didn't exist as Blueprints, we could write our own classes in C++ and have them appear as Blueprints. We found and installed a plugin that implemented RESTful API calls to a server in Blueprints. The plugin was slightly outdated, so we had to recompile the plugin with the engine version we were using at the time. So we have basic grasp on how to implement new functionality in Blueprint by a plugin.

3.2 Blueprint

In this section we will try to give a simple explanation of how some the different elements of the Blueprints system works. This won't be a complete Blueprint guide, but more of a general explanation that would help one understand the basics of the source code.

Blueprint is the visual scripting language in Unreal Engine 4. You can use it to create fairly complicated gameplay elements and game logic. In Blueprints you don't write code as text. Instead we are using Nodes with Wires between them. There are two main differences between the different wires. There is the sequence of execution wires that are white, and the data flow wires which are colored depending on data type. Reference the Unreal Engine documentation about variables¹ to see what colors are associated with what variable types.

One exception is the wire associated with event dispatchers, which you connect by a thin red wire. This wire will trigger an event when the node is called, but the event might not execute immediately, thus it doesn't represent the sequence of execution as the white wire does, but is rather a variable to a function.

¹<https://docs.unrealengine.com/latest/INT/Engine/Blueprints/UserGuide/Variables/index.html> - Unreal Engine documentation on variables

3.2.1 Classes

In order to make a new Blueprint object in Unreal Engine, you have to extend some existing class. This is equal to a superclass in c++, java, etc. The new Blueprint will contain some basic functionality and variables as found in the selected superclass. Some of the classes work quite differently from each other. For instance, a Player Character base class will support movement and control, while a Widget Blueprint will support UI elements and touch interaction.

The most basic Blueprint class is an actor. It is simply an object that will exist in the world with a piece of code attached to it. One can also extend own classes in a hierarchy of objects. Further information about Blueprint classes can be found in the Unreal Epic documentation on Class Blueprint².

3.2.2 Variable

Blueprints handle variables through getter and setter nodes. As you can see in Figure 3 you can get the variable and assign a new value to it. In the figure, the first thing that happens is that we are assigning NewInt to the value of NewInt (Done as an example with little practical use). Next, we assign NewBool to true. Given that we are not using another variable as input, it counterparts hardcoding in a value in C++. The last node that you see is a branch node taking the NewBool variable as input. It works like an if/else statement and will either execute the code associated with the True or False execution outputs.

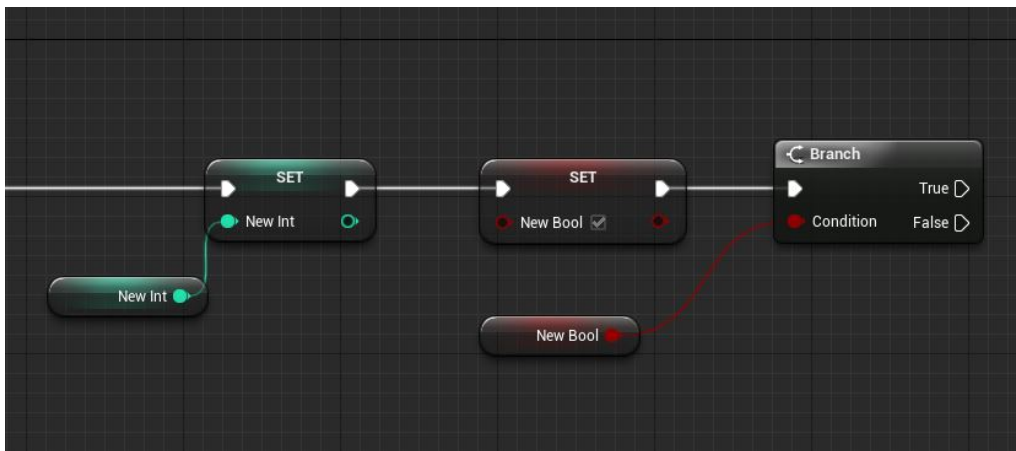


Figure 3: Variables in Blueprint

3.2.3 Function

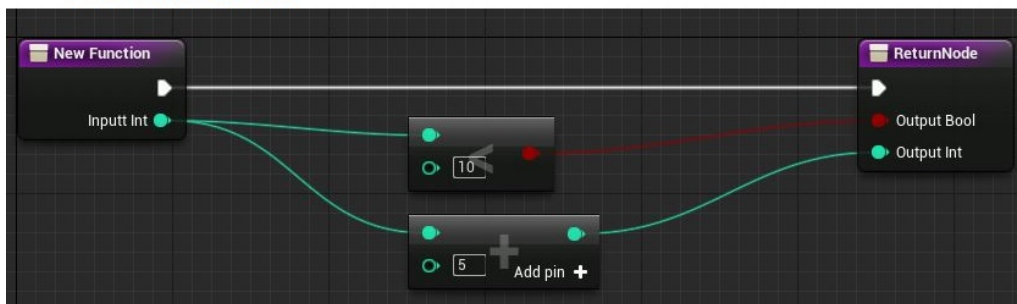
Functions in blueprint is used the same way as you would use a function in any other language. You are able to have many different input parameters, and many different return values. The example in Figure 4 have one parameter and two return values. What

²<https://docs.unrealengine.com/latest/INT/Engine/Blueprints/UserGuide/Types/ClassBlueprint/index.html> - Unreal Engine Documentation on Class Blueprint

it simply does is to check if the InputInt is below 10, and it also adds 5 to the value of what InputInt is, and returns the true/false for the check and the new value.

A compiled function appear as a node accessible elsewhere in the Blueprint. One notable unique feature is the "Target" pin on the New Function node when we are calling it. This defaults to "self" and refer to which Blueprint you are calling the function on. By default, the function is called on the blueprint that is doing the calling. But one could also call the inherited Apply Damage function from within a Player Character, and set target another Player Character. As such, the target is the one that would take damage.

The Function



Calling the function

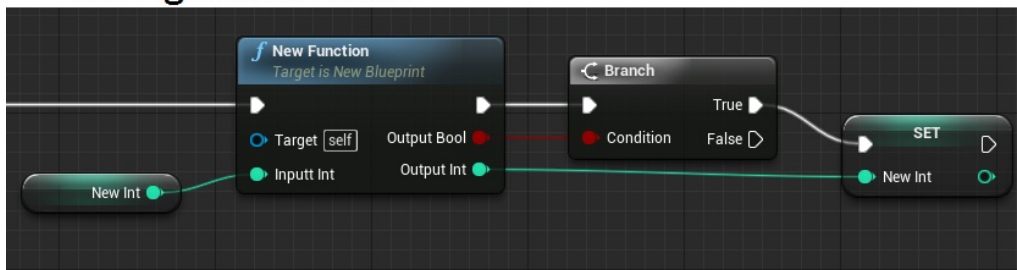


Figure 4: An example of a function and how to call it.

3.2.4 Event

An event is an entry point for code. Mainly it's used for responding to engine events such as collision, and replication across multiple computers. Events can utilize latent functions and delay, they do not return anything to the caller and as such stand separate from their caller. You call an event much like a function, except you are not guaranteed it will execute and return immediately. All events, including custom events, in a blueprint exist in the same space. This mean that multiple events can work on identical code, and even merge execution path.

3.2.5 Macro

A macro differ from functions and events in that upon compile, the content of the macro get copied in and replace the macro node where it's used. This mean that a macro that's used multiple places, get compiled multiple times. Because a macro is copied in on compile, it will face the limitations of it's parent execution. If it's called upon from a function,

it can't contain latent functions or delay. If it's called from within an event, it can. A macro in reality doesn't return any values, despite how you can give it an output execution path that pass along variables. It would be more along the lines of creating temporary values, and then keeping a reference to them after the macro have executed.

3.2.6 Collapsed Code

Especially useful for handling complexity is the ability in Blueprints to collapse pieces of code into what appears as one node. This is done simply by marking some code, and if it fit the criteria you can right click and collapse code. This allow you to easily manage sections of code, because they only appear as one node, making it easy to follow the path of execution. Alternatively you can collapse it to a function. This will create a function of the code you had selected and can be called like any other function.

3.2.7 Actor Component

An Actor Component is something that was introduced in Unreal Engine 4.7. Basically an Actor Component is some predefined behavior that you have made. You can add it as a component to your Blueprint and the behavior you made will now work as if part of the Blueprint.³

3.2.8 Widget

As mentioned in subsection 3.2.1 there are many different Blueprints and they behave differently. The Widget blueprint is a particularly interesting class to extend.

Unreal Engine 4 has a visual UI designing tool called **Unreal Motion Graphics UI Designer (UMG)** and at the core of UMG are widgets. A widget is something that defines different HUD elements such as buttons, sliders, check boxes, *etc.* These elements are designed using the designer tab in the editor window. The buttons are added from the palette panel. For instance, when executing code as a result of touch events on buttons, you first enable an "On Click Event" for the particular button you have selected. The editor then change to the graph tab where you have the Event node to start the execution.

Figure 5 is an example on how you can arrange the different elements in the designer tab in the editor. The image is taken from our game, and consist of several widget components to make up the widget. For details about what the different widgets are in this picture, see Figure 19 in section 5.3.

3.2.9 Construction Script

The construction script is a special section of Blueprints, and some Blueprints, such as Level Blueprint, doesn't have any construction script. A construction script allow us to run snippets of code as we construct the object. This applies for editing in the editor as well. An example is having a flower object that when placed, create several flower meshes of varying size, rotation and position. The flower object could potentially get it's position and determine the color of the flowers based on that.

³https://wiki.unrealengine.com/Videos/Player?series=PLZ1v_N0_01gYeJX3xX44yz0b7_kTS7FPM&video=qr4ZjieAQKY - Video explaining actor component

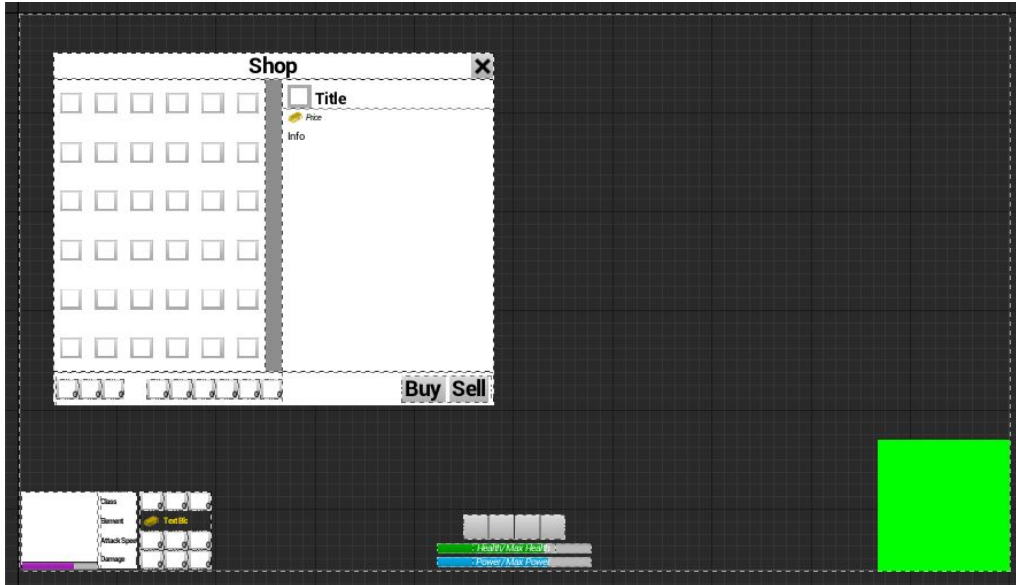


Figure 5: An example of how you may arrange your HUD in a widget.

3.3 Version Control

We had decided on using Git as our version control, and decided on Bitbucket rather than Github for hosting our repository. These are the main sites for hosting Git repositories. Bitbucket allow closed repositories and have close integration with Confluence and Jira, which we intended to use as well. Github also offer closed repositories as part of a free student pack, and because we didn't use Jira or Confluence as active as we should, there is no real advantage one way or another.

3.4 Tracking and documentation

For time tracking and daily reports we used Toggl and Google Documents. Google Documents would allow all of us to join and see what was written in the short reports as well as write themselves if something was missing or wrong. Toggl is a simple time tracker that let us specify what we're working on, when we were working, and on what project we were working.

Confluence and Jira was used briefly as wiki documentation and issue tracking. It was never a main focus and as such Jira was rarely used other than a burst of activity. Confluence was set up as a simple game wiki describing gameplay, abilities and enemies. It does not contain any code, but rather basic information, background story and lore that one would be interested in as a player.

3.5 Blender

“Blender is the free and open source 3D creation suite. It supports the entirety of the 3D pipeline. —modeling, rigging, animation, simulation, rendering, compositing and motion tracking, even video editing and game creation. Advanced users employ

Blender's API for Python scripting to customize the application and write specialized tools; often these are included in Blender's future releases. Blender is well suited to individuals and small studios who benefit from its unified pipeline and responsive development process."[\[2\]](#)

Blender is released under the GNU General Public License (GPL, or "free software").[\[3\]](#)

4 Development Process

This chapter contains an overview of how we worked throughout the project, including what it was like working with the different tools.

4.1 Working Hours

We decided on working ours from 09:00 to 15:00. Starting off with daily scrum meeting over Skype. Weekly Scrum meetings would be on Mondays after our meeting with Simon McCallum as the project supervisor. The weekly scrum being on Monday would allow us to utilize the weekend if we were behind on work.

4.2 Development Tools

This section is about the bigger tools we used, and what it was like to work with them.

4.2.1 Unreal Engine 4

At first, we had a hard time getting around and doing what we wanted in Unreal Engine. Because we weren't used to the Blueprint visual scripting language in Unreal Engine 4, it would often be quite hard to follow the call and what it was doing. The Blueprint language has a unique problem compared to regular C++ code. It needs to be visually structured properly in order to find what variables go where.

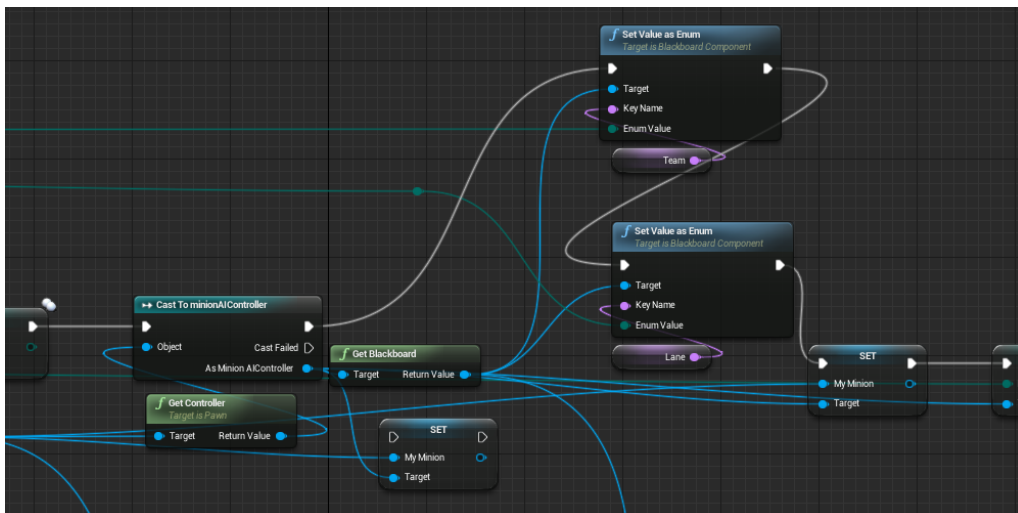


Figure 6: An example of badly written Blueprint code.

As is evident in Figure 6, it's not easy to follow where all the variables come from. This would be similar to what most of the code looked like in the beginning, if not worse.

Later on we found the "reroute node", which is essentially just a stepping stone for the wires. This node would allow us to branch out wires that would be needed later, but without having it pass through all nodes around.

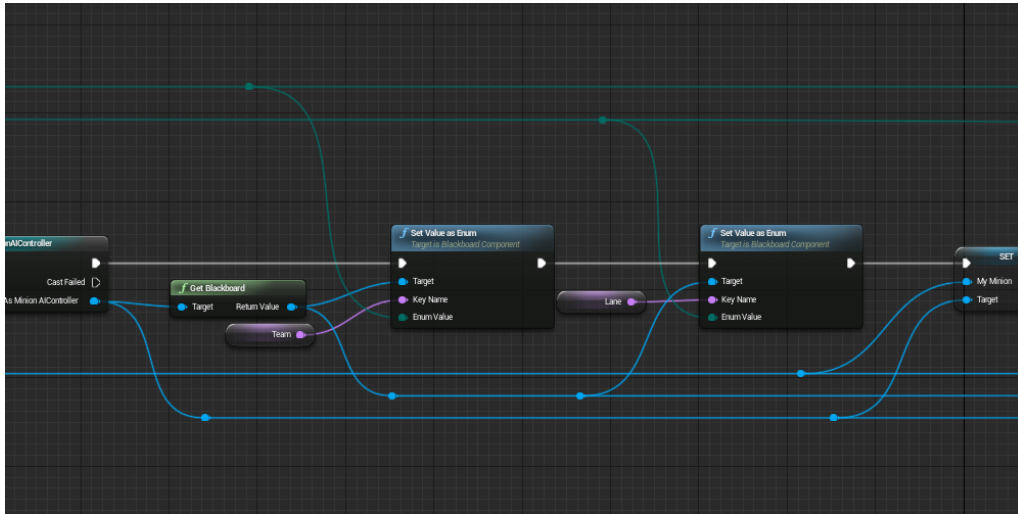


Figure 7: An example of improved Blueprint code.

As you can see in Figure 7, it's not pretty, but you've avoided a large amount of confusion based on how the wires between nodes aren't intersecting and overlapping nodes as much as previously. As such the readability of the code have increased, and it's easier to work with later on. This snippet of code is still lacking however.

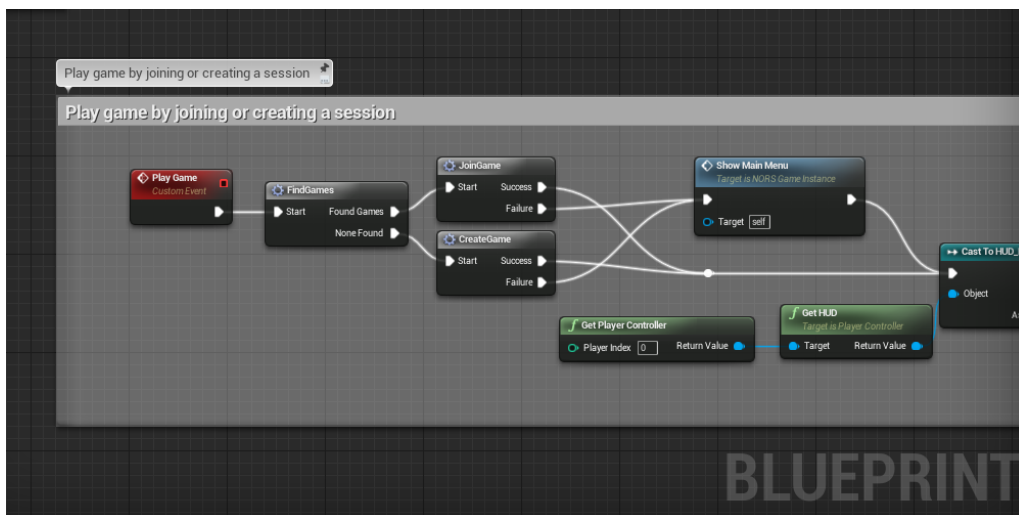


Figure 8: An example of good blueprint code.

Figure 8 is from a different section of the code. This is what one would consider good code. To start off, the code snippet is commented. Second, the code utilize macros for readability of the code. This section of the code also doesn't pass variables from start to

end, but access them individually inside the macros.

As time progressed and we figured out how to work with Blueprints, we went from bad Blueprint code to better and eventually good Blueprint code. One of the most noticeable steps in better structured code was finding the "reroute" node. But later on we also started using functions more often, and commented better when we knew what we were doing and that it would work.

4.2.2 Blender

We decided to use blender, because it's free, open-source and have all the function we needed for an external 3D application. Some assets were acquired from BitGem¹ that needed minor edits to suit our needs. This was done in Blender v2.74. We will show parts of the process of editing a model using the Viking model from the game. We started by importing the 3D model as a .fbx file, and Figure 9 illustrates how that appears in Blender.

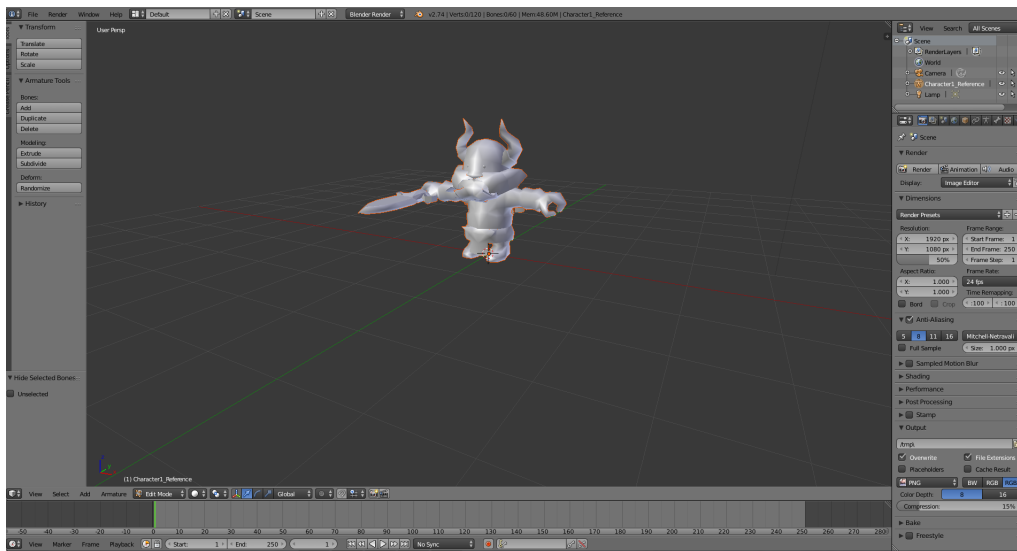


Figure 9: Imported 3d model: viking

We now have a polygon model with an armature rigged and weight painted. An armature is Blender's understanding of skeleton. As seen in Figure 10, an armature consists of the bones and joints between them. In order to animate the mesh, we transform bones using scale, rotation and location.

The vertices of the mesh are weight painted to the bones to determine how close they should follow the movement of the bone. Fingers for instance, have to move quite close to the bones in the finger, while the chest piece of a suit of armor might not. Upon changing the model the weight paint got distorted, and had to be retouched, depicted in Figure 11. Red means a stronger connection, and blue or green is a weaker connection.

Every 3D model needs some kind of texture. To map the texture to the model we need

¹<http://shop.bitgem3d.com/> - Bitgem shop

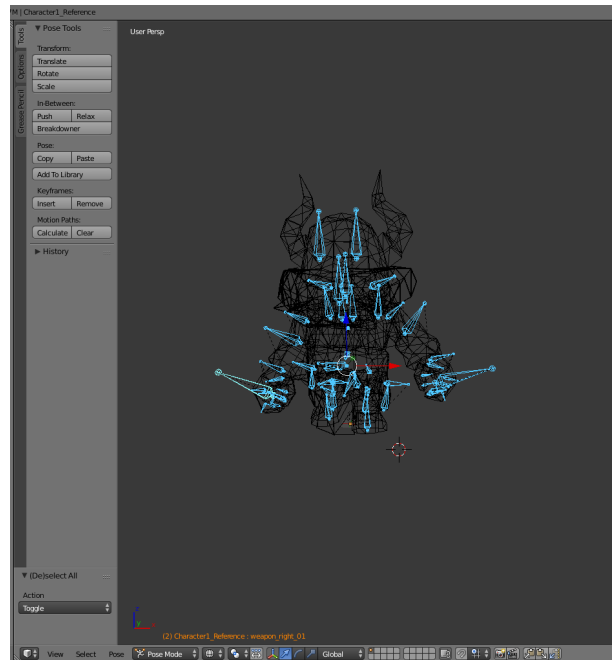


Figure 10: Armature: viking

an UV layout. Essential what this means is taking our 3D model and flatten it out so we can map a 2D picture to the surface of the models. To make a UV layout we need to cut the model open using seams. Seams marks an edge on the model where you want the UV unwrap to cut the model so it can be flattened as seen in Figure 12.

The asset came with some animations. We added some unique animations, for example the ranger class needed a custom animation when he fires his bow. This animation was done in blender by animating the bones from the armature in Figure 10. We will not explain every detail of the animation process. But as seen in Figure 13 the keys are the yellow diamond shaped symbols and represent every transformation a bone has. The orange line between them represent a graph that add an interpolation between keys in the transformation of the frames.

To properly export the assets to Unreal Engine 4, the transformation done in Blender need to be cleared or applied. Blender's default import algorithm usually scales down and rotate the model towards the camera in the scene. Moving the asset around in Blender can cause distortion when exporting to UE4. By clearing the transformation, the asset will go back to the original transformation prior to import. Applying transformations bake the data with the asset as seen in Figure 14. The asset is now exported as shown in Figure 15.

4.2.3 Version control

We used Git with Bitbucket as our version control. In hindsight, this was a bad idea. All Blueprint files in Unreal Engine 4 are binary asset files, and as such, cannot be merged or resolved other than overwriting one or the other. This is partly a flaw in Unreal Engine 4 for not having an option to save Blueprints as XML. Being such a powerful engine

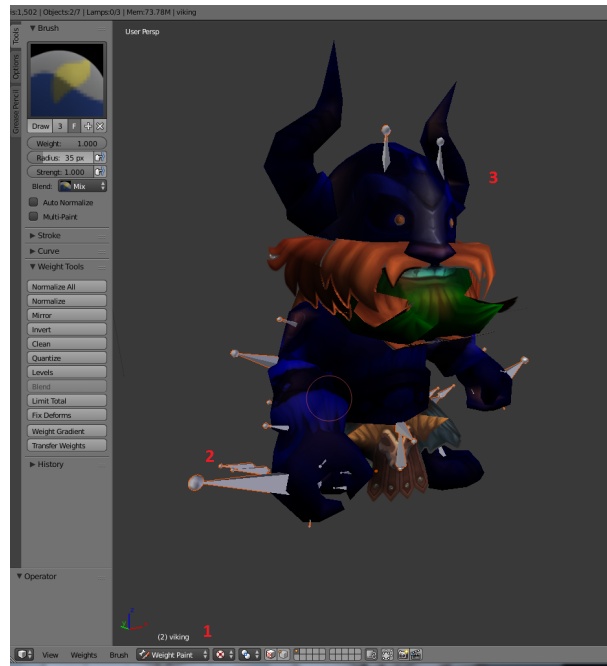


Figure 11: Weight-Paint: viking

that specifically made Blueprint to allow designers to program, it would likely cause merge errors at some point in which the designers would have to handle, or bring in the programmers.

After certain files grew in size and neared completion, we pushed them off to a shared Dropbox folder instead of on Bitbucket. A 200 megabyte map file would add 1 gigabyte to a git repository in 5 minor edits. This was the case which prompted the move of files to Dropbox.

Early on the binary files was a big issue, because everyone was working on a few of the same files. This was mostly because we didn't have much structure going at the time. This part was rather troubling in the start, because we would constantly 'lock' certain files verbally. And they wouldn't necessarily get unlocked when they were done. A bit later when we got objects and classes branching out, it became more safe to work on objects and classes, and with fewer merge errors as a result.

4.3 Project Workflow

This section will be describing the project workflow during our development process.

4.3.1 Scrum

We quickly decided to go for scrum as our system development method. Scrum allows for changes to be made under the development process. And since we are going to make a game, it's bound to come some changes along the way, as issues arise or new features get suggested. Scrum consist of sprints, where our sprints will last one week to start with. This is subject to change if needed later in the project.

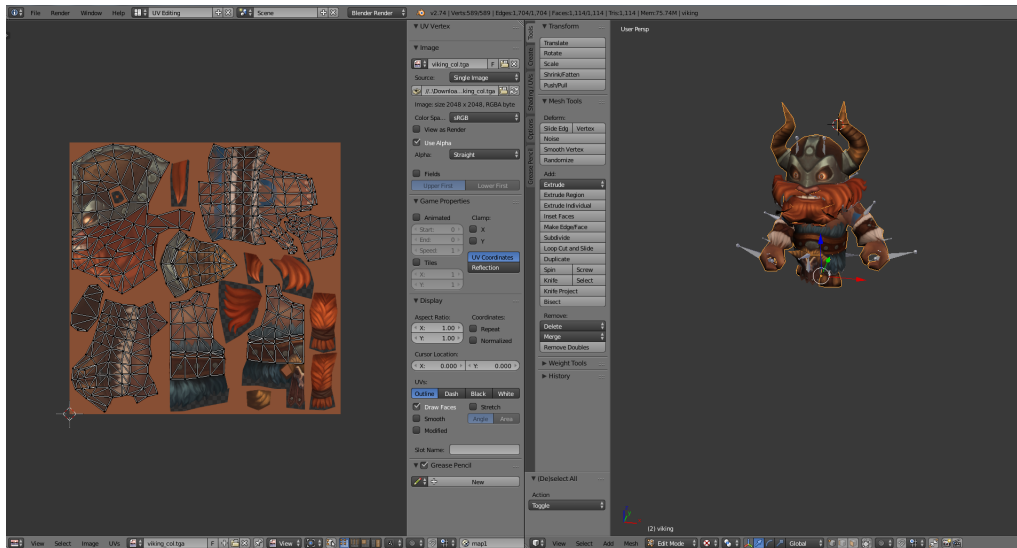


Figure 12: Viking UV: viking

The tasks will be of various sizes. In the early stages of the project, we will focus on setting up the developing environments separately to each machine. Later on in the project we will have smaller tasks, such as fixing features in the game. As well as bigger tasks such as Networking between Client and Server.

Since we are rotating on who will be project manager, we will also rotate on who will be the Scrum master. The project manager and scrum master are not usually the same person, as they have different roles entirely. But we've decided on essentially merging the two roles and treat them as one, because of the limited number of people and the lack of a proper company structure. That way everyone will get experience as both Project Manager and Scrum Master.

In Scrum there is a Sprint planning meeting and End sprint meeting for each Scrum Sprint. The sprint planning meeting is where the team discuss what to do in the upcoming sprint. End sprint meeting consist of the Sprint Review and the Sprint Retrospective meetings. In the Sprint Review meeting we show what we did do and what we did not manage to do this sprint. The Sprint Retrospective meeting we simply reflect on what went well and what can be improved.

We plan to merge the Sprint planning meeting and the End sprint meeting into one meeting. We do this to save time, and minimize the number of meetings we need to have. It also opens up the possibility of working through the weekend, instead of the usual work days of Monday-Friday.

4.3.2 Schedule

Reference the appendix D for Gantt Chart of the schedule we had prepared initially. Sadly, issues with player movement and networking, as well as the whole engine working slightly different than expected, prevented us from being able to follow the schedule as planned. For instance, networking in Unreal Engine 4 is something that should be worked

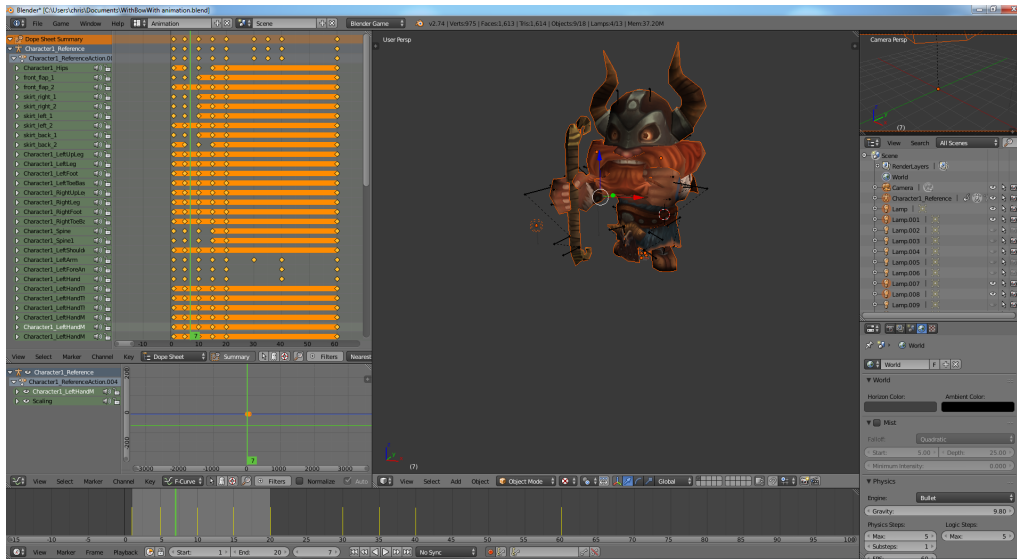


Figure 13: Viking Animation: viking

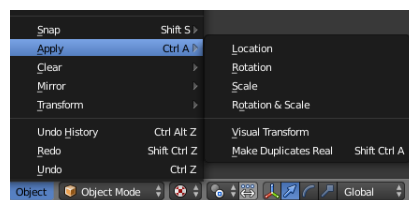


Figure 14: Viking Apply: viking

on from the very beginning, otherwise most basic functions would need rewriting to support the online client/server setup.

4.4 Development Workflow

The workflow in Project NORS starts with the Scrum Planning Meetings. We would plan for the features to be made during the sprint and decide what would get priority. We would then start working on the features by ourselves or in groups to do pair programming on sensitive files. Given that Blueprints are binary files, we rely on frequent pushes and pulls in Git to have as few large merge errors as possible.

We had Daily Scrum Meetings and would generally decide and inform the team what files we would be working on during the meeting. We would try to avoid working on the same files as much as possible to avoid merge conflicts.

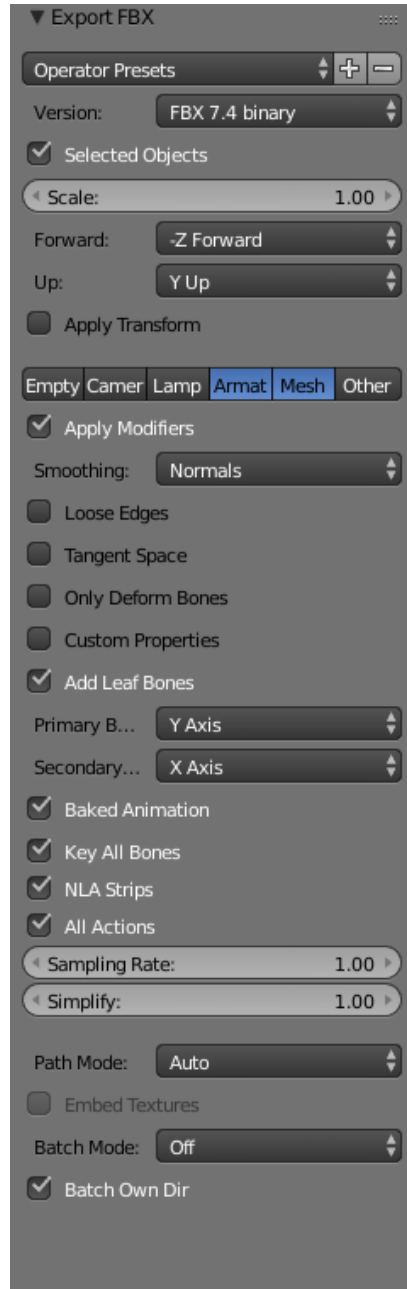


Figure 15: Viking Fbx Export: viking

5 Implementation

This chapter is about how we implemented several core features of the game, as well as issues and bugs we encountered along the way.

5.1 Spells

Each class have four spells and since we have tree different classes, that makes it 12 different spells. Because the spells share many of the same data types: damage, cost, range, *etc.*, we chose to use a table in Unreal Engine 4 to store and read the necessary information. By doing this we got a simple and clear overview over the data for the spells, as seen in Figure 16.

Name	Damage	Cost	Range	Radius	Cooldown	DamageBuildings	AttackName	AttackDescription	WhatAbility	WhatClass
MageQ	10.000000	10.000000	500.000000	500.000000	4.000000	False	MageQ	This is the mage class Q ability	NewEnumerator0	NewEnumerator0
MageW	10.000000	10.000000	10.000000	10.000000	3.000000	False	MageW	This is the mage class W ability	NewEnumerator1	NewEnumerator0
MageE	10.000000	10.000000	10.000000	10.000000	3.000000	False	MageE	This is the mage class E ability	NewEnumerator2	NewEnumerator0
MageR	10.000000	10.000000	1.000000	600.000000	3.000000	False	MageR	This is the mage class R ability	NewEnumerator3	NewEnumerator0
WarriorQ	10.000000	10.000000	100.000000	100.000000	3.000000	False	WarriorQ	This is the warrior class Q ability	NewEnumerator0	NewEnumerator1
WarriorW	10.000000	10.000000	10.000000	10.000000	3.000000	False	WarriorW	This is the warrior class W ability	NewEnumerator1	NewEnumerator1
WarriorE	10.000000	10.000000	10.000000	10.000000	3.000000	False	WarriorE	This is the warrior class E ability	NewEnumerator2	NewEnumerator1
WarriorR	10.000000	10.000000	3.000000	500.000000	3.000000	False	WarriorR	This is the warrior class R ability	NewEnumerator3	NewEnumerator1
RangerQ	10.000000	10.000000	10.000000	10.000000	3.000000	False	RangerQ	This is the ranger class Q ability	NewEnumerator0	NewEnumerator2
RangerW	10.000000	10.000000	10.000000	10.000000	3.000000	False	RangerW	This is the ranger class W ability	NewEnumerator1	NewEnumerator2
RangerE	10.000000	10.000000	10.000000	10.000000	3.000000	False	RangerE	This is the ranger class E ability	NewEnumerator2	NewEnumerator2
RangerR	10.000000	10.000000	10.000000	10.000000	3.000000	False	RangerR	This is the ranger class R ability	NewEnumerator3	NewEnumerator2

Figure 16: The data table for the spells, note this is not balanced.

Due to the fact that spells share similar variables, but work differently, we chose to create one blueprint for each spell. This let us easily code up the effect of each spell independently, without interruption or merge errors between users. Each blueprint hold an Actor Component that read it's values from the data table on startup. The spell will then refer to the Actor Component for it's data. The Actor Component would also handle casting verification by checking if the Player Character have enough Mana to cast the spell, if the spell is on cooldown or if the player is dead.

The engine sequence when using a spell is described in Figure 17. Initially the client inform the server that an ability is used. The Server initiate a function on the Player Pawn class that live on the server and begin by checking what class the player is. It's then followed by what spell it's casting. Then it proceed to spawn a Blueprint of the correct spell. The spawning of a new Blueprint is replicated to all clients so that the effect is visible for all. The class and spell type that was cast is stored in an Enum.

The benefit of doing it this way is that the spells stand independent of both the champion class and each other. So we can change, test and work on the spells without interfering with the functionality of other spells. It could also allow us to use the spells for other ingame events, if so desired. The table would allow us to easily change and balance the attributes for each spell.

Downside to this specific implementation is the network overhead. We could save network traffic by having the Client itself check weather or not it met the requirements

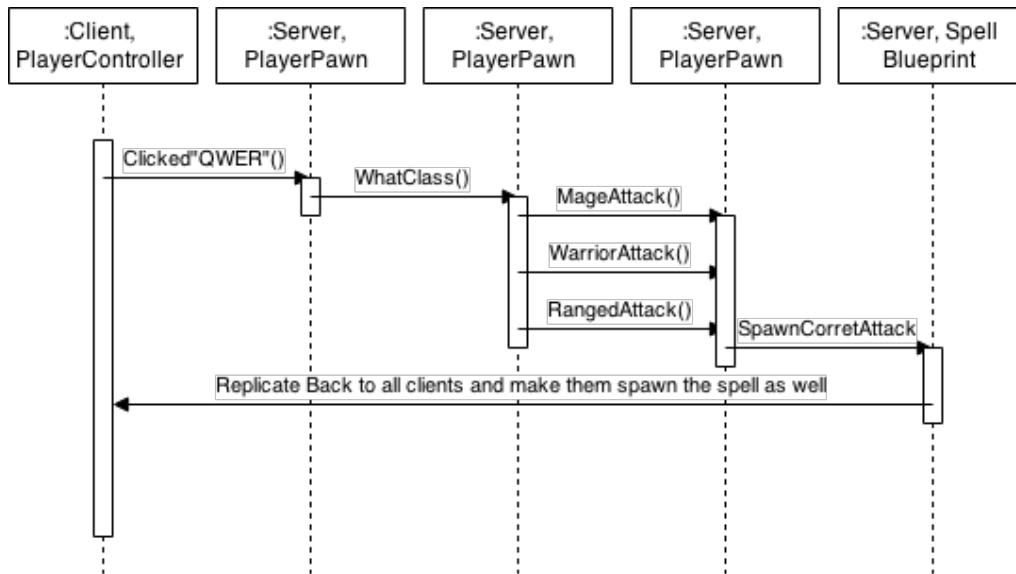


Figure 17: The program flow when casting a spell.

to cast the spell. The spell doesn't exist until the server spawn it, and at that point you've already replicated actions from client to server.

The problem here is if the client somehow bypass the verification to cast the spells. The fix in such a case would be to have redundant checks. Having the client first check if it could cast the spell, and then proceed to have the server verify the checks when the Spell spawn. This would create processing overhead rather than network overhead, and would be secure against client bypassing the verification.

The spell require basic information from the caster in order to verify it. As such, we initially ran a function to set the caster on the spell immediately after creating it. It worked but complicated the process of creating a spell. Later on we decided to use the Instigator pin when spawning the Blueprint. The spell would then get the Instigator internally and cast it to Player Pawn to access the required information.

5.2 Items

Items is an essential element in our game. Different items enhance different abilities of the player. Different item combinations will provide for different strengths and weaknesses for the player, that will again result in different gameplay and play styles. An item is bought from the Shop using Gold that is acquired during the game. Gaining a gold lead would as such lead to an item advantage later in the game. Items are stored in the Player's Inventory as described in subsection 5.3.1, and is bought from the shop as described in subsection 5.3.2.

In UE4, all information about the items are hold in a data table, which you can see in Figure 18. The problem, which can be seen in the same figure, is that the image tab on all the items is set to none. For some unexplained reason the table items can not hold

an image, which is a 2DTexture. Our workaround for this problem was to read in each row from the table, and put the information into an item structure. Then we read from an array of 2DTexture and combine the first row in the table with the first 2DTexture in that array.

There is also a possibility to read the information from file, which would be the optimal solution. We chose the easier solution because of time constrains and convenience, but if we where to develop our game further we will implement it. This will also make it easier to balance the items, as mentioned in section 7.3.

The screenshot shows the Unreal Engine 4 Item Editor interface. The top part is a table listing various items. The bottom part shows the 'Row Editor' for the 'Health_Potion' item, with fields for Title, Price, InfoText, Consumable, AttributeChange, and AttributeTarget.

Name	Image	Title	Price	InfoText	Consumable	AttributeChange	AttributeTarget
Health_Potion	None	Health Potion	20	Restores 10 health points	True	10.000000	NewEnumerator0
Power Stone	None	Power Stone	25	Restores 10 power points	True	10.000000	NewEnumerator1
Helmet of Life	None	Helmet of Life	50	This is your basic helmet. It helps protect a	False	25.000000	NewEnumerator2
Power Ring	None	Power Ring	50	This ring is imbued with magical powers. It	False	25.000000	NewEnumerator3
Fit of Fury	None	Fit of Fury	50	Hit harder with these awesome gloves. It's	False	25.000000	NewEnumerator4
Dragon Bone Armor	None	Dragon Bone Armor	50	This armor is made by the bones of fallen d	False	0.250000	NewEnumerator5
Axe of the Wolf	None	Axe of the Wolf	250	The wolf clan is a horde of barbarians. This	False	75.000000	NewEnumerator6
Staff of Wisdom	None	Staff of Wisdom	250	This staff once belonged to an old wizard. It	False	0.750000	NewEnumerator7
Amulet of Eternal Life	None	Amulet of Eternal Life	1000	The red stone in this amulet is said to have	False	0.250000	NewEnumerator8
Amulet of Immense Power	None	Amulet of Immense Power	1000	The blue stone in this amulet was believed	False	0.250000	NewEnumerator9
Evil Eye	None	Evil Eye	1000	Because everyone knows that it's a curse,	False	0.250000	NewEnumerator10
Automatic Crossbow	None	Automatic Crossbow	1000	A crossbow that reloads itself, meaning you	False	0.250000	NewEnumerator11

The 'Row Editor' for 'Health_Potion' shows the following values:

- Title: Health Potion
- Price: 20
- InfoText: Restores 10 health points
- Consumable:
- AttributeChange: 10.0
- AttributeTarget: Health

Figure 18: This is a view of the table in UE4 with all the items in our game.

5.3 GUI

GUI stands for Graphical User Interface. This is everything the player can interact with, or information that the player can find useful during game play. All GUI elements are made using widgets, widgets are described briefly in subsection 3.2.8.

Figure 19 shows the layout of our ingame GUI elements.

1. Player's experience bar. It represents progress to the next level.
2. Here is an image of the players skin. (this is currently not implemented. Read more about skins in section 8.2).
3. This display general information about the players champion. Including the champions class, element, attack speed(AS) and attack damage(AD).
4. Part of the players inventory. See subsection 5.3.1.
5. Amount of gold the player has.
6. This is the players main inventory. See subsection 5.3.1.
7. The four spells the player can use. See section 5.1.
8. Health and power bar.
9. The mini-map. See subsection 5.3.3.



Figure 19: Explanation of the different GUI elements.

5.3.1 Inventory

The inventory is a reoccurring game element in many games. This is where the player places items bought from the shop. Most other MOBAs, like DotA2, HoN and LoL, have six inventory slots to place all kinds of items. We do it a little bit differently. In addition to the six main inventory slots, we have three separate inventory slots just for consumables. Consumables are items that are one-time use, such as the health potion. These consumables are stackable, with a max stack count of 10 to balance it.

Meanwhile, the six main inventory slots work the same way as in the other games. An item affects certain attributes as long as you possess the item. We do not have any items that can be used other than the consumables. The reason for splitting the consumable from non-consumable items, is to maintain the possibility to heal yourself and regain power later in the game. As the game progresses, the player will have to buy items to increase health, damage, *etc.* Thereby occupying the six main inventory slots. If we used the same inventory models as other MOBAs, there would be no room for consumables and the player would have to rely on the health they currently have. We want to let the players use consumables later during gameplay instead.

5.3.2 Shop

The shop is where players buy all the items. Figure 20 shows the layout of the shop. On the left side is a list of all the items available for purchase. When the player clicks on an item, the right part of the screen will show information about that item. The information includes the name of the item, the cost and what attribute it affects. Down in the bottom left is the player's inventory, where they can see what items they have bought. The green highlight indicates the selected item in the inventory. To sell an item, the player must select that item and click sell, at which point the player will get a portion of the item's price back. The shop is only available at the player's start position, indicated by the stone circle shown in Figure 20.



Figure 20: An example of the shop.

5.3.3 Minimap

Due to some issues with Unreal Engine 4, we couldn't set a orthographic camera as the camera for the Minimap. Instead it's a regular camera with low Field of View placed up above the arena. Because of the size in our world environment, the Minimap can't show the entire map with details such as player position. We could have the Minimap show a smaller area following just around the player, but this was not implemented.

5.4 Blueprint Structure

This section describes some of our blueprints in more detail. A Blueprint can require other assets to work properly, depending on the heritage of the Blueprint class.

5.4.1 Pawns and Character

The Player Pawn and Character blueprint consist of several other components in order to get animation, collision and looks to work properly.

Physic Asset - Unreal Engine 4 automatically create a Physics Asset when importing a mesh from Blender. This is based upon the structure of the model armature. This process is described in the subsection [4.2.2. Physic Asset](#) in itself is described in subsection [5.6.1](#).

Skeleton - The basic skeleton is based on a simplified version of the human bone structure. Mainly one bone per limb that is to be transformed. Each bone in the skeleton can rotate, scale and move. Animations is made by applying transform operations on different bones in our model. The model mesh is mapped to the bones by weight painting, which is done in Blender or other 3D modeling software.

Skeletal Mesh - Is often used in Unreal Engine 4 to represent a moveable character or pawn that can be animated through it's bone structure. A skeletal mesh has a group of polygons that are weight painted upon our skeleton. The model should follow some

guideline for effective use. Such as not having any lose polygons and normals being correctly aligned.

Texture - The Texture is essential just an image that is mapped to the UV layout. An UV layout is the mapping between the surface area of model and the texture. Because the UV layout is mapped to the direction of the normals, it is important to have them correctly aligned. You can blend different textures, UV layouts, *etc.* into a Material. This is mostly used to combine textures with bump maps, normal maps and exclusion maps. The material work as a texture that give an enhanced depth field and emulate higher resolution of the model. High resolution textures demand more memory and processing on the GPU, so it's not without drawbacks.

Anim Sequence - The Anim Sequence (short for Animation Sequence) is a collection of frames in which one or more bones in the skeleton is transformed from the basic pose. The collection of these frames chained correctly make up the animation. The Anim Sequence is used inside the Animation Blueprint for animating a character.

5.4.2 Champion and Proxy Pawn

Champion is a playable pawn which interact within the game through a player controller. Due to problems with networking as mentioned in subsection 5.8.3, we had to create a Proxy Pawn. The Proxy Pawn is an invisible Pawn that has a camera and functions associated with it. The player control the proxy, which replicates data to the server. The server spawn a Champion that mirror the Proxy Pawn, and is updated along with the Proxy Pawn.

The Proxy Pawn have a Player Controller that only exist on the client, which run events triggered by user interaction. The actual Champion Blueprint is controlled by an AI Controller. The AI Controller again follow instructions from the Player Controller. We are doing a linear interpolation of 50% between the Champion and Proxy Pawn positions. This would be changed based on latency between client and server.

Given the network overhead caused by moving the champion, we set the replication to update 60 times each second maximum. This is due to the framerate of the game being capped at 60 frames per second. It also have highest priority to avoid making the game feel unresponsive.

5.4.3 BaseMinions

BaseMinions is the superclass for the JungleCreep and Minion_FootSoldier Blueprints, with Pawn as it's superclass. Illustrated in Figure 21. It is defined as a non-playable pawn which is possessed by an AI Controller. The AI Controller gives instruction to this pawn. The BaseMinion class have functions that are mostly empty, but get overridden in the subclasses. Using a superclass prevents us from having to copy and paste code and functionality for each subclass, and changes in the superclass affect the subclasses accordingly.

Initially we didn't use a class hierarchy because of how Blueprint was a new language for us. But when it came to implementing the JungleCreep Blueprint, we found it could utilize most of the same functions as the current Minion_FootSoldier. We reworked that

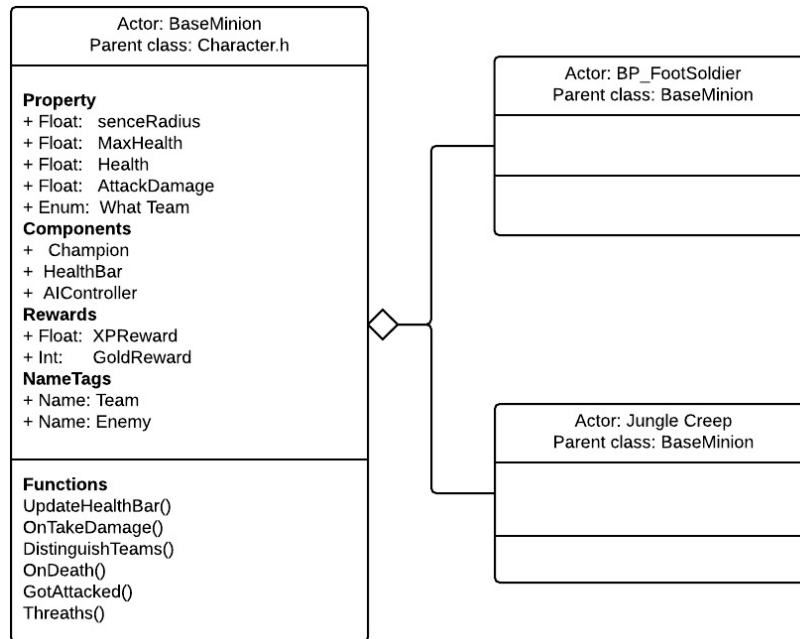


Figure 21: UML of the relation between Base Minion and subclasses

class, at the time called Minion, to a superclass named BaseMinion.

Using the superclass we could easily implement the JungleCreep Blueprint, not only because we could reuse code, but because the rest of the project could treat the JungleCreep as it's superclass. We didn't have to specify to check for JungleCreeps or Minion_FootSoldier, we could only check again BaseMinion and it would be handled appropriately.

Figure 22 illustrates how to create a subclass of an existing class.

5.4.4 Minion_FootSoldier

Minion_FootSoldier is the basic units of each team. The Minion_FootSoldier spawn in groups around the Main Hall and move towards the appointed lane, attacking enemy units and towers along the lane towards the enemy Main Hall.

The player should not have problem facing a group of minions. They are both weaker and slower than champions. They have also been given a smaller size to help the player understand that the player is superior. Minion_FootSoldier have sword and shield and can only attack when within melee range which is set at 200 units.

Figure 23 shows an image of the skeleton mesh of Minion_FootSoldier. The model contain fewer polygons than the Champion mesh to improve performance.

Minion_FootSoldier helps create momentum towards destroying buildings and advance on the battlefield. Since Towers will target them first, they will soak a lot of damage, which would otherwise been done to the player. The Minion_FootSoldier have an AI Controller who possesses him and give him instruction on what he should do and behave based upon a behavior tree 26. Minion_FootSoldier is set up by a hierarchy of

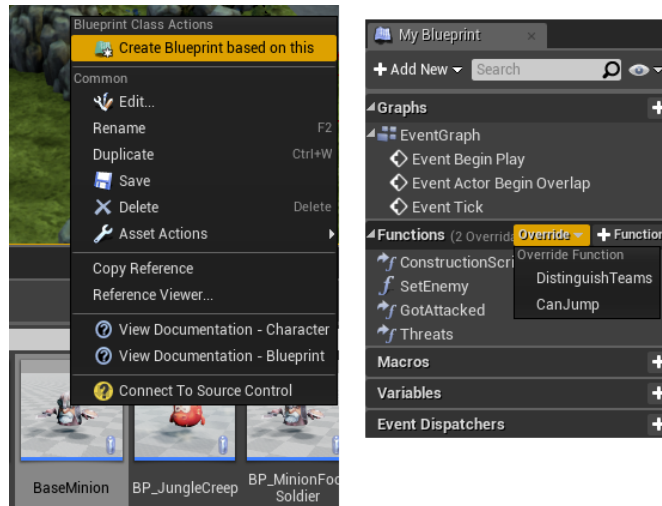


Figure 22: Create subclass based on existing Blueprint Class

Blueprints, as demonstrated in Figure 24.

1. **BP_MinionFootSoldier** The blueprint. Parent class.
2. **BP_BaseMinion** Parent of BP_MinionFootSoldier.
3. **minionAIController** Executes AI instruction, and controls the BP_MinionFootSoldier.
4. **MinionFootSoldier_Skeleton** Makes the BP_MinionFootSoldier the able to have motion and be animated.
5. **MinionFootSoldier_PhysicsAsset** Rigid body of the model.
6. **BP_MinionFootSoldier_SKMesh** Skeleton mesh with 663 Triangles and 612 vertices.
7. **MinionFootSoldier_Texture** With one UV Channel and one alpha channel.
8. **MinionFootSoldier_Anim** Animation sequence where he lifts his sword and shield

In subsection 5.4.1 we give an syllabus of assets general used with character and pawns.

5.5 AI

The basic for a good AI is that it maximize user experience at as little computation cost as possible. We would also try to find an approach that is simple, easy to reuse, possible to scale, great to debug, and fulfill the requirement of the game mechanics of a standard MOBA. A Finite State Machine was an option, but we wanted a structure that was more responsive to the environment and was easy to reuse and scale.

In a paper written by Christopher Dragert, Jorg Kienzle, Clark Verbrugge they describe some key concept of reusable AI as:

“Players expect non-player characters (NPCs) to intelligently react to player actions while exhibiting appropriate behaviors depending on their role within the game context.”



Figure 23: Image of Minion_FootSoldierMesh

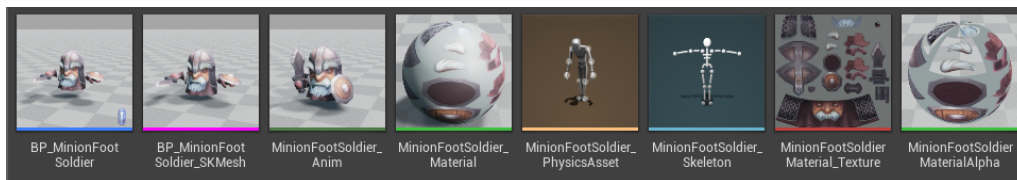


Figure 24: Unreal Engine Components that make up the Foot Soldier

“This type of AI is referred to as computational behaviour, distinguishing it from classical AI approaches. In the context of game development, efficiency and testability are paramount, strongly constraining design approaches. The easiest approach is to employ arbitrary code expressed through a custom scripting context or a relatively simple tree or graph structure, such as a decision tree.” [4]

Based on this text we starting moving towards Behavior Tree. Behavior Tree is simple, easy to debug and scalable. The Behavior Tree is described in further detail in subsection 5.5.5. Since we try to mirror the behavior of an medieval soldier we believe that the player will feel a connection towards the decision the Minion will take when faced with choices. We tried to avoid complex behaviors by themselves, to make it easy to debug and reuse the behaviors.

5.5.1 BaseMinions

The base minion contain the most basic AI, and will attack enemies if one of these conditions are fulfilled:

1. They were attacked by someone. Attacks the closest target.
2. They have vision of an enemy.
3. They have valid paths towards target.

5.5.2 Minion_FootSoldier

Minion_FootSoldier inherit the basic AI from the BaseMinion. The MinionFootSoldier AI is based upon moving from on point to another and kill enemy's structures or units on the way. The AI will priority's the structures before units and units before champions. Even if this is the priority's queue, it is ways for the minion to change targets or priority based upon Algorithm 1.

```

Data: Damage instigator(Attacker)
Result: Set current target
if Target equals none;
then
  if Attacker  $\in$  ActorsInVicinity;
  then
    Line collision check in eye height from yourself to attacker;
    if Attacker match blocked hit;
    then
      | Move towards target and attack
    end
  else
    | Reset HP
  end
end

```

Algorithm 1: Pseudocode for GotAttacked

The above pseudocode is triggered by the Unreal Engine call "Apply Damage" which contain several variables such as damage, type and instigator. The algorithm rely only on the instigator, as that is the enemy that attacked.

5.5.3 Jungle Creeps

JungleCreeps also have the extended AI of the superclass BaseMinion, but have it's own AI Controller. They are not fully implemented yet, as we planned to have some of them roaming around. Outside of roaming, they would only be standing still at a given spot, and respond to being attacked similarly to the Minion_FootSoldier.

5.5.4 Pathfinding

In a perfect world we could have used Floyd-Warshall ¹ algorithm and precomputed every path in the search space, and then put that data into a look-up table. This algorithm is definitely the fastest way to generate path at runtime, much faster than any A* implementation.[5] But our search space is to big and it would have required a lot of memory allocation.

As such, we needed an different solution. There are currently three primary search spaces available that are of interest.

1. **Grid Search Space**
2. **Waypoint Graph**
3. **Navigation Mesh**

¹http://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm - Roy-Floyd-Warshall algorithm

Grid Search Space is dividing the current map into equal sized tiles, the number of nodes (n) is equal to $\text{mapSize}/\text{tileSize}$. A map that is 2×2 with a tile-size of 1×1 will generate 4 nodes. This creates the most overhead in terms of memory.

Waypoint Graph uses path from node to node to describe where the AI can move, if he want to move from point A to point B he needs to follow a certain path, and this behavior can create a zigzag effect. The player would not find it likely that a human person would do this in the real world. The number of nodes (n) is equal to the number of path point needed to define a graph that is suitable for the AI to reach his objectives. This has the potential to be the least number of nodes since you specify which path the AI should take from where he is to the place he will go.

Navigation Mesh uses walkable areas with non-overlapping polygons [6] as shown in Figure 25. This is a very logical way to think about path finding since we as humans does not generate multiple way points in our head an manually walk towards them in a fixed manner and when we reach the point we turn towards next point before starting to walk in a straight line. Instead we set a goal and look for opportunities to walk the fastest way to that point as long as nothing blocking our path.

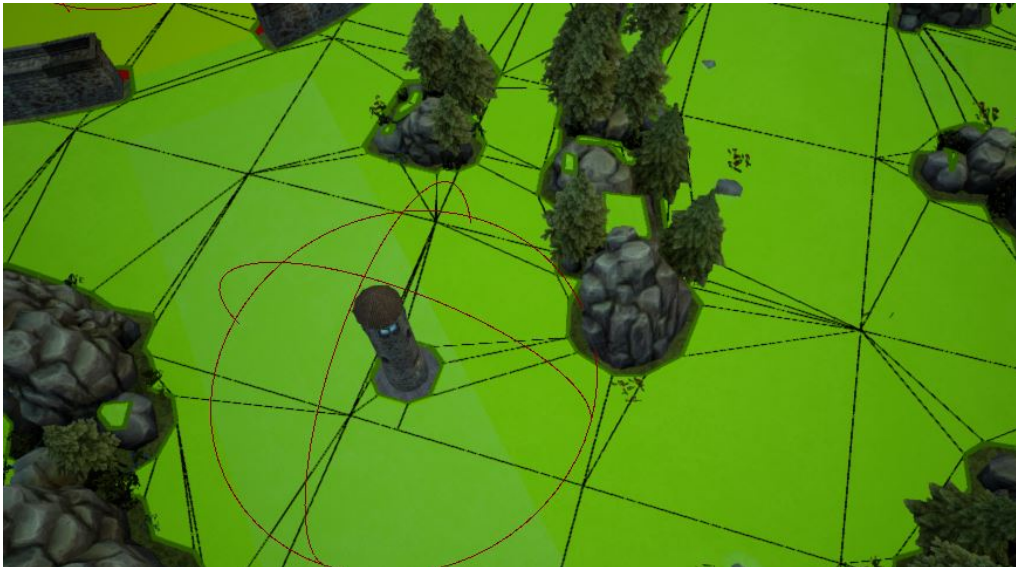


Figure 25: Navmesh Polygon graph.

Since Unreal Engine 4 already implements a Navigation Mesh, we did not need to implement our own. We customized the settings to our needs. Such as to avoid floating islands of navigation space, valid navigation space inside buildings or minor ground terrain counting as unavailable.

Waypoint is an Actor Blueprint that only has a location and a sprite (that shows in the editor but not on runtime). We use Waypoints to specify a point on the map we want the the AI to be able to move to. Since Waypoint is a Blueprint we can add variables to it. Later we renamed the blueprint to Checkpoint. We decided to use int to specify the position of the checkpoint on the map. The checkpoints are spawned on towers and

other structures when the game starts. The code trigger in the Level Blueprint. The AI Controller then uses the Navmesh to find the best path to the next Checkpoint.

5.5.5 Behavior Tree

A Behavior Tree is a hierarchy of data structures starting at a root node and branches down to leaf nodes of behaviors. A behavior is a predefined set of instruction that the AI will execute before moving on to the next node. The Behavior Tree limit the execution of behavior with preconditions that have to be met.

A precondition specify if the Behavior Tree should run or exclude a section of the tree. As an example see Algorithm 2.

```

if Enemy->isSet();
then
  | Add node to execution stack, run child nodes preconditions.
end
else
  | Check the precondition of next sibling.
end

```

Algorithm 2: Pseudocode for Behavior Tree

When the Behavior Tree has reached the end of the tree, it will chose the highest-priority behaviors from the stack to run.

Advantages of using Behavior Tree:

- “Doesn’t need to remember what behaviors were previously running in order to determine what behaviors should execute on a given frame”.^[5]
- Behaviors from different branches in the Behavior tree are independent from each other.
- Easy to add extra functionality because every node is separated.
- Its easy to debug, particularly so when using a graphical editor such as Unreal Engine 4.

5.5.6 Behavior Tree in Unreal Engine 4

Behavior tree in Unreal Engine 4 differ in some regards from what was described above. Epic Games say “*There are three critical ways in which the UE4 implementation of Behavior Trees differs from "standard" behavior trees*”^[7]. What they mean by "standard" is a bit hard to discern because there are no general or specific standard Behavior Tree. Based on further reading, we can however tell what they’ve done as improvements to their implementation.

Advantages of Unreal Engine 4 Behavior Tree^[7]

- *Event-driven behavior trees avoid doing lots of work every frame. Instead of constantly checking whether any relevant change has occurred, the behavior trees just passively listen for "events" which can trigger changes in the tree.*
- *Performance Improvements - Since the code does not have to iterate through the entire tree every tick, performance is much better! Conceptually, instead of constantly asking*

"Are we there yet"?, we can just rest until we are prodded and told "We are there!"

- Better debugging, because you can step through the Behavior tree's execution list.

Since we are using Behavior-Tree-Task-Node(BTT) we could easily reuse parts of the tree in different behavior tree structures. BTT uses Blackboard KeySelector to reference data from the shared blackboard. Those blackboard gives us relevant information gathered and shared by multiple AI structures. Having behavior made in advance make it easier to understand what kind of data we need to share in our blackboard. Re-usable BBT helps us develop part of our AI only one time, and share its behavior.

In Unreal Engine, a complete Behavior Tree is built up of the following components:

- **Blackboards** - Are a data structure that stores the data of an AI Controller and are referenced by the behavior tree. It has a list of keys. Each key represent a data structure inside Unreal Engine.
- **Behavior tree** - Is where we will define our nodes and set conditions, this is our graphical representation of the Behavior Tree.
- **Services** - "Are special nodes associated with any composite node (Selector, Sequence, or Simple Parallel), which can register for callbacks every X seconds and perform updates of various sorts that need to occur periodically." [7]
- **Selectors** - Is an branch type of the Behavior Tree. It lets the children node execute each of its child behaviors in order until it finds a child node that successfully return.
- **Sequence** - Executes each of their children until all of them return succeed. This can chain together different behaviors to act like a bigger behavior.
- **The Behavior-Tree-Task-Nodes** - This is where our actual code lies and is always a child of a sequence or a selector.
- **Decorator** - Function that queries the blackboard for values as an observer. This works as a precondition only that it does not check values at tick, but wait for events.

MinionBehaviorTree as referenced in Figure 26

1. **minionBlackboard (Blackboard)**: The shared data between the AI structures. Data keys is shown in point 10.
2. **Selector with SearchPossibleTargets (Service)**: Will execute function Choose best target(Algorithm 4) on the AIController every 0.90 second to 1.10 second.
3. **When target is set (Decorator)** Event based condition. Will execute children if target is set.
4. **Setting first checkpoint (Decorator)** event based condition. Will execute its children when tree starts. Because point 3 has no target and Setting first checkpoint is the next in priority queue and target location is not set.
5. **When there is no target (Decorator)** Event based condition. Will execute its children if none before has valid conditions. The When there is no target is set is redundant, but it makes it more readable, re-usable and easier to debug.
6. **BTT_MoveToEnemy** Tells the AI controller to move the pawn towards the EnemyTarget in the blackboard.

7. **BTT_AttackEnemy** Execute Apply Damage as described in sub-chapter 3.2.3 to the EnemyTarget, if the target is within AttackRange.
8. **BTT_MinionNextCheckpoint** Sets the first target location to the first Checkpoint, gets AttackDamage from the minion property and set it to Attack Damage in the blackboard.
9. **BTT_MoveToLocation** Tells the AI controller to move to the location in TargetLocation in the blackboard. When the unit reaches it's TargetLocation or within 200 units, a success message will then update the checkpoints, se Algorithm 3
10. **Blackboard keys** in the minionBlackboard. These keys can only be a part of UE4 variables.

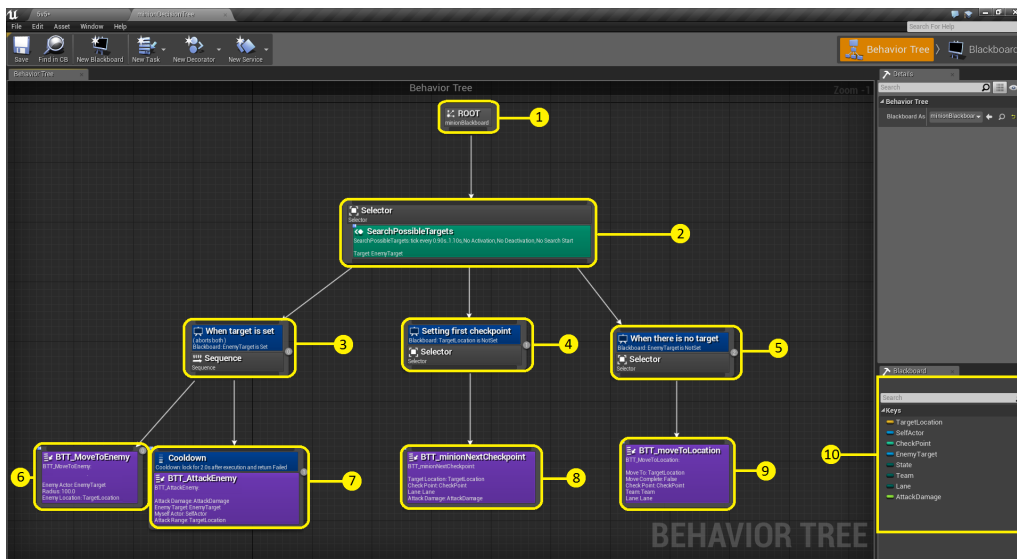


Figure 26: minionBehaviorTree

BBT pseudocode described in more detail

Data: Team

Result: Set next checkpoint

if Team is blue **then**

 | Increment checkpoint;

else

 | Decrement checkpoint;

end

Algorithm 3: Pseudocode for MoveToLocation->UpdateCheckpoint

We have three Behavior Trees in our project. The minionDecisionTree, JungleCreep-DecisionTree and ChampionAIDecisionTree. The minionDecisionTree is illustrated in Figure 26.

5.5.7 AI controller

AI Controllers are non-physical Actors that can possess a Pawn. The difference between Player Controller and AI Controller is that the AI Controller is more focused on responding to input from the environment and game world. We want the AI Controller to handle and gather data about the environment and game world, and react accordingly to that information. Such as updating variables in the Behavior Tree. The AI Controller connects the AI functionality and controls the possessed Pawn in return. As illustrated in Figure 27.

Advantages of using AI Controllers

- Updates the Controller before the possessed Pawn to minimize latency between input processing and Pawn movement.
- Programmers and designers can work on two different objects without causing conflicts.
- Easy to use on multiple objects.
- Make it more viable to use subclasses, because functionality in the AI Controller does not get affected by the possessed pawn's subclass.

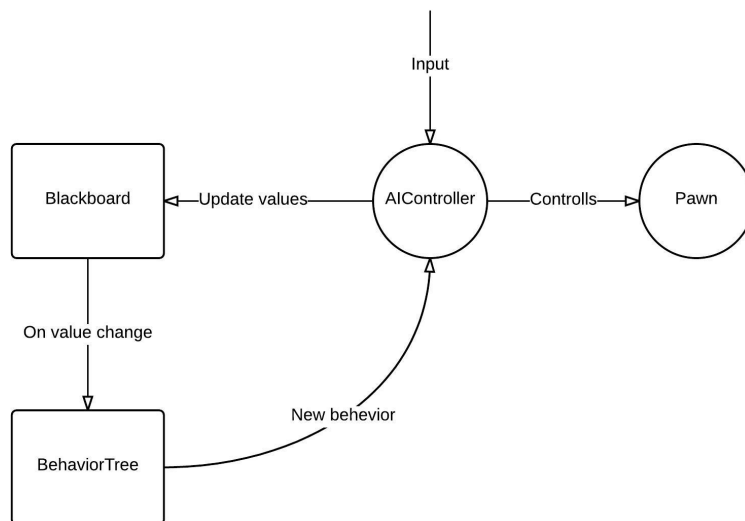


Figure 27: Sequence of events in the AI Controller.

5.5.8 Minion AI Controller

This is essentially the Minion Foot Soldier's brain. Without the AI Controller, the Minion Foot Soldier blueprint would only be a rigid body doll solely interacting with the physics of the game. This AI Controller work closely with its Behavior Tree of the Minion Foot Soldier as illustrated in Figure 28. It is mainly getting a function call at a given interval. The function is to choose the best target to follow and attack. See the Algorithm 4.

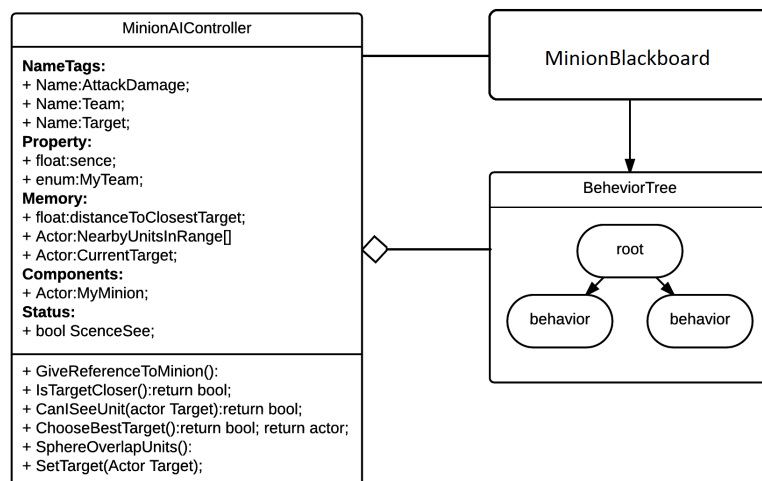


Figure 28: UML diagram of the Minion AI Controller.

5.5.9 Tower Targeting

The towers in MOBAs are very similar to the towers in Tower Defense games when it comes to the AI. They have very simplified AI where they have a priority queue of actors to shoot at. The queue is filled up by the "onComponentBeginOverlap" event and emptied by the "onComponentEndOverlap" event or death of actor. When an actor triggers the overlap it will be added to an array of actors, where minions are the first to be targeted. If no minions are nearby, or an enemy Champion has attacked another champion in vicinity of the tower, the tower will shoot at the champion.

This would be achieved by event dispatcher on Champions when they are attacked, that inform all friendly towers to prioritize the attacking Champion. If the tower is within range to fire at the enemy champion, it will proceed to do so.

Pseudocode for adding target to the queue of targets is found in Algorithm 5. Pseudocode for attacking the target is found in Algorithm 6

5.5.10 Conclusion

As described in Game AI pro: "We succeed any time that the user thinks about and responds to the AI as if it were real, even if the underlying algorithms is quite simple"[5]. Several of the AI in the game behave simply and cause users to respond appropriately. Others are more complex and require a bit more testing and polishing before we can claim to have succeeded with them.

5.6 Physics

This section describe physics in Unreal Engine, especially collision and how we added a custom trace channel for simple collision of clickable objects.


```

Data: Enemy units
Result: Set current target
Search for enemy units in a given volume;
if Units found;
then
    for unit ∈ array do
        Line collision check in eye height from yourself to potential target;
        if Current unit match blocked hit;
        then
            if Current unit is closer than previews current target;
            then
                Set current unit to current target;
                break;
            end
        end
    end
end

```

Algorithm 4: Pseudocode for ChooseBestTarget

```

Data: Enemy unit on overlap
Result: Add unit to Targets
if Unit is valid then
    Add unit to Targets;
end

```

Algorithm 5: Pseudocode for Begin overlap

Unreal Engine uses PhysX for physic calculation. PhysX is an physic engine that has been in development since 2004 by a firm called Ageia. In 2008 Nvidia acquired PhysX in a merger and it's now known as Nvidia PhysX. Nvidia implemented PhysX to run on top of CUDA architecture enabled GPUs for hardware acceleration and are exclusive to Nvidia GPU's [8].

5.6.1 Collision

Collision Responses and Trace Responses form the basis for how Unreal Engine 4 handles collision and ray casting during runtime. Upon creating an object in Unreal Engine 4, we have to specify which object type it belongs to. This simplifies collision later on as it automatically belong to a predefined collision group. We can change that in the collision tab.

Simple Collision vs Complex Collision - Simple Collision uses simple low polygon meshes to define an area and check for collision inside its volume. Complex collision usually use per poly geometry. That means it will search trough the geometry of the model and build its volume from every polygon instead of a simple geometry shape, cylinder, sphere or a box. The fewer polygons, the faster the collision detection. If it's not crucial for the game to know that it hit inside the geometry of your original mesh, then simple collision is the way to go.

- **Example of Complex Collission** - In most shooter games, it is not acceptable to shoot next to the body and count as a hit. You require more detailed collision.

```

Data: Targets
Result: Fire at Targets
while Unit  $\in$  Targets do
  | if recharging then
  | | do nothing
  | else
  | | if Unit is valid then
  | | | Fire at first target in Targets;
  | | | delay(3 seconds);
  | | end
  | end
end

```

Algorithm 6: Pseudocode for fire at targets

And often more so when the different regions of the mesh have different damage ratings.

- **Example of Simple Collision** - In the event that the user should select something. We don't need a complex collision, as worst case the user will click between the mesh and miss. As such we wrap the model in a basic geometric shape, or even a simplified mesh of the model, that the user can click.

Custom Object Channels and Trace Channels - Unreal Engine 4 comes with 6 Object Response Channels and 2 Trace Response Channels by default. We desired a custom channel to determine if an object is clickable or not. One could simply use the visibility with the camera, but then the user could select any visible actor (and the environment contain roughly four thousand of those). As such we created our own channel as illustrated in Figure 29.

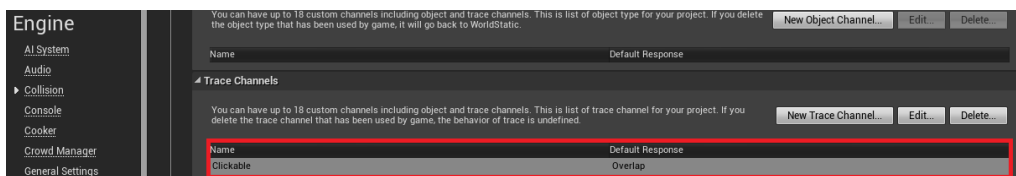


Figure 29: Creating of new Trace Channel

Creating a clickable object with simple collision - Here is a short walkthrough on the process of making an object clickable and wrapping it in simple collision rather than a complex collision. We start by making a colliding cube (or other simple shape) around the mesh as illustrated in Figure 30. Then we proceed to specify what channels it should collide with. As illustrated in Figure 31, we ignore everything but the custom Clickable Channel.

5.7 User Data

We intended to have a user profile for the different users. This profile would contain information such as friends, any bought items, history of matches and stats of said matches. This would all be stored in a database on a remote server and would be accessed through

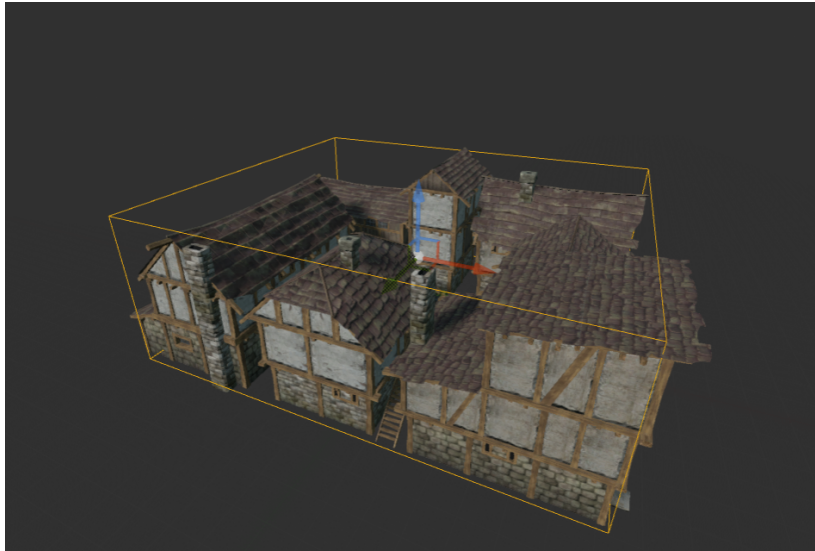


Figure 30: Simple Collision box

web requests to PHP pages. As Such a basic database was put up early on, containing only Username, Password and Email along with an unique identifier. Due to complications and priority elsewhere, it was never expanded upon.

Having this database accessible through PHP calls, we were able to authenticate users from within the game. We could also easily expand and make website that utilize the PHP pages to authenticate the user. The site would be a more detailed summary of the user profile, and have extended functionality compared to the game itself. For instance, changing the password would be handled through the website and not in the game.

5.7.1 Security

Limitations of outdated PHP on the school's webserver meant we couldn't make use of certain security features in more recent versions of PHP. Some of which included password-hashing and hash-verifying. Because we didn't have access to those, we decided on a more generic hash function and string compare. The Password is hashed using the Username as quick and easy salt.

We would ideally have wanted to hash the data before even sending it to the PHP server, but that were not a priority since our focus in the bachelor project was gameplay and not security. Everything is sent to the server using HTTPS to encrypt the content and POST rather than GET to prevent it from being cached, remain in browser history and avoid bookmarking the page. As such, we considered it safe enough to not warrant any more attention until after more important and pressing features were completed.

5.7.2 Blueprint Solution

We installed and configured a plugin called VaRest² in Unreal Engine 4. The plugin exposes functionality that allow one to make RESTful API calls to a remote host. Using the

²<https://github.com/ufna/VaRest/wiki> - Git repository for VaRest

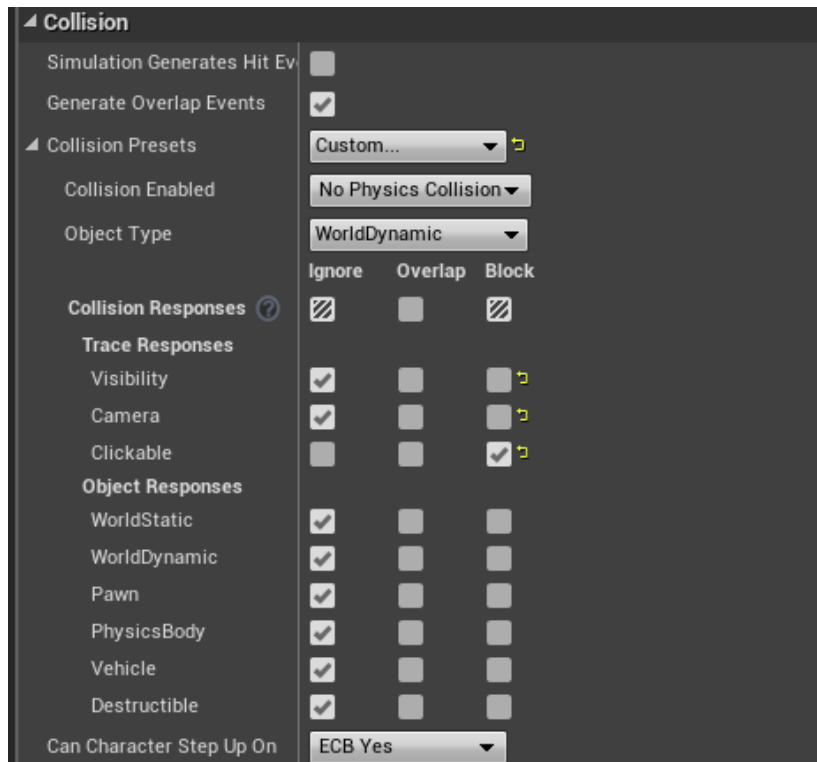


Figure 31: Collision options for objects

plugin, we build a JSON file which we then send to the webserver. Using event dispatcher in Blueprints, which is quite similar to function pointers, we run a specific function when we get a response from the server.

One notable problem with the usage of plugins to expose functionality to Blueprints in a Blueprint only project, is that the project require some code in order to include the plugin. This is currently a bug in Unreal Engine 4, and a very simple workaround is to add an empty C++ class to the project.

5.8 Ingame Networking

Ingame networking is the topic of how different events in the game is to be mirrored over the network. This is sent over the network to other clients by Unreal Engine 4 automatically once a connection is established. The act of mirroring an event between the users is called Replication.

On every actor that rely on a Controller you can specify some options to optimize replication overhead, and secure that only intended information is sent over the network.

- **Net Cull Distance Squared** - This specifies the max distance from the client's view-point that this actor is relevant and will be replicated. That means that objects that are far away from the camera will not be replicated. This also helps prevent siphoning unwanted information.
- **Net Update Frequency** - Maximum update frequency that the Blueprint is consid-

ered for replication.

- **Net priority** - Set the priority the Blueprint has when considered for replication. The highest number has the highest priority.

5.8.1 Replicated Variables

We can make variables replicate between users automatically. There is the option to set it to "replicates" in the editor, as well as "RepNotify". The "None" option don't replicate the variable automatically at all.

- **Replicated** - Means when the variable is changed it will automatically change the same variable on all the connected clients. Getter would get updated value, setter would override the current value.
- **RepNotify** - Will create a function that is called when the variable is updated. This can be used to optimize code as you won't have to constantly ask a variable for its value. Instead, the function is called when "Current Health" is updated, and update the values in the healthbar accordingly.

5.8.2 Replicated Functions

Unreal Engine provide us with built in functionality for replicating entire functions. This is done by creating a custom Event and deciding on how the event should be replicated. The options work a bit differently and serve different purposes.

- **Run on Server** - Is the function type that is only run on server, regardless of who calls upon the function. This is useful for things that we require the server to handle. The act of damaging enemies is run on server. This is to make sure the client don't just add crazy amounts of damage to his attacks and kill everything.
- **Multicast** - Is often called after **Run on Server**. This type of event will first run on the server, and then ask all the clients to run the same code. For instance, destroying an actor that has died.
- **Run on Owning Client** - This function type is called when only a single Client should get updated with information, but the call comes from the server. For instance a death message would be authenticated on the server, but only the dying player should get the function call.

By default they are not reliable, but important functions can be set to reliable in the editor. This guarantees the function will run on the client. Minor things, such as position that get updated constantly, don't need the overhead of being reliable because of the constant updates.

It's important to note that some Replicated Events might call upon a specific function that is of importance, or you only want either the Clients or the Server to run that particular function. In such a case you can guard the function at its beginning by an Authority check. The Authority Check work as a switch between Client and Server, gathering the data from the date on the game. Since this is built into Unreal Engine 4, it is a reliable security feature to make sure only the server and client run certain aspects of the code.

5.8.3 Problems

One major problem is that some of these events and variables aren't immediately intuitive, and some require chaining to work. When spawning a new object, you need the server to be the authority, so you call a "Run on Server" event. But at the same time, you want all clients to be aware of this new object, so the "Run on Server" event immediately call a "Multicast" event. This might seem slightly redundant and complicated instead of having a "Multicast from Server" kind of function call.

Another problem we encountered was movement. While there is initially a setting called "Replicate Movement" for objects, it required us to use a Movement Input to control the character. Movement Input would be similarly to adding force in physics. You add movement input forward and it will move forward. Which is a common way of doing movement with joysticks or key presses to trigger the movement in different directions and rotations. However, we were using navigation where the character would get a destination and navigate there using the Navigation Mesh generated by Unreal Engine 4. The Solution was to make a Proxy Pawn between the player controller and the champion they moved around.

5.9 Buildings

This section contain basic information of the buildings as seen in the game.

5.9.1 Tower

The Tower is the most essential for gameplay so far. Towers will damage any enemy unit that come within the range of the tower, and prioritize Minions before Champions. The Algorithm mentioned in subsection 5.5.9 is triggered by a collision with an overlapping sphere around the tower, adding them to a list of current targets. It then removes them from this list when they leave the collision sphere. When units die and get removed from the game, they count as leaving the collision, and as such is removed from the list of targets. The tower have a short delay between shooting targets.

5.9.2 Main Hall

The main objective of the game is to destroy the enemy's Main Hall. As such, this building is a grand structure protected by towers. The Blueprint of the object is quite simple. It have some simple code for taking damage and updating health, and upon reaching 0 health the game will end. We implemented a clickable collision box around the building to avoid a complex trace when the user would click on the building to attack it. As mentioned in the subsection 6.1.2 about deployment errors, there was some issues with the Main Hall during packaging the project for release.

5.9.3 Barracks

So far in the project, the Barracks is of no importance. They have the basic data such as health, healthbar and will get destroyed when reaching 0 or less health. But they don't currently add any gameplay element. This is due to time limitation, as we had planned for it to have some functionality as mentioned in subsection 2.1.4, regarding Minion

Commander.

5.10 Animation

Animation in UE4 is done in an animation Blueprint. The animation Blueprint is a composition of animation sequences, montages and Blendspaces. For a pawn to have animation he would need an skeleton and skeleton mesh to create the basic of hierarchy to run the correct animation as shown in Figure 32.

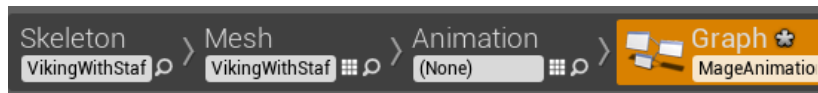


Figure 32: Animation Structure

BlendSpace - We only use BlendSpace to blend from idle-walk-run animation sequences based upon the movement component. BlendSpace is a 1D graph that takes animation sequence and blend the space between them giving them a good transitions between sequence as shown in Figure 33. When movement speed rises the graph also increases it's value.

Montages - Montages are very agile when it comes to combine animation sequence and loops them when a given event is triggered. We made two custom event, BasicAttack and Ultimate that is triggered from the champion blueprint. Inside StartUltimateAnimation it switches on what type the champion is and then gets the AnimInstance and then trigger the correct event for the ultimate as shown in Figure 34. Point 1 starts the event and point 2 start the montage. The basic attack animation is based upon the attack speed of the champion. If the attack speed increases the animation also speeds up.

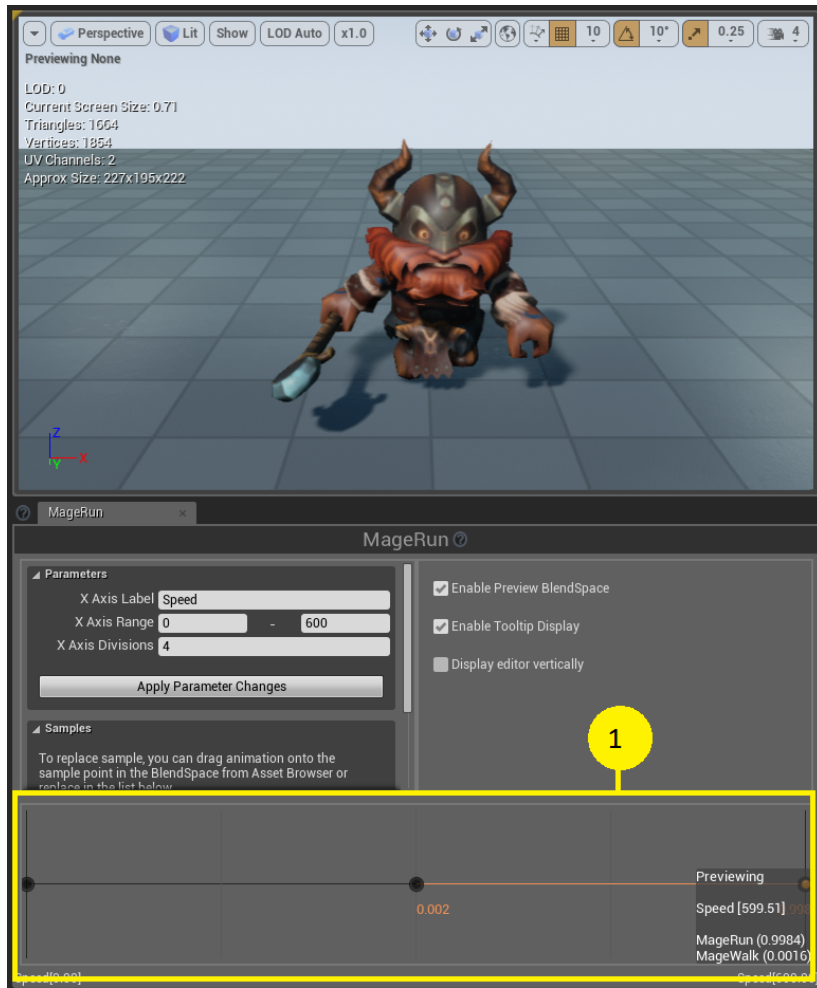


Figure 33: Blendspace

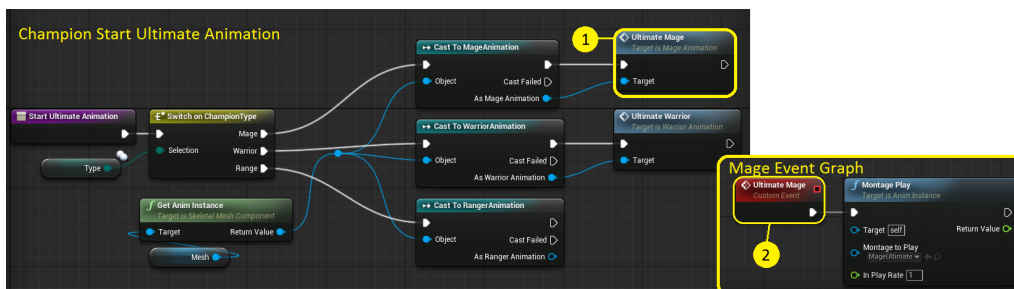


Figure 34: ChampionStartAnimation

6 Deployment

This chapter will cover the deployment process for Project NORS.

6.1 Building

Building the game is done using the option in the Unreal Engine 4 editor. It will automatically package the project and create an .exe file to launch the game. The .exe file needs to be run with the required assets in its folder and subfolders.

6.1.1 Different Machines

We tested the .exe file that's built under the Shipping release mode. It worked on machines that didn't have the Unreal Engine 4 Editor installed, and we were able to join up with them and play a game together, while on the same local network.

6.1.2 Errors

The following errors we encountered while packaging the game.

Network Errors - Initially we had an error with networking automatically failing on a packaged project. This was solved early on by simply including a few packages in a build file that the Unreal Engine 4 Editor created automatically when packaging the project. The packages were specifically the OnlineSubsystem and OnlineSubsystemNull packages that govern the default Blueprint setup for online gaming, as well as the "Null" subsystem which doesn't rely on external server or programs. Later on we changed to using Steam as the subsystem in order to get networking to work.

Building Displacement - Another error we encountered with the packaged project was that the Main Hall in each base, got moved to position 0.0.0. We tried fixing this ourselves, by checking if any of the functions that were running in the object happened to move the object by some error. In the end, we didn't find any solution and reported this to Epic Games. This was apparently something they had a case for on Jira and it would be fixed in a later release of the engine. We did not find any solution to the problem, and as such, the Main Hall was rebuilt using an existing working structure as a basis. It is possible that the issue was related to accessing the position of the building in the Level Blueprint.

Plugin Not Included - As mentioned in the subsection [5.7.2](#) under User Data, we had some issues where plugins that we used didn't get included in the project upon build. Because these were third party community plugins, Epic Games were reluctant to provide any support on the basis that it was the plugin that was faulty. Eventually it came forth that when having an all Blueprint project, community plugins don't get included in the built package. This had to do with how it's packaged and without any code files, it will do a default packaging of blueprints only. When you add code files, the editor will

automatically generate a solution that is used for packaging the project. The solution also include plugins in the list of files to be packaged.

7 Testing and User Feedback

This is the chapter about testing and user feedback.

7.1 Testing

Every game or other software that are in the making should go through a testing phase. By testing, we have a chance to let others than ourselves try and test out what we are making. This is a good thing because we get people that have never seen our product before to test it. They won't necessarily know how to do everything, and do things in a way that we didn't plan for. They can answer things such as "was the game fun?" or "was this button easy enough to hit?". This is something we need real people to do, and preferably someone different each time, to see if you have managed to fix the problems that we wanted to be tested.

When it comes to game testing, there are usually two groups that make up a testing session, the testers and the instructors. [9] The testers are there to play/test the things that the developer wants to test. And the instructors are there to make sure that the testers act within the boundaries of the test. To get the most authentic test the instructor will silently sit and watch the tester do his work while taking notes for themselves.

7.1.1 Testing with kids

On May 5th we had some kids testing our game. It was an interesting experience to see someone beside ourselves play the game.

In the beginning of the play testing session we had some problems getting the games to connect to each other. As the game is, the only way to connect and find each other works over LAN. Even if we were connected to the same SSID, it was not necessary the same access point. Because the school (GUC) have many overlapping access points. This was quickly solved by setting up a new router as an access point that we could connect to.

When that was sorted out, we could begin the play testing. There was only one of the kids that had previous experience with MOBA games, so we had to do some explanation of the game mechanics before they could start. It did not take long before another problem emerged which was lag. This made the game totally unplayable, and we had to restart the game, but only to replicate the lag problem. A quick workaround to this problem was to let the kids play in the UE4 editor and play alone against the minions. They did seem to enjoy the experience more when they played against each other, the kids did not mind that they had to play alone for a while. We later figured out why there was massive lag, see subsection 5.4.2 to read about it.

Overall they seemed to enjoy it, despite the bugs and lack of gameplay elements in the game. One thing that seemed to amaze them most was the possibility to buy different

items to get stronger. Since this is not balanced correctly it made them very strong.

7.1.2 Testing with Game Programming students

On May 9th we had another testing session, but this time with some of the other Game Programming students. We were a bit more prepared then last time. This time we knew that it worked over the LAN, and we had prepared a Google form for them to fill inn. Read more about the Google form in section 7.2.

During testing we gave all players 10000 gold to start with, to let them test all the items available. Since the items is not balanced, this meant that the players where able to kill minions and buildings way to quickly and finish the game within minutes. Thereby we didn't get to test all the different gameplay elements we wanted to test out.

Since the game have no way of telling or showing how the spells works, so we could not sit silently in the background and watch them play. Some explanation had to be done before they could start. The shop wasn't discovered before we told them about it either, so there is clearly a need for more/better graphical way to show the players how things work.

7.2 User Feedback

Getting feedback from our users is a crucial thing and there are many ways of going about it. The way we did was to use a Google form¹.

By using a form it's easy to collect the information we want to get from the test. It's also easy for the user to fill inn, making it more plausible that they actually do it. From the data we collected, we were able to make different diagrams to easily show and analyze the data. In Figure 35 we are showing some diagrams from the play testing with the other students.

7.3 Balancing

There are many ways you can go about balancing a game. We had balancing in mind from the start of the development (see subsection 2.1.2). By thinking about balancing early on in the process, it will make the balancing job easier as we go along.

We never got to the actual balancing part due to limited time. There are several things we could have done if we where to balance the game. First off, unreal have build in support for reading in values into tables from files. [10] By having this we could easily read values and use them to set variables such as health, damage *etc*. When everything is in the same place it is much easier to find and change the values that needs to be changed. This also means that we won't have to build a new version of the game just to change a single variable somewhere.

We have many different elements in our game that needs to be balanced, and most of them can be balanced by tweaking numbers.

¹<https://docs.google.com/forms/d/18hP94-2oHahK1MOJwNbgirHRbHZsU7PwOBSuvGdUBiY/viewform> - Form we used when we had playtesting

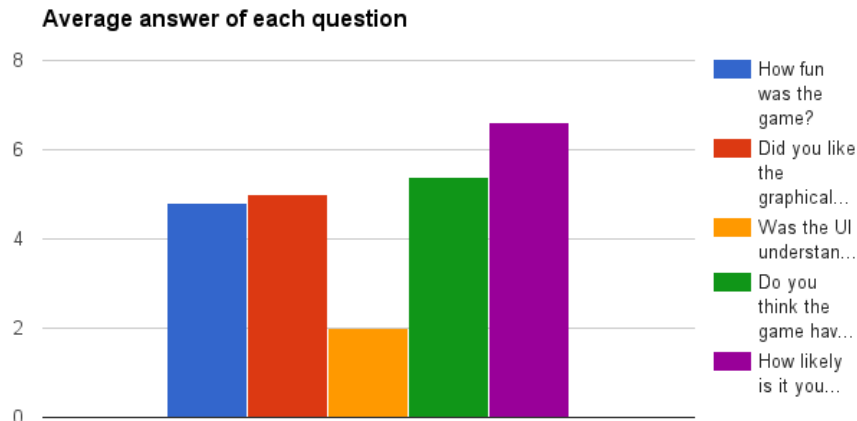


Figure 35: The average score for each answer.

- **Spells** - The spells are overflowing with different variables and ways of balancing. Everything from damage, range and cooldown, to cost, radius, *etc.* Since we have different "categories" of spells, a natural thing to balance on is damage, cost and cooldown. A stronger spell needs more mana than a weaker spell, deal more damage, and should have a longer cooldown.
- **Items** - Much like spells, items also contains many different variables and different things we can tweak to balance. But mostly we would tweak the cost for each item, depending on how powerful it is. As of now we don't have that many items and have decided for ourselves how much attack damage, attack speed *etc.* the item would give. To try and balance this, more play testing would have been needed.
- **Champions** - When it comes to balancing the different champions/classes in the game, first off we would start at looking at the spells. Since none of the champions shares spells, getting the spells balanced would have taken us a long way. This is also something that will be tweaked in the active lifetime of the game.
- **Buildings** - Other than the tower damage, there isn't much more than health that we would balance when it comes to the buildings. We would balance this by play testing.
- **Jungle Creeps/Minions** - There are a few thing to take into consideration when we would start to balance the minions. If we would set the health to high we would only end with a to hard early game, but to low and the end game would suffer. The damage can't be to high or else there won't be a prolonged battle, but more of a slaughter. How much gold they give when killed is something that will be balanced around the items, because most of the players income will come from the minions.

In the beginning of the play testing, we would start by setting the numbers really high, and test them. If they turned out to be to high, we would set them down to half the value and test again. If they then turn out are to low we would add half the value.

By repeating this enough times we will eventually end up approximately where we want to be.

8 Discussion

Discussion about the project, how it went, how it could be different and what could be expanded upon in the future.

8.1 Different approaches

A section on what we could have, or should have done differently when thinking in retrospect of how the project went.

8.1.1 Testing and User Feedback

We should have been more effective by testing more often. This would have revealed errors early on, so that they could have been fixed in the finished product. We would also get more continuous feedback on improvements and new ideas for new features. With more testing and feedback, balancing would also be more in focus.

8.1.2 Class Structure

We should have started off by creating base classes for different objects, to avoid having to repeat the same code over and over. Both buildings and units have health, so that could be its own base class. Alternatively, we could have used the Components feature in Unreal Engine, which lets you add a complete set of code to an already existing object.

8.1.3 Code Limitation

We shouldn't have limited ourselves to mainly working in Blueprint, but should also have programmed in C++. This was however one of the learning goals with the project, and if we hadn't used Blueprint mainly, it wouldn't have been as successful as it was. While it has some limitations, Epic Games are constantly improving and fixing the language.

8.2 Future Development

In our design document we explain a lot of different features and content we wanted to add in our game. Due to the time constraint, we did not get to implement everything we had planned. Therefore we add those to future plans for our game.

8.2.1 Features

Our main focus for future development would be to fix issues that we didn't have time to fix in time. One of the main features we were going to add to assert our difference from all the MOBA clones out there, was the possibility to affect the environment during gameplay. This is explained in more detail in [Appendix C](#).

As mentioned in [section 7.1](#), we did not prioritize visual effects or audio. This had

a negative effect during testing, since the players did not get enough audiovisual feedback. Because of this, we will prioritize some basic visual effects earlier in future projects to make testing easier early on.

8.2.2 Ingame Content

One of the easier parts of future development would be to expand on the ingame items and get them to work properly. As planned previously, we would also want to have different textures and skins and unlockable content in the future. This should be reasonably easy to implement, but it would require one to either make, or buy more art assets.

8.2.3 Future Bachelor Project

While the amount of work required to finish off the features we had planned, as well as the polishing and bugfixing of the game would have amounted to the work expected from a bachelor project, it wouldn't be recommended. The game has structural flaws that one would preferably have reworked entirely.

9 Conclusion

When we started this project, we were very enthusiastic about creating a good game that many people would find enjoyable. We had many good ideas of how to make a new and exciting MOBA game. In retrospect, we saw that we really over scoped. We new that this would be a taunting task when we started, but we did not realize just how much. At least we made a playable demo of our game, and those who tested it found it entertaining. With future development, we are certain we would have a really good game that many people would decide to buy and play on a regular basis.

Unreal Engine 4 was a good tool to use, and it is beneficial for us to learn such a sophisticated game engine. By learning the visual scripting language, it will be easier to work with designers who doesn't have much programming knowledge. It also gives us an insight in workings of a larger game engine, which will be helpful if we are going to develop our own engine. Since larger engines, such as UE4 and Unity, is getting more available for independent developers, the community around these engines will grow and more larger companies will make use of them.

As a group, we worked very well. We did not work as much as we could have, but we were faced with new tools and a completely new programming language. We had some set-backs that reduced motivation and caused less work being done. We achieved our goal of learning Unreal Engine 4 and Blueprint, since we had to dig deep in to the workings of the engine.

Overall we made something, that with a bit of polish, we can be proud of. We got an insight on how to work in groups on larger projects, which will be beneficial for later work.

Bibliography

- [1] Jesse, S. 2008. *The Art of Game Design: A book of lenses*. Morgan Kaufmann Publishers.
- [2] About blender. <http://www.blender.org/about/>. (Visited Mai 2015).
- [3] License blender. <http://www.blender.org/about/license/>. (Visited Mai 2015).
- [4] Dragert, C., Kienzle, J., & Verbrugge, C. 2012. Reusable components for artificial intelligence in computer games. In *Proceedings of the Second International Workshop on Games and Software Engineering: Realizing User Engagement with Game Engineering Techniques*, GAS '12, 35–41, Piscataway, NJ, USA. IEEE Press. URL: <http://dl.acm.org/citation.cfm?id=2663700.2663708>.
- [5] Rabin, S. 2014. *Game AI Pro: Collected Wisdom of Game AI Professionals*. CRC Press.
- [6] Map representations. <http://theory.stanford.edu/~amitp/GameProgramming/MapRepresentations.html>. (Visited Mai 2015).
- [7] Inc, E. G. How unreal engine 4 behavior trees differ. <https://docs.unrealengine.com/latest/INT/Engine/AI/BehaviorTrees/HowUE4BehaviorTreesDiffer/index.html>.
- [8] Kumar, K. 2013. *Learning Physics Modeling with PhysX*. Packt Publishing Ltd.
- [9] Game testing. http://www.gamasutra.com/view/feature/130745/better_games_through_usability_.php?print=1. (Visited Mai 2015).
- [10] Reading data from table. <https://docs.unrealengine.com/latest/INT/Gameplay/DataDriven/index.html>. (Visited Mai 2015).

A Project NORS: Terminology

This appendix will contain terminology and description of the words used in the thesis. We will assume the reader have basic programming knowledge and will as such only describe things that are either specific to the game or the Blueprint language we used.

Unreal Engine Terminology

- **Epic Games** - The company that own and develop Unreal Engine.
- **UE** - Refer to Unreal Engine, the engine we're using to create our game. More Specifically we used Unreal Engine 4.7.x
- **Blueprint** - The visual scripting language in Unreal Engine 4
- **A Blueprint** - Most often refer to an object that have code associated with it, or the code itself.
- **Level Blueprint** - This is the blueprint is associated with the Level or Map.
- **Node** - Refer to one of the many boxes that makes up the blueprint.
- **Wire** - The lines connecting the nodes in a blueprint, either passing variables and data, or as the sequence of execution.

Genre

- **MOBA** - Multiplayer Online Battle Arena. This is the genre of the game.
- **ARTS** - Action Real Time Strategy. An alternative name to the genre.
- **Bots** - Generally refer to AI controlled opponents and not real players.

Games

- **LoL** - Refer to the game League of Legends.
- **DotA** - Refer to the game Defense of the Ancients.
- **HoN** - Refer to the game Heroes of Newerth.
- **HoS** - Refer to the game Heroes of the Storm.

Map Environment

- **Lane** - Between each team's base, there is a road. This is referred to as a lane. In the most common map there are 3 lanes, referred to as top, mid and bot.
- **Jungle** - This refer to any ground between or around the lanes.
- **Bushes** - The terrain element bush that hides you from enemy vision.
- **Spawn** - Refer to the area at which the player or NPC unit is placed at during it's creation. For some units, they will also reappear at the spawn area after being killed. In which case they are respawned.

Buildings

- **Main Hall** - This is the main building. Destroying this will end the game and winner/loser is decided by whoever still have theirs standing. Different games name their building differently.
- **Towers** - Throughout the world each team have several towers placed. These have to be destroyed in turn before the team can attack the Main Hall.
- **Barracks** - This is where the Minion Commander spawns from. Destroying this will greatly decrease the force in which the minions push.

Units

- **Champion** - This is the unit controlled by the player. It have a set of skills and stats that it use in battle. Depending on the game, it can also be called Hero.
- **Minions** - Each team will spawn soldiers on their team that will aid in attacking the enemy. They be a small squad that attack push in each their own lane
- **Minion Commander** - This is a Minion that is superior in strength and with a more advanced AI. It will have the ability to affect the environment similar to the Champions.
- **Jungle Creeps** - Refer to monsters in the jungle that are not affiliated to any team.

Abilities

- **Q** - Refer to the ability bound to the "Q"
- **W** - Refer to the ability bound to the "W"
- **E** - Refer to the ability bound to the "E"
- **R** - Refer to the ability bound to the "R"
- **Ulti or Ult** - This refer to the R ability. In most games this ability is significantly stronger than the others, and is therefore called the "ultimate" ability. Which is shortened to "ult" or "ulti".
- **Cooldown** - This is the delay between each time you can use the same ability. Some abilities have longer cooldowns than others.
- **Mana, Power and Energy** - This refer to the cost of using an ability. Most commonly it's called Mana, and abilities cost a given amount of mana to be activated.
- **XP** - Stands for experience points and is used to level up the champion.

B Project NORS: Project Plan

The following is the project plan for Project NORS as written before the start of the project. Certain goals, features and milestones were shifted or removed. Other features came to life during the project development.

Project **NORS**

Jonathan Ness, Aleksander Olsen, Marius Rødland, Chris Sand

1. GOAL AND FRAMES

1.1. Background

1.2. Project goal (Effect and Result)

Effect

Results

1.3. Frames

2. SCOPE

2.1. Field

2.2. Limitation

2.3. Task Description

3. PROJECT ORGANIZATION

3.1. Responsibilities and roles

3.2. Routines and rules

4. PLANNING, FOLLOW-UP AND REPORTING

4.1. The main divisions of the project

4.2. Plan for meetings and when to make decisions

5. ORGANISATION OF QUALITY ASSURANCE

5.1. Documentation, use of standards and source code

5.2. Configuration management

5.3. Analyse the risks(identify, analyse, measures, follow-up)

6. PLAN OF ATTACK

Gantt-form of planned progress

Comments to the Gant-form.

1. GOAL AND FRAMES

1.1. Background

Since all of us in the group is interested in MOBA¹ games, and that our bachelor's degree in Game Programming lead up to us working on a big project, we decided on making a MOBA style game.

We have most experience with C++, and as such we decided on using Unreal Engine 4 (4.7.x) because it allow us to program in C++ and override limitations in the engine if need be. Though the majority of the programming will be done with the engine's visual scripting language Blueprint.

1.2. Project goal (Effect and Result)

Effect

- We want to make 60% of the people who try the game, become regular returning players
- Our goal is to have 20% of the users become paying users to some degree.
- We want 5% of the userbase to be generating content for the game. (Skins, Hats, Voiceovers, etc) We will use the same business model as Vale does with DotA2²

Results

- We will make our 3D models openly available to make it easier for the users to generate custom skins and textures, as well as eventually edit the 3D models as well.
- We want to give the regular players a free bonus of some sort to encourage them to come back and play more.
- By having some regular sales (weekly or monthly) to encourage the users to become paying users to benefit from a sale.

1.3. Frames

We have a limited timeframe of early January to mid May to complete the task. We also have several documents which have to be written in LaTeX and handed in by the 15th of May.

¹https://en.wikipedia.org/wiki/Multiplayer_online_battle_arena - MOBA - Multiplayer Online Battle Arena

² <http://www.dota2.com/workshop/> - DotA2 on User Generated Content.

2. SCOPE

2.1. Field

Since we are taking a bachelor's degree in Game Programming, we're making a bigger game project as our Bachelor thesis.

2.2. Limitation

We have chosen to work in Unreal Engine 4 (4.7.x), and as such we face most of the limitations present there. We can edit the source code to overcome those limitations, but it's unlikely that we'll have enough time to do that properly.

We have decided on using Bitbucket for version control and Toggle for time tracking. We will try out Jira as bug tracking and Scrum board, and Confluence as a Wiki and partial documentation. Some LaTeX compiler will be used to write up the thesis itself.

Our main focus will be the technical aspect, but we will as much as possible put time into creating an aesthetically pleasing game environment.

2.3. Task Description

The task is to make a MOBA game for our bachelor thesis. In order to do so we have decided to work in Unreal Engine 4 (4.7.x) with its benefits and limitations.

MOBA is a type of game where two teams of players battle against each other on a map. Each user control their own "hero/champion" and fight the other teams "heroes/champions". The objective of the game is usually to destroy the enemy's main base building.

We are going to make a MOBA game with some tweaks to it, so it won't become another MOBA clone. We have come up with these changes that will make this game more unique.

- Parts of the environment is going to be changed by some elemental powers (fire, water, air and nature.) that the characters have. E.g. there will be possible to freeze water so you will have a harder time to move your character around when you are on the frozen water.³

³ http://divinity.wikia.com/wiki/Divinity:_Original_Sin_Environmental_Effects - Similar Concept

- The game will have a currency system like DotA2⁴ and LoL, two of the bigger MOBA games. Here you slowly accumulate income over time, as well as earn it from killing enemy units. We will also incorporate control points that will increase the income over time when controlled by your own team.
- MOBA games have minions⁵ that constantly push against your enemy's defenses, and work as the main source of income from killing them. We want to give them a smarter AI, to avoid having them only run to their death immediately, but will also utilize the environment to their advantage, as well as trying to gain exp and grow stronger themselves.
- In most of the MOBA games out there you pick a predefined hero/champion⁶ with a specific set of skills. We will rather focus on having the user "build" up their Hero/Champion by deciding on their fighting style (Melee, Ranged or Magic) and what kind of element (Fire, Water, Air or Nature) they're going to use.

3. PROJECT ORGANIZATION

3.1. Responsibilities and roles

The fact that one person has something as their responsibility does not mean that others can't work on that field. Having the responsibility simply means that this person will be the one making the decisions on issues in that field. It's also safe to assume that the person with a particular responsibility will focus their attention on that field.

The responsibilities are divided up as such, but is in no way final, and are subject to change on a "I want to work with..." basis.

- Aleksander
 - Design
- Chris
 - Artificial Intelligence
- Marius
 - Network
- Jonathan
 - Webpage

⁴ <http://dota2.gamepedia.com/Gold> - Currency in Defence of the Ancients 2 - DotA2

⁵ <http://leagueoflegends.wikia.com/wiki/Minion> - Minions in League of Legends - LoL

⁶ <http://www.heroesofnewerth.com/heroes/> - Heroes in Heroes of Newerth - HoN

3.2. Routines and rules

See appendix A - Group Rules.

4. PLANNING, FOLLOW-UP AND REPORTING

4.1. The main divisions of the project

We quickly decided to go for scrum as our system development method. We chose this because scrum allows for changes to be made under the development process. And since we are going to make a game, its bound to come some changes along the way as issues arise or new features get suggested.

At least in the beginning, the sprint will last one week. This is subject to change if needed later in the project.

The tasks will be of various sizes. But in the early stages of the project, we will focus on setting up the developing environments separately to each machine, and make sure the source control doesn't break the workflow. Later on in the project we will have smaller tasks, such as fixing features in the game in Unreal Engine 4. As well as bigger tasks such as Networking between Client and Server.

Since we are rotating on who will be project manager, we will also rotate on who will be the Scrum master. The project manager and scrum master are not usually the same person, as they have different roles entirely. But we've decided on essentially merging the two roles and treat them as one because of the limited number of people, and the lack of a proper company structure. That way everyone will get experience as both Project Manager and Scrum Master.

In Scrum there is something called Sprint planning meeting and End sprint meeting. The sprint planning meeting is where the team discuss what to do in the upcoming sprint. End sprint meeting consist of two other meetings called Sprint Review and Sprint Retrospective. In the Sprint Review meeting we show what we did do and what we did not manage to do this sprint. The Sprint Retrospective meeting we simply reflect on what went well and what can be improved.

We plan to merge the Sprint planning meeting and the End sprint meeting into one meeting. We do this to save time, and minimize the number of meetings we need to have. It also opens up the possibility of working through the weekend, instead of the usual work days of Monday-Friday.

4.2. Plan for meetings and when to make decisions

As decided in the rules, we will have a meeting on Mondays. This will consist of the Sprint Review, Sprint Retrospective and Sprint Planning meeting. As well as change of Project Manager, which goes on rotation as per the rules.

There will also have an independent meeting with our internal supervisor, Simon McCallum. This will happen weekly on Monday at 9 AM. Later in the project this might change to biweekly meetings.

In the event that we need additional meetings, it's the Project Manager's job to arrange, with supervisor if necessary, as well as inform the rest of the group. Any bigger decisions will have to be done on a proper meeting so that it can be voted on. Smaller decisions can be decided with a majority of the votes on Skype etc, but shall also be written down on the meeting log.

5. ORGANISATION OF QUALITY ASSURANCE

5.1. Documentation, use of standards and source code

We haven't decided on a specific coding standard as we mostly have similar style of coding. But in the event that we disagree on any aspects of the code, we will refer to the Google C++ Style Guide⁷.

Most of the work will however be in Unreal Engine's Blueprint. We will solve any disagreement on coding style there by a majority of vote. Naming Conventions will be as defined by Tom Looman on his page⁸.

When it comes to documentation, every public function and classes will have to be commented and documented. Internal functions and classes should also be commented, but this is more for internal use during development.

These points are also written down in the Rules document.

5.2. Configuration management

We have chosen to use bitbucket as our repository. We chose Bitbucket over Github because we are simply more used to how it works. The advantage Bitbucket have over Github is it's integration with Confluence and Jira. Other than that there is no real advantage choosing the one over the other, especially when we can get a private repository in Github as well.

⁷ <http://google-styleguide.googlecode.com/svn/trunk/cppguide.html> - Google's C++ Style Guide

⁸ <http://www.tomlooman.com/ue4-naming-convention/> - Unreal Engine 4 - Naming Convention

5.3. Analyse the risks(identify, analyse, measures, follow-up)

Technology, Business, As the project group

Description	Probability (1-10)	Severity (1-10)
1: Epic Games stop supporting UE4 or license it differently.	1	7
2: The unique mechanics doesn't work the way we intended, and it becomes just another DotA/LoL clone.	3	7
3: We could potentially get a harsh community full of flaming and swearing.	4	4
4: The MOBA genre is full of games, making it impossible to make it unique and interesting without having an existing franchise to expand on.	4	8
5: Bitbucket have some problems and our project gets deleted.	2	4
6: We are not able to get the correct 3d models and the games ends up looking horrible.	7	5
7: Some of our hardware stop working, our laptops/server etc.	4	9
8: We have to moderate the User Generated Content, and it end up consuming way too much time.	3	6
9: The game might be hacked, allow for users to gather data about other users or generally break the game.	7	4
10: If we don't balance the game correctly we can end up with Pay to Win.	7	5

Dealing with the risks:

4: We have done some research on MOBA games currently out there and have found several aspects in which we intend to deviate from the rest of the genre from. See appendix B - Charts.

6: Since this is a bachelor thesis the actual game does not count that much, but it's still worth putting some time and effort in to it. Having a good looking game will be encouraging for us, and we will have something to show on a future job application.

7: This will be a major crisis, if we don't have any equipment to work on, nothing will be done. Some of us have other pc's to work on, but for those that doesn't this will become a real problem. Either they will have to buy a new laptop or borrow one from someone.

9: We will encrypt and hash any sensitive user information we end up storing. So the hacker will only get encrypted information if he breaks in. In the case someone hacks our game and we find out about it, it will be top priority to close the hole and patch it.

10: We will avoid any kind of friction that will directly affect the battles between players. If we only provide the user with cosmetic microtransactions, it will be impossible to buy an advantage in-game.

6. PLAN OF ATTACK

Gantt-form of planned progress


























	 Task Mode	Task Name	Durat	Start	Finish	Predecessor
1		▾ January	13 days	Thu 15.01.15	Sat 31.01.15	
2		Project Plan	10 days	Thu 15.01.15	Wed 28.01.15	1
3		Basic menu system	13 days	Thu 15.01.15	Sat 31.01.15	
4		Testing of modules	6 days	Mon 26.01.15	Sat 31.01.15	
5		▾ February	21 days	Mon 02.02.15	Sat 28.02.15	
6		Map structure	11 days	Mon 02.02.15	Sat 14.02.15	3
7		Simple Gameplay	11 days	Mon 16.02.15	Sat 28.02.15	6
8		▾ Basic AI	21 days	Mon 02.02.15	Sat 28.02.15	4
9		Tower AI	6 days	Mon 02.02.15	Sat 07.02.15	
10		Junge creep AI	5 days	Tue 24.02.15	Sat 28.02.15	6
11		Minions AI	7 days	Mon 16.02.15	Tue 24.02.15	6
12		▾ March	22 days	Mon 02.03.15	Tue 31.03.15	5
13		Champion AI	12 days	Mon 02.03.15	Tue 17.03.15	6
14		▾ Network	12 days	Mon 02.03.15	Tue 17.03.15	5
15		Server	11 days	Mon 02.03.15	Sat 14.03.15	
16		Client	11 days	Mon 02.03.15	Sat 14.03.15	
17		Testing of network	2 days	Mon 16.03.15	Tue 17.03.15	15;16
18		Multiplayer gameplay	10 days	Wed 18.03.15	Tue 31.03.15	17;13
19		▾ April	22 days	Wed 01.04.15	Thu 30.04.15	12
20		Playtesting	22 days	Wed 01.04.15	Thu 30.04.15	18
21		User feedback	22 days	Wed 01.04.15	Thu 30.04.15	18
22		Ai Balancing	22 days	Wed 01.04.15	Thu 30.04.15	18
23		▾ Mai	11 days	Fri 01.05.15	Fri 15.05.15	19
24		Polishing	11 days	Fri 01.05.15	Fri 15.05.15	21

Fig1.

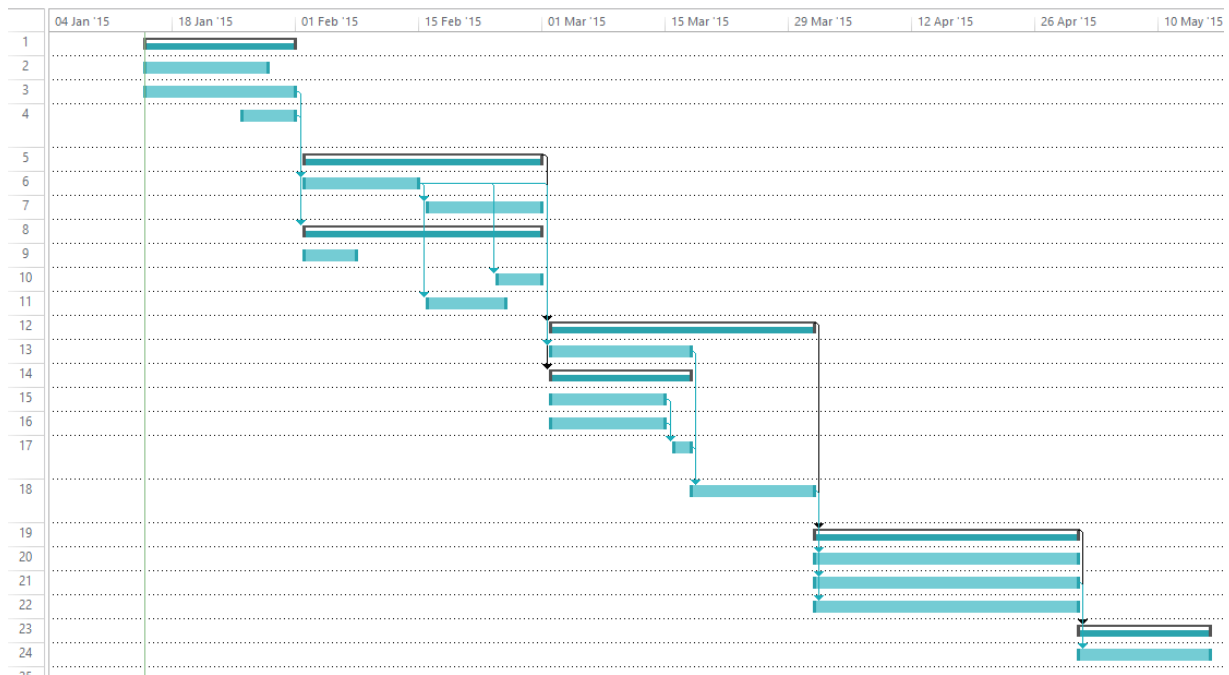


Fig2.

(The line numbers in Fig 2 corresponds to the line numbers in Fig 1.)

Comments to the Gant-form.

This Gantt-form only explains how we will make our game, not how we will write the thesis. The actual writing of the thesis will have to be done parallel to the game.

The milestones to the project is at the ending/starting of each month, so we split the Gantt-form into bulks containing each month. Where we will focus on new parts for each month. As shown in Fig1. We tried to make a estimation on how long each thing will take, but this might change fast when we have little to none experience on estimating things.

We have tried specifying as best as possible what components which are required before we can work on different components. For instance, multiplayer gameplay would require us to have networking working, as well as AI for Champions. We can't do proper AI on champions and units, before we have a map structure for pathfinding.

One thing that the Gant-form does not describe is who many will work on the same thing. We will decide this as we go on, depending on who wants to do what, or what works better in a practical way. When we are starting on the network (15,16) it's practical to be more than one working, since we need a server and the client to talk to eachother.

C Project NORS: Design Document

The initial Design Document for Project NORS.

Map

The main map of the game will be like the “Summoners rift” map in League of Legends.¹ There will be 3 lanes the players can take to get to their objectives. The setting of the map will be in an open landscape outside the town York, where the Vikings and the Saxons are fighting over the control over the city.

If there is time, we will try to make another map with two lanes. This map will be an urban map, where the players have to fight over control over the city.

Heroes

Each side will have three main classes. An up close and personal class, that does melee damage. An ranged class that throws axes or shoots arrows. And a magi class that can cast magic spells, possibly with area of effect.

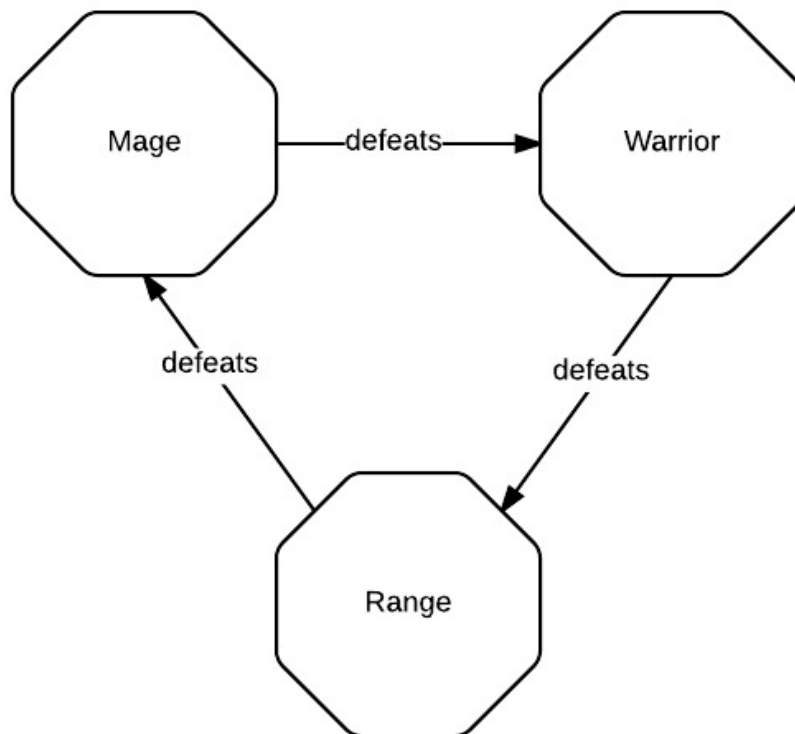


Fig. 1

We want to keep the classes balanced to some sort, so we went with the rock, paper scissors method. Where one class is strong against one class and weaker against another. As shown in Fig. 1 the warrior is stronger against the Ranged than it is against the Mage.

¹ http://leagueoflegends.wikia.com/wiki/Summoner%27s_Rift

Elements

With the power of the elements, you will be able to change the environment around you and increase your own abilities as well.

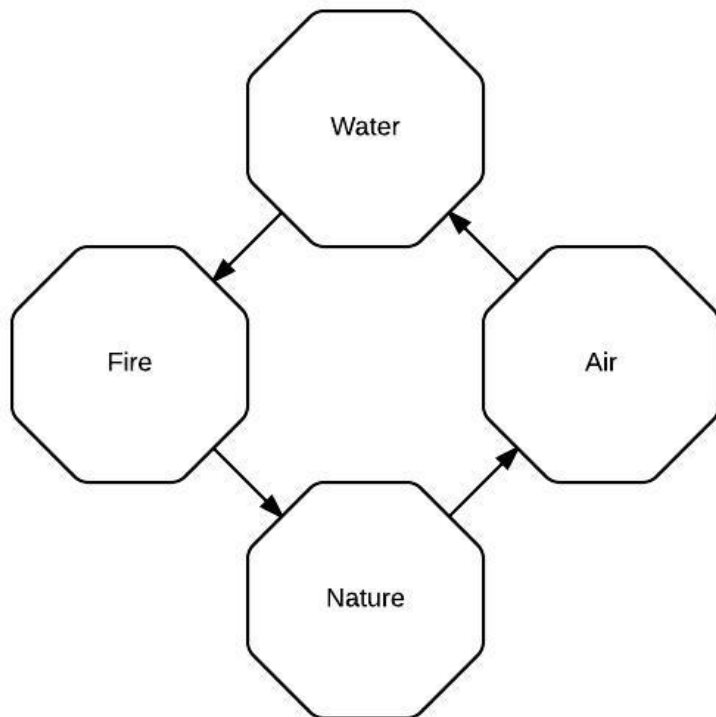


Fig.2 Made with lucidcharts

Rock Paper Scissors on the Elements

Water is able to put out fire, hence it's stronger against Fire.

Fire is able to scorch and burn the earth, hence it's strong against Nature.

Nature is able to block and cover for the wind, hence it's strong against Air

Air is able to freeze the water to ice and snow, hence it's strong against Water.

Other combinations won't have any preference to either side.

Bonuses from each Elements

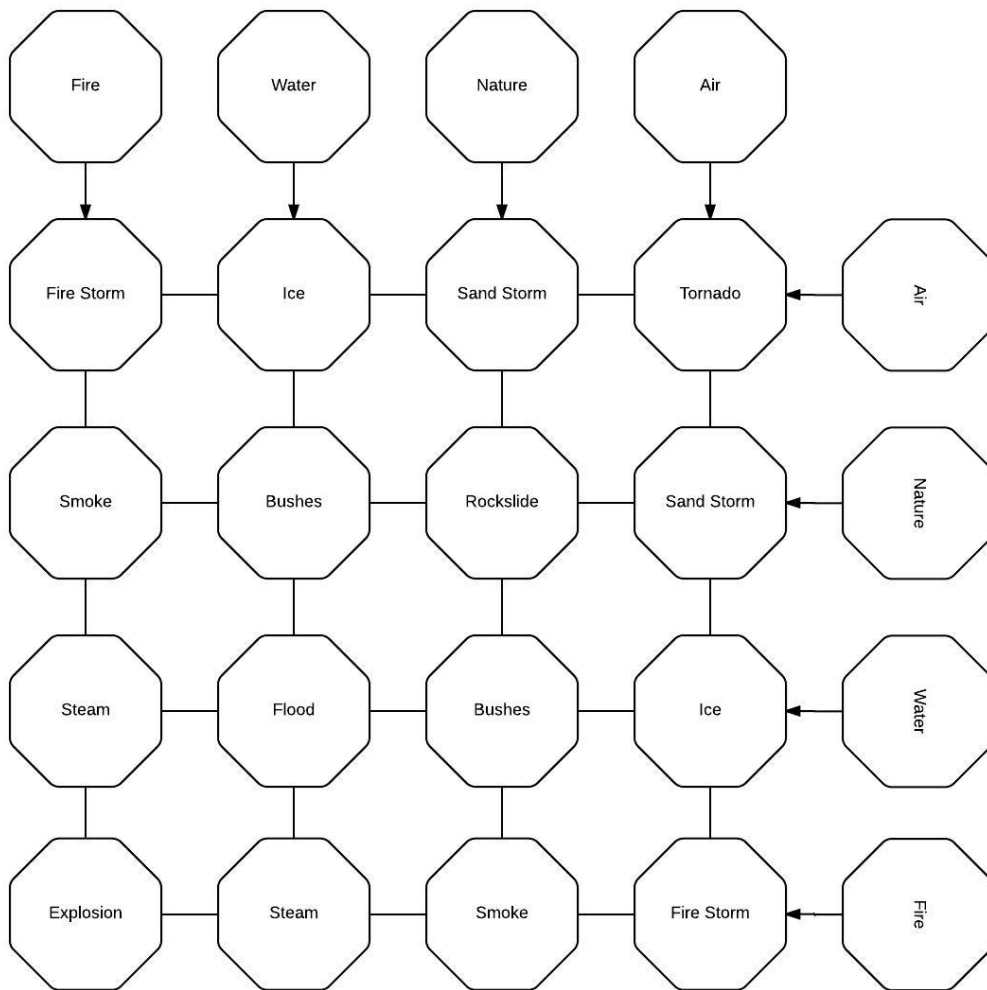
Water will provide you with healing bonus by cleansing the wounds.

Fire will provide you with a damage bonus by burning things in your way

Nature will give you a defensive bonus by being sturdy as rocks.

Air will give you a movement bonus by moving like the wind.

Combinations of elements



As the figure show, combining certain elements will provide you with certain special effect kind of attacks. The attacks will each have their own characteristics and work differently from the usual attacks.

Water and Air will provide you with Ice, which can affect the movement of the opponent by being slippery on the ground.

AI

In our game we want the AI to the minions/jungle creeps to be a bit smarter than they usually are in a "normal" MOBA game.

Minions: Instead of just running to their death against the closest tower, we want the minions to try to find optional routes. This is where a problem occurs, if there is no minions to take tower damage. So to work around this we come up with a solution where the minions will gain strength when they kill other minions. With increased power comes increased self-confidence and the minions will try to venture into the “jungle” to find gold and glory (or the death by jungle creeps).

Minion Commander: A Minion Commander would be stronger and smarter than the regular minion again. And it would possess elemental abilities that would let it affect the environment to its advantage.

Jungle creeps: In most MOBA games the jungle creeps have a camp where they stand still and wait. We think this is boring and want our jungle creeps to roam around. So instead of having to run around to the next camp after the other, you have to do a bit more searching to find them. We will also play with the idea that you can somehow mindcontrol the jungle creep. If you kill a jungle creep with the Nature element that jungle creep will spawn in the next minion wave.

Items

To give the game replay value and to give the players some way of getting stronger we need to have items in our game. If you look at League of Legends they have 192 items² you can buy, we think this is a bit too much, since we want the game to be easy to learn but hard to master. There should not be that many items, because that makes the game way too hard to learn.

Every champion will be able to buy 4-5 items, so you have to choose wisely, which makes it hard to master. So by choosing between a limited set of items, you have to be extra careful on what you want to buy.

The idea we have for items is that there is that there will be one item that increases one stat. E.g. there will be one item that increases your health, and one that increases your defence abilities and so on. So instead of having a bunch of items that increases a bunch of stats, you will have to be more thoughtful about what you will buy.

Tutorial

There will be a tutorial level, but we will focus more on showing off the interactions with the elements during the gameplay, rather than always telling the user that they can interact here and there.

This can be done by having a commander minion in every 5th wave or so, which will carry one element and affect the terrain accordingly. Showing the users that a given element affects the environment in a given way.

² <http://gameinfo.eune.leagueoflegends.com/en/game-info/items/>

Technology

- Unreal Engine 4 (4.7.0)

We chose UE4 because we feel it has a more “professional” feeling about it. Having an already made engine for us, make the job a lot easier, with not having to make an engine as well.

- Blender

Blender is a free to use 3d model program that we can try to make some simple models in. It's a good choice for us since it's free to use.

- Paint.net

This is also a free, a bit limited image editing tool. If you only are going to do very simple things, but requires that it is a bit more advanced than ms paint, this is your goto choice.

- Bitbucket

We need to source control our project, we simply chose Bitbucket as our repo because, we are used to it. It holds no real upper hand over GitHub or other similar services, except that it integrates good with Confluence and Jira.

- Sourcetree

Most of us use Sourcetree as our source control tool. It has an easy GUI and we know it pretty well. It uses Git in the background to do the merging and that kind of stuff.

- Git Bash

Some of us are professional and use the terminal to do their source control, and since we are all using Windows Git Bash is the program that Git gives out.

- Google Drive

We are going to write our thesis in LaTeX, but a lot other stuffs, just needs to be written down, like this. It's free and we can work simultaneously on the same document, making it really practical to use.

- Lucidchart

This is a plugin we're going to use for certain graphical representations of data. Such as rock-paper-scissors with the different classes, and the element combinations.

- LaTeX

We chose to write our thesis in LaTeX, mostly it's very practical to learn if we are going to continue our education. It's also relatively easy to make a good structured document, which is good for us.

- Jira
Jira is a issue tracking product made by Atlassian. It provides an easy way for us to do bug tracking and project management.
- Confluence
This is a powerful wiki site, that we will use to document our game, as we progress.
- Toggl
Toggl is a time tracking app/web page/desktop program that we use to track the time we use to do our tasks.
- MS Paint
This is a very simple image editing tool provided by Microsoft. We mostly use this when we just need to crop, resize e.g. an image.

Aesthetic

We will use Low Poly Models in order to avoid high system requirements for the game. Making it something that everyone can play as long as they have a computer. To get a personal feeling to the game, we will try to use hand painted textures to give it an unique style.

Lore

Our game is set during the Anglo-Saxons period. During this time, the Great Britain is taking shape and the vikings are plundering and pillaging all over. The vikings have also set their mind on the British Isles, so they start to invade the Saxons. This is also a time where the people worshipped different gods, and there are those who have been blessed by these gods. This enables them to control different natural elements, and have enhanced strength, agility and dexterity. These people are called Heroes or Champions by common folks, and are set at the forefront at the war.

Depending on the god the hero worship, he/she can control one of the four elements; fire, water, air or nature. With the help of this power, the hero must defeat the other heroes and destroy the enemy's place of worship and destroy their link to their gods. Thereby, they lose their powers and can easily be defeated.

D Project NORS: Gantt chart

Appendix containing only the Gantt Chart as made during the Project Plan at the beginning of the project.

Project NORS Gantt diagram.


























	 Task Mode	Task Name	Durat	Start	Finish	Predecessor
1		▾ January	13 days	Thu 15.01.15	Sat 31.01.15	
2		Project Plan	10 days	Thu 15.01.15	Wed 28.01.15	
3		Basic menu system	13 days	Thu 15.01.15	Sat 31.01.15	
4		Testing of modules	6 days	Mon 26.01.15	Sat 31.01.15	
5		▾ February	21 days	Mon 02.02.15	Sat 28.02.15	
6		Map structure	11 days	Mon 02.02.15	Sat 14.02.15	3
7		Simple Gameplay	11 days	Mon 16.02.15	Sat 28.02.15	6
8		▾ Basic AI	21 days	Mon 02.02.15	Sat 28.02.15	4
9		Tower AI	6 days	Mon 02.02.15	Sat 07.02.15	
10		Junge creep AI	5 days	Tue 24.02.15	Sat 28.02.15	6
11		Minions AI	7 days	Mon 16.02.15	Tue 24.02.15	6
12		▾ March	22 days	Mon 02.03.15	Tue 31.03.15	5
13		Champion AI	12 days	Mon 02.03.15	Tue 17.03.15	6
14		▾ Network	12 days	Mon 02.03.15	Tue 17.03.15	5
15		Server	11 days	Mon 02.03.15	Sat 14.03.15	
16		Client	11 days	Mon 02.03.15	Sat 14.03.15	
17		Testing of network	2 days	Mon 16.03.15	Tue 17.03.15	15;16
18		Multiplayer gameplay	10 days	Wed 18.03.15	Tue 31.03.15	17;13
19		▾ April	22 days	Wed 01.04.15	Thu 30.04.15	12
20		Playtesting	22 days	Wed 01.04.15	Thu 30.04.15	18
21		User feedback	22 days	Wed 01.04.15	Thu 30.04.15	18
22		Ai Balancing	22 days	Wed 01.04.15	Thu 30.04.15	18
23		▾ Mai	11 days	Fri 01.05.15	Fri 15.05.15	19
24		Polishing	11 days	Fri 01.05.15	Fri 15.05.15	21

Fig 1

Figure 1 is the written form of the Gantt chart.

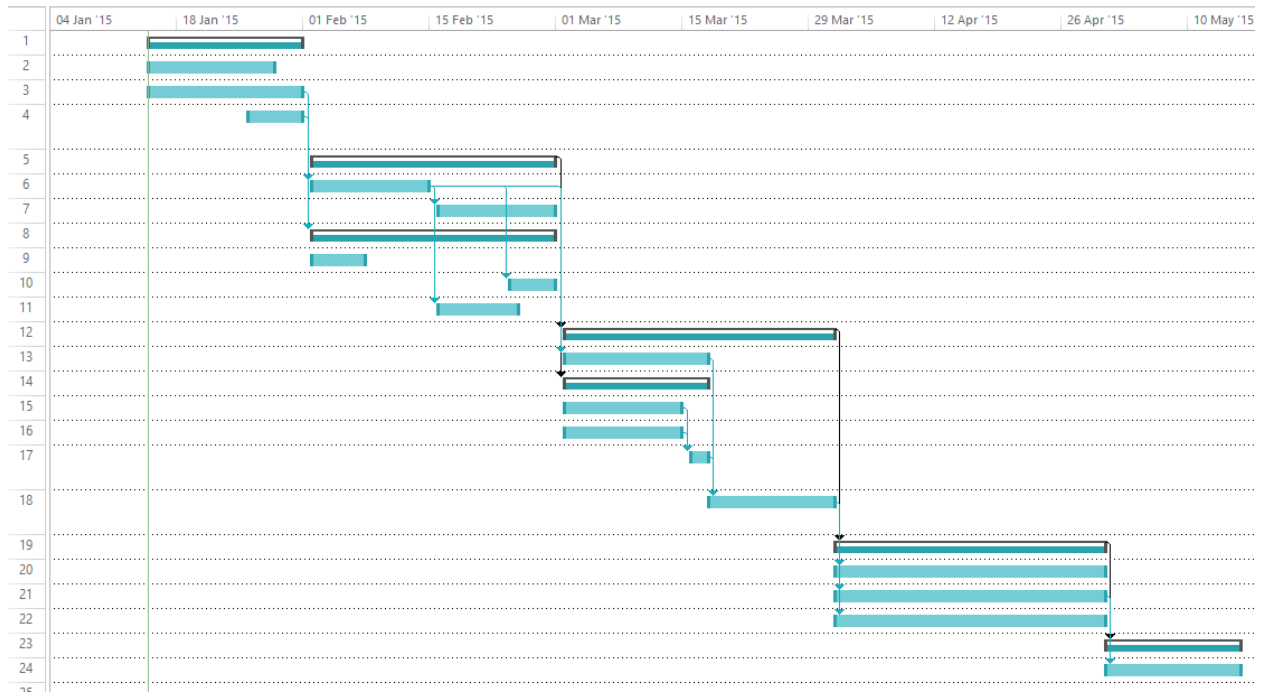


Fig2

Figure 2 is the graphical form of Gantt chart

E Project NORS: Confluence Wiki

PDF printout of the Confluence Wiki for the Project.

1. Project NORS Home	2
1.1 About the Project	2
1.2 Basic Game Details	2
1.2.1 Items	3
1.2.1.1 Amulet of Eternal Life	3
1.2.1.2 Amulet of Immense Power	3
1.2.1.3 Automatic Crossbow	4
1.2.1.4 Axe of the Wolfs	4
1.2.1.5 Dragon Bone Armor	4
1.2.1.6 Fist of Fury	4
1.2.1.7 Frying Pan	5
1.2.1.8 Health Potion	5
1.2.1.9 Helmet of Life	5
1.2.1.10 Power Elixir	5
1.2.1.11 Power Ring	5
1.2.1.12 Staff of Wisdom	6
1.2.2 Maps	6
1.2.3 Structures	6
1.2.3.1 Barracks	6
1.2.3.2 Main Hall	7
1.2.3.3 Towers	7
1.2.3.4 Walls	7
1.2.4 Units	7
1.2.4.1 Champions	8
1.2.4.1.1 Abilities	9
1.2.4.1.2 Elemental Passive	10
1.2.4.2 Jungle Bosses	11
1.2.4.3 Jungle Monsters	11
1.2.4.4 Minion Commander	11
1.2.4.5 Minions	11
1.2.4.6 Shopkeepers	12

Project NORS Home

Project NORS is the Bachelor project for Jonathan Ness, Aleksander Olsen, Marius Rødland and Chris Sand.

The project is the development of a MOBA game in Unreal Engine 4.

[More about the Project](#)

[Basic description of the Gameplay](#)

About the Project

Project NORS is the bachelor project for Jonathan Ness, Aleksander Olsen, Marius Rødland and Chris Sand.

The Project is to create a MOBA styled game in Unreal Engine 4 that we will be proud to show off and display in a portfolio.

The game will have certain unique elements intended to make it stand out in the Genre, but will stick to a "Easy to learn, hard to master" kind of philosophy along the way. This to avoid adding complicated mechanics that add little to the game.

More detailed description of the gameplay can be found [here](#).

Webpage for Updates and Signing up for Alpha/Beta -testing: www.tinyurl.com/ProjectNORS

Basic Game Details

Login

There will be a login when you start the game, this because the user will store some statistics and data about their games.

When we get to it, the different users will also be able to buy and create different skins and 3D models to use ingame, this will then be linked to their specific account.

Game Start

When starting the game, you will select what Map you want to play, and then if you wish to invite and play with friends, or join random people.

There will then be a queuing system for creating a game. When a match have been found, each player select what kind of Champion they want, and what element they want to possess.

Maps

There will be a standard map that consist of 3 main paths between the bases, as well as some jungle between the lanes where indifferent monsters lurk.

We intend to make a single lane tutorial kind of map as well, used for learning basic controls when you're new to the game.

Gameplay

The basic gameplay is that each player control their own champion and fight in a team battle. The controls base themselves on mouse for movement, and specific keys for activating different abilities.

The abilities will be based on a chosen element, and these elements will also have some effect on the environment.

The goal of the game is to destroy the Enemy's main base, while protecting your own.

Upon killing enemy units or monsters, you will gain some income that can be used to buy and upgrade your own gear.

Units

There will be several different units for the player to chose from.

- Ranger
- Mage
- Melee

As well as several non playable units that contribute to the gameplay in various ways

- Minions
- Minion Commanders
- Jungle Monsters
- Boss Monsters
- Shopkeepers

Items

Here is a list of the current items in the game.

- Health Potion
- Power Elixir
- Helmet of Life
- Power Ring
- Fist of Fury
- Dragon Bone Armor
- Axe of the Wolfs
- Staff of Wisdom
- Amulet of Eternal Life
- Amulet of Immense Power
- Frying Pan
- Automatic Crossbow

All icons provided under the Creative Commons 3.0 BY license.

More info and icons available at game-icons.net

Amulet of Eternal Life



Price: 1000 gold

The red stone in this amulet is said to have fallen from the sky, and forged in the lava pits underneath the mountain of the gods. Increases health by 25%.

Icon made by Lorc, <http://lorcblog.blogspot.com>

Amulet of Immense Power



Price: 1000 gold

The blue stone in this amulet was retrieved from the deepest pit in the deepest ocean. Then it was imbued with magic by the most powerful wizards of ancient times. Increase power by 25%.

Icon made by Lorc, <http://lorcblog.blogspot.com>

Automatic Crossbow



Price 1000 gold

A crossbow that reloads itself, meaning you can attack faster. Increase attack speed by 25%.

Icon made by Carl Olsen, <https://twitter.com/unstoppableCarl>

Axe of the Wolfs



Price: 250 gold

The wolf clan is a horde of barbarians. This axe is there weapon of choice. Increase attack damage by 75.

Icon made by Lorc, <http://lorcblog.blogspot.com>

Dragon Bone Armor



Price: 50 gold

This armor is made by the bones of fallen dragons. It is lighter and stronger than any other armor. Increase attack speed by 25.

Icon made by Lorc, <http://lorcblog.blogspot.com>

Fist of Fury



Price: 50 gold

Hit harder with these awesome gloves. Increase attack damage by 25.

Icon made by Lorc, <http://lorcblog.blogspot.com>

Frying Pan



Price: 1000 gold

Because everyone knows that a frying pan is the ultimate weapon. Increase attack damage by 25%.

Icon made by Aleksander Olsen.

Health Potion



Price: 20 gold

Restores 10 health points.

Icon made by Aleksander Olsen.

Helmet of Life



Price: 50 gold

This is your basic helmet. It helps protect against blows to the head. Increase health by 25.

Icon made by Delapouite, <http://delapouite.com>

Power Elixir



Price: 20 gold

Restores 10 power points

Icon made by Aleksander Olsen.

Power Ring



Price: 50 gold

This ring is imbued with magical powers. Increase your power by 25.

Icon made by Lorc, <http://lorcblog.blogspot.com>

Staff of Wisdom



Price: 250 gold

This staff once belonged to an old wise turtle. They say his speed was unmatched by anyone. Increase attack speed by 75.

Icon made by Lorc, <http://lorcblog.blogspot.com>

Maps

Single Path

The map with a single path between the bases will mainly be used for tutorials and learning the game.

That aside, it will be a playable map with a given number of champions on each side. Given how simple the map in itself is, it's unlikely to have much Jungle area or Jungle Creeps

Three Paths

This will be the main Map to play the game on. It will feature 3 paths between the bases, as well as some divider diagonally along the middle.

The space between the lanes will make up the jungle area where the Jungle Monsters roam around.

Structures

Every team will have several structures over the map that are vital to the gameplay.

Inside Team Base

Each Team's Base will contain the **Main Hall** building that is the goal of the game. Upon destroying the enemy's core building, the game is won. At the same time, you have to defend your own from attacking.

The **Main Hall** will be defended by towers in the immediate proximity. **Towers** fire at the enemies within range, targeting minions to start with and enemy Champions if a battle occur under the tower.

The Base will also contain the **Barracks**, this is where the minions spawn from.

Base itself will be surrounded by **walls**, so that there are limited entrances or exits.

Outside of Base

The different paths between the bases will have several **Towers** placed throughout the path. They will work the same way as the towers inside the main base.

This will mainly be the structures outside each team's base.

Barracks

The barracks are buildings within the walls of each base. They are responsible for spawning the elite minion commanders in the game. By destroying the enemy team's Barracks, they are unable to respawn their commander, which would've provided morale and firepower to the minions around it.

Main Hall

The core is the game deciding building.

Once a Main Hall is destroyed, the game is over. The team who still have their Main Hall building are victorious.

To Main hall is that base's headquarters. When it is destroyed that team have been defeated.

Towers

The towers are defensive outposts that attack enemy units within range using electric lightning.

This magic attack is unique to the towers, and as such, none of the other elements have a particularly good defense against it and cannot be countered truly.

Once a tower is destroyed, it cannot be rebuilt by the players, and will remain as a rubble for the rest of the game.

Walls

There will be walls surrounding the team's bases. These walls are indestructible even by Champions, and the only way around them, is literally to walk around them to the different openings.

Units

Playable Units

The playable units will be what we refer to as Champions.

There are 3 types of champions based on their attack style

- Ranged
- Melee
- Magic

And each of them can pick one element out of the following:

- Fire
- Air
- Nature
- Water

The combination of type and element give some variation, and their synergy between different champions would be important to the gameplay.

The different elements can either boost each other when friendly, or counter each other if enemies.

Non Playable Units

Active Combat Units

Active combat units are the units that actively join the combat. These are the Minions from each team, as well as the Minion Commanders that spawn every few waves.

The [Minions](#)' main goal is to push the different paths back to the enemy base.

[Minion Commanders](#) will prioritize to attack champions and change the terrain to their advantage based on their element. If they survive to grow stronger, they will start to roam around in the jungle.

Passive Combat Units

Passive Combat units are the [Jungle Monsters](#) who roam around protecting their territory, as well as the [Boss Monsters](#) who only sit at their base.

Non-Combatant Units

Non combatant units are each team's [Shopkeeper](#) that sell them their items.

Champions

Classes

The champion will feature 3 different classes combined with 1 of 4 different elements.

- [Ranged](#)
- [Melee](#)
- [Magic](#)

The elements that can be wielded are

- [Fire](#)
- [Water](#)
- [Nature](#)
- [Air](#)

Who each give a [passive bonus](#), as well as specific ones depending on the [Champion class](#)

Ranged Champion

The ranged champion will have the main focus of dealing direct damage over a distance.

Their defensive stats are average as a base

Attack damage is average as a base

They have a slightly higher movement speed and attack speed.

[Ranger Abilities](#)

Melee Champion

The Melee champion will have the focus of being the "tank" one, standing in the middle of the battle.

They will have higher defense as well as decent direct damage.

Movement speed is slower with an average attack speed.

[Warrior Abilities](#)

Mage Champion

The magic champion will have area of effect spells and buff/debuff spells as their main role.

They have low armor and relatively high attack speed, and they work as the supporting role.

They also have the best opportunity to affect most of the environment at during battles, because they both damage and hit the terrain at the

same time.

Mage Abilities

Abilities

Every champion will have 4 abilities to use, as well as a passive modifier as a result of the chosen element. In general, the abilities follow a pattern.

- Q - Damage enemies
- W - Getaway / Movement utility
- E - Special to class and element
- R - Large amounts of damage

Warrior

Active Q - Stab

The warrior will thrust his sword in front of him, dealing damage to the unfortunate souls in his path.

Active W - Dash'n Slash

With the help of powerful leg muscles, the warrior leaps forwards towards an enemy and slashes him with his sword.

Active E - Shield

The warrior will shield himself, blocking incoming damage. The elements will also provide a side effect.

Element	Effect
Fire	Burns Nearby Enemies
Nature	Extra strong shield
Water	Increased Healing
Air	Pushback on enemies

Active R - Tornado

The warrior is a gentle soul and have been doing ballet for many years. With this he have learned to spin fast and a lot, like a tornado! By applying this skills to the battlefield, he can make a sword tornado out of him self, dealing loads of damage to nearby enemies.

Ranger

Active Q - Snipe

The Ranger will fire of a stronger attack in the dirrection he chooses.

Active W - Sneak

After years living in the wilderness the Ranger can move around the map unseen for a while.

Active E - Trap

Traps is a big part of the hunt, therefore the Ranger is an expert on it! He can place out taps that have different effects depending on what element he belongs to.

Element	Effect
Fire	Burn Damage
Air	Slow Movement
Nature	Root enemy
Water	Poison Damage

Active R - Barrage

After years of practice with the bow and arrow, the Ranger can shoot out arrows at a blazing speeds, but with some redused damage.

Mage

Active Q - Elemental Strike

The mage will strike a small area with elemental damage, dealing damage to everyone within the area, as well as changing the environment accordingly.

Active W - Teleport

The mage teleports to the mouse location

Active E - Buff / Debuff

The Mage will buff allies, or debuff enemies. The effect is based on the elements

x	Allied	Enemy
Fire	Extra Burn Damage	Damage over time
Air	Increased Attack Speed	Decreased Attack Speed
Nature	Shield	Decrease Damage
Water	Healing over time	Decrease Healing

Active R - Elemental Blast

The mage will blast a large area, dealing large amounts of elemental damage, as well as changing the environment.

Elemental Passive

All champions will select one element to go with their champions. The elements have roles they are best suited for, but can be used as one wish.

Fire - Mainly Damage Dealers

Passive: Extra burn damage when you deal damage.

Water - Support and healing

Passive: Increased Passive Healing.

Nature - Mainly for the Tank

Passive: Increased armor as a shield.

Air - Agility and utility

Passive: Increased movement speed.

Jungle Bosses

These are massive Boss Monsters that usually require a majority of the team to defeat.

They will give the entire team a bonus of some kind that can easily turn the tide of the battle.

There will most likely be 2 Jungle Bosses on the 3 Path map.

In the event that there is any jungle area in the single path map, there might be one boss there as well.

Jungle Monsters

The Jungle Monsters are passive combat units.

They are passive because they don't actively search out enemies to fight. They roam around and defend their territory from enemies. If the enemies are Champions, Minions, Minion Champions or other Jungle Monsters doesn't matter.

If the Jungle Monsters get more kills, they will grow stronger and expand their territory. This is likely to cause them to engage in more battles with other units as they now roam further around.

Jungle Monsters will come in a variety of types. Some will be highly defensive, while others will be sneaky and do large amounts of damage.

Certain Jungle Monsters might give special effects to the player when they are killed.

Minion Commander

The Minion Commanders will be stronger minions that spawn every 5th wave or so.

The Minion Champions will be Mages that wield one element decided randomly at spawn.

If their elements defeat the opponents Minion Champion the pushing of waves can easily turn the tides. Because of this, one can't simply leave the minions alone because it'll be even.

If the Commander successfully push the wave so that it grow stronger, it might be daring enough to venture into the Jungle to try and get some extra experience.

Minions

Minions

The minions are the main source of income for the player, as well as the basic ally in keeping pressure at the enemy.

The basic minions will spawn and mainly push down each a given lane. In the event that they spot an enemy on the way, they will chase that one down and attack it for a certain distance. This could also include Jungle Monsters that have grown strong and daring, hence they invade lanes in search of prey.

They have neutral attacks without any element affiliation.

They come as ranged or melee units, with the Mage one being the Minion Commander and only spawning every 5th wave or so.

Shopkeepers

Each team will have their own shopkeeper located at the base. They don't participate in combat, but will sell the respective teams their gear.

Despite being two different units, they will sell the exact same gear.

List of available gear can be found [here](#).

F Project NORS: Business Model

In the event that we follow up after the project and build a company around it, the following is the reasoning and conclusions we've made in relation to what business model we would chose for the game.

Business Model

Data from the industry

Out of the MOBA games listed on Wikipedia¹, 23 are free to play, 6 are pay to play, and the original Genre defining DotA is a mod which in itself is free to play, but require base games in order to play properly. Even if the list is incomplete, it features some of the bigger MOBAs around, and is therefore a reasonable average to the MOBA genre.

This isn't that surprising, when there is research that show Free to Play games have a total revenue that's as much as three times what Pay to Play games make in the same region². Even World of Warcraft by Blizzard Entertainment, one of, if not the biggest monthly subscription based game^{3,4}, now offer a limited free to play alternative⁵ and have some microtransactions.⁶

Average Revenue Per User

It should be noted that the revenue by the different free to play games vary heavily depending on how it's monetized. League of Legends is on average earning less than a third of what World of Tanks earn per user, despite having roughly five times as many active players⁷. However, the source of this data warn not to focus solely on generating revenue.

“Focus on user experience, not revenue. Sure, we've all heard the arguments about the exploitative aspects for free-to-play. But if it all it ever did was suck the life out of customers, there would be none. Instead, we see an audience of dedicated gamers growing across genres. Take your audience serious by providing a rewarding experience, and they will reward you in turn.”

This fit well with how there are now users who live off of generating content for free to play games such as Dota2 and Team Fortress 2.⁸ Some data show that allowing for User Generated Content can actually increase active users dramatically.⁹

¹ https://en.wikipedia.org/wiki/List_of_multiplayer_online_battle_arena_games - List of some MOBAs

² <http://www.superdataresearch.com/market-data/mmo-market/> - F2P revenue vs P2P revenue

³ <http://www.polygon.com/2014/11/19/7250737/world-of-warcraft-warlords-draenor-10-million-subscribers>

⁴ http://en.wikipedia.org/wiki/World_of_Warcraft - Data on World of warcraft

⁵ http://www.wowwiki.com/World_of_Warcraft_Starter_Edition - World of Warcraft F2P component

⁶ <https://us.battle.net/support/en/article/world-of-warcraft-shop> - World of Warcraft Microtransactions

⁷ <http://www.superdataresearch.com/blog/mmo-arpu/> - Average Revenue Per User for different games.

⁸ <http://www.theverge.com/2013/1/8/3852144/gabe-newell-interview-steam-box-future-of-gaming> - UGC

⁹ <http://caas.raptr.com/the-secret-behind-team-fortress-2s-post-f2p-success/> - User Generated Content

Conclusion

As such, we would likely decide on a free to play model, supporting User Generated Content for monetization, rather than adding friction that one pay to remove. This because we don't have to 'break' our game in order to earn money, but also get to give back to the community.

G Project NORS: Grouprules

The following is the group rules that we made in the start of the project.

Group Rules - Project NORS

Work flow:

- Work from 09:00 - 15:00 - mon-friday. But flexible if something interrupts.
- We will have daily Scrum meetings at 09:00, this will be hold on skype. The meeting will be done as defined on Wikipedia¹.
- We will have sprints lasting one week, so the monday meetings will function as a Sprint planning meeting. (If needed the sprints can last longer.)
- We will still have some other classes, so the hours will be flexible around them.
- You have to log when you work in Toggle and link to resources used (sites for information, videos, free assets, etc.)

- Meeting at least once a week (Mondays) to discuss what has been done and what shall be done that week.
- It will be written notes from every meeting.
- If you can't turn up for a weekly meeting, you have to let the Project Manager know.
- On the start of each monday meeting, we will have a code review.

Code:

- If conflicts occur about the structure of code, we will use google code standard as reference.²
- Brackets shall be on the same line (e.g. } else {)
- Don't push something that is not commented and documented, code or blueprints.
- If you find some code, that have been done in a "bad" way, you have to inform the author of the code, and/or fix it yourself with proper comments.

- Unreal Engine 4 Naming Convention will be as defined by Tom Looman on his page.³
- Unreal Engine 4 Blueprint Style disagreements will be solved by a majority of vote on a full meeting.

¹ http://en.wikipedia.org/wiki/Scrum_%28software_development%29

² <http://google-styleguide.googlecode.com/svn/trunk/cppguide.html> - Google C++ Style Guide

³ <http://www.tomlooman.com/ue4-naming-convention/> - UE4 Naming Convention.

Project manager:

- We will circle the responsibility of being the project manager (PM). Rotation alphabetically based on last name. (N, O, R, S)
- The responsibility of the PM will be:
 - The PM will be the moderator when we have meetings.
 - If a meeting is needed outside the weekly meeting, it's the PM's responsibility to arrange a time and place that works for everyone in the group.
 - The PM will be responsible to decide who does what.

Buying of assets, licences, hardware, etc.

- In the event that we will buy some assets or licences, the cost will be divided among all members of the group. Shared "pot 'o gold" kind of way.
- There will not be any actual money usage unless everyone have agreed on it, in a meeting.

Decisions:

- Big decisions should be discussed where every party member gets to speak their case.
- They will be voted on based on a majority of votes, and the PM count as 2 votes.
- Decisions should not be brought back up and re-decided unless some new information have appeared. (For instance, not bring up past weeks decision after PM change)
 - In the event that there is strong disagreement with PM having two votes at a certain topic, we will schedule a meeting with our Supervisor and he will get the deciding vote.

Non contributors:

- If no work can be shown at a meeting, without a valid explanation, a warning will be given
- With 3 repeated warnings within a span of a month, external supervisor will be brought in.
- Beyond this, there is a risk of being removed from the group, depending on explanations

H Project NORS: Daily Scrum Log

The following is the log of our daily scrum meetings.

14.01.2015

Current Project Manager - Jonathan Ness

Early startup phase, mostly research

Shared

Will do:

Meeting with Simon / Mariusz to fix error with grouping.

Aleksander Olsen

Have done:

Installed Unreal and played around some, also watched some more tutorial.

Will do:

Called in sick today, will get better.

Chris Sand

Have done:

Found research on differences between the different MOBAs

Will do:

Will continue to find differences between the different MOBAs, to map out the genre to find out how we differentiate from the competition

Jonathan Ness

Have done:

Worked on the Business Model with research as to what options are valid.

Got Unreal Engine up and working.

Will do:

Started to work on the project plan.

Research

Marius Rødland

Have done:

Worked on the project plan, started to generally fill in things.

Will do:

Will continue to work on the project plan

Date 15.01.2015

Current Project Manager - Jonathan

Still early startup.

Aleksander Olsen

Have done:

Called on sick.

Will do:

Started to look at getting the UE4 on to our repo

Chris Sand

Have done:

Made charts showing our game in context to other games. And learn to use HTML import to import data to our spreadsheets.

Will do:

Get more data for different types of moba, set up AI diagrams showing template decision tree.

Jonathan Ness

Have done:

Did a little on risk analysis.

Will do:

More risk analysis and general Project Plan work.

Marius Rødland

Have done:

Continued to work on the project plan.

Will do:

Will have a look on a app called Asana that we can use as a scrum board. Will continue to work on the project plan as well.

Date 16.01.2015

Current Project Manager - Jonathan

Aleksander Olsen

Have done:

Stated to get the repo up and going.

Will do:

Getting the repo up and running.

Chris Sand

Have done:

Made more diagrams about the MOBA genre, looked for 3d models to use.

Started to look at champion AI.

Will do:

Finishing the project plan.

Jonathan Ness

Have done:

Worked on the project plan, and did research about the MOBA genre.

Will do:

Finishing the project plan.

Marius Rødland

Have done:

Mostly the project plan, did not get time to look at the app.

Will do:

Finishing the project plan.

Date 19.01.2015

Current Project Manager - Aleksander Olsen

Global Game Jam this weekend.

Shared

Will do:

Global Game Jam 23-25 January

Getting to know LaTeX

Aleksander Olsen

Have done:

Built his own Unreal Engine 4

Will do:

Continue with setting up repo for project

Design document

Chris Sand

Have done:

Working on project plan.

Found 3D models to use in game.

Made some texture examples.

Researching other MOBA games in difference to ours.

Will do:

Introduction to JIRA and agile project management

Jonathan Ness

Have done:

Project plan

Will do:

The website with a Google Form for recruiting alpha/betatesters.

Create a shared email for the Project.

Marius Rødland

Have done:

Working on project plan.

Will do:

Design document.

Date 20.01.2015

Current Project Manager - Aleksander Olsen

Shared

Will do:

Global Game Jam 23-25 January.

Getting to know LaTeX.

Play around in UE4.

Aleksander Olsen

Have done:

Created repo for project.

Will do:

Design document.

Chris Sand

Have done:

Working on 3D modeling.

Will do:

Introduction to JIRA and agile project management.

Jonathan Ness

Have done:

The website with a Google Form for recruiting alpha/betatesters.

Create a shared email for the Project.

Will do:

Marius Rødland

Have done:

Design document.

Will do:

Design document, LaTeX and UE4.

Date 21.01.2015

Current Project Manager - Aleksander Olsen

Shared

Will do:

Global Game Jam 23-25 January.

Getting to know LaTeX.

Play around in UE4.

Aleksander Olsen

Have done:

Will do:

Fix git repo.

Chris Sand

Have done:

Will do:

Game design.

Jonathan Ness

Have done:

Will do:

Look at materials in UE4.

Marius Rødland

Have done:

Design document.

Will do:

LaTeX.

Date 22.01.2015

Current Project Manager - Aleksander Olsen

We will focus on the design document so that we have done everything that is to be handed in on 28/1. This way, we can focus on getting to know UE4.

Shared

Will do:

Global Game Jam 23-25 January.

Getting to know LaTeX.

Play around in UE4.

Aleksander Olsen

Have done:

Getting to know UE4.

Will do:

Design document.

Chris Sand

Have done:

Design document.

Will do:

Design document.

Jonathan Ness

Have done:

Made a dynamic material that go from grass to burnt ground based on 2 parameters

Will do:

Design document

Getting to know UE4

Marius Rødland

Have done:

Getting to know LaTeX.

Will do:

Design document

Date 23.01.2015

Current Project Manager - Aleksander Olsen

Shared

Will do:

Everyone is off to Global Game Jam on Hamar.

Aleksander Olsen

Have done:

Will do:

Chris Sand

Have done:

Will do:

Jonathan Ness

Have done:

Will do:

Marius Rødland

Have done:

Will do:

Date 27.01.2015

Current Project Manager - Marius Rødland

Aleksander Olsen

Have done:

Made a simple map and went through the project plan

Will do:

Pausing menu.

Chris Sand

Have done:

Made a simple map and went through the project plan

Will do:

How to spawn the minions and start on the minions AI

Jonathan Ness

Have done:

Made a simple map and went through the project plan

Will do:

Starting screen.

Marius Rødland

Have done:

Made a simple map and went through the project plan

Will do:

Making the camera movements, and a final read through the project plan

Date 28.01.2015

Current Project Manager - Marius Rødland

Aleksander Olsen

Have done:

Making a pause menu

Will do:

will continue to make it better

Chris Sand

Have done:

Spawning of minions and pathfinding.

Will do:

Make the minions run to different places.

Jonathan Ness

Have done:

Made a simple start menu.

Will do:

Continue on the start menu, and improve it.

Marius Rødland

Have done:

Camera and camera movements.

Will do:

Make the camera better and add a dude.

Date 29.01.2015

Current Project Manager - Marius Rødland

Aleksander Olsen

Have done:

Did some more menu stuff.

Will do:

Have to do math, because of task.

Chris Sand

Have done:

More minions AI, waypoint and s circular mesh

Will do:

Make them follow the waypoints. Figure out how to do events.

Jonathan Ness

Have done:

Read through the project plan and made sure it was good enough to deliver.

Will do:

Will continue to play with the menu.

Marius Rødland

Have done:

Looked ai network, and started on the player.

Will do:

More on the player, hopefully he will be ready today.

Date 30.01.2015

Current Project Manager - Marius Rødland

Aleksander Olsen

Have done:

Did Math.

Will do:

Will finish the menu.

Chris Sand

Have done:

Make them follow the waypoints

Will do:

More AI

Jonathan Ness

Have done:

Did something in the menu, but not much because of the karrieredag.

Will do:

Marius Rødland

Have done:

Did something, karrieredag took a lot of time.

Will do:

Movable character

Date 03.02.2015

Current Project Manager - Chris Sand

Aleksander Olsen

Have done:

Will do:

Chris Sand

Have done:

Merge project over to main

Will do:

fix bug in behavior tree

Jonathan Ness

Have done:

setting up server - particle research

Will do:

will do GPU

Marius Rødland

Have done:

server

Will do:

movement of player

Date 04.02.2015

Current Project Manager - Chris Sand

Aleksander Olsen

Have done:

made a minimap

Will do:

logging

Chris Sand

Have done:

Merge project over to main

Will do:

fix bug in behavior tree

Jonathan Ness

Have done:

GPU

Will do:

tower

Marius Rødland

Have done:

movement of player

Will do:

continue with the same

Date 05.02.2015

Current Project Manager - Chris Sand

Aleksander Olsen

Have done:

Research logging

Will do:

same as yesterday

Chris Sand

Have done:

made ai minions work

Will do:

start with other type of ai and get minions to work properly

Jonathan Ness

Have done:

tower, particle effect. Resource of network

Will do:

Marius Rødland

Have done:

Research network. Made the champion walk with point and click

Will do:

More network

Date 06.02.2015

Current Project Manager - Chris Sand

Shared

Will do:

Aleksander Olsen

Have done:

setting up saved files

Will do:

design level

Chris Sand

Have done:

gave minions the possibility to see enemies and to change states

Will do

start with other type of ai and get minions to work properly

Jonathan Ness

Have done:

tower fire

Will do:

work with network

Marius Rødland

Have done:

Movement

Will do:

Movement

Date 10.02.2015

Current Project Manager - Jonathan Ness

Aleksander Olsen

Have done:

Designed level

Will do:

Re-design level

Chris Sand

Have done:

Research on Fog-of-war and collision detection on Minions

Will do

Implement Fog-of-War

Jonathan Ness

Have done:

Messed up for everyone by setting up network.

Will do:

Fix player movement on client-server

Marius Rødland

Have done:

Movement

Will do:

Health on champions

Date 11.02.2015

Current Project Manager - Jonathan Ness

Aleksander Olsen

Have done:

Designed level

Will do:

More level design

Chris Sand

Have done:

Minion targeting and following of enemies. As well as research on Fog-of-War.

Will do

Merge work and fix Fog-of-War

Jonathan Ness

Have done:

Worked on getting client to replicate rotation of character

Will do:

Continue on that fix.

Marius Rødland

Have done:

Implemented health on champions

Will do:

Different attributes on champions. Possibly take damage

Date 12.02.2015

Current Project Manager - Jonathan Ness

Aleksander Olsen

Have done:

Level Design.

Will do:

Redo Minimap

Chris Sand

Have done:

Merge branches and update engine.

Will do

Minion spawn on runtime and follow path to enemy base.

Jonathan Ness

Have done:

Fixed player rotation, tower firing at both client and server, and writing of Confluence.

Will do:

Fix on Tower firing and more work in Confluence and Jira.

Marius Rødland

Have done:

Dealt damage to champions on tower hit.

Will do:

UI on healthbar over champions, and look at lobby.

Date 13.02.2015

Current Project Manager - Jonathan Ness

Aleksander Olsen

Have done:

Worked on Minimap

Will do:

Will continue on Minimap as well as other GUI elements

Chris Sand

Have done:

Minion spawn

Will do

Set up of AI on more units

Jonathan Ness

Have done:

Minimal UI research

Will do:

Work in Confluence.

Marius Rødland

Have done:

UI research

Will do:

Will try implementing UI and possibly give Minions health

Date 17.02.2015

Current Project Manager - Aleksander Olsen

Aleksander Olsen

Have done:

Worked on different solutions for the Minimap.

Will do:

Will continue on Minimap as well as other GUI elements.

Chris Sand

Have done:

Looked at masking for fog of war.

Will do

Minion spawning in right location with right waypoints.

Jonathan Ness

Have done:

Researching different aspects of the game.

Will do:

set up LaTeX and get to know that.

Marius Rødland

Have done:

Minimap

Will do:

Health bar and pause menu

Date 18.02.2015

Current Project Manager - Aleksander Olsen

Aleksander Olsen

Have done:

Made health bar and power bar.

Will do:

Will continue with GUI elements.

Chris Sand

Have done:

Minion spawn in right place and go to right lane.

Will do

More AI.

Jonathan Ness

Have done:

Lekte i LaTeX. Pushet et felles document.

Will do:

Documentation.

Marius Rødland

Have done:

Pause screen. Health bar over champions.

Will do:

More health bar. Gold.

Date 19.02.2015

Current Project Manager - Aleksander Olsen

Aleksander Olsen

Have done:

HUD elements.

Will do:

More HUD elements.

Chris Sand

Have done:

Researching decision tree for AI.

Will do

More on minion AI.

Jonathan Ness

Have done:

LaTeX research.

Will do:

Game lobby.

Marius Rødland

Have done:

Health bars overhead + champion stats.

Will do:

More health bar.

Date 24.02.2015

Current Project Manager - Marius Rødland

Aleksander Olsen

Have done:

Tried migrating

Will do:

Hud, inventory, gold and the rest.

Chris Sand

Have done:

Made a gamestate BP, same as some global. Tower attack the right minions and minions attack eachother.

Will do:

Migrating things.

Jonathan Ness

Have done:

Healthbar, migrating to 4.6

Will do:

Multiplayer

Marius Rødland

Have done:

migrating, healthbars, some network

Will do:

Multiplayer

Date 25.02.2015

Current Project Manager - Marius Rødland

Aleksander Olsen

Have done:

HUD, inventor, gold

Will do:

Hud, inventory, gold and shop.

Chris Sand

Have done:

DID NOT SHOW

Will do:

DID NOT SHOW

Jonathan Ness

Have done:

Player movement with networking.

Will do:

Network playing.

Marius Rødland

Have done:

Player movement with networking.

Will do:

Network playing.

Date 26.02.2015

Current Project Manager - Marius Rødland

Aleksander Olsen

Have done:

Inventory, mostly done!

Will do:

Will start on shop

Chris Sand

Have done:

Fixed bugs on minions, looked at some lobby

Will do:

Post processing with fog of war

Jonathan Ness

Have done:

Healthbar, working properly

Will do:

Spawning on different teams

Marius Rødland

Have done:

Networking, minions fixes

Will do:

Spawning on different teams

27.02.2015

Current Project Manager - Marius Rødland

Aleksander Olsen

Have done:

Made a simple shop

Will do:

Make the shop do something, like buy something.

Chris Sand

Have done:

Fog of war

Will do:

More minions

Jonathan Ness

Have done:

More network, champion spawning

Will do:

More correct spawning

Marius Rødland

Have done:

More network, champion spawning

Will do:

More correct spawning

03.03.2015

Current Project Manager - Chris Sand

Aleksander Olsen

Have done:

Worked with shop

Will do:

Make the shop do something, like buy something.

Chris Sand

Have done:

Put tower in global arrays, made targetpoint that spawns on towers. Made minions follow its own lane, tower, minions all have lanes enum. Found bug (fire before all is initialized)

Will do:

Fix eventuel bugs with minions, try fog of war tutorial, and fix movement of heroe.

Jonathan Ness

Have done:

More network, champion spawning

Will do:

Lobby

Marius Rødland

Have done:

Shop, network

Will do:

Spells

04.03.2015

Current Project Manager - Chris Sand

Aleksander Olsen

Have done:

Make the shop do something, like buy something.

Will do:

sell stuff, make last part.

Chris Sand

Have done:

fixed destroyed actor bug, worked on fix for movement for player.

Will do:

continue on tutorial.

Jonathan Ness

Have done:

trying to get to the guy selling the module

Will do:

talking to the dude. look at the pluggin.

Marius Rødland

Have done:

spells

Will do:

continue on spells.

Date 05.03.2015

Current Project Manager - Chris Sand

Shared

Will do:

Working on getting a playable demo

Aleksander Olsen

Have done:

Shop, get working.

Will do:

get it to work.

Chris Sand

Have done:

fixed so movement is smooth on server.

Will do:

port to champion, work on hiding other components from client via server.

Jonathan Ness

Have done:

Plugin: VaRest, 4.7.2 how to use it.

Will do:

Send login to php script.

Marius Rødland

Have done:

Champion spells

Will do:

Champion spells, help chris on porting movement to champion.

Date 06.03.2015

Current Project Manager - Chris Sand

Shared

Will do:

working on demo

Aleksander Olsen

Have done:

Shop with network

Will do:

get it to work.

Chris Sand

Have done:

Ported movement to character, made very good fog of shadow for players,

Will do:

gone tank fog of war out to other factors.

fix a minion behavior(bug)

trying to hide other components from client via server.

Jonathan Ness

Have done:

Database: User login

Will do:

UI of user login

Marius Rødland

Have done:

Ported movement to champion, spells on champion.

Will do:

continue on it.

Date 10.03.2015

Current Project Manager - Jonathan Ness

Shared

Will do:

working on demo

Aleksander Olsen

Have done:

Fixed inventory so that one can buy stuff.

Will do:

Fix so that items actually do what they state, and selling of item.

Chris Sand

Have done:

Removed memory leak, fixed minion movement issue and fog of war.

Will do:

Change to event driven behavior tree.

Jonathan Ness

Have done:

Tried porting the Lobby module to 4.7

Will do:

Make the lobby module work in 4.7

Marius Rødland

Have done:

Spells

Will do:

More spells

Date 11.03.2015

Current Project Manager - Jonathan Ness

Shared

Will do:

working on demo

Aleksander Olsen

Have done:

User can now sell stuff, but don't get paid

Will do:

Will fix so that we get paid, and using of items.

Chris Sand

Have done:

Decision tree made to event based

Will do:

Spells

Jonathan Ness

Have done:

Renamed elements of the Lobby for clarity.

Will do:

Make the lobby work.

Marius Rødland

Have done:

Spells

Will do:

More Spells

Date 12.03.2015

Current Project Manager - Jonathan Ness

Shared

Will do:

working on demo

Aleksander Olsen

Have done:

Fixed the shop so that selling is properly.

Will do:

Fix the shop up properly.

Chris Sand

Have done:

Holding mouse down will also move character, fixed so that you can click minions. Limited movement during spells.

Will do:

Spells, want ranger spells. And fix minion spawn points.

Jonathan Ness

Have done:

Made some lobby functionality work.

Will do:

Make the lobby functionality actually join players together.

Marius Rødland

Have done:

Spells

Will do:

More spells

Date 17.03.2015

Current Project Manager - Aleksander Olsen

Aleksander Olsen

Have done:

Fixing bug with XP, repairing after renaming folder and changing health variables.

Will do:

Attributes.

Chris Sand

Have done:

DID NOT SHOW

Will do:

DID NOT SHOW

Jonathan Ness

Have done:

Lobby.

Will do:

Lobby

Marius Rødland

Have done:

Help fixing bugs and health.

Will do:

Spells.

Date 18.03.2015

Current Project Manager - Aleksander Olsen

Aleksander Olsen

Have done:

Attributes.

Will do:

Attributes.

Chris Sand

Have done:

Minions attack and on click. Decals.

Will do:

Agro system.

Jonathan Ness

Have done:

Lobby and network.

Will do:

Lobby and network.

Marius Rødland

Have done:

Spells.

Will do:

Spells.

Date 19.03.2015

Current Project Manager - Aleksander Olsen

Aleksander Olsen

Have done:

Attributes.

Will do:

Base attack (auto attack).

Chris Sand

Have done:

Agro system.

Will do:

Agro system. Minion AI.

Jonathan Ness

Have done:

Network.

Will do:

Networking.

Marius Rødland

Have done:

Ranged attack.

Will do:

R attack ranged

Date 24.03.2015

Current Project Manager - Marius Rødland

Aleksander Olsen

Have done:

Design

Will do:

Design

Chris Sand

Have done:

Looked more at fog of war

Will do:

Look more into fog of war

Jonathan Ness

Have done:

Design

Will do:

Joining games by ip

Marius Rødland

Have done:

Design

Will do:

More spells

Date 25.03.2015

Current Project Manager - Marius Rødland

Aleksander Olsen

Have done:

Design

Will do:

Healthbar over buildings

Chris Sand

Have done:

Fog of war, raycasting and stuffs

Will do:

Fog of war, more raytracing

Jonathan Ness

Have done:

Fixed loads of stuffs in relation with packaging of the project

Will do:

Shield stuffs

Marius Rødland

Have done:

Champion spells

Will do:

Champion spells

Date 26.03.2015

Current Project Manager - Marius Rødland

Aleksander Olsen

Have done:

Health replication

Will do:

Animation

Chris Sand

Have done:

Raycasting around the character and cast shadows

Will do:

Copying to the real project, and write how it works down

Jonathan Ness

Have done:

Shields

Will do:

Fix stuffs on shield

Marius Rødland

Have done:

Some spells and design

Will do:

More design and spells

Date 08.04.2015

Current Project Manager - Marius Rødland

Aleksander Olsen

Have done:

Easter

Will do:

Items

Chris Sand

Have done:

Easter

Will do:

Jungle creeps and champion AI

Jonathan Ness

Have done:

Easter

Will do:

Mini Map

Marius Rødland

Have done:

Easter

Will do:

Spells

Date 10.04.2015

Current Project Manager - Marius Rødland

Aleksander Olsen

Have done:

Information about your stats, class, and improved items.

Will do:

More items

Chris Sand

Have done:

Added a check for vision between to target(primarily minions) Moved Alsensing to the Ai controller. Added distance prefere check.

Will do:

Add the junglecreep camp and working on a cleaner Minion blueprint.

Jonathan Ness

Have done:

Minimap

Will do:

Errors related to the packaging of the project

Marius Rødland

Have done:

Spells, and made a new repo with a better git ignore file

Will do:

Spells

Date 14.04.2015

Current Project Manager - Chris Sand

Aleksander Olsen

Have done:

Long meeting, fixed bugs with the group

Will do:

Items and stuff

Chris Sand

Have done:

customize behavior tree move to.

Will do:

find more bugs in minions add jungle creep

Jonathan Ness

Have done:

win condition

Will do:

fix Ui after win condition

Marius Rødland

Have done:

Started writing on the project

Will do:

spells, more writing

Date 15.04.2015

Current Project Manager - Chris Sand

Shared

Will do:

Fix network(last try)
writing reports

Aleksander Olsen

Have done:

made items done, fixing icons

Will do:

iconsa til itemsa.

Chris Sand

Have done:

Animation, fixed bugg in choose best target, where closest target didnt always return true when first target was set. testing network

Will do:

basic attack for champion, network, writing report and putting out creep minions

Jonathan Ness

Have done:

testing network

Will do:

UI

Marius Rødland

Have done:

testing network, changing element.

Will do:

more spells, writing report

Date 17.04.2015

Current Project Manager - Chris Sand

Aleksander Olsen

Have done:

Will do:

Chris Sand

Have done:

Junglee creep, animation for death. Turn around when basic attacking, redesigned behavior tree. written report.

Will do:

fix some minor bug in behavior tree, some minor bugs in jungle creep, more animation and more typing. Fix icons to spells

Jonathan Ness

Have done:

fixed UI.

Will do:

Writing report.

Marius Rødland

Have done:

Spells, done.

Will do:

report writing

Date 21.04.2015

Current Project Manager - Jonathan Ness

Aleksander Olsen

Have done:

Work on writing the thesis

Will do:

Write more on the thesis as well as finish up icons in the shop.

Chris Sand

Have done:

Finished jungle creeps, attack animation based on attack speed, and basic attack range. Also wrote on the thesis.

Will do:

Will look on animation for spells and write on the thesis.

Jonathan Ness

Have done:

Wrote on the thesis.

Will do:

Write more on the thesis.

Marius Rødland

Have done:

Wrote on the thesis.

Will do:

Finish up spells with cooldown and mana cost, and write more on the thesis

Date 22.04.2015

Current Project Manager - Jonathan Ness

Aleksander Olsen

Have done:

More work on items and icons.

Will do:

Will finish up with icons on items, will write on thesis if time.

Chris Sand

Have done:

Wrote about collision and some of the functionality in Unreal Engine.

Will do:

Will write more about collision, and about AI

Jonathan Ness

Have done:

Wrote on the thesis about User Data and security.

Will do:

Write more on the thesis.

Marius Rødland

Have done:

Fixed spells with cost.

Will do:

Finish up spells with cooldown. Writes in report if given time.

Date 23.04.2015

Current Project Manager - Jonathan Ness

Aleksander Olsen

Have done:

Finished up items with icons, icons for gold

Will do:

Write on theiss, write about items in Confluence.

Chris Sand

Have done:

Fixed structure of enemies. Using a baseclass for enemies, junglecreeps now work

Will do:

Will make weapon for mage, and write about physics in thesis.

Jonathan Ness

Have done:

Wrote and researched thesis.

Will do:

Write more on the thesis. Visual effects for spells.

Marius Rødland

Have done:

Fixed Cooldowns on spells, started looking at the last warrior spell. Wrote a little on thesis.

Will do:

Will finish the last spell and write thesis.

Date 24.04.2015

Current Project Manager - Chris Sand

Shared

Will do:

Playtesting

Aleksander Olsen

Have done:

Writes about items in confluence

Will do:

More rapport.

Shop, making so you can only buy inside spawn.

Chris Sand

Have done:

Making pipeline between blender and unreal engine 4, made animation for mage. Modeled mage and starting on ranger animation and model.

Will do:

add autoattack range checks and playtesting. continue on ranger. Trying custome animation done in blender.

Jonathan Ness

Have done:

Daily scrum log to latex

Will do:

Visual effect for spells.

Marius Rødland

Have done:

Done with all spells. Switching team(ui)

Will do:

Working on recent bugg with playing with 2 clients because of updates.

Date 28.04.2015

Current Project Manager - Aleksander Olsen

Shared

Will do:

Write the report.

Aleksander Olsen

Have done:

Done with spawn area, heal and shop.

Will do:

Write about GUI.

Chris Sand

Have done:

Research game balancing with A.I.

Will do:

Write about balancing.

Jonathan Ness

Have done:

Set-up report layout.

Will do:

Write in the report.

Marius Rødland

Have done:

Set-up report layout.

Will do:

Write more on introduction.

Date 29.04.2015

Current Project Manager - Aleksander Olsen

Shared

Will do:

Write the report.

Aleksander Olsen

Have done:

Wrote about GUI.

Will do:

Write more about GUI.

Chris Sand

Have done:

Wrote about AI.

Will do:

Write more about behavior tree.

Jonathan Ness

Have done:

Fix bug i packaging the project.

Will do:

Write about bugs concerning release of project.

Marius Rødland

Have done:

Wrote about introduction.

Will do:

Write more introduction and spells.

Date 30.04.2015

Current Project Manager - Aleksander Olsen

Shared

Will do:

Write the report.

Aleksander Olsen

Have done:

Fixing spawn with shop.

Will do:

Get spawn fix and write about GUI.

Chris Sand

Have done:

Wrote about minions, jungle creeps and AI.

Will do:

Write about behavior tree and animation.

Jonathan Ness

Have done:

Wrote about networking. Try to fix buildings spawn at 0.

Will do:

Write more.

Marius Rødland

Have done:

Done with introduction and started spells.

Will do:

Write more about spells.

Date 1.05.2015

Current Project Manager - Aleksander Olsen

Shared

Will do:

Write the report.

Aleksander Olsen

Have done:

Fix spawn area and wrote about GUI

Will do:

Fix map with added spawn area functions.

Chris Sand

Have done:

Will do:

Jonathan Ness

Have done:

Wrote about deployment and looking through for errors.

Will do:

Fix main base that spawn at zero.

Marius Rødland

Have done:

Wrote about spells and looking through for errors.

Will do:

Write more about spells and technology.

Date 05.05.2015

Current Project Manager - Marius Rødland

Shared

Will do:

More bug fix'es

Aleksander Olsen

Have done:

Bug fix'es on the game

Will do:

Write more on GUI and shop

Chris Sand

Have done:

Bug fix'es on the game

Will do:

Writing about animations

Jonathan Ness

Have done:

Bug fix'es on the game

Will do:

Instructions in the game

Marius Rødland

Have done:

Bug fix'es on the game

Will do:

Will write about how blueprints works

Date 06.05.2015

Current Project Manager - Marius Rødland

Shared

Will do: Fix more bugs in the code around 1500

Aleksander Olsen

Have done:

Wrote more about GUI and joined the game testing

Will do:

Write more about the GUI and shop

Chris Sand

Have done:

Wrote about AI controller, and blueprints

Will do:

Will write about the behavior tree and animation for champion

Jonathan Ness

Have done:

Wrote about Ue and how it was to use it

Will do:

Will write more about the different functionality in ue4

Marius Rødland

Have done:

Wrote about Blueprints how it works

Will do:

Will write more about the blueprints

Date 07.05.2015

Current Project Manager - Marius Rødland

Aleksander Olsen

Have done:

Wrote about the GUI and inventory, and did more bug fixing

Will do:

Will write more about the inventory, then the shop and if time the map.

Chris Sand

Have done:

Wrote about the ai controller, generally about skeletons, mesh and so on.

Will do:

Behavior tree, then some about navmesh.

Jonathan Ness

Have done:

Bug fixing, wrote some about the different function types in Blueprint.

Will do:

Write in the conclusion and discussion part.

Marius Rødland

Have done:

Wrote about things we use in blueprints land.

Will do:

Will write about the testing we did.

Date 08.05.2015

Current Project Manager - Marius Rødland

Aleksander Olsen

Have done:

Wrote about inventory, started on the shop with pictures.

Will do:

Will continue on the shop, and map if time

Chris Sand

Have done:

Wrote about behavior tree

Will do:

Finish behavior tree, nav mesh and pathfinding.

Jonathan Ness

Have done:

Made Terminology an appendix, and starting to write about future development

Will do:

Will write more about future development

Marius Rødland

Have done:

Testing with the kids.

Will do:

User feedback and balancing.

Date 12.05.2015

Current Project Manager - Chris Sand

Aleksander Olsen

Have done:

allocated time to reviewing own text

Will do:

Future plans, writing more about items

Chris Sand

Have done:

Writing more about PhysX, blender.

Will do:

Writing more about how we used stuff in our project and blender pipeline

Jonathan Ness

Have done:

Writed about intruduction, rewriting sentences

Will do:

reviewing requirment and design.

Marius Rødland

Have done:

Writed about testing on saturday.

Will do:

Testing about game.

Date 13.05.2015

Current Project Manager - Chris Sand

Aleksander Olsen

Have done:

writed about ui, shop and inventory.

Will do:

Writing more about items and map

Chris Sand

Have done:

Added more refs to images, writed about minions and heritage, fixed some structures. added more to physx and stuff.

Will do:

Write animation, behavior tree, and physx done.

Jonathan Ness

Have done:

reviewed and maked changes. Added more text to do.

Will do:

Reviewing implementation.

Marius Rødland

Have done:

Almost done with balancing

Will do:

Re-writing technical design.

I Project NORS: Meeting Notes

The following is the short snippets of review written after all Weekly Scrum Meetings, Meetings with our supervisor and other more important meetings.

[Meeting 1 \(08.01.15\):](#)

[Summary](#)

[Meeting 2 \(11.01.15\):](#)

[Summary](#)

[Meeting 3 \(12.01.15\):](#)

[Summary:](#)

[Meeting with Simon 1 \(1.19.15\):](#)

[Summary:](#)

[Meeting with Simon 2\(26.01.15\):](#)

[Summary:](#)

[Weekly Scrum Meeting\(26.01.15\):](#)

[Meeting with Simon 3\(02.02.15\):](#)

[Summary:](#)

[Weekly scrum meeting\(02.02.15\)](#)

[Meeting with Simon 4\(09.02.15\):](#)

[Summary:](#)

[Weekly scrum meeting\(09.02.15\)](#)

[Meeting with Simon 5\(16.02.15\):](#)

[Summary:](#)

[Weekly scrum meeting\(16.02.15\)](#)

[Meeting with Simon 6\(23.02.15\):](#)

[Summary:](#)

[Weekly scrum meeting\(23.02.15\)](#)

[Meeting with Simon 7\(02.03.15\):](#)

[Summary:](#)

[Weekly scrum meeting\(02.03.15\)](#)

[Meeting with Simon 8\(09.03.15\):](#)

[Summary:](#)

[Weekly scrum meeting\(09.03.15\)](#)

[Meeting with Simon 9\(16.03.15\):](#)

[Summary:](#)

[Weekly scrum meeting\(16.03.15\)](#)

[Meeting with Simon 10\(23.03.15\):](#)

[Summary:](#)

[Weekly scrum meeting\(23.03.15\)](#)

[Extraordinary Pre Easter Scrum Planning Meeting \(26.03.2015\)](#)

[Meeting with Simon 11\(07.04.15\):](#)

[Weekly scrum meeting\(07.04.15\):](#)

[Meeting with Simon 12\(13.04.15\):](#)

[Weekly scrum meeting\(13.04.15\):](#)

[Weekly scrum meeting\(13.04.15\):](#)

[Weekly scrum meeting\(27.04.15\):](#)

[Meeting with Simon 13\(04.05.15\):](#)

[Weekly scrum meeting\(04.05.15\):](#)

[Meeting with Simon 13\(11.05.15\):](#)

[Summary:](#)

[Weekly scrum meeting\(11.05.15\):](#)

Meeting 1 (08.01.15):

- Unreal Engine?
- MOBA themed.
 - New minions behavior.
 - Cars theme?
- Mobile game.
- Genre
 - MOBA
 - FPS
 - Platform

Summary

Since we did not have a group, we got together and made a group. We then discussed what we want to do for our "bacheloroppgave". Suggestions to use unreal engine were made, and a MOBA themed game was the most positive suggestion. Other suggestions were a FPS game or a platform puzzle game, but no specific ideas came to mind.

So we decided we want to make a MOBA game using unreal engine.

We decided also to arrange a meeting with Simon to asses the feasibility and possibility to do this for our "bacheloroppgave". We will meet with him on Monday (12.01.15).

Meeting 2 (11.01.15):

- More defining of group rules
- Brainstorming on MOBA theme
 - Viking, saxon, english men themed.
 - The vikings against the saxon, with the english men as neutral as mobs.
 - Take over bases for currency.
 - The elements.
 - At the start of the game, choose your fighting style (ranged, melee), and then what element you want to control.
 - Elements and their use affecting the map and terrain
 - fire, water, lightning, ice, metal, etc.
 - Set fire to the grass
 - Freeze water, make it slippery
 - Water on earth to make muddy and slowing.
 - Lightning on water to stun area.
- Cars.
 - Mecha?
 - Vrom?

Summary

Rules have been defined more, see rule document

Agreed upon heavy use of elements in the game environment and it's effects.

So far the viking, saxons and english men theme is the one we are currently deciding on. Not a final decision though.

Meeting 3 (12.01.15):

- Grade: B
- Design document
 - What is different from other MOBAs?
 - Currency
 - Control of elements
- Business model
 - Friction
- Development process
- Balancing
 - A.I.
 - Possible game testing.

What to do this week:

- Design document
 - Research MOBA games. (CHRIS)
 - Create a chart.
 - Common, uncommon, rare.
 - Development method (Decided on scrum) (MARIUS)
 - Deciding on map styles, general gameplay.
 - Business model. (JONATHAN)
 - Play around with the unreal engine (ALL)
 - Setting up repository (ALEKS)

Summary:

We had a meeting with Simon today. We discussed the feasibility of our project and what to do to earn a B. We need to have a way to “measure” success of our game with our changes, over other MOBA games. We need to research other MOBA games and make a chart where we classify the different types. Then we have to place our game in that chart and discuss our game contra the others. We also discussed including a business model if we want to commercialize our project.

This week we will start with design document. By the end of this work day, we will have sent a mail to Simon with our group, short name (for website) and so that we can have a key for the unreal engine. For documentation, we will use a combination of Confluence, Jira and LaTeX. We started to use Toggl for time tracking.

We have decided to use the short name the_nors.

Meeting with Simon 1 (1.19.15):

- Talk about using LaTeX for the bachelors.
 - There is a Repo on BitBucket and instructions on Fronter
 - We can start and get familiar with LaTeX now.
- Info on the Project setup.
 - Write about section rather than steps
 - Topic vs narrative (Don't write the steps in a tutorial)
 - Things you'd want to know before you started
- Check out Game logging in Unreal
- Technology, aesthetics, story, mechanics
- Possible to find Artists, playtesters, etc. from GlobalGameJam
 - Need a basic webpage by end of week.

Summary:

We should look into LaTeX, setup a basic webpage with a Google form for alpha/betatesters and have fun on the Global Game Jam. Game logging should be figured out for Unreal Engine and we should log everything from the very start.

Meeting with Simon 2(26.01.15):

- Creating company as external in project agreement.
 - With shares?
- Confluence
- Jira

Summary:

We talked about the project agreement and decided we want try to make an own "company" so that the agreement is with our "company" and us students as individuals. Therefore we will own the rights to our game, and not to an external party.

We also got a confluence and jira group to use in our project.

Weekly Scrum Meeting(26.01.15):

- Make map.
- Character with movement.
- Towers.
- Basic AI (Minions)
- Menu/interface

Meeting with Simon 3(02.02.15):

- Look at attaching camera to objects.
- Game testing survey form.
 - Different for each build
- Unreal logging system.
- Config files
- Project agreement:
 - Create a “company”.
 - 600 shares each
- What to write in the bacheloroppgave:
 - Bugs
 - Decisions
 - Logging

Summary:

We shall sign an agreement by next monday meeting where we establish a “company”. This “company” will hold the IP for the project.

We will make a logging system to make issue tracking easier.

We will start using confluence and jira.

Weekly scrum meeting(02.02.15)

- Minions AI
- Elements
- Simple network
- Simple logging
- Setup and start with confluence and jira

Meeting with Simon 4(09.02.15):

- Have the contract
- Write agreement about voting/sharing
- Research networking
- Collision detection minions
-

Summary:

We will sign the agreement and decide on voting within our group.

Weekly scrum meeting(09.02.15)

- Make networking work.
- More AI
- Level design
- Movement (champion/camera) / champion stats.

Meeting with Simon 5(16.02.15):

- Map
 - Show math
 - 24800 total map
 - 256 minimap
 - 20 degree
 - find Z
 - $(f/128 = z+f/12400)$
- Lighting
 - single light with mask
 - vs
 - multiple dynamic light
 - fog of war
- Decisions

Summary:

Weekly scrum meeting(16.02.15)

- Minions attack other minions and towers.
- Towers attack enemies.
- GUI (health, inventory, map etc.)
- Hero stats
- Set up LaTeX

Meeting with Simon 6(23.02.15):

- Name of entity
 - NORS

Summary:

We had a not so good work week last week. Jonathan was working at Huset because of the party week. Aleksander was sick half the week. Marius and Chris went away to a cabin on friday.

Weekly scrum meeting(23.02.15)

- Migrate to 4.6
 - Voted no 3 against 1 because we couldn't simply copy files to old version.
 - Stick to 4.7 and latest release, hoping unstable functions don't get removed.
- Make a demo
 - Make player attack
 - Minions attack
 - Tower attack
- HUD
 - Inventory
 - Display money
- Xp and money from minion/tower

Meeting with Simon 7(02.03.15):

- Digging in to network
 - not sending rotation (your own)
 - check if rotation is set to 0 or not set at all.
 - technique for debugging
- Load and save game
 - Load stuff from config file(items)
 - For debugging
- Section on security.
- Buying premade stuff(plugin)
 - CEO:
 - The faster the better (buy it)
 - Educator:
 - Want you to learn stuff
 - useful if you can learn from it
 - Supervisor:
 - For good grade need to show level of understanding
 - What can you write about
 - Show that you can learn

Weekly scrum meeting(02.03.15)

- Movement over network fix
- Shop
 - buy and put in inventory
 - item buffs
- Basic attack from champion
- Minion gameplay
- fix

Meeting with Simon 8(09.03.15):

- Progress report
- Website / SQL server.

Weekly scrum meeting(09.03.15)

- Make a playable demo
 - Lobby
 - Minion behavior
 - Spells
 - Items

Weekly scrum meeting(16.03.15)

- Spells
- Highlight selection
- Animation
- Minion do damage
- Lobby
 - Type select
 - elements etc.

Meeting with Simon 10(23.03.15):

- Networking trouble
 - use wireshark?
 - Port scanning
 - Always have a lobby
- Minion aggro
- Design
 - Buy assets?
- Balancing
 - Do big jumps
 - Double/half
 - Define success
 - Result & conclusion
 - Stopping criteria:
 - Time
 - Until close enough
 - Write how and what
- Health regen on tower not at frontline
 - Can repair towers if attacked and not at frontline
 - Visually with workers/repairers
- **Next week meeting: Tuesday 7th**

Weekly scrum meeting(23.03.15)

- Continue from last week
- Next meeting with Simon will be 07.04, because of easter.

Extraordinary Pre Easter Scrum Planning Meeting (26.03.2015)

- We will have to decide what we are going to do through the easter holiday.
- We will all have to write in confluence about what we have been working with so far.
 - **Chris:** Minions and Jungle creeps.
 - **Marius:** Make the rest of the spells and put visual effects on them.
 - **Jonathan:** Making of the jungle. Changing of environment.
 - **Aleksander:** Making of items. We will go for different "gear" pieces, like "Boots of Speed", "Legs of Damage" etc.

Meeting with Simon 11(07.04.15):

- Discussing easter

Weekly scrum meeting(07.04.15):

- Spells
- Minions
- Jungle Creeps
- Networking
- Items

Meeting with Simon 12(13.04.15):

- Write and send to Simon feedback.

Weekly scrum meeting(13.04.15):

- Write report
- Items
- Minions
- Networking
- Win scenario + respawn

Weekly scrum meeting(13.04.15):

- Write report
- Items
- Minions
- Networking
- Win scenario + respawn

Weekly scrum meeting(27.04.15):

We finished networking.

- Spawn area
- Write report

Meeting with Simon 13(04.05.15):

- Testing: 18:00 - 05.05.15 Tuesday
- Fix bugs today, then only work with report.

Meeting with Simon 13(11.05.15):

- Questions about writing style

Summary:

Simon talked about how to write the bachelor thesis. He looked quickly through the report and gave a few notes about what we should do and what we shouldn't do.

Weekly scrum meeting(11.05.15):

- Write everything to Thursday.
- On Thursday we will look through together for errors and changes that needs to be done.

J Project NORS: Toggle Time Tracking

The following is monthly printout of our tracked time from Toggl.

Detailed report



2015-01-01 - 2015-01-31

Total 219 h 26 min

Date	Description	Duration	User
01-12	Project Managing as Project Manager :D Bachelor NORS	0:28:23 14:56-15:24	Spoki0
01-12	searching for resources Bachelor NORS	0:27:02 14:58-15:25	Kress92 Cs
01-13	Project plan Bachelor NORS - [mobile]	3:45:00	Rodland92
01-13	Setting up UE4, Testing and watching tutorials Bachelor NORS	6:00:00 09:00-15:00	D3stny
01-13	Setting up UE4 Bachelor NORS	0:37:00 09:10-09:47	Spoki0
01-13	Research on Business Model Bachelor NORS	2:32:12 09:48-12:20	Spoki0
01-13	Project plan Bachelor NORS - [mobile]	1:15:52 14:31-15:46	Rodland92
01-13	Finding out how we can make diagrams of generic play styles Bachelor NORS	0:23:30 15:56-16:19	Kress92 Cs
01-14	Daily Scrum Meeting Bachelor NORS	0:20:00 09:00-09:20	Spoki0
01-14	Scrum meeting og project plan Bachelor NORS - [mobile]	5:46:48 09:05-14:52	Rodland92
01-14	Finding out how we can make diagrams of generic play styles Bachelor NORS	0:00:04 09:11-09:11	Kress92 Cs
01-14	Reading resources Bachelor NORS	0:20:00 09:20-09:40	Spoki0
01-14	Møte med Mariusz, fix groups Bachelor NORS	0:20:00 10:10-10:30	Spoki0
01-14	Research Bachelor NORS	4:10:46 10:41-14:52	Spoki0
01-14	finding out where our game is on the charts Bachelor NORS	1:07:30 10:59-12:07	Kress92 Cs
01-14	finding out where our game is on the charts Bachelor NORS	0:07:58 12:07-12:15	Kress92 Cs
01-14	finding out where our game is on the charts Bachelor NORS	0:28:59 12:23-12:51	Kress92 Cs
01-15	Daily Scrum meeting Bachelor NORS	0:20:00 09:00-09:20	Rodland92
01-15	Daily Scrum Meeting Bachelor NORS	0:30:00 09:00-09:30	Spoki0

01-15	Project plan Bachelor NORS - [mobile]	5:26:20 09:21-14:47	Rodland92
01-15	Working on Project Plan Bachelor NORS	0:52:00 09:37-10:29	Spoki0
01-15	finding out where our game is on the charts Bachelor NORS	3:34:00 10:26-14:00	Kress92 Cs
01-15	Working on Project Plan Bachelor NORS	1:50:44 10:42-12:33	Spoki0
01-15	Making of Gantt diagram Bachelor NORS	1:42:57 12:33-14:16	Spoki0
01-15	Setting up UE4 project with git Bachelor NORS	2:27:49 15:05-17:33	D3stny
01-15	Basic learning of Unreal engine 4 Bachelor NORS	2:46:00 17:25-20:11	Kress92 Cs
01-15	Setting up UE4 project with git Bachelor NORS	1:16:07 19:21-20:38	D3stny
01-16	Scrum meeting Bachelor NORS - [mobile]	0:24:31 09:12-09:36	Rodland92
01-16	Scrum meeting Bachelor NORS	0:23:58 09:12-09:36	D3stny
01-16	Finishing project plan Bachelor NORS - [mobile]	0:22:24 09:36-09:59	Rodland92
01-16	learning to create AI bot Bachelor NORS	1:12:43 09:49-11:02	Kress92 Cs
01-16	Setting up UE4 project with git (it takes a really long time) Bachelor NORS	1:47:35 10:41-12:28	D3stny
01-16	Project plan Bachelor NORS - [mobile]	1:35:11 12:03-13:39	Rodland92
01-16	Daily Scrum Meeting Bachelor NORS	0:30:00 21:00-21:30	Spoki0
01-19	Meeting with Simon Bachelor NORS	0:53:00 09:00-09:53	Spoki0
01-19	Meeting with Simon Bachelor NORS	0:45:00 09:00-09:45	Rodland92
01-19	Meeting with Simon Bachelor NORS	0:50:00 09:00-09:50	D3stny
01-19	Weekly Scrum Meeting Bachelor NORS	1:27:10 09:54-11:21	Spoki0
01-19	Design Document Bachelor NORS	0:58:05 10:05-11:03	Rodland92
01-19	Continue set up UE4 with git and test if working Bachelor NORS	0:43:00 10:17-11:00	D3stny
01-19	Looking at different resources on the web and how to implement them in our project Bachelor NORS	2:37:00 10:47-13:24	Kress92 Cs

01-19	More on Design Document Bachelor NORS	2:30:00 13:30-16:00	Rodland92
01-19	Scrum meeting + Design document + Project Bachelor NORS	4:30:00 13:30-18:00	D3stny
01-19	Makes Webpage Bachelor NORS	2:42:59 13:32-16:15	Spoki0
01-19	Git n stuff Bachelor NORS	0:37:00 16:30-17:07	Spoki0
01-20	Daily Scrum Meeting Bachelor NORS	0:15:00 09:00-09:15	Spoki0
01-20	Scrum meeting Bachelor NORS	0:15:00 09:00-09:15	D3stny
01-20	Design document Bachelor NORS - [mobile]	3:08:00 09:22-12:30	Rodland92
01-20	Learning UE4 Bachelor NORS	2:16:32 10:33-12:50	Spoki0
01-20	The 4 elements of a game Bachelor NORS	3:40:00 11:04-14:44	Kress92 Cs
01-20	Played around in UE4 Bachelor NORS	2:00:00 14:00-16:00	Rodland92
01-20	Learning UE4 Bachelor NORS	2:55:53 14:06-17:02	Spoki0
01-20	Getting to know UE4 Bachelor NORS	2:45:30 15:07-17:53	D3stny
01-20	Learning UE4 Bachelor NORS	2:09:45 19:47-21:57	Spoki0
01-20	Daily scrum meeting Bachelor NORS	0:15:00 21:00-21:15	Rodland92
01-20	Learning UE4 Bachelor NORS	0:46:56 22:07-22:54	Spoki0
01-21	Daily Scrum Meeting Bachelor NORS	0:30:00 09:00-09:30	Spoki0
01-21	Scrum meeting Bachelor NORS	0:25:00 09:00-09:25	D3stny
01-21	LaTeX Bachelor NORS - [mobile]	4:01:32 09:45-13:46	Rodland92
01-21	Making diagram Bachelor NORS	2:27:59 09:46-12:14	Kress92 Cs
01-21	Fixing UE4 after peeps made it crash Bachelor NORS	1:19:21 11:37-12:56	Spoki0
01-21	Working with UE4 and git support Bachelor NORS	2:19:18 12:16-14:35	D3stny
01-21	Learning UE4 Bachelor NORS	1:54:00 12:56-14:50	Spoki0
01-21	Learning about NavMesh and bot spawning Bachelor NORS	1:38:00 15:26-17:04	Kress92 Cs

01-21	Working with UE4 and git support	0:54:14	D3stny
	Bachelor NORS	17:04-17:58	
01-21	Daily scrum meeting	0:15:00	Rodland92
	Bachelor NORS	21:00-21:15	
01-22	Daily Scrum Meeting	0:15:00	Spoki0
	Bachelor NORS	09:00-09:15	
01-22	Daily scrum meeting	0:16:00	Rodland92
	Bachelor NORS	09:02-09:18	
01-22	Daily scrum meeting	0:16:00	Rodland92
	Bachelor NORS	09:02-09:18	
01-22	Daily scrum meeting	0:23:00	D3stny
	Bachelor NORS	09:07-09:30	
01-22	Design document	3:39:06	Rodland92
	Bachelor NORS	09:22-13:01	
01-22	Writing about mechanics	4:16:00	Kress92 Cs
	Bachelor NORS	10:18-14:34	
01-22	Div document editing	4:05:00	Spoki0
	Bachelor NORS	11:25-15:30	
01-22	Writing lore for design document	1:42:56	D3stny
	Bachelor NORS	12:21-14:04	
01-22	LaTeX	2:17:28	Rodland92
	Bachelor NORS - [mobile]	13:03-15:20	
01-23	Writing about mechanics	5:04:00	Kress92 Cs
	Bachelor NORS	10:14-15:18	
01-26	Meeting with Simon	1:08:53	Rodland92
	Bachelor NORS - [mobile]	09:03-10:12	
01-26	Meeting with Simon	1:10:00	Spoki0
	Bachelor NORS	09:05-10:15	
01-26	Weekly Scrum Meeting	0:57:00	Spoki0
	Bachelor NORS	11:20-12:17	
01-26	Daily scrum meeting	1:06:53	D3stny
	Bachelor NORS	11:20-12:27	
01-26	Weekly scrum meeting	0:45:00	Rodland92
	Bachelor NORS - [mobile]	11:30-12:15	
01-26	Making map	1:20:26	Spoki0
	Bachelor NORS	12:18-13:38	
01-26	Setting up project and creating map level	1:19:40	D3stny
	Bachelor NORS	12:29-13:49	
01-26	Setting up simple map	1:18:40	Rodland92
	Bachelor NORS - [mobile]	12:31-13:49	
01-26	Finishing up project plan	1:03:14	Spoki0
	Bachelor NORS	13:54-14:57	
01-26	Discussing project plan	1:00:50	D3stny
	Bachelor NORS	13:56-14:57	
01-26	Final touches on Project plan and simple map	1:00:00	Rodland92
	Bachelor NORS	14:00-15:00	

01-26	Meeting with Simon Bachelor NORS	1:00:00 21:15-22:15	D3stny
01-27	Daily scrum meeting Bachelor NORS	0:06:00 09:00-09:06	Rodland92
01-27	Daily Scrum Meeting Bachelor NORS	0:08:00 09:00-09:08	Spoki0
01-27	Starting to look at camera movement Bachelor NORS	0:54:00 09:11-10:05	Rodland92
01-27	Camera movement Bachelor NORS	1:42:00 10:12-11:54	Rodland92
01-27	Trying to get bot spawn with AI Bachelor NORS	2:11:53 10:29-12:41	Kress92 Cs
01-27	Make pause menu Bachelor NORS	0:26:28 13:08-13:35	D3stny
01-27	Make pause menu Bachelor NORS	2:14:31 14:24-16:38	D3stny
01-27	Trying to get bot spawn with AI Bachelor NORS	5:20:00 15:24-20:44	Kress92 Cs
01-27	Intro Menu UE4 Bachelor NORS	2:20:00 20:53-23:13	Spoki0
01-28	Daily scrum Bachelor NORS	0:10:00 09:00-09:10	D3stny
01-28	Daily Scrum meeting Bachelor NORS	0:11:00 09:02-09:13	Rodland92
01-28	Daily Scrum Meeting Bachelor NORS	0:11:00 09:04-09:15	Spoki0
01-28	Try to find out how to make the AI move towards a point Bachelor NORS	6:58:00 09:11-16:09	Kress92 Cs
01-28	Camera movements Bachelor NORS	1:00:24 09:37-10:37	Rodland92
01-28	Moving AI to spawnPoints Bachelor NORS	2:57:57 09:50-12:48	Kress92 Cs
01-28	Make pause menu Bachelor NORS	0:08:49 10:13-10:21	D3stny
01-28	Finishing up project plan Bachelor NORS	5:19:15 10:53-16:12	Spoki0
01-28	Looked at networking Bachelor NORS	1:53:12 11:07-13:00	Rodland92
01-28	Getting a movable character Bachelor NORS	1:45:27 13:01-14:46	Rodland92
01-28	Moving AI to spawnPoints Bachelor NORS	4:16:56 14:27-18:44	Kress92 Cs
01-29	Daily scrum Bachelor NORS	0:30:00 09:00-09:30	D3stny

01-29	Daily Scrum Meeting Bachelor NORS	0:31:00 09:02-09:33	Spoki0
01-29	Daily Scrum meeting Bachelor NORS	0:22:00 09:08-09:30	Rodland92
01-29	waypoint and looking through events Bachelor NORS	4:54:00 09:18-14:12	Kress92 Cs
01-29	Getting a movable character Bachelor NORS	0:22:00 09:30-09:52	Rodland92
01-29	spawning ai Bachelor NORS	2:41:12 09:35-12:17	Kress92 Cs
01-29	waypoint and looking through events Bachelor NORS	3:17:00 19:14-22:31	Kress92 Cs
01-29	Intro Menu UE4 Bachelor NORS	0:54:25 22:19-23:13	Spoki0
01-30	Daily Scrum meeting Bachelor NORS	0:10:00 09:00-09:10	Rodland92
01-30	Daily scrum Bachelor NORS	0:11:45 09:03-09:15	D3stny
01-30	Getting a movable character Bachelor NORS	1:30:00 09:12-10:42	Rodland92
01-30	Work more on behavior tree Bachelor NORS	5:54:00 09:21-15:15	Kress92 Cs
01-30	Getting a movable character Bachelor NORS	4:16:15 10:43-14:59	Rodland92
01-30	Intro Menu UE4 Bachelor NORS	2:04:08 10:58-13:02	Spoki0
01-30	Make pause menu Bachelor NORS	0:49:45 13:29-14:19	D3stny
01-30	Make pause menu Bachelor NORS	1:49:21 15:56-17:45	D3stny
01-30	Make pause menu Bachelor NORS	3:30:35 18:11-21:42	D3stny

Detailed report



2015-01-01 - 2015-01-31

Total 219 h 26 min

Date	Description	Duration	User
01-12	Project Managing as Project Manager :D Bachelor NORS	0:28:23 14:56-15:24	Spoki0
01-12	searching for resources Bachelor NORS	0:27:02 14:58-15:25	Kress92 Cs
01-13	Project plan Bachelor NORS - [mobile]	3:45:00	Rodland92
01-13	Setting up UE4, Testing and watching tutorials Bachelor NORS	6:00:00 09:00-15:00	D3stny
01-13	Setting up UE4 Bachelor NORS	0:37:00 09:10-09:47	Spoki0
01-13	Research on Business Model Bachelor NORS	2:32:12 09:48-12:20	Spoki0
01-13	Project plan Bachelor NORS - [mobile]	1:15:52 14:31-15:46	Rodland92
01-13	Finding out how we can make diagrams of generic play styles Bachelor NORS	0:23:30 15:56-16:19	Kress92 Cs
01-14	Daily Scrum Meeting Bachelor NORS	0:20:00 09:00-09:20	Spoki0
01-14	Scrum meeting og project plan Bachelor NORS - [mobile]	5:46:48 09:05-14:52	Rodland92
01-14	Finding out how we can make diagrams of generic play styles Bachelor NORS	0:00:04 09:11-09:11	Kress92 Cs
01-14	Reading resources Bachelor NORS	0:20:00 09:20-09:40	Spoki0
01-14	Møte med Mariusz, fix groups Bachelor NORS	0:20:00 10:10-10:30	Spoki0
01-14	Research Bachelor NORS	4:10:46 10:41-14:52	Spoki0
01-14	finding out where our game is on the charts Bachelor NORS	1:07:30 10:59-12:07	Kress92 Cs
01-14	finding out where our game is on the charts Bachelor NORS	0:07:58 12:07-12:15	Kress92 Cs
01-14	finding out where our game is on the charts Bachelor NORS	0:28:59 12:23-12:51	Kress92 Cs
01-15	Daily Scrum meeting Bachelor NORS	0:20:00 09:00-09:20	Rodland92
01-15	Daily Scrum Meeting Bachelor NORS	0:30:00 09:00-09:30	Spoki0

01-15	Project plan Bachelor NORS - [mobile]	5:26:20 09:21-14:47	Rodland92
01-15	Working on Project Plan Bachelor NORS	0:52:00 09:37-10:29	Spoki0
01-15	finding out where our game is on the charts Bachelor NORS	3:34:00 10:26-14:00	Kress92 Cs
01-15	Working on Project Plan Bachelor NORS	1:50:44 10:42-12:33	Spoki0
01-15	Making of Gantt diagram Bachelor NORS	1:42:57 12:33-14:16	Spoki0
01-15	Setting up UE4 project with git Bachelor NORS	2:27:49 15:05-17:33	D3stny
01-15	Basic learning of Unreal engine 4 Bachelor NORS	2:46:00 17:25-20:11	Kress92 Cs
01-15	Setting up UE4 project with git Bachelor NORS	1:16:07 19:21-20:38	D3stny
01-16	Scrum meeting Bachelor NORS - [mobile]	0:24:31 09:12-09:36	Rodland92
01-16	Scrum meeting Bachelor NORS	0:23:58 09:12-09:36	D3stny
01-16	Finishing project plan Bachelor NORS - [mobile]	0:22:24 09:36-09:59	Rodland92
01-16	learning to create AI bot Bachelor NORS	1:12:43 09:49-11:02	Kress92 Cs
01-16	Setting up UE4 project with git (it takes a really long time) Bachelor NORS	1:47:35 10:41-12:28	D3stny
01-16	Project plan Bachelor NORS - [mobile]	1:35:11 12:03-13:39	Rodland92
01-16	Daily Scrum Meeting Bachelor NORS	0:30:00 21:00-21:30	Spoki0
01-19	Meeting with Simon Bachelor NORS	0:53:00 09:00-09:53	Spoki0
01-19	Meeting with Simon Bachelor NORS	0:45:00 09:00-09:45	Rodland92
01-19	Meeting with Simon Bachelor NORS	0:50:00 09:00-09:50	D3stny
01-19	Weekly Scrum Meeting Bachelor NORS	1:27:10 09:54-11:21	Spoki0
01-19	Design Document Bachelor NORS	0:58:05 10:05-11:03	Rodland92
01-19	Continue set up UE4 with git and test if working Bachelor NORS	0:43:00 10:17-11:00	D3stny
01-19	Looking at different resources on the web and how to implement them in our project Bachelor NORS	2:37:00 10:47-13:24	Kress92 Cs

01-19	More on Design Document Bachelor NORS	2:30:00 13:30-16:00	Rodland92
01-19	Scrum meeting + Design document + Project Bachelor NORS	4:30:00 13:30-18:00	D3stny
01-19	Makes Webpage Bachelor NORS	2:42:59 13:32-16:15	Spoki0
01-19	Git n stuff Bachelor NORS	0:37:00 16:30-17:07	Spoki0
01-20	Daily Scrum Meeting Bachelor NORS	0:15:00 09:00-09:15	Spoki0
01-20	Scrum meeting Bachelor NORS	0:15:00 09:00-09:15	D3stny
01-20	Design document Bachelor NORS - [mobile]	3:08:00 09:22-12:30	Rodland92
01-20	Learning UE4 Bachelor NORS	2:16:32 10:33-12:50	Spoki0
01-20	The 4 elements of a game Bachelor NORS	3:40:00 11:04-14:44	Kress92 Cs
01-20	Played around in UE4 Bachelor NORS	2:00:00 14:00-16:00	Rodland92
01-20	Learning UE4 Bachelor NORS	2:55:53 14:06-17:02	Spoki0
01-20	Getting to know UE4 Bachelor NORS	2:45:30 15:07-17:53	D3stny
01-20	Learning UE4 Bachelor NORS	2:09:45 19:47-21:57	Spoki0
01-20	Daily scrum meeting Bachelor NORS	0:15:00 21:00-21:15	Rodland92
01-20	Learning UE4 Bachelor NORS	0:46:56 22:07-22:54	Spoki0
01-21	Daily Scrum Meeting Bachelor NORS	0:30:00 09:00-09:30	Spoki0
01-21	Scrum meeting Bachelor NORS	0:25:00 09:00-09:25	D3stny
01-21	LaTeX Bachelor NORS - [mobile]	4:01:32 09:45-13:46	Rodland92
01-21	Making diagram Bachelor NORS	2:27:59 09:46-12:14	Kress92 Cs
01-21	Fixing UE4 after peeps made it crash Bachelor NORS	1:19:21 11:37-12:56	Spoki0
01-21	Working with UE4 and git support Bachelor NORS	2:19:18 12:16-14:35	D3stny
01-21	Learning UE4 Bachelor NORS	1:54:00 12:56-14:50	Spoki0
01-21	Learning about NavMesh and bot spawning Bachelor NORS	1:38:00 15:26-17:04	Kress92 Cs

01-21	Working with UE4 and git support	0:54:14	D3stny
	Bachelor NORS	17:04-17:58	
01-21	Daily scrum meeting	0:15:00	Rodland92
	Bachelor NORS	21:00-21:15	
01-22	Daily Scrum Meeting	0:15:00	Spoki0
	Bachelor NORS	09:00-09:15	
01-22	Daily scrum meeting	0:16:00	Rodland92
	Bachelor NORS	09:02-09:18	
01-22	Daily scrum meeting	0:16:00	Rodland92
	Bachelor NORS	09:02-09:18	
01-22	Daily scrum meeting	0:23:00	D3stny
	Bachelor NORS	09:07-09:30	
01-22	Design document	3:39:06	Rodland92
	Bachelor NORS	09:22-13:01	
01-22	Writing about mechanics	4:16:00	Kress92 Cs
	Bachelor NORS	10:18-14:34	
01-22	Div document editing	4:05:00	Spoki0
	Bachelor NORS	11:25-15:30	
01-22	Writing lore for design document	1:42:56	D3stny
	Bachelor NORS	12:21-14:04	
01-22	LaTeX	2:17:28	Rodland92
	Bachelor NORS - [mobile]	13:03-15:20	
01-23	Writing about mechanics	5:04:00	Kress92 Cs
	Bachelor NORS	10:14-15:18	
01-26	Meeting with Simon	1:08:53	Rodland92
	Bachelor NORS - [mobile]	09:03-10:12	
01-26	Meeting with Simon	1:10:00	Spoki0
	Bachelor NORS	09:05-10:15	
01-26	Weekly Scrum Meeting	0:57:00	Spoki0
	Bachelor NORS	11:20-12:17	
01-26	Daily scrum meeting	1:06:53	D3stny
	Bachelor NORS	11:20-12:27	
01-26	Weekly scrum meeting	0:45:00	Rodland92
	Bachelor NORS - [mobile]	11:30-12:15	
01-26	Making map	1:20:26	Spoki0
	Bachelor NORS	12:18-13:38	
01-26	Setting up project and creating map level	1:19:40	D3stny
	Bachelor NORS	12:29-13:49	
01-26	Setting up simple map	1:18:40	Rodland92
	Bachelor NORS - [mobile]	12:31-13:49	
01-26	Finishing up project plan	1:03:14	Spoki0
	Bachelor NORS	13:54-14:57	
01-26	Discussing project plan	1:00:50	D3stny
	Bachelor NORS	13:56-14:57	
01-26	Final touches on Project plan and simple map	1:00:00	Rodland92
	Bachelor NORS	14:00-15:00	

01-26	Meeting with Simon Bachelor NORS	1:00:00 21:15-22:15	D3stny
01-27	Daily scrum meeting Bachelor NORS	0:06:00 09:00-09:06	Rodland92
01-27	Daily Scrum Meeting Bachelor NORS	0:08:00 09:00-09:08	Spoki0
01-27	Starting to look at camera movement Bachelor NORS	0:54:00 09:11-10:05	Rodland92
01-27	Camera movement Bachelor NORS	1:42:00 10:12-11:54	Rodland92
01-27	Trying to get bot spawn with AI Bachelor NORS	2:11:53 10:29-12:41	Kress92 Cs
01-27	Make pause menu Bachelor NORS	0:26:28 13:08-13:35	D3stny
01-27	Make pause menu Bachelor NORS	2:14:31 14:24-16:38	D3stny
01-27	Trying to get bot spawn with AI Bachelor NORS	5:20:00 15:24-20:44	Kress92 Cs
01-27	Intro Menu UE4 Bachelor NORS	2:20:00 20:53-23:13	Spoki0
01-28	Daily scrum Bachelor NORS	0:10:00 09:00-09:10	D3stny
01-28	Daily Scrum meeting Bachelor NORS	0:11:00 09:02-09:13	Rodland92
01-28	Daily Scrum Meeting Bachelor NORS	0:11:00 09:04-09:15	Spoki0
01-28	Try to find out how to make the AI move towards a point Bachelor NORS	6:58:00 09:11-16:09	Kress92 Cs
01-28	Camera movements Bachelor NORS	1:00:24 09:37-10:37	Rodland92
01-28	Moving AI to spawnPoints Bachelor NORS	2:57:57 09:50-12:48	Kress92 Cs
01-28	Make pause menu Bachelor NORS	0:08:49 10:13-10:21	D3stny
01-28	Finishing up project plan Bachelor NORS	5:19:15 10:53-16:12	Spoki0
01-28	Looked at networking Bachelor NORS	1:53:12 11:07-13:00	Rodland92
01-28	Getting a movable character Bachelor NORS	1:45:27 13:01-14:46	Rodland92
01-28	Moving AI to spawnPoints Bachelor NORS	4:16:56 14:27-18:44	Kress92 Cs
01-29	Daily scrum Bachelor NORS	0:30:00 09:00-09:30	D3stny

01-29	Daily Scrum Meeting Bachelor NORS	0:31:00 09:02-09:33	Spoki0
01-29	Daily Scrum meeting Bachelor NORS	0:22:00 09:08-09:30	Rodland92
01-29	waypoint and looking through events Bachelor NORS	4:54:00 09:18-14:12	Kress92 Cs
01-29	Getting a movable character Bachelor NORS	0:22:00 09:30-09:52	Rodland92
01-29	spawning ai Bachelor NORS	2:41:12 09:35-12:17	Kress92 Cs
01-29	waypoint and looking through events Bachelor NORS	3:17:00 19:14-22:31	Kress92 Cs
01-29	Intro Menu UE4 Bachelor NORS	0:54:25 22:19-23:13	Spoki0
01-30	Daily Scrum meeting Bachelor NORS	0:10:00 09:00-09:10	Rodland92
01-30	Daily scrum Bachelor NORS	0:11:45 09:03-09:15	D3stny
01-30	Getting a movable character Bachelor NORS	1:30:00 09:12-10:42	Rodland92
01-30	Work more on behavior tree Bachelor NORS	5:54:00 09:21-15:15	Kress92 Cs
01-30	Getting a movable character Bachelor NORS	4:16:15 10:43-14:59	Rodland92
01-30	Intro Menu UE4 Bachelor NORS	2:04:08 10:58-13:02	Spoki0
01-30	Make pause menu Bachelor NORS	0:49:45 13:29-14:19	D3stny
01-30	Make pause menu Bachelor NORS	1:49:21 15:56-17:45	D3stny
01-30	Make pause menu Bachelor NORS	3:30:35 18:11-21:42	D3stny

Detailed report



2015-03-01 - 2015-03-31

Total 353 h 38 min

Date	Description	Duration	User
03-02	Meeting with Simon Bachelor NORS	0:59:00 09:00-09:59	Spoki0
03-02	Weekly Meeting with Simon Bachelor NORS	1:00:00 09:00-10:00	Rodland92
03-02	Meeting Bachelor NORS	1:00:00 09:00-10:00	Kress92 Cs
03-02	Meeting with Simon Bachelor NORS	0:50:00 09:05-09:55	D3stny
03-02	Shop Bachelor NORS	2:30:00 10:00-12:30	D3stny
03-02	Pair programming Bachelor NORS	1:40:00 10:05-11:45	Spoki0
03-02	trying to find "destroy actor" blackboard bug also general discussion in team Bachelor NORS	2:59:00 10:05-13:04	Kress92 Cs
03-02	Weekly Scrum Meeting Bachelor NORS	0:20:00 11:45-12:05	Spoki0
03-02	Pair programming Bachelor NORS	1:20:00 12:05-13:25	Spoki0
03-02	weekly scrum Bachelor NORS	0:16:16 12:43-12:59	D3stny
03-02	Network Bachelor NORS	1:00:00 14:30-15:30	Rodland92
03-02	Updating engine Bachelor NORS	0:47:10 15:15-16:02	Spoki0
03-02	Trying to fix networking movement Bachelor NORS	1:04:56 16:02-17:07	Spoki0
03-02	Minions, go in right lane, fixed bug,added towers to array etc. Bachelor NORS	4:42:00 17:12-21:54	Kress92 Cs
03-03	Daily Scrum meeting Bachelor NORS	0:25:00 09:00-09:25	Rodland92
03-03	Daily Scrum Meeting Bachelor NORS	0:33:00 09:00-09:33	Spoki0
03-03	Daily Scrum Log Bachelor NORS	0:24:27 09:01-09:25	D3stny
03-03	Fixed Toggl Bachelor NORS	0:05:51 09:25-09:30	Rodland92
03-03	fixed some bugs Bachelor NORS	1:04:00 09:30-10:34	Kress92 Cs

03-03	Spells on champions	1:48:24	Rodland92
	Bachelor NORS	09:31-11:19	
03-03	looking through different types of textures that can be used.	0:00:15	Kress92 Cs
	Bachelor NORS	10:33-10:33	
03-03	Lobby plugin	0:31:46	Spoki0
	Bachelor NORS	10:34-11:05	
03-03	looking through different types of textures that can be used.	0:12:23	Kress92 Cs
	Bachelor NORS	10:42-10:54	
03-03	Shop	2:12:19	D3stny
	Bachelor NORS	11:48-14:01	
03-03	Spells on champions	1:03:46	Rodland92
	Bachelor NORS	14:00-15:03	
03-03	Shop	0:57:12	D3stny
	Bachelor NORS	14:16-15:13	
03-04	Daily Scrum Meeting	0:20:00	Spoki0
	Bachelor NORS	09:00-09:20	
03-04	Daily Scrum Log	0:12:51	D3stny
	Bachelor NORS	09:06-09:19	
03-04	Daily Scrum meeting	0:12:31	Rodland92
	Bachelor NORS	09:06-09:19	
03-04	Working on proxy pawns and copy from server: movement and transforms.	1:06:00	Kress92 Cs
	Bachelor NORS	09:18-10:24	
03-04	Spells on champions	5:05:17	Rodland92
	Bachelor NORS	09:54-14:59	
03-04	VaRest plugin - Rebuild to UE4.7.1	1:03:26	Spoki0
	Bachelor NORS	10:01-11:05	
03-04	Working on proxy pawns and copy from server: movement and transforms.	0:43:00	Kress92 Cs
	Bachelor NORS	10:51-11:34	
03-04	Shop	3:31:56	D3stny
	Bachelor NORS	11:28-15:00	
03-04	VaRest plugin - Rebuild to UE4.7.1	1:11:57	Spoki0
	Bachelor NORS	12:23-13:35	
03-04	Working on proxy pawns and copy from server: movement and transforms.	1:56:00	Kress92 Cs
	Bachelor NORS	13:08-15:04	
03-04	VaRest Plugin - Test functionality	0:41:30	Spoki0
	Bachelor NORS	13:35-14:16	
03-04	Working on proxy pawns and copy from server: movement and transforms.	3:29:00	Kress92 Cs
	Bachelor NORS	16:34-20:03	
03-04	Researching network	0:33:32	D3stny
	Bachelor NORS	17:07-17:40	
03-04	Shop	3:24:35	D3stny
	Bachelor NORS	17:47-21:11	
03-04	VaRest Plugin - Test functionality	0:15:00	Spoki0
	Bachelor NORS	18:30-18:45	

03-04	VaRest Plugin - Test functionality	2:25:16	Spoki0
	Bachelor NORS	20:25-22:50	
03-04	Spells on champions	0:48:35	Rodland92
	Bachelor NORS	20:56-21:45	
03-04	Shop	1:15:00	D3stny
	Bachelor NORS	21:45-23:00	
03-05	Daily Scrum meeting	0:17:22	Rodland92
	Bachelor NORS	09:00-09:17	
03-05	Daily Scrum Meeting	0:17:00	Spoki0
	Bachelor NORS	09:00-09:17	
03-05	Daily Scrum Log	0:14:28	D3stny
	Bachelor NORS	09:02-09:16	
03-05	Port it over to champion	2:48:57	Kress92 Cs
	Bachelor NORS	09:20-12:09	
03-05	Porting	1:28:58	Rodland92
	Bachelor NORS	10:20-11:48	
03-05	VaRest Plugin - Getting it to work	1:54:00	Spoki0
	Bachelor NORS	10:26-12:20	
03-05	Updating	1:02:54	D3stny
	Bachelor NORS	11:08-12:10	
03-05	Spells on champions	2:54:43	Rodland92
	Bachelor NORS	11:49-14:43	
03-05	Shop/GUI	1:56:34	D3stny
	Bachelor NORS	12:11-14:07	
03-05	PHP Server and MySQL Database	1:40:10	Spoki0
	Bachelor NORS	12:20-14:00	
03-05	decals	0:51:00	Kress92 Cs
	Bachelor NORS	14:08-14:59	
03-05	Shop	0:35:20	D3stny
	Bachelor NORS	14:28-15:03	
03-05	Shop	1:07:36	D3stny
	Bachelor NORS	15:14-16:22	
03-05	PHP Server and MySQL Database	1:37:39	Spoki0
	Bachelor NORS	15:55-17:32	
03-05	Researching network	2:52:48	D3stny
	Bachelor NORS	16:22-19:14	
03-05	decals:fog of war	5:57:00	Kress92 Cs
	Bachelor NORS	20:23-02:20	
03-06	Daily Scrum Log	0:24:38	D3stny
	Bachelor NORS	08:58-09:23	
03-06	Daily Scrum meeting	0:25:10	Rodland92
	Bachelor NORS	08:59-09:24	
03-06	Daily Scrum Meeting	0:26:00	Spoki0
	Bachelor NORS	09:00-09:26	
03-06	Spells on champions	2:20:00	Rodland92
	Bachelor NORS	09:45-12:05	

03-06	PHP Server and MySQL Database	1:51:00	Spoki0
	Bachelor NORS	10:12-12:03	
03-06	Shop	1:25:47	D3stny
	Bachelor NORS	12:05-13:31	
03-06	Shop	1:27:46	D3stny
	Bachelor NORS	16:24-17:51	
03-06	Shop	1:30:00	Rodland92
	Bachelor NORS	16:30-18:00	
03-07	Fixing after someone broke plugin	0:30:00	Spoki0
	Bachelor NORS	13:30-14:00	
03-07	Spells on champions	0:41:07	Rodland92
	Bachelor NORS	13:50-14:31	
03-07	Shop	2:47:10	D3stny
	Bachelor NORS	15:05-17:52	
03-08	Shop	0:23:38	D3stny
	Bachelor NORS	12:40-13:03	
03-08	Login/Register on intro screen	1:14:41	Spoki0
	Bachelor NORS	18:47-20:02	
03-08	Buying Lobby Module	0:30:00	Spoki0
	Bachelor NORS	21:09-21:39	
03-09	Meeting with Simon	0:34:00	Spoki0
	Bachelor NORS	09:00-09:34	
03-09	Weekly Meeting with Simon	0:30:00	Rodland92
	Bachelor NORS	09:00-09:30	
03-09	Meeting with Simon	0:27:10	D3stny
	Bachelor NORS	09:04-09:31	
03-09	weekly scrum	1:14:05	D3stny
	Bachelor NORS	09:31-10:45	
03-09	Weekly Scrum Meeting	1:25:00	Spoki0
	Bachelor NORS	09:35-11:00	
03-09	Jira and Confluence	1:00:53	Spoki0
	Bachelor NORS	11:09-12:10	
03-09	Setting up JIRA	0:58:20	D3stny
	Bachelor NORS	11:48-12:46	
03-09	Spells on champions	2:45:00	Rodland92
	Bachelor NORS	13:45-16:30	
03-09	Looking into Blueprints of Lobby System	2:45:59	Spoki0
	Bachelor NORS	13:50-16:35	
03-09	Inventory	1:13:11	D3stny
	Bachelor NORS	16:28-17:41	
03-09	Porting Lobby System to 4.7	0:15:45	Spoki0
	Bachelor NORS	16:36-16:51	
03-09	Inventory	0:23:40	D3stny
	Bachelor NORS	18:34-18:58	
03-09	Weekly Scrum Meeting	1:25:00	Rodland92
	Bachelor NORS	21:35-23:00	

03-09	Inventory Bachelor NORS	0:49:20 21:42-22:31	D3stny
03-10	Daily Scrum Log Bachelor NORS	0:10:00 09:00-09:10	D3stny
03-10	Daily Scrum Meeting Bachelor NORS	0:08:00 09:00-09:08	Spoki0
03-10	Daily Scrum meeting Bachelor NORS	0:10:00 09:00-09:10	Rodland92
03-10	Improve behavior tree Bachelor NORS - [mobile]	5:05:00 09:07-14:12	Kress92 Cs
03-10	Spells on champions Bachelor NORS	3:26:00 09:25-12:51	Rodland92
03-10	Making Lobby Work Bachelor NORS	2:17:35 10:35-12:53	Spoki0
03-10	Inventory Bachelor NORS	2:36:22 11:24-14:01	D3stny
03-10	Shop Bachelor NORS	0:22:29 15:14-15:36	D3stny
03-10	Making Lobby Work Bachelor NORS	3:14:32 15:29-18:44	Spoki0
03-10	Shop Bachelor NORS	0:59:03 16:53-17:52	D3stny
03-10	Spells on champions Bachelor NORS	1:00:00 20:30-21:30	Rodland92
03-11	Daily Scrum Log Bachelor NORS	0:10:00 09:00-09:10	D3stny
03-11	Daily Scrum Meeting Bachelor NORS	0:15:00 09:00-09:15	Spoki0
03-11	Daily Scrum meeting Bachelor NORS	0:10:00 09:00-09:10	Rodland92
03-11	Getting lobby system to work Bachelor NORS	2:39:41 09:23-12:03	Spoki0
03-11	Spells on champions Bachelor NORS	2:31:00 09:30-12:01	Rodland92
03-11	studying the example of turn based strategy on ue4 Bachelor NORS	1:15:00 10:41-11:56	Kress92 Cs
03-11	Lynkurs rapportskriving Bachelor NORS	1:13:00 12:02-13:15	Rodland92
03-11	Lecture on writing the bachelor thesis Bachelor NORS	1:16:41 12:04-13:20	Spoki0
03-11	Lynkurs rapportskriving Bachelor NORS	0:59:17 12:04-13:03	D3stny
03-11	Spells on champions Bachelor NORS	2:22:28 13:45-16:07	Rodland92
03-11	Work on lobby system Bachelor NORS	1:26:00 14:20-15:46	Spoki0

03-11	Learn the case of spells, and learn the structure	1:21:00	Kress92 Cs
	Bachelor NORS	15:15-16:36	
03-11	Minor Update of Webpage	0:13:30	Spoki0
	Bachelor NORS	15:46-15:59	
03-11	Shop	0:35:33	D3stny
	Bachelor NORS	15:49-16:25	
03-11	Shop	0:22:39	D3stny
	Bachelor NORS	17:30-17:52	
03-11	Trying to make the Q for warrior	4:27:00	Kress92 Cs
	Bachelor NORS	19:11-23:38	
03-12	Daily Scrum Meeting	0:17:22	Spoki0
	Bachelor NORS	08:59-09:17	
03-12	Daily Scrum meeting	0:17:41	Rodland92
	Bachelor NORS	09:00-09:17	
03-12	Daily Scrum Log	0:05:57	D3stny
	Bachelor NORS	09:10-09:16	
03-12	Work on lobby system	1:59:56	Spoki0
	Bachelor NORS	09:29-11:28	
03-12	Spells on champions	4:31:00	Rodland92
	Bachelor NORS	09:29-14:00	
03-12	Shop	2:15:36	D3stny
	Bachelor NORS	11:22-13:37	
03-12	Transision between attack, finnished warrior Q-attack to charge target.	2:43:48	Kress92 Cs
	Bachelor NORS	11:22-14:05	
03-12	Pair programming on movement buff/debuff	0:57:00	Spoki0
	Bachelor NORS	12:02-12:59	
03-12	Mail to simon	0:16:21	Spoki0
	Bachelor NORS	13:30-13:46	
03-12	Shop	0:44:08	D3stny
	Bachelor NORS	20:11-20:55	
03-12	Making static minion and with basic behavior to test spells on, working on let the ai take over the character.	4:21:00	Kress92 Cs
	Bachelor NORS	20:39-01:00	
03-12	Attribute changes	1:08:40	D3stny
	Bachelor NORS	20:57-22:05	
03-13	Looking at projektile move ment and the example cowboy	2:02:00	Kress92 Cs
	Bachelor NORS - [mobile]	09:32-11:34	
03-13	Spells on champions	0:18:51	Rodland92
	Bachelor NORS	11:30-11:48	
03-13	Attribute changes	2:40:29	D3stny
	Bachelor NORS	11:48-14:29	
03-13	Attribute changes	2:33:00	D3stny
	Bachelor NORS	15:45-18:18	
03-13	Spells on champions	1:36:37	Rodland92
	Bachelor NORS	16:00-17:36	

03-13	Work on lobby system Bachelor NORS	1:28:46 16:58-18:27	Spoki0
03-13	Looking at projekte move ment and the example cowboy Bachelor NORS - [mobile]	0:00:00 18:59-18:59	Kress92 Cs
03-14	Shop Bachelor NORS	1:54:24 12:54-14:49	D3stny
03-15	Work on lobby system Bachelor NORS	4:24:38 13:15-17:39	Spoki0
03-16	Weekly Scrum Meeting Bachelor NORS	1:33:35 09:00-10:33	Spoki0
03-16	scrum meeting, disscussion. planing Bachelor NORS	2:32:00 09:00-11:32	Kress92 Cs
03-16	Weekly Scrum Meeting Bachelor NORS	1:30:00 09:00-10:30	Rodland92
03-16	Fixing XP bug Bachelor NORS	1:01:44 09:02-10:03	D3stny
03-16	weekly scrum Bachelor NORS	0:31:27 10:04-10:35	D3stny
03-16	Work on lobby system Bachelor NORS	4:19:54 11:45-16:04	Spoki0
03-16	Fix bug after folder changes Bachelor NORS	0:36:21 11:55-12:31	D3stny
03-16	Attribute changes Bachelor NORS	2:02:21 12:33-14:35	D3stny
03-17	Daily Scrum Meeting Bachelor NORS	0:10:00 09:00-09:10	Spoki0
03-17	Daily Scrum Log Bachelor NORS	0:09:31 09:00-09:10	D3stny
03-17	Daily Scrum meeting Bachelor NORS	0:08:00 09:02-09:10	Rodland92
03-17	Spells on champions Bachelor NORS	3:18:00 09:30-12:48	Rodland92
03-17	Work on lobby system Bachelor NORS	1:53:00 11:16-13:09	Spoki0
03-17	Making minion attack Bachelor NORS	2:15:06 12:29-14:44	Kress92 Cs
03-17	Work on lobby system Bachelor NORS	2:08:14 13:38-15:46	Spoki0
03-17	Making ai attack tower, minion on clicked get new materials, making minion attack tower Bachelor NORS	3:04:21 14:44-17:49	Kress92 Cs
03-17	Jira and Confluence Bachelor NORS	1:10:13 16:52-18:03	Spoki0
03-17	decals and fun stuf Bachelor NORS	0:20:14 17:52-18:12	Kress92 Cs

03-17	Attributes Bachelor NORS	2:05:37 20:19-22:24	D3stny
03-17	Test of LAN Networking Bachelor NORS	1:22:12 21:18-22:40	Spoki0
03-17	Updating Engine Bachelor NORS	0:34:54 21:18-21:52	Spoki0
03-18	Daily Scrum Meeting Bachelor NORS	0:11:00 09:00-09:11	Spoki0
03-18	Daily Scrum meeting Bachelor NORS	0:10:00 09:00-09:10	Rodland92
03-18	Daily Scrum Log Bachelor NORS	0:05:42 09:06-09:12	D3stny
03-18	Minions attack and trying to add decal to minion Bachelor NORS	11:18:00 09:08-20:26	Kress92 Cs
03-18	Spells on champions Bachelor NORS	6:01:12 09:30-15:31	Rodland92
03-18	updating unreal Bachelor NORS	1:03:26 10:18-11:22	D3stny
03-18	Saving Username Bachelor NORS	2:39:54 11:08-13:48	Spoki0
03-18	Attributes Bachelor NORS	5:25:21 12:14-17:39	D3stny
03-18	Spells on champions Bachelor NORS	0:51:31 18:20-19:11	Rodland92
03-19	Daily Scrum meeting Bachelor NORS	0:25:00 08:59-09:24	Rodland92
03-19	Daily Scrum Log Bachelor NORS	0:20:38 08:59-09:20	D3stny
03-19	Daily Scrum Meeting Bachelor NORS	0:20:00 09:00-09:20	Spoki0
03-19	Research Character Movement Bachelor NORS	0:10:00 09:40-09:50	Spoki0
03-19	Spells on champions Bachelor NORS	1:15:00 10:00-11:15	Rodland92
03-19	Networking Bachelor NORS	4:06:05 10:23-14:29	Spoki0
03-19	Base attack Bachelor NORS	0:28:10 12:13-12:41	D3stny
03-19	Base attack Bachelor NORS	2:44:36 13:23-16:08	D3stny
03-19	Saving Username Bachelor NORS	1:30:36 14:40-16:10	Spoki0
03-20	Champion Selection Bachelor NORS	2:36:12 11:21-13:57	Spoki0
03-20	Base attack Bachelor NORS	0:24:13 13:23-13:47	D3stny

03-20	Base attack Bachelor NORS	1:26:39 14:22-15:49	D3stny
03-20	Spells on champions Bachelor NORS	1:20:57 14:27-15:48	Rodland92
03-20	Spells on champions Bachelor NORS	0:47:27 19:35-20:22	Rodland92
03-21	Base attack Bachelor NORS	1:11:50 12:20-13:32	D3stny
03-23	Meeting with Simon Bachelor NORS	0:45:00 09:00-09:45	Spoki0
03-23	Weekly Scrum Meeting Bachelor NORS	0:45:00 09:00-09:45	Rodland92
03-23	Meeting with Simon Bachelor NORS	0:37:42 09:06-09:44	D3stny
03-23	Weekly Scrum Meeting Bachelor NORS	0:30:00 09:45-10:15	Spoki0
03-23	Continued meeting Bachelor NORS	0:25:39 09:46-10:11	D3stny
03-23	Integrating of Assets Bachelor NORS	4:30:00 10:30-15:00	Spoki0
03-23	Redesign towers and bases Bachelor NORS	3:00:00 12:00-15:00	D3stny
03-23	Dessign Bachelor NORS	2:59:00 12:01-15:00	Rodland92
03-24	Daily Scrum Meeting Bachelor NORS	0:10:00 09:00-09:10	Spoki0
03-24	Daily Scrum meeting Bachelor NORS	0:10:00 09:00-09:10	Rodland92
03-24	subtract from the mask. object out of line of sight Bachelor NORS	3:27:15 10:27-13:54	Kress92 Cs
03-24	Material and Particle fixes Bachelor NORS	1:31:13 11:16-12:48	Spoki0
03-24	Design Bachelor NORS	1:42:54 12:10-13:53	D3stny
03-24	Research on networking Bachelor NORS	2:58:13 13:45-16:43	Spoki0
03-24	Spells on champions Bachelor NORS	2:58:47 14:00-16:58	Rodland92
03-24	Design Bachelor NORS	2:53:50 14:21-17:14	D3stny
03-24	Package Sniffing over LAN Bachelor NORS	0:51:10 16:43-17:34	Spoki0
03-24	subtract from the mask. object out of line of sight Bachelor NORS	1:35:27 17:01-18:36	Kress92 Cs
03-24	Fix healthbar Bachelor NORS	0:23:21 17:32-17:55	D3stny

03-24	Test of LAN Networking Bachelor NORS	0:47:24 17:37-18:24	Spoki0
03-24	Fix healthbar Bachelor NORS	1:20:10 18:39-20:00	D3stny
03-24	Fix of Plugin for release Bachelor NORS	0:55:53 19:55-20:50	Spoki0
03-25	Daily Scrum Meeting Bachelor NORS	0:18:00 09:00-09:18	Spoki0
03-25	Daily Scrum meeting Bachelor NORS	0:20:00 09:00-09:20	Rodland92
03-25	Daily Scrum Log Bachelor NORS	0:15:00 09:00-09:15	D3stny
03-25	Design, Fences Bachelor NORS	1:20:00 09:30-10:50	Rodland92
03-25	subtract from the mask. object out of line of sight Bachelor NORS	1:19:17 09:43-11:02	Kress92 Cs
03-25	Fixing Building Healthbar Bachelor NORS	0:21:28 10:00-10:21	Spoki0
03-25	Shield Bar Bachelor NORS	0:39:03 10:21-11:00	Spoki0
03-25	custome project, making the line find wich side and draw it correctly Bachelor NORS	4:12:43 12:58-17:11	Kress92 Cs
03-25	Shield Bar Bachelor NORS	2:45:30 13:25-16:11	Spoki0
03-25	Healthbar Bachelor NORS	2:02:46 13:52-15:54	D3stny
03-25	Design, Stone Walls Bachelor NORS	2:00:00 14:00-16:00	Rodland92
03-25	Warrior E Spell Bachelor NORS	1:06:14 16:11-17:17	Spoki0
03-25	custome project, making the line find wich side and draw it correctly Bachelor NORS	0:00:15 17:11-17:11	Kress92 Cs
03-26	Daily Scrum meeting Bachelor NORS	0:10:00 09:00-09:10	Rodland92
03-26	Daily Scrum Meeting Bachelor NORS	0:08:00 09:00-09:08	Spoki0
03-26	Daily Scrum Log Bachelor NORS	0:15:00 09:00-09:15	D3stny
03-26	custome project, making the line find wich side and draw it correctly Bachelor NORS	7:56:02 09:07-17:04	Kress92 Cs
03-26	Design, Stone Walls Bachelor NORS	0:36:02 09:25-10:01	Rodland92
03-26	Overall design Bachelor NORS	4:03:00 10:01-14:04	Rodland92

03-26	Warrior E Spell Bachelor NORS	0:31:18 10:35-11:06	Spoki0
03-26	Extraordinary Scrum meeting before Easter Bachelor NORS	1:56:18 11:06-13:03	Spoki0
03-26	Designing jungle meeting Bachelor NORS	1:23:13 11:18-12:41	D3stny
03-26	Pre-Easter meeting Bachelor NORS	0:51:44 12:41-13:33	D3stny
03-26	Warrior E Spell Bachelor NORS	1:22:37 13:03-14:25	Spoki0
03-26	Fixing healthbar Bachelor NORS	1:56:00 13:34-15:30	D3stny
03-27	Migrate Bachelor NORS - [mobile]	2:35:47 09:49-12:24	Kress92 Cs
03-31	Map Environment Bachelor NORS	3:28:12 12:44-16:12	Spoki0

Detailed report



2015-04-01 - 2015-04-30

Total 279 h 12 min

Date	Description	Duration	User
04-01	Map Environment Bachelor NORS	3:27:32 13:54-17:21	Spoki0
04-02	fix bug in behavior tree Bachelor NORS	7:05:00 09:42-16:47	Kress92 Cs
04-02	Map Environment Bachelor NORS	3:58:53 12:46-16:44	Spoki0
04-02	Fixing healthbar Bachelor NORS	0:57:45 14:17-15:15	D3stny
04-02	Items and attributes Bachelor NORS	0:13:12 15:50-16:03	D3stny
04-02	custome project, making the line find wich side and draw it correctly Bachelor NORS	0:00:42 18:11-18:12	Kress92 Cs
04-05	Updating Engine Bachelor NORS	1:49:37 12:23-14:12	Spoki0
04-05	Map Environment Bachelor NORS	1:39:02 14:12-15:51	Spoki0
04-05	Map Environment Bachelor NORS	2:43:05 20:15-22:58	Spoki0
04-05	Fixing Git in relation to size limit Bachelor NORS	1:58:37 22:58-00:57	Spoki0
04-07	Meeting With Simon Bachelor NORS	0:45:00 14:45-15:30	Spoki0
04-07	Meeting with Simon Bachelor NORS - [mobile]	0:44:52 14:51-15:36	D3stny
04-07	Weekly Scrum Meeting Bachelor NORS	0:45:00 15:30-16:15	Spoki0
04-07	Weekly scrum meeting Bachelor NORS - [mobile]	0:37:52 15:41-16:18	D3stny
04-08	Tracking vision of minion, moving code to controller instead of minion. Bachelor NORS	7:45:00 08:25-16:10	Kress92 Cs
04-08	Daily Scrum Meeting Bachelor NORS	0:07:00 09:00-09:07	Rodland92
04-08	Daily Scrum Meeting Bachelor NORS	0:07:00 09:00-09:07	Spoki0
04-08	Updating and dl'ing new repo Bachelor NORS	0:24:00 09:30-09:54	Rodland92
04-08	Spells on champions Bachelor NORS	6:19:48 09:54-16:13	Rodland92

04-08	Webpage update Bachelor NORS	0:23:13 11:01-11:24	Spoki0
04-08	Minimap Bachelor NORS	3:30:07 11:24-14:54	Spoki0
04-08	Items and item functions Bachelor NORS	1:42:44 13:35-15:18	D3stny
04-08	Research Bachelor NORS	1:17:00 14:58-16:15	Spoki0
04-09	Updating Repo and Engine Bachelor NORS	1:00:17 13:13-14:13	Spoki0
04-09	Champ info Bachelor NORS	1:15:02 15:48-17:03	D3stny
04-09	Champ info Bachelor NORS	1:34:52 18:45-20:20	D3stny
04-10	Daily Scrum Meeting Bachelor NORS	0:09:00 09:00-09:09	Spoki0
04-10	Reaserch about AI Bachelor NORS	1:30:00 09:30-11:00	Kress92 Cs
04-10	Fixing errors in Build Bachelor NORS	1:53:05 10:27-12:20	Spoki0
04-10	AI Brain Bachelor NORS	8:14:05 11:01-19:15	Kress92 Cs
04-10	Items and item functions Bachelor NORS	1:39:47 13:18-14:58	D3stny
04-13	Weekly Scrum Meeting Bachelor NORS	1:00:00 09:00-10:00	Spoki0
04-13	Weekly scrum meeting Bachelor NORS	2:36:02 09:18-11:54	D3stny
04-13	Meeting With Simon Bachelor NORS	0:15:00 13:00-13:15	Spoki0
04-13	Win conditions Bachelor NORS	1:01:45 14:29-15:31	Spoki0
04-13	Win conditions Bachelor NORS	1:57:58 16:29-18:27	Spoki0
04-14	custome move AI to, debugging minion, debugging spawningpoints, towers etc Bachelor NORS	9:19:00 07:43-17:02	Kress92 Cs
04-14	Daily Scrum Meeting Bachelor NORS	0:20:00 09:00-09:20	Spoki0
04-14	Daily Scrum Meeting Bachelor NORS	0:05:00 09:10-09:15	Rodland92
04-14	Spells on champions Bachelor NORS	4:30:00 09:50-14:20	Rodland92
04-14	Items and item functions Bachelor NORS	1:28:52 11:37-13:06	D3stny
04-14	Win conditions - UI Fixing Bachelor NORS	4:20:01 11:38-15:58	Spoki0

04-14	Items and item functions Bachelor NORS	1:16:38 14:11-15:28	D3stny
04-14	Writing in LaTeX Bachelor NORS	0:45:00 14:25-15:10	Rodland92
04-14	Merge Bachelor NORS	0:18:25 15:58-16:16	Spoki0
04-15	importing, skeletons, fbx, setting texture, Animations Bachelor NORS - [mobile]	8:02:43 09:44-17:46	Kress92 Cs
04-15	Networking Bachelor NORS	3:57:57 10:00-13:57	Spoki0
04-15	Network Bachelor NORS	4:00:00 10:00-14:00	Rodland92
04-15	Items and item functions Bachelor NORS	2:42:27 11:00-13:43	D3stny
04-15	Changing of elements and Class Bachelor NORS	2:34:00 14:00-16:34	Rodland92
04-15	Items and item functions Bachelor NORS	1:07:13 19:49-20:56	D3stny
04-15	Animations Bachelor NORS - [mobile]	3:56:00 20:04-00:00	Kress92 Cs
04-16	Daily Scrum Meeting Bachelor NORS	0:30:00 09:00-09:30	Spoki0
04-16	Daily Scrum Meeting Bachelor NORS	0:29:15 09:05-09:34	Rodland92
04-16	Daily Scrum Log Bachelor NORS	0:20:39 09:16-09:37	D3stny
04-16	animations, turn champion towards enemy, Bachelor NORS - [mobile]	5:10:00 09:17-14:27	Kress92 Cs
04-16	Spells on champions Bachelor NORS	5:00:00 10:00-15:00	Rodland92
04-16	UI Bachelor NORS	3:32:00 11:00-14:32	Spoki0
04-16	Change animation blueprints, montage (research) add dead animation Bachelor NORS - [mobile]	2:45:00 17:47-20:32	Kress92 Cs
04-16	Changing behavior tree to purely event driven. removed state Bachelor NORS - [mobile]	2:14:00 22:46-01:00	Kress92 Cs
04-17	Daily Scrum Meeting Bachelor NORS	0:25:00 09:00-09:25	Rodland92
04-17	Daily Scrum Meeting Bachelor NORS	0:30:00 09:00-09:30	Spoki0
04-17	Writing, animasjon Bachelor NORS	2:54:00 09:03-11:57	Kress92 Cs
04-17	Writing in LaTeX Bachelor NORS	5:00:00 09:30-14:30	Rodland92

04-17	Writing, animasjon Bachelor NORS	2:59:13 11:57-14:56	Kress92 Cs
04-17	LaTeX Bachelor NORS	0:30:00 12:00-12:30	Spoki0
04-20	Weekly Scrum Meeting Bachelor NORS	0:35:00 09:20-09:55	Spoki0
04-20	Report meeting Bachelor NORS	2:25:02 09:52-12:17	D3stny
04-20	LaTeX Bachelor NORS	2:39:00 09:55-12:34	Spoki0
04-20	LaTeX Bachelor NORS	0:50:00 13:00-13:50	Spoki0
04-20	Report meeting Bachelor NORS	2:41:00 13:00-15:41	D3stny
04-21	Daily Scrum Meeting Bachelor NORS	0:20:00 09:00-09:20	Spoki0
04-21	Daily Scrum Log Bachelor NORS	0:15:00 09:00-09:15	D3stny
04-21	Daily Scrum Meeting Bachelor NORS	0:16:00 09:00-09:16	Rodland92
04-21	Spells on champions Bachelor NORS	4:00:00 09:30-13:30	Rodland92
04-21	GUI Work Bachelor NORS	0:40:00 11:00-11:40	Spoki0
04-21	Writing, range basic attack Bachelor NORS - [mobile]	3:56:38 11:10-15:06	Kress92 Cs
04-21	Updating engine Bachelor NORS	0:34:04 11:41-12:15	Spoki0
04-21	Writing of thesis Bachelor NORS	0:35:53 12:15-12:51	Spoki0
04-21	Writing of thesis Bachelor NORS	1:15:24 13:58-15:13	Spoki0
04-21	Spells on champions Bachelor NORS	1:00:00 14:00-15:00	Rodland92
04-21	Item icon design Bachelor NORS	1:10:03 14:11-15:21	D3stny
04-21	GUI Work Bachelor NORS	0:16:05 16:11-16:27	Spoki0
04-22	Daily Scrum Meeting Bachelor NORS	0:15:00 09:00-09:15	Spoki0
04-22	Daily Scrum Meeting Bachelor NORS	0:15:00 09:00-09:15	Rodland92
04-22	Daily Scrum Log Bachelor NORS	0:15:00 09:00-09:15	D3stny
04-22	Cooldown on spells Bachelor NORS	5:46:23 09:25-15:11	Rodland92

04-22	JungleCreep, basic attacks, etc	5:09:00	Kress92 Cs
	Bachelor NORS	09:42-14:51	
04-22	LaTeX - Design Document	2:35:31	Spoki0
	Bachelor NORS	11:45-14:20	
04-22	Item icon design	2:33:58	D3stny
	Bachelor NORS	12:09-14:42	
04-22	Thesis Reading	1:26:43	Spoki0
	Bachelor NORS	14:20-15:46	
04-22	Item icon design	0:28:24	D3stny
	Bachelor NORS	14:59-15:27	
04-22	Rebasing minion to baseclass	6:08:47	Kress92 Cs
	Bachelor NORS	17:25-23:34	
04-23	Daily Scrum Meeting	0:20:00	Rodland92
	Bachelor NORS	09:00-09:20	
04-23	Daily Scrum Meeting	0:15:23	Spoki0
	Bachelor NORS	09:00-09:15	
04-23	Daily Scrum Log	0:16:00	D3stny
	Bachelor NORS	09:03-09:19	
04-23	Spells on champions	4:17:24	Rodland92
	Bachelor NORS	09:36-13:53	
04-23	Thesis Writing	4:06:34	Spoki0
	Bachelor NORS	10:22-14:28	
04-23	Writing about items in confluence	2:17:53	D3stny
	Bachelor NORS	12:13-14:31	
04-23	Player Spawn	0:58:18	Spoki0
	Bachelor NORS	14:28-15:26	
04-24	Daily Scrum Meeting	0:30:00	Rodland92
	Bachelor NORS	09:00-09:30	
04-24	Daily Scrum Meeting	0:30:00	Spoki0
	Bachelor NORS	09:00-09:30	
04-24	Daily Scrum Log	0:29:55	D3stny
	Bachelor NORS	09:04-09:34	
04-24	Shortcut errors	0:28:58	Rodland92
	Bachelor NORS	09:30-09:58	
04-24	Contacted Simon	0:25:00	Spoki0
	Bachelor NORS	10:15-10:40	
04-24	Including steam to the project	5:30:00	Rodland92
	Bachelor NORS	10:30-16:00	
04-24	Networking	3:55:59	Spoki0
	Bachelor NORS	10:44-14:40	
04-24	Make spawn area	3:26:44	D3stny
	Bachelor NORS	12:46-16:12	
04-24	Bugfixing for actual Networking	2:08:22	Spoki0
	Bachelor NORS	14:41-16:49	
04-27	Weekly Scrum Meeting	3:00:00	Rodland92
	Bachelor NORS	09:00-12:00	

04-27	Meeting with Simon Bachelor NORS	0:45:00 09:00-09:45	Spoki0
04-27	Thesis Structuring Bachelor NORS	2:05:00 09:45-11:50	Spoki0
04-27	Weekly scrum meeting Bachelor NORS	1:30:00 10:30-12:00	D3stny
04-27	Weekly Scrum Meeting Bachelor NORS	0:15:00 11:50-12:05	Spoki0
04-27	Writing in LaTeX Bachelor NORS	1:18:26 12:30-13:48	Rodland92
04-27	Fixing spawn area Bachelor NORS	1:00:00 13:00-14:00	D3stny
04-28	Daily Scrum Meeting Bachelor NORS	0:10:00 09:00-09:10	Rodland92
04-28	Daily Scrum Meeting Bachelor NORS	0:10:00 09:00-09:10	Spoki0
04-28	Daily Scrum Log Bachelor NORS	0:10:00 09:00-09:10	D3stny
04-28	Writing in LaTeX - Introduction Bachelor NORS	5:33:42 09:30-15:03	Rodland92
04-28	Writing GUI Bachelor NORS	2:00:00 11:00-13:00	D3stny
04-28	Bugfixing for released project Bachelor NORS	1:13:32 14:48-16:02	Spoki0
04-28	Bugfixing for released project Bachelor NORS	0:22:14 18:09-18:32	Spoki0
04-28	Bugfixing for released project Bachelor NORS	1:17:20 20:28-21:45	Spoki0
04-29	Daily Scrum Meeting Bachelor NORS	0:10:00 09:00-09:10	Spoki0
04-29	Daily Scrum Meeting Bachelor NORS	0:10:00 09:00-09:10	Rodland92
04-29	Daily Scrum Log Bachelor NORS	0:10:00 09:00-09:10	D3stny
04-29	Writing in LaTeX - Introduction Bachelor NORS	4:17:25 09:30-13:47	Rodland92
04-29	Bugfixing for released project Bachelor NORS	1:03:57 10:59-12:02	Spoki0
04-29	Thesis Writing - Replication Bachelor NORS	3:01:00 12:03-15:04	Spoki0
04-29	Fixing spawn area Bachelor NORS	1:13:00 12:47-14:00	D3stny
04-29	Fixing main attack bug Bachelor NORS	0:11:36 15:24-15:35	D3stny

04-29	Bugfixing for released project	0:32:00	Spoki0
	Bachelor NORS	15:35-16:07	
04-29	Fixing spawn area	0:58:33	D3stny
	Bachelor NORS	15:54-16:52	
04-29	Writing in LaTeX - Spells	1:48:35	Rodland92
	Bachelor NORS	19:30-21:18	
04-30	Daily Scrum Meeting	0:15:00	Spoki0
	Bachelor NORS	09:00-09:15	
04-30	Daily Scrum Meeting	0:10:00	Rodland92
	Bachelor NORS	09:00-09:10	
04-30	Daily Scrum Log	0:12:21	D3stny
	Bachelor NORS	09:04-09:16	
04-30	Writing in LaTeX - Spells	4:20:00	Rodland92
	Bachelor NORS	09:20-13:40	
04-30	Bugfixing for released project	1:06:29	Spoki0
	Bachelor NORS	10:15-11:21	
04-30	Thesis Writing - Deployment	2:24:53	Spoki0
	Bachelor NORS	11:21-13:46	
04-30	Fixing spawn area	0:39:03	D3stny
	Bachelor NORS	12:28-13:07	
04-30	Writing GUI	2:08:28	D3stny
	Bachelor NORS	13:07-15:16	
04-30	Thesis Reading	1:39:12	Spoki0
	Bachelor NORS	13:46-15:25	

Detailed report



2015-05-01 - 2015-05-31

Total 277 h 27 min

Date	Description	Duration	User
05-01	Daily Scrum Meeting Bachelor NORS	0:10:00 09:00-09:10	Spoki0
05-01	Daily Scrum Meeting Bachelor NORS	0:13:00 09:00-09:13	Rodland92
05-01	Writing in LaTeX - Spells Bachelor NORS	5:04:00 09:30-14:34	Rodland92
05-01	Bugfixing for released project Bachelor NORS	4:45:00 12:00-16:45	Spoki0
05-01	Trying out latex and compile and write about general AI and blender Bachelor NORS	2:47:00 20:22-23:09	Kress92 Cs
05-04	Meeting with Simon Bachelor NORS	0:50:00 09:00-09:50	Spoki0
05-04	Weekly Scrum Meeting Bachelor NORS	0:54:00 09:00-09:54	Rodland92
05-04	Bugs, meetings, packaging, camera and targeting of minion glitchx Bachelor NORS - [mobile]	5:44:00 09:03-14:47	Kress92 Cs
05-04	Fixing Bugs for playtesting Bachelor NORS	4:53:00 10:00-14:53	Rodland92
05-04	Pair programming Bugfix for release Bachelor NORS	3:47:03 10:03-13:50	Spoki0
05-04	Fixing bugs and getting ready for testing Bachelor NORS	4:12:34 10:23-14:35	D3stny
05-04	Bugs, meetings, packaging, camera and targeting of minion glitchx Bachelor NORS - [mobile]	8:59:00 15:48-00:47	Kress92 Cs
05-04	Fixing more bugs Bachelor NORS	7:00:00 16:00-23:00	Rodland92
05-04	Pair programming Bugfix for release Bachelor NORS	1:15:21 19:22-20:37	Spoki0
05-05	Daily Scrum Meeting Bachelor NORS	0:10:00 09:00-09:10	Spoki0
05-05	Daily Scrum Meeting Bachelor NORS	0:10:00 09:00-09:10	Rodland92
05-05	Daily Scrum Log Bachelor NORS	0:10:00 09:00-09:10	D3stny
05-05	fixing bugs Bachelor NORS	7:30:00 09:00-16:30	Kress92 Cs
05-05	Writing about Ai controllers Bachelor NORS - [mobile]	6:11:37 09:08-15:20	Kress92 Cs

05-05	Thesis Writing - Deployment Bachelor NORS	2:33:52 10:30-13:03	Spoki0
05-05	Writing GUI Bachelor NORS	1:51:31 13:18-15:10	D3stny
05-05	Thesis Writing - Technical Bachelor NORS	1:20:00 14:40-16:00	Spoki0
05-05	Bug fixes Bachelor NORS	1:35:00 15:20-16:55	Rodland92
05-05	Fixing bugs and getting ready for testing Bachelor NORS	1:30:00 15:30-17:00	D3stny
05-05	Debugging for release build Bachelor NORS	0:49:00 16:00-16:49	Spoki0
05-05	Gametesting Bachelor NORS	1:28:53 16:57-18:26	Spoki0
05-05	Playtesting Bachelor NORS	1:20:00 17:00-18:20	Rodland92
05-05	Playtesting Bachelor NORS	1:00:00 17:00-18:00	D3stny
05-05	Game testing with kids Bachelor NORS	2:00:00 17:00-19:00	Kress92 Cs
05-06	Daily Scrum Meeting Bachelor NORS	0:10:00 09:00-09:10	Rodland92
05-06	Daily Scrum Meeting Bachelor NORS	0:10:00 09:00-09:10	Spoki0
05-06	Daily Scrum Log Bachelor NORS	0:10:00 09:00-09:10	D3stny
05-06	Fixing togg Bachelor NORS	0:25:00 09:20-09:45	Rodland92
05-06	Writing in LaTeX - Blueprints Bachelor NORS	5:18:00 09:50-15:08	Rodland92
05-06	Making diagram Write about Blueprints Structure Bachelor NORS	8:05:00 10:00-18:05	Kress92 Cs
05-06	Thesis Writing - Technical Bachelor NORS	1:29:31 10:09-11:38	Spoki0
05-06	Writing GUI Bachelor NORS	3:04:05 11:43-14:47	D3stny
05-06	Bugfixing for released project Bachelor NORS	5:32:44 11:44-17:16	Spoki0
05-06	Bug fixes Bachelor NORS	2:37:00 15:30-18:07	Rodland92
05-06	Fixing bugs Bachelor NORS	3:28:14 16:03-19:32	D3stny
05-07	Daily Scrum Meeting Bachelor NORS	0:10:00 09:00-09:10	Spoki0
05-07	Daily Scrum Meeting Bachelor NORS	0:15:00 09:00-09:15	Rodland92

05-07	Writing in LaTeX - Testing Bachelor NORS	5:41:22 09:30-15:11	Rodland92
05-07	Thesis Writing - Terminology Bachelor NORS	1:35:59 10:25-12:00	Spoki0
05-07	Writing about AI General And blender Bachelor NORS - [mobile]	11:19:00 11:12-22:31	Kress92 Cs
05-07	Thesis Writing - Discussion Bachelor NORS	2:25:25 12:00-14:26	Spoki0
05-08	Daily Scrum Meeting Bachelor NORS	0:13:00 09:00-09:13	Rodland92
05-08	Writing in LaTeX - Testing Bachelor NORS	5:40:30 09:30-15:10	Rodland92
05-08	Daily Scrum Meeting Bachelor NORS	0:15:00 11:00-11:15	Spoki0
05-08	Thesis Writing Bachelor NORS	1:25:29 14:00-15:25	Spoki0
05-09	Writing about AI General Bachelor NORS	6:06:36 15:28-21:34	Kress92 Cs
05-09	Playtesting Bachelor NORS	2:30:00 17:01-19:31	Rodland92
05-09	Playtesting Bachelor NORS	2:45:00 17:15-20:00	Spoki0
05-10	Making psudocode structures Bachelor NORS	3:18:36 20:10-23:28	Kress92 Cs
05-11	Meeting with Simon Bachelor NORS	0:59:00 09:00-09:59	Spoki0
05-11	Weekly Scrum Meeting Bachelor NORS	2:30:00 09:00-11:30	Rodland92
05-11	Meeting with Simon Bachelor NORS	1:00:00 09:00-10:00	D3stny
05-11	Meeting with Simon, Team meeting deciding what to write Bachelor NORS	5:40:00 09:20-15:00	Kress92 Cs
05-11	Weekly Scrum meeting Bachelor NORS	1:20:00 10:00-11:20	Spoki0
05-11	Weekly Scrum Bachelor NORS	1:00:00 10:00-11:00	D3stny
05-11	Writing in LaTeX - Testing Bachelor NORS	4:39:16 12:00-16:39	Rodland92
05-11	Writing GUI Bachelor NORS	0:47:22 13:28-14:15	D3stny
05-11	Thesis Writing - Implementation Bachelor NORS	1:15:00 13:45-15:00	Spoki0
05-11	Thesis Writing - Introduction Bachelor NORS	0:47:49 15:00-15:47	Spoki0

05-11	Thesis Writing - Requirements and Design	1:06:14	Spoki0
	Bachelor NORS	15:47-16:54	
05-12	Daily Scrum Meeting	0:17:00	Spoki0
	Bachelor NORS	09:00-09:17	
05-12	Daily Scrum Meeting	0:10:00	Rodland92
	Bachelor NORS	09:00-09:10	
05-12	Daily Scrum Log	0:15:00	D3stny
	Bachelor NORS	09:00-09:15	
05-12	Writing about Behavior tree	2:48:00	Kress92 Cs
	Bachelor NORS	09:23-12:11	
05-12	Writing in LaTeX - Testing with 2. years	3:22:00	Rodland92
	Bachelor NORS	09:30-12:52	
05-12	Thesis Writing - Requirements and Design	2:46:00	Spoki0
	Bachelor NORS	10:10-12:56	
05-12	Blender writing	1:24:00	Kress92 Cs
	Bachelor NORS	11:00-12:24	
05-12	Writing about inventory and shop	1:12:52	D3stny
	Bachelor NORS	12:52-14:05	
05-12	Writing in LaTeX - Testing with 2. years	5:00:00	Rodland92
	Bachelor NORS	14:00-19:00	
05-12	Writing about inventory and shop	0:22:56	D3stny
	Bachelor NORS	17:04-17:27	
05-12	Writing about Behavior tree	3:12:00	Kress92 Cs
	Bachelor NORS	18:32-21:44	
05-12	Writing about inventory and shop	1:14:00	D3stny
	Bachelor NORS	19:25-20:39	
05-12	Thesis Writing - Technical	1:57:08	Spoki0
	Bachelor NORS	20:24-22:21	
05-13	Daily Scrum Meeting	0:30:00	Spoki0
	Bachelor NORS	09:00-09:30	
05-13	Daily Scrum Meeting	0:13:00	Rodland92
	Bachelor NORS	09:00-09:13	
05-13	Daily Scrum Log	0:30:00	D3stny
	Bachelor NORS	09:00-09:30	
05-13	Writing in LaTeX - Finished testing and user feedback, going over old stuff	5:50:00	Rodland92
	Bachelor NORS	09:30-15:20	
05-13	Thesis Writing - Development	2:00:00	Spoki0
	Bachelor NORS	10:00-12:00	
05-13	Writing about AIController And minionStructure	7:56:00	Kress92 Cs
	Bachelor NORS	10:01-17:57	
05-13	Write about items	0:55:51	D3stny
	Bachelor NORS	11:05-12:01	
05-13	Thesis Writing - Implementation	4:44:00	Spoki0
	Bachelor NORS	12:30-17:14	

05-13	Write about items Bachelor NORS	0:59:03 19:03-20:03	D3stny
05-13	Thesis Writing - Implementation Bachelor NORS	1:24:56 22:21-23:46	Spoki0
05-13	Going through all, checking references Bachelor NORS	0:56:00 22:34-23:30	Rodland92
05-14	Thesis Writing - Implementation Bachelor NORS	0:59:23 10:00-10:59	Spoki0
05-14	Writing about Behavior tree Bachelor NORS	13:25:00 10:34-23:59	Kress92 Cs
05-14	Going through the Thesis Bachelor NORS	6:30:00 11:00-17:30	Spoki0
05-14	Final wrap up Bachelor NORS - [mobile]	12:59:00 11:00-23:59	Rodland92
05-14	Finishing the thesis Bachelor NORS	13:00:00 11:00-00:00	D3stny
05-14	Going through the Thesis Bachelor NORS	5:05:00 18:54-23:59	Spoki0