DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics: Games Engineering

Serious Game: Ancient Battles in VR

Steen Müller

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics: Games Engineering

Serious Game: Ancient Battles in VR

Serious Game zu antiken Schlachten in VR

Author:	Steen Müller
Supervisor:	Prof. Gudrun Klinker, Ph. D.
Advisor:	Dipl.-Inf. Univ. David A. Plecher, M.A.
Submission Date:	15.11.2017

I confirm that this master's thesis in informatics: games engineering is my own work and I have documented all sources and material used.

Munich, 15.11.2017 Steen Müller

Acknowledgments

I would like to thank my thesis advisor Dipl.-Inf. Univ. David Plecher. He allowed me to choose my own topic to work on and provided me with helpful guidance whenever I needed it.

I would also like to thank Eric Massenberg and Thi Van Khanh Tran for providing assets for my thesis as well as continued support. Additionally, I would like to thank the participants of my survey for their time.

Finally, I want to express my gratitude to my parents for supporting me through my years of study.

Abstract

This work discusses the possiblities for a serious game about ancient battles. The game should teach the player about the tactics which were used in the battle. Additionally, the reasons for using a Virtual Reality headset are explained and the design consequences are discussed. A prototype is implemented in Unity3D using a HTC Vive and its touch controllers.

Diese Masterarbeit diskutiert die Möglichkeiten eines Serious Game über Antike Schlachten. Das Spiel lehrt dem Spieler die Taktiken, die von den Kommandeuren verwendet worden sind. Die Gründe für die Verwendung von Virtual Reality werden erklärt und darausfolgende Designentscheidungen werden diskutiert. Abschließend wird ein Prototype in Unity3D mit der HTC Vive erstellt.

Contents

1. Introduction

The consume of video games grew extremely in the last decade and shows the potential of games to entertain people. People are having fun during the interaction with these fictional creations. There is a lot of research about the methods which make games fun but it is not completely discovered why some games are more fun than others. But games could also be used for educational purposes. The combination of something which people like to do and something which people have to do seems to be an interesting approach. The best parts of a fun game and the information you want to teach to somebody could result in new and interesting ways of teaching and learning.

1.1. Goal of the Thesis

The thesis tries to implement a Serious Game about the used tactics in ancient battles. To build a solid foundation for the game, existing games are analyzed. After that, requirements for the game are formulated. The used technologies and hardware choice are explained. The resulting constraints from this technologies and the timeframe are discussed. The implementation will afterwards be tested by a small-scale user study to test the serious game.

1.2. Outline

Each chapter matches closely to one of the goals of the thesis. After an introduction in the first chapter, the second chapter defines the terms Serious Game and Virtual Reality. The growing interest in these topics in the last years is explained. The possibilities for a combination of the two topics are explored. As a last point, a game itself is discussed and how it differentiates from visualizations.

The third chapter discusses existing works. In the first part, existing games with the same topic are analyzed based on their efficiency in teaching battle tactics. The second part contains a discussion about other related work like visualizations or Serious Games in Virtual Reality but not with the same topic.

The analysis and the definition of the resulting requirements are presented in the fourth chapter. At first, the learning content including the actual historical battle are analyzed. The following sections debate the various possibilities for the visualization of the content. Other areas of game development like gameplay, Artificial Intelligence or User Interface are discussed.

The fifth chapter shows the actual implementation of all important areas of the game. This ranges from the graphical representation, the interaction through the controllers, the replay mode, the User Interface, the gameplay to the Artificial Intelligence in the game.

During the sixth chapter, the Serious Game is evaluated by a user study. The goals of the study are explained and the results are discussed.

The thesis ends with a conclusion and an outlook of the possible future work to improve this game.

2. Terms and Definitions

In this chapter two important terms for this master thesis are defined. Additionally, the relation to the topic of this thesis is explained.

2.1. Serious Game

The term Serious Game was first formally defined in 1970 by Abt [Abt87]. Due to the lack of an existing video game industry the term was not connected to video games but more to Pen-and-Paper games. In 2002 Sawyer provided an updated definition of the term Serious Games which directly connected it to video games[SR08]. He connected video games with a serious part like distributing knowledge or teaching about an existing model.

This thesis will use the following definition of Serious Games from Djaouti [DAJ11]: "Any piece of software that merges a non-entertaining purpose (serious) with a video game structure (game)." The definition contains two parts, the first one is the non-entertaining purpose. Most often this relates to the distribution of knowledge or the management of specific processes. The second part consists of the video game structure. This video game structure helps to catch the attention of the user. Additionally, the reward structure of a video game is used, so that the user will continue to use the Serious Game application and continue to learn more content and consolidate existing learning content.

Serious Games represent a growing market which is estimated to reach a market volume of 5450 millions by 2020 [mar]. Many companies like Audi [stra] or Allianz [strb] already use Serious Games applications.

As seen above, the topic Serious Games represents an interesting area for research. The combination of learning content with video games could lead to exciting new games which could partially replace existing learn methods.

To show these exciting possibilities, this thesis will try to create a Serious Game about ancient battles. The game should teach the user general knowledge about these battles regarding the location or time frame, but more importantly about the used tactics and provides various perspectives to watch the battle. The game should offer the possibility to just watch the battle but also take part as one of the commanders. This allows the player to explore the tactics firsthand. By changing the movement of the units, the player can try to change the course of the events of the battle and win as the historically losing side. But he can also switch to the winning side and try to win even more decisively. The new possibilities due to the game structure allow the player to learn more about the battle and the used tactics. The balance between game content, e.g. the possibility to play as the losing side and win the battle, and learning content, e.g. the historical correctness, is one of challenges of this thesis.

2.2. Virtual Reality

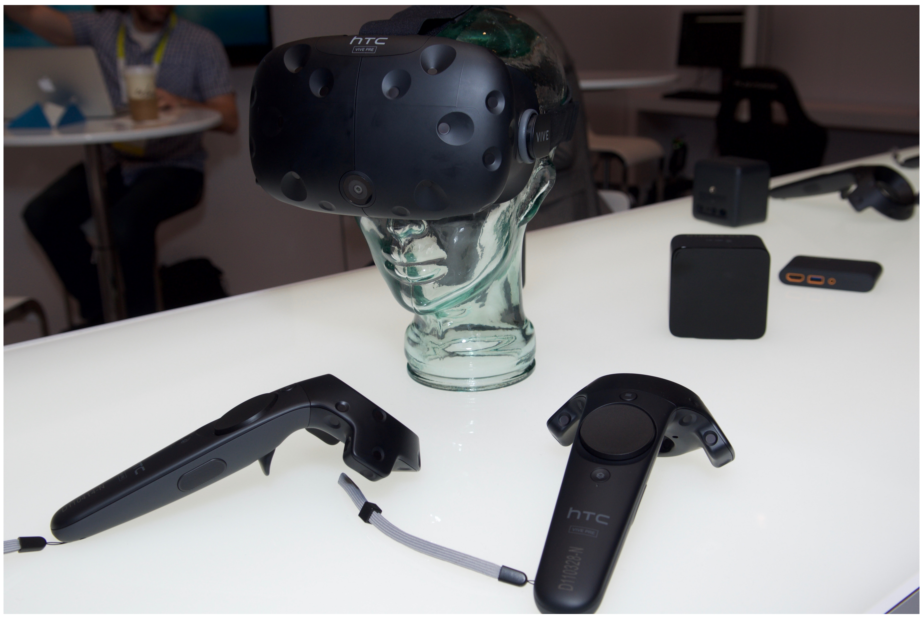


Figure 2.1.: Vive and its controllers, in the back base stations

Source: [ETC]

Virtual Reality (VR) is defined as a computergenerated reality [Pro]. This reality can be viewed on a head mounted display (HMD) or specific rooms which are called Cave Automatic Virtual Environment (CAVE). The possible forms of interaction depend on the used head mounted display. Some can detect the motion of the headset and calculate the respective changes to the virtual reality. Others use additional hardware like a 3D mouse. Some head mounted displays require a powerful external computer to calculate the video and audio streams for them (e.g. HTC Vive [HTC]), while mobile solutions often utilize the calculation power and the screen of a smartphone (e.g. Samsung Gear VR [Sam]).

Augmented Reality (AR), another topic which is often mentioned, differs from Virtual Reality by the fact that it only produces a computer generated picture which modifies the existing reality, so that additional information can be conveyed. Virtual Reality produces a computer generated picture and only shows the picture and not the reality in front of you. An example for an augmented Reality device would be the Microsoft HoloLens [Mic].

The popularity of the HTC Vive and the Oculus Rift after the launch of the sale of the consumer versions in 2016 shows the potential of the market. The market is estimated to grow to $48,5 billion by 2025 [Gra]. The two biggest companies in this market area are the HTC Corporation with the HTC Vive and the Oculus VR LLC with the Oculus Rift.

This thesis will develop the game for the HTC Vive and its controllers. The headset and the controllers are motion tracked by the base stations. Each controller features a trackpad, a trigger in the back, haptic feedback and 24 sensors. The development of a control scheme for this new interaction method will be one of the challenges for this thesis.

The tracking of the headset and the controllers allows an experience which is way more immersive than the standard experience in front of a monitor. The headset is closed at the sides in order to prevent light from shining through which also helps to focus on the screens. But this also poses problems like motion sickness. Some people experience motion sickness due to the imperfect movement alignment of the head and limbs with their real counterparts.

2.3. Serious Game in Virtual Reality

The combination of both components can result in great Serious Games. With the advantages of Virtual Reality like new control methods and increased immersion, the effect of Serious Games could be greatly amplified. If these new methods can be researched and used to create better and more effective Serious Games, a lot of people would benefit from that. Serious Games are a growing market as shown above and will influence a lot of people in the coming years especially in corporate environments.

A problem with Serious Games in Virtual Reality is the cost of the Virtual Reality headset. Most people will not buy a headset because they are rather expensive and require a good Personal Computer to run on. But in a corporate environment, this could change. If the employees of the company greatly benefit from the knowledge, which is transfered through the Serious Game, the employer will set up a Virtual Reality environment at the workspace for his employees.

2.4. Games versus Visualization

Another point worth mentioning is the difference between games and visualizations.

A game in itself is a form of structured play. Play is a range of activities which are voluntary and intrinsically motivated [Gar90]. The game itself should be rewarding enough to be played by an user. In game design this can often be achieved by using specific structures and reward mechanism. But in the context of Serious Game there is a balance problem. Serious Games contain a serious purpose like teaching the tactics of an Ancient Battle. The developer has to balance the amount of attention on the serious purpose with the rewarding, intrinsically motivating parts. If he focuses too much on the serious purpose, the game is not fun enough to play and people will not recognize it as a game. If he focuses too much on the game part, people will not actually learn the serious aspiration and the purpose of the Serious Game is lost.

Another important part of game is the interactivity in comparison to visualization. If you want to teach people e.g. about the tactics Hannibal used in the Battle of Cannae, you can show them a documentation. But the documentation lacks the interactivity. The user does not actually give any commands or can try another strategy to see how that one works. Games feature this decision making and it is a really important component of them. It also allows the player to actually take part in the battle so that he feels more immersed into it.

But this also poses a problem in the context of Serious Games. If you try to add as many decisions as possible to the game to enable the player the possibility to make as many decisions as possible, the historic authenticity will most likely suffer from that. Also, if there are a lot of decisions, the player may not actually learn something about the real historic course of events but just about his battle in Cannae. The decision making has to be constrained to some degree in order to guarantee the accuracy of the historic course of the events. Finding a balance in this respect poses one of the major challenges in this thesis.

3. Related Work

This chapter discusses related work and tries to define the borders of the topic. Also, differences to existing games are mentioned. After extensive research through various online distribution methods, no exact match for this combination of ancient battles and Virtual Reality could be found. There are some existing games which fit either Virtual Reality or ancient battles but not both. The first section will discuss these games. The games will be analyzed to see if they are suitable as a learning environment for tactics in ancient battles. Additionally, some other related applications in the Virtual Reality field are discussed in the second section of this chapter.

3.1. Related Existing Games

The following games are related to the theme ancient battles. This section will discuss how close they match real existing battles and if they are able to show the used tactics in these battles.

3.1.1. Alexander

Alexander is a Real Time Strategy game which was developed by GSC Game [GSC] and published by Ubisoft. It was developed in 2004 as a tie-in to the film with the same name. The film tries to show a mostly realistic representation of the life of Alexander, but there are some differences to the real life of Alexander. The game was not well received and scored a rating of 56 on Metacritic [Met].

During the campaign of the game, you follow Alexander on his campaign through the Persian Empire until India. After finishing Alexander's campaign, you can fight in three additional campaigns as the Indian, Persian or Egyptian Empire.

The game is developed with the same engine and gameplay principles as Cossacks. Due to this, the population limits and the amount of units in this game are rather high compared to other strategy games like Age of Empires. This can be seen on the website where the developer promotes his game with battles with up to 64000 units. The player can group units to brigades which allows to command these unit amounts more easily. The use of these brigades is closer to the historical reality, where sometimes thousands

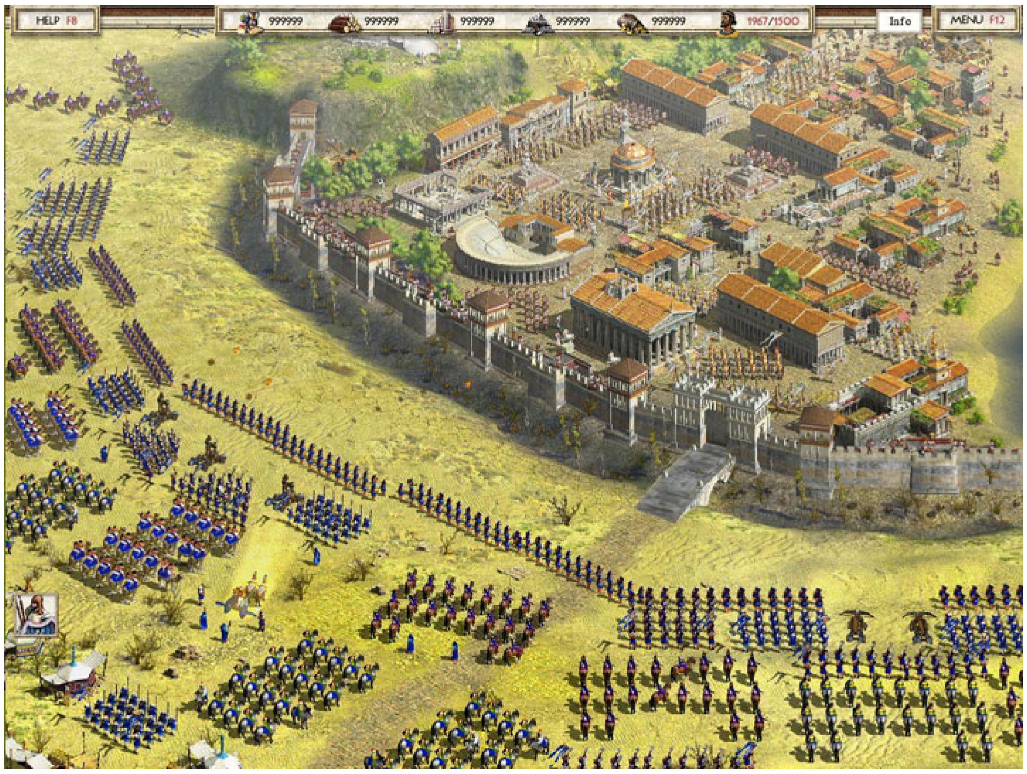


Figure 3.1.: Screenshot from the game Alexander (2004)

Source: [GSC]

of soldiers fought against each other, than the battles in Age of Empire where only dozens of units fight against each other.

Due to the lack of owning the game and the lack of existing video material, the game's controls can only be rated by existing reviews of Alexander. According to reviews from IGN [But] or Gamespot [Bee], the controls often do not work. The battles of Gaugamela or Issus, where Alexander won due to his strategies, do not show the need for his used tactic in the game. The units often do not act according to their commands and it is often just enough to send all units to the enemy and wait for the victory. Tactical possibilities like flanking and formations are not even needed because the player can win even without them.

To sum the game up, it uses some historical correct facts like the place of some battles and the amount of units but is not able to teach the player about the used strategy of Alexander.

3.1.2. Ultimate Epic Battle Simulator

The Ultimate Epic Battle Simulator [Bria] was developed by Brilliant Game Studios [Brib]. The game is a sand box for battle simulation. It allows the player to let various humans like archers, roman soldiers or Chuck Norris fight against each other. Other units like orcs, trolls or chickens are also available. The developer designed the simulator to not limit the number of units but the performance will decrease at a certain point.

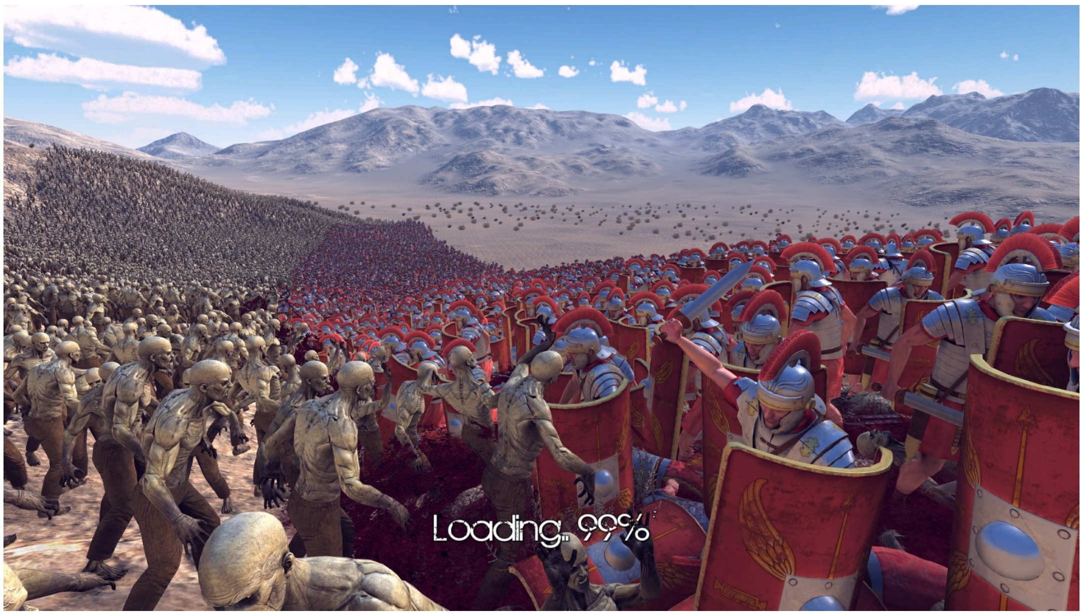


Figure 3.2.: Screenshot from the game Ultimate Epic Battle Simulator

Source: [Edu17]

After starting the battle, it is not possible for the player to control the units anymore except for fighting as one single soldier. The player is not able to arrange a strategy for his units. This dramatically reduces the use cases of this simulator as a learning environment. It is not possible to build custom maps to recreate ancient battles because you are restricted to the set of maps which the developer provides. Due to these constraints which are imposed by the game, it is not suitable as a learning environment about ancient battles.

3.1.3. Total War

The Total War Series [SEG] was developed by Creative Assembly [Cre17]. A total of 11 parts have been developed in the last 17 years. The series is about the combination of round based management of territories and real-time battles between various factions.

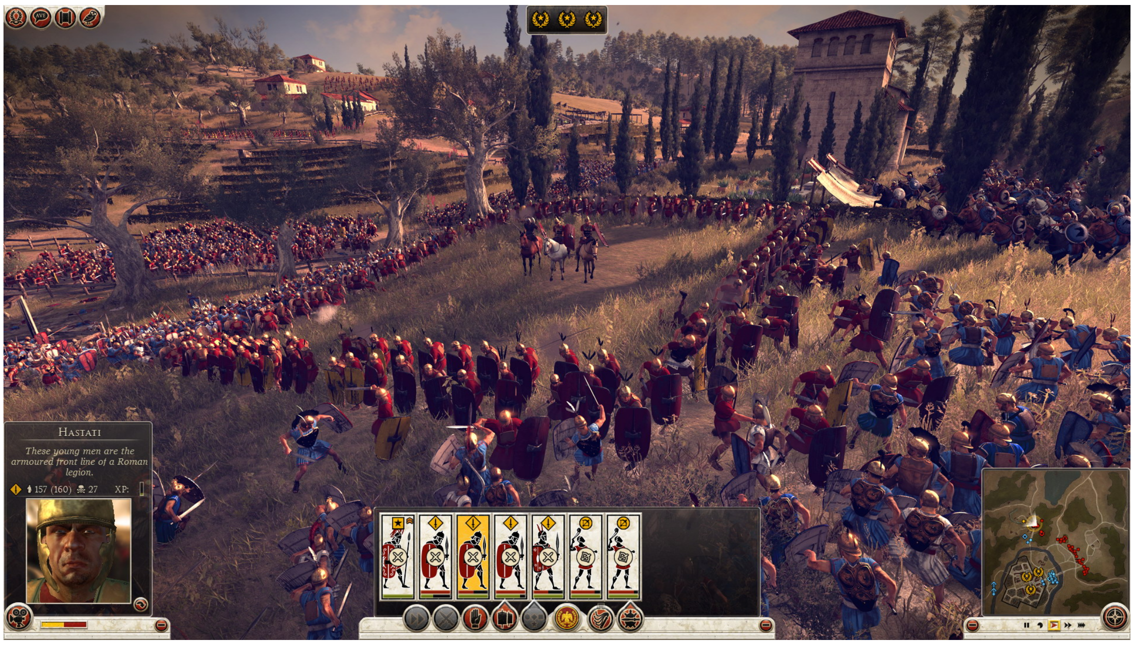

HASTATI
These young men are the armoured front line of a Roman legion.
157 (160)
27
XP:

Figure 3.3.: Screenshot from the game Total War: Rome 2

Source: [M13]

This thesis will concentrate on the game Total War: Rome 2. It is a remake of the Game Rome : Total War which was developed in 2004. The game features a large campaign which starts in 272 BC and lasts for 300 years. But also some historic scenarios were added to the game like the Battle of the Teutoburg Forest. These scenarios only feature a battle and no round based management.

The game contains around 500 different unit variants [Sch]. In the real-time battles, each unit is represented by a number of soldiers and you can field up to 40 units. You only control the unit as a whole. Each unit has various values like morale, fighting strength or movement speed. Additionally, you can choose different formations for your units to form your army exactly like you want. Units whose morale is too low will flee from the battle. The goal of a battle is to kill the enemy or make them flee.

The battles are comparably good at representing ancient battles due to formations, morale and the use of the real information about these units which are visible in

Screenshot 3.3. But there are also some inconsistencies like the amount of soldiers in most battles. The game reduces the amount of soldiers for performance reasons to a few thousand in most battles but in ancient battles during the Roman Empire often the number of participating soldiers was in the tens of thousands (e. g. Battle of Cannae 120 000 Soldiers). Additionally, the game represents cavalry very strong. This is partially correct as outflanking an enemy phalanx unit is rather strong and creates high casualties, but in the game cavalry was able to win against almost any kind of infantry. This is shown during multiplayer matches where most players relied mainly on cavalry and some "support" infantry. Still, the battles in Total War: Rome 2 represent the best approach from the analyzed games for showing the tactics in ancient battles.

The television show Time Commanders [BBC] was a game show which aired on BBC. The participants in the show were briefly introduced to the units and the terrain of one ancient battle and were then asked to develop a strategy. After developing a strategy, it was tested in a match versus a bot commander. During the match, military specialists explained the difference to the real historic battle. The show used the Total War Engine with slight modifications to show the battles and have a real-time visualization of strategy and the reaction of the AI. This shows that the historic background of the game was rather good as it could be used in a television show.

3.2. Other related Applications

This section discusses other related applications and their usefulness to the project.

3.2.1. Serious Game in Virtual Reality

A Serious Game for travel training for persons with Autism Spectrum Disorder was developed [Ber+15]. This game uses Virtual Reality to immerse the players into the world. As seen in their paper, they use a virtual environment in a Virtual Reality headset to display tasks to the player. These tasks should help players with ASD to interact in the real world with more confidence.

The game is not directly related to the topic as it does not feature any ancient battles or tactics. But they use a Virtual Reality headset to increase the immersion of their serious game.

3.2.2. Rome Reborn

Rome Reborn is a visualization of the ancient Rome over a timespan of circa 1550 years from 1000 B.C. to 550 A.D.[Fri]. It shows the buildings in the city and provides explanations for various things. It uses the Virtual Reality headset for immersion purposes and it can not be considered as a game because it lacks interactivity with the content.

3.3. Summary

The goal of this chapter was to find an existing Serious Game which relates to the topic of ancient battles and uses a Virtual Reality headset like the HTC Vive in its setup. The existing games with a relation to ancient battles did not have a serious purpose and were missing the Virtual Reality headset. Out of these games Total War: Rome 2 has the best approaches for providing a correct, historic experience for ancient battles. The author was not able to find an existing Serious Game about ancient battles. An existing Serious Game which uses a Virtual Reality headset was shortly analyzed but found unsuitable for the topic of this thesis.

4. Analysis and Definition of Requirements

In the following sections, an existing battle will be analyzed and the requirements for the serious game will be discussed. The goal is to develop a Serious Game about ancient battles. For this purpose a battle simulator part with an artificial intelligence needs to exist so that the player can test the strategy. It would be also beneficial to build a replay part where the player can watch the strategy which the commander used.

4.1. Learning Content

The serious purpose of the learning game is the distribution of knowledge. The knowledge in this case is an understanding of the tactics which were used by ancient commanders to win battles. Due to time constraints, it is advisable to focus on one specific battle and its tactics. If it is possible to teach a player about one specific battle and its tactics, this can be replicated for other battles as well.

Due to the mentioned reasons, this thesis will concentrate on one specific battle by an ancient commander. One option would be Alexander the Great who had several interesting battles like the one near Issus or Marathon. Another interesting commander would be Hannibal who led the troops during the second Punic War. Multiple interesting battles like the one at Lake Trasimene are available for an analysis. Also the Battle at Cannae is interesting because it represents a prime example for a pincer movement and is still considered as one of the best executions of this tactic. It was decided that this thesis will focus on the Battle of Cannae due to its good documentation by various sources and its importance for warfare. The battle will be further analyzed in the next section.

The main focus of this thesis are on the one hand the used tactics in that battle and on the other hand additional topics which are also considered to the list. One topic could be the communication on the battlefield. It was often rather hard for a commander to react to enemy movements and give commands after the battle had started. Due to the enormous number of ca. 86 000 soldiers on the roman side, it was hard to give orders to units. Additionally, it is rather hard to reach cavalry which are already using horses to move over greater distances. One solution were instruments but it remains to be seen if it is possible to include these in the game mechanics while still making it fun to play.

Another topic are basic information about the different units which took part in the battle. This information could range from equipment and experience in battle to general role in a battle. Also some additional information about the previous battles and Hannibals march through Italy could be shown. An interesting part is the connection between the real data and the values of the units in the game which are defined by combat system.

4.2. Historical Battle Of Cannae

In this section the Battle of Cannae will be analyzed. For this purpose the units of both sides, the location and the course of events will be discussed. The following sections will use the information from the book [Dal03].

4.2.1. Units (Roman Side)

The size of the Roman Army was estimated to approximately 63000 infantry soldiers and 6000 cavalry soldiers and additional soldiers as guards in the camp. The army consisted of three big unit groups:

- Light Infantry: Velites
- Heavy Infantry: Hastati, Principes, Triarii
- Cavalry: Eques

Light Infantry

The light infantry in the Roman Army was called Velites. Those wore only light armor because they moved around a lot and had to be mobile. As a helmet, they used a wolf skin helmet. They carried around 7 javelins, which are light throwing spears, and a parma shield, which is a small round shield, into the battle. They engaged in the typical skirmisher tactics by just throwing spears at the enemy and evading the enemy by running away if he charged at them. These spears from the skirmishers were the main weapon of the Roman Army versus the elephants in Hannibals Army although most of them already died during the march through the Alps.

Their role in combat was to hide the movement of the heavy infantry and cavalry by building a small wall and attracting the attention of the enemy by throwing spears at them. Later on in the battle, after they had thrown their spears, they supported the main army by closing gaps and transporting equipment. There were circa 24 000 velites in the Battle of Cannae [Dal03, P. 58].

Heavy Infantry

The heavy infantry represented the main part of the Roman Army. They consisted of three groups.

The first group, the Hastati, included the inexperienced and young soldiers, which were often not wealthy. Their equipment consisted of a chain mail body armor. The senate gave them a gladius, a small stabbing sword, and the scuta, the distinctive roman shield. Additionally to the gladius, they carried several pila into the battle. A pilum is a throwing spear, which the hastati used in close range. They were thrown against shields to make them unusable due to the added weight of the spear.

The role of the Hastati was not to win the fight for the army but to wear the enemy down and exhaust them. Then the stronger and more experienced troops would win the fight.

The second group, the Principes, were similar to the Hastati, but more experienced and wealthier. This lead to better equipment. Also, they were older than the Hastati.

Their role was to win the fight versus the enemy after the hastati had worn them down. If they were not successful, the third group of the heavy infantry came into play.

The Triarii were the most experienced soldiers in the army. They were often wealthy which lead to the best possible equipment. They were equipped in similar matter as the other two groups but they fought with spears as a main weapon instead of the gladius like hastati and principes.

If the triarii actually fought in a battle, then the last reserves of the Roman Army would had been mobilized. They only represented approximately 1/5 of the heavy army and were placed in the back of the army as they were considered the elite units.

The typical composition of a roman legion for that battle consisted of 1440 hastati, 1440 principes and 600 triarii [Dal03, P.58].

Cavalry

The cavalry soldiers in the Roman Army were called Equites. They were equiped with spears, swords, small round shields and some kind of body armor. It is not clear, which one was used due to missing documentation and contradicting sources. The participation in the cavalry instead of the infantry was considered more prestigious and was paid better, but the soldier had to pay for his own horse and equipment which was only possible for the wealthiest among the citizens.

Normally, the job of the cavalry was to protect the infantry from flanking attacks. The heavy infantry was the main force of the Roman Army and often won the fight for them. Due to this fact and the fact that the roman cavalry lost most of their battles during the Second Punic War, some considered the roman cavalry rather weak. But the losses during the Second Punic War can the attributed to the sheer number advantage (during the Battle of Cannae, 6000 Equites fought against around 10 000 cavalry units in Hannibals Army). During Scipio's campaign in Iberia, the cavalry won against the carthagian cavalry only if they had been deployed properly and led by a competent leader.

Organization

The army was organized in the following way. One century consists of a group of 72 man with a depth of 10 layers. Each century had its own centurion and some additional commanders. A maniple consisted of two century. Each of the 10 cohorts in a legion cosisted of a maniple of Velites, Gastati, Principes and a century of Triari. Each legion contained 10 cohorts and one ala. An ala is a unit of 300 Equites. This results in a size of circa 5300 man.

4.2.2. Units (Carthaginian Side)

The soldiers on the carthaginian side are a mix of a great amount of different culture groups. Due to this fact, it is rather hard to describe them in a short way. This also poses a problem for the game because it is rather difficult to represent all culture groups and their equipment adequate without using a great amount of time to model and texture the respective items. For this purpose, the game will only display a selected amount of groups in the game. The groups were chosen to be similar to the roman ones.

The carthaginian strategy requires at least four different groups:

- Cavalry
- Line Infantry
- Lybians
- Skirmishers

The cavalry consisted of three different culture groups. The Numidians fought more like mounted skirmishers and did not wear any armour. The other groups are the Celts and the Iberians which both fought as melee cavalry. They carried a small round shield, a spear and a sword.

The line infantry consisted mostly of Celts and Iberians. Most of them were not as experienced as the lybian soldiers in the army but still carried similar equipment to the roman troops. The main differences were the Falclata Espanada, a long slashing sword, and the scutarii, an oblong shield. The Lybians will be represented similar in the game.

Multiple culture groups were part of the skirmishers. The Balearian slingers were one group which used slings to throw stones at the enemy. The spearmen part of the skirmishers were a mix of various racial groups like Moors or Spaniards. Additionally, most skirmishers also carried a sword to defend themself after they had thrown their spear. Due to the wide range of different equipment and differences in race the equipment can not really be standardized.

4.2.3. Location

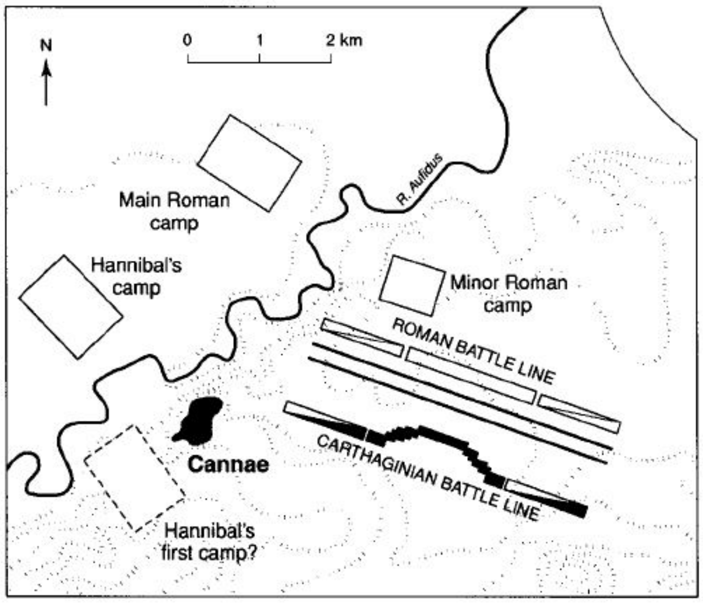


Figure 4.1.: Map of Kromayer's Theory

Source: [Dal03, p.34]

The location of the battle is still argued about. One of the most common views is the placement on the right bank of the Ofanto by Kromayer. The area is rather flat and the height of the slope is just 66 centimeter per 100 meter. The placement works pretty well according to the sources and does not provide one side with a major disadvantage which shows that it is the most plausible theory. But it has not actually been proven that it is the side of the battle.

It is rather difficult to exactly determine the position because the landscape changed a lot in the last 2000 years. Today, the area consists of mostly fields and does not provide any hints about the battle.

4.2.4. Course of Events

The description is taken from the book[Dal03, p.40-41].

The fight started with the usual deployment of the skirmishers. These stayed in front of the main battle line and allowed the heavy infantry in the back to position themselves without really being paid attention to. The battle of the skirmishers did not really grant one side a significant advantage over the other side. Shortly after, the skirmishers withdrew from the frontline and fell back.

The celtic and iberian heavy cavalry on the left wing started to fight versus the roman citizen cavalry, while the Numidians had been engaging with the roman ally cavalry. The heavy cavalry was able to to defeat the citizen cavalry and heavily exposed the flank of the Roman Army. The Numidians were only able to pin down the roman ally cavalry.

In the meantime, the roman heavy infantry marched forward into the carthaginian forces. The Celts and Iberians were placed in a rather thin cresent shape which slowly fell back to lure the Roman Army in. Hannibal placed the Lybians which were some of his strongest and most experienced soldiers at each side of his army as a column.

When the Roman Army marched forward, the Lybians faced inwards and attacked each side of the Roman Army. Additionally, the heavy cavalry attacked the rear end of the Roman Army, after helping the Numidians beating the roman ally cavalry.

4.3. Visualization of Units

The Battle of Cannae featured approximately 120 000 man. It is not possible to show a detailed version of every soldier in the game due to performance reasons. It is generally possible to show thousands of soldiers on screen and calculate them individually as it is shown in Total War: Rome 2, but Virtual Reality requires a higher amount of frames per second and thus performance than games on a monitor. The higher performance leads to reduced motion sickness.

Additionally, the actual state of the battle and the performance of single units is not really clear if you show too many units. It is often advisable to reduce the number of units to introduce clarity with regard to performance. Also, it is rather hard to command circa 70 000 units if each units needs to receive its own command.

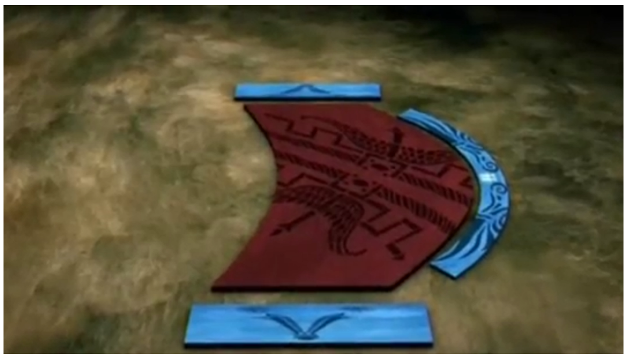


Figure 4.2.: Screenshot from a Documentation about Hannibal

Source: [Bed05]

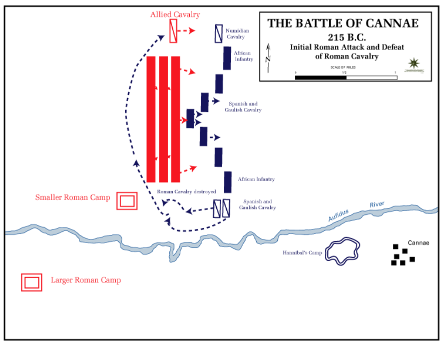


Figure 4.3.: Deployment of troops at the Start of the Battle of Cannae

Source: [The]

4.3.1. Reduction to rectangles

Due to the mentioned reasons above, a reduction from a realistic amount of soldiers was needed. One solution to this problem which is often used in documentaries is to only show the units as rectangle in their respective colors as visible in figure 4.2 and figure 4.3. These representations work quite well in documentaries because they allow the director to simply show the general strategy to the viewer but they are mostly not historically correct. The movement is simplified and the exact number of soldiers can only be guessed by the size of the respective rectangle.

For my purposes this representation is not suitable because it represents a two dimensional field. Due to the motion tracking of the Virtual Headset, the player is able to just move around and watch the battle from different angles. If the battle is just represented in two dimensions, this is not really possible.

Additionally, this representation does not really show the historical division into multiple legions. This missing division makes it hard to actually control the units in an interesting way. The Roman Army is represented with basically five unit blocks in figure 4.3. If you want to show the tactic which Hannibal used, you need a greater amount of units than that.

Due to these reasons, the author decided to use another representation which is discussed in the next section.

4.3.2. Graphical Representation as figures

Another possible representation is the reduction to a set of figures like in chess. During this reduction the army is represented by a number of figures which can can be moved on their own and embodies a set amount of soldiers. This allows to split the army into smaller groups like the legions in the Roman Army which increases the historic authenticity. Also the tactic can better be shown with a greater amount of units.

A graphic style which is similar to carved wooden figures would be helpful as it reduces the amount 3D modeling as well as removing the need for animations. In the time frame of this thesis it is not really possible to generate enough models and their animations.

Each unit should contain some unit specific items to show their role. Iconic items like the gladius sword and the scutum shield should be placed at the units to achieve two goals. One is to generate knowledge about the equipment of the units in that time period but also to make it possible to differentiate between the various unit types in the army. Additionally, a banner or flag would be helpful to show their association to a respective side. Also it could be used to convey game related information like experience.

4.4. Visualization of Terrain

The exact location of the battle can not be determined. Also, the landscape changed during the last 2000 years so todays landscape information can not be used as an exact replication of the location of the battle. The description of the location by the historic sources is rather broad and does not really mention specifics. The general difference in height on the battlefield was around 66 centimeter per 100 meter [Dal03, P.34]. This low difference makes the height an unimportant factor in the visualization of the map. A flat map representation of the map would be the best possible solution. It can also generate a "general"-like feeling for the player since he commands his unit around the battlefield.

4.5. Combat Model

The game needs to feature a combat model for the battle simulation part. The player should be able to fight his own battle versus the artificial intelligence. For the fights between units in these battles the damage on the units has to be determined. The combat system should calculate the battles in a somewhat historically correct matter. The replay part can feature this combat system as well but this is considered a "nice-to-have" feature because the replay part will mostly be scripted. The focus lies on the battle simulation part for this combat model.

The combat model should include values and calculations for various values like existing soldiers in a unit group, morale, ranged fights and melee fights. Also, the possibility for flanking should be given. Various stances like a defensive greek-style phalanx would be nice but are not required for the gameplay.

4.6. Gameplay

The gameplay should feel similar to a turn based strategy game like chess. The game uses turns and not real time for its movement advancements and fights. During the replay part, the player should be able to step forward and backward in the simulation of the battle.

In the battle simulation part the player should be able to give movement commands and attack nearby enemies. Additional features like stances for units, faster movement for a morale cost or partial retreats would be nice to have, but are not required.

4.7. User Interface

The user interface should contain information about the current player aswell as the state of the battle. Each unit should contain information about its respective values like morale, health or attack damage which are used by the backing combat model. The movement paths and targets for attacks should be shown in the interface.

Due to the use of a Virtual Reality headset the use of typical two dimensional user interface is not advisable as it is not easily controllable by the player who uses the controllers of the HTC Vive. The typical two dimensional user interface is attached to the screen, so a selection of elements is rather hard with controllers as they are represented as tracked objects in the playing space.

In Virtual Reality games it is often more helpful to use diegetic user interfaces instead [Ant]. These are user interface elements which are directly connected to the world and the user can interact with them. An example for this is the health bar element in Dead

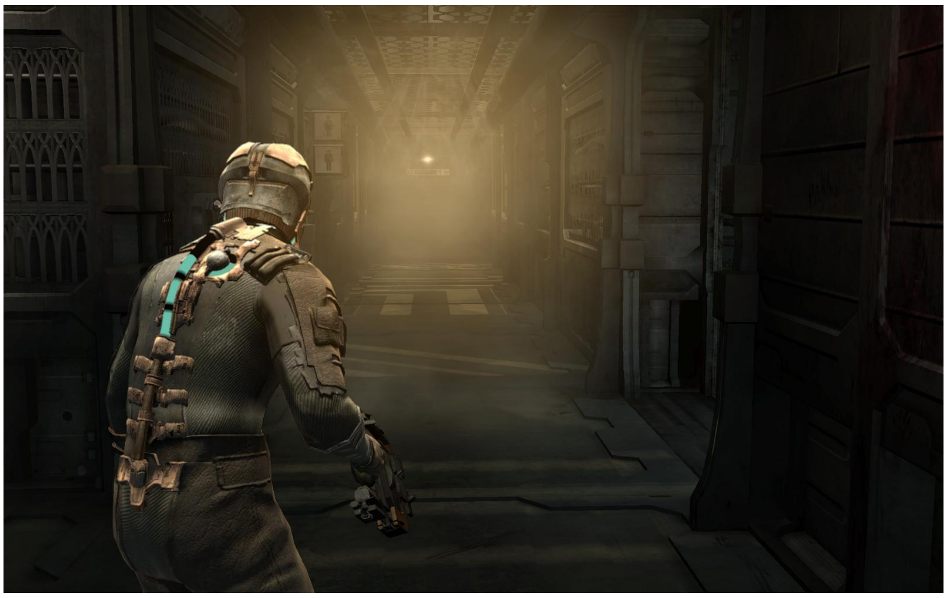


Figure 4.4.: Screenshot of the game Dead Space

Source: [Liu06]

Space in figure 4.4 which is shown at the back of the suit of the main character. The typical separation of the bar into four segments and as shrinking when low on health is still present, but its inclusion in the world creates a better immersion. Interactive elements can often be represented by real world interactions e.g. if you want to leave the current level, you open the door and try to go through it. The interaction is well-known to the player from his real world experiences and often does not need to be taught. The inclusion of these kind of interface elements would be advisable.

4.8. Artificial Intelligence

The battle simulation requires an artificial intelligence component. A basic requirement for a game is the interactivity with the medium, so that the player can actually choose an action. If the battle simulation part only has a step by step replay of the real steps of Hannibal, then the player can easily find a tactic which cancels out the replay steps. But if there is an artificial intelligence, which tries to react to the players actions, the player has to actually think about the used strategy. Therefore, an artificial intelligence is needed.

The units can move according to a set of movement rules, so the artificial intelligence has to find a way to find possible movement spots. In this case, this pathfinding is easier than normal because the player units rely on the same movement rules. Additionally, the respective stance or other actions has to be determined.

The goal of the Serious Game is to teach the player about the used tactics in the battle. Due to that, the artificial intelligence has to act like the ancient commanders and use their tactics. This is a rather hard requirement because it is difficult to teach an artificial intelligence a high level thought like "Use the cavalry on right flank to distract the enemy cavalry there" in a real time environment like a game.

The difficulty should not be set too high. It should still be fun for the player to play against it and the artificial intelligence should not react perfectly to the player movements. Finding the right solution for this problem is rather interesting and should be balanced around the players previous experience in strategy games.

A last requirement is the calculation time. Finding a perfect turn in a chess game is possible but too expensive to actually calculate it. The solution has to be found in a timely manner so that the player does not have to wait too long until his next turn. A good enough solution that fits the requirements is mostly sufficient.

4.9. Virtual Reality

Augmented Reality, e.g. a setup with a HoloLens which projects 3D models of units on various markers, was looked at but the missing haptic feedback of the three dimensional models of the units was considered too big of a problem. Augmented Reality could create some interesting new mechanics and experiences for games but the author considered Virtual Reality to be superior for this Serious Game due to its value in increasing immersion and allowing easier switches between different perspectives.

The game will use the Virtual Reality headset HTC Vive. This leads to some constraints for the design of the game. One constraint is the general performance and movement of the player. If one of them feels off, most players will experience motion sickness from the game. Another constraint is in the control of the game. The use of the controllers as visible in 2.1 is desirable as it allows to use the whole playing area instead of just a seated experience while using a keyboard or a gamepad. If you want to use these controllers, you have to find new ways for user interface elements as discussed in 4.7 and discover possibilities to convey the usual mouse and keyboard movements into the Virtual Reality setting since this translation had been proven to be difficult. Furthermore it should be tested how to make use of the different features the controllers offer, e. g. the touch pad.

4.10. General Goals

There are some general goals which should be reached. One goal is to reach a performance of 90 frames per second which corresponds to circa 11 milliseconds render time per frame. This leads to the most comfortable and fewest motion sickness inducing experience [BG].

Another goal of the game is the avoidance of any crashes. If the user tries to perform any action, the game should not crash.

5. Approach

This chapter will discuss the actual implementation and design choices during the development of the Serious Game.

5.1. Graphic Assets

Every game needs a graphical representation of most of its components. For this purpose, various objects were modeled or modified and textured. The author decided to go for a rather low polygon style which helps reducing the amount of time to model the actual object as well as the reduced render load on the engine. In the end, the amount of visible soldiers is around 350 at the maximum, so a reduction in detail is important to just provide a clearer overview over the battle. Additionally, a rather low polygon style fits with the intended carved wooden figures look.

5.1.1. Acknowledgements Graphic Assets

The time frame of approximately 4,5 months for developing the Serious Game reduced the possible count of graphic assets which the author could produce due to the accounted time which the other areas of development needed. Additionally, the skill set of the author led to increased times for the production of said assets. Therefore, the author decided to use some existing graphic assets. The acknowledgements for graphic assets can be seen in table 5.1.

5.1.2. Models

The figures need a representation of the human soldier as well as horses for the cavalry. Human bodies have been modeled many times before, so the use of an existing model seems to be a good approach. The used model [Yia17] provides a rather good base object, but the author decided to reduce the polygon count even further by decimating the object in blender. This option tries to reduce the amount of faces by dissolving triangles.

The resulting model can be seen in figure 5.1. After this step, the author decided that a rig for body is needed. This helps to move the bones of units to a pose for their

Asset	Author	Reference	License (if applicable)
Male Human Model	Panayiotis Yianni	[Yia17]	CC-BY-SA 3.0
Horse	Tom Higgins	[Hig17]	
Tileable Grass Textures	Van Khanh Tran	[Tra]	
Tileable Water Textures	Van Khanh Tran	[Tra]	
Wood Texture	Eric Massenberg	[Mas]	
Banner Textures	Eric Massenberg	[Mas]	
Parma Shield Texture	Eric Massenberg	[Mas]	
Scutum Shield Texture	Eric Massenberg	[Mas]	
Falcata Sword Texture	Eric Massenberg	[Mas]	
Gladius Sword Texture	Eric Massenberg	[Mas]	
Hasta Spear Texture	Eric Massenberg	[Mas]	
Pilum Spear Texture	Eric Massenberg	[Mas]	
Swordbelt Texture	Eric Massenberg	[Mas]	
Scutarii Texture	Eric Massenberg	[Mas]	
Verutum Spear Texture	Eric Massenberg	[Mas]	

Table 5.1.: Acknowedgements Graphic Assets

purpose as a soldier, otherwise each pose has to be modeled in blender which increases workload compared to a rig. Rigging is a rather time consuming work, so the author decided to use an auto rigging service provided by Mixamo[Mix23]. The website takes a 3D model in T-Pose and after setting some reference points for the elbows, chin, knees, groin and wrists the rig will be calculated. This does not provide a perfect rig, but is close enough to use in the game for the purposes of this thesis. Animations were considered, but for the purpose of this thesis not found important enough to actually implement them. Body armor, especially in the chest area, was another component which was considered, but, due to the possible need to provide a different body armor for each pose in the game, not found important enough.

The horse was taken from the Unity Asset Store [Hig17]. This horse was provided with a rig which was helpful, because the auto rigging service only works on human models. But the face count of the horse was too high, so the author decided to decimate the model as well. The resulting model can be seen in figure 5.2.

5.1.3. Roman Equipment

The sources for the roman equipment provide a far better overview over the used weapons and armor than the Carthaginian Army. This is due to the low amount of variety in the soldier types in the Roman Army in comparison to the Carthaginian

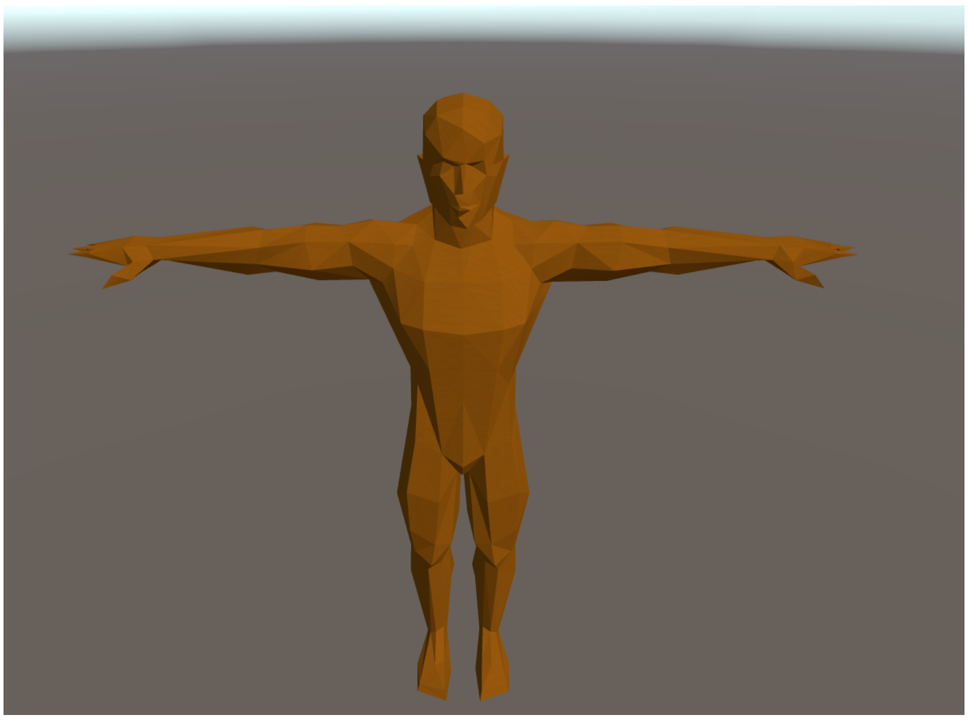


Figure 5.1.: Screenshot of the Human Body Model

Army where every culture group has a different set of weapons and armor. Although the game in this thesis wants to provide historic authenticity, the game will only use a reduced set of armor and weapons. This is due to the increased amount of 3D modeling and texturing needed for this while not providing a real benefit for showing the typical weapons and armor. Additionally, some weapons do not vary a lot according to the descriptions in the sources or are not clearly defined, like the shields of the Equites, so the author decided to reuse some weapons like spears for multiple units.

The models were made by the author but do not claim to be completely historically accurate due to the restrictions of a modern modeling environment as well as some stylistic decisions. Another restriction is the amount of polygons which can be used for each item. If the polygon count gets too high in the game, the performance will suffer from the increased render load. With this restrictions in mind the author tried to recreate the items as historically as possible with the least amount of polygons while staying at the same stylistic approach.

After some considerations, the following list of items were modeled by the author during the development and textured by Eric Massenberg:

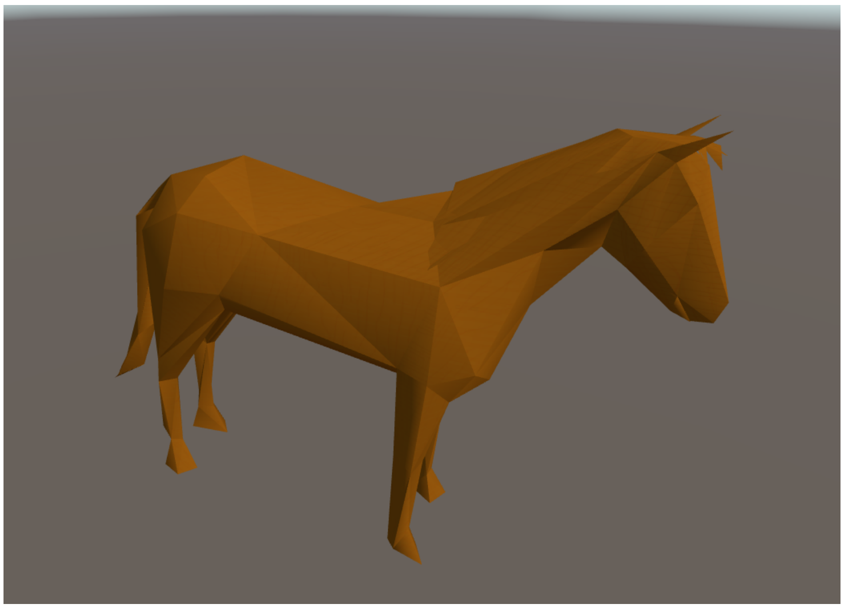


Figure 5.2.: Screenshot of the Horse Model

- Pilum: A throwing spear used by the Hastati and Principes
- Hasta: A rather long stabbing spear which was used by the Triarii and the Equites
- Gladius: A rather small sword which everyone in the game carries except the Velites
- Belt: A belt with a scabbard to carry the Gladius
- Scutum: The typical roman, rectangular, semi-cylindrical shield
- Parma: A small round shield carried by the Velites and Equites
- Montfort Helmet: A typical helmet for a soldier in the Roman Army

5.1.4. Carthaginian Equipment

The Carthagian Army used a lot of different weapons and armor types in their army. Most culture groups had very different sets of weapons even if they served in the same role e.g. the Balears with sling shots and the Moors with javelins. Additionally, the clothing differed from being naked like some of the celtic warriors to various kinds of armor. Some troops in Hannibals army also took the equipment of dead roman soldiers and wore it. Due to this enormous variety in equipment, the author decided to reduce this to a small subset due to the increased workload for modeling every kind of weapon.

This also helps the player to better understand the different roles in a battle like skirmisher or cavalry by providing a clearer identification with the respective role if there is only one type of equipment for each role. For this purpose, the author tried to mirror the roles and equipment to their roman counterpart.

The list of weapons and shields on the Carthagian side is as follows:

- Scutarii: An oval shield
- Espada Falcata: A slashing sword similar to Gladius
- Verutum: A short throwing spear
- Caetra: A small round shield for mobile troops

5.1.5. Triplanar Shading

The figures should be made out of wood to represent a carved wooden figure like in a wooden chess game. For this purpose, various ways for applying a wood texture on the figure have been analyzed.

One way to implement the wood texture would be a three dimensional texture [SM00]. Textures can be interpreted as a function which maps a two dimensional position from the respective uv map to the position on the texture map. Instead of a mapping from two dimensions to two dimension you can also use a mapping from three dimensions to a three dimensional texture. This kind of texture is defined as a three dimensional block. The three dimensional position of a point is taken and mapped to the three dimensional texture, so that the function results in a single color value. This technique removes the need for an UV map but also requires potentially a lot of disc space as the space requirement is increased by the factor eight if you double the resolution. The general idea is pretty applicable for wooden figures as they are carved out of a wooden block. The problem was the generation of a good looking three dimensional wood texture. Some papers [Liu+16] already discussed the implementation of such a texture but the missing source code and the wrong graphical style led to the discard of said paper. Experiments with the creation of a three dimensional wood texture led to non-desirable results. Therefore, the author decided to look for another approach to this topic.

Another approach to this problem would be the typical two dimensional texture approach. For each unit, the wood has to be UV mapped to a wood texture. This requires quite a lot of work although the same tileable wood texture can be used for

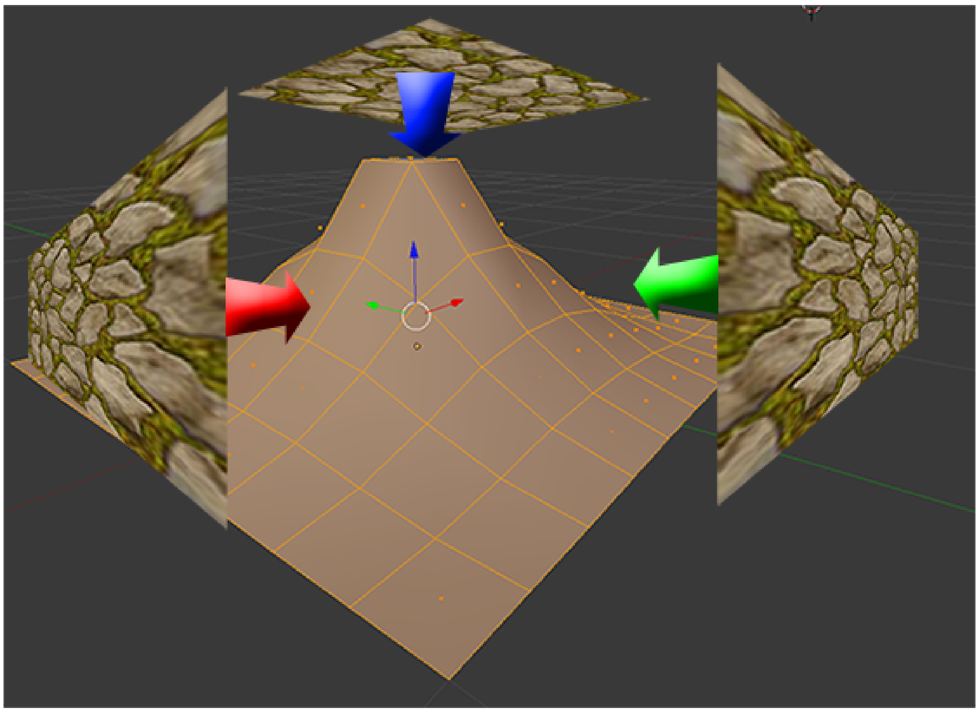


Figure 5.3.: Explanation Triplanar Shading

Source: [Owe]

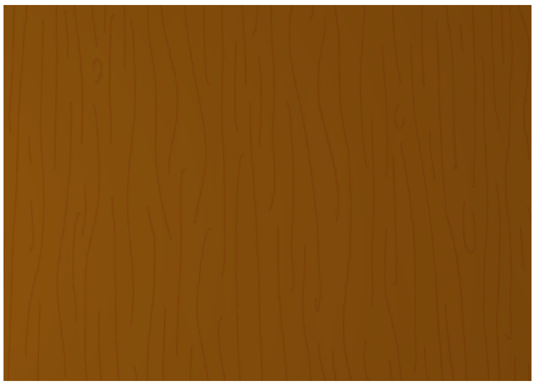
every model. Still the uv mapping work was considered rather large and another approach was tested.

Triplanar shading is another possible solution to this problem which removes the need for uv mapping. Triplanar shading takes three different input diffuse maps for the top, the side and the front. The shader then takes the world position from a fragment to determine the uv coordinates on the respective texture and sample it. Then the resulting color is blended based on the world normal of the object.

Listing 5.1: Triplanar Surface Shading

```
half2 yUV = IN.worldPos.zx / _TextureScale;
half2 xUV = IN.worldPos.zy / _TextureScale;
half2 zUV = IN.worldPos.yx / _TextureScale;

half3 yDiff = tex2D (_DiffuseMapTop, yUV);
half3 xDiff = tex2D (_DiffuseMapSide, xUV);
```


a) Wood Texture with Scale 1

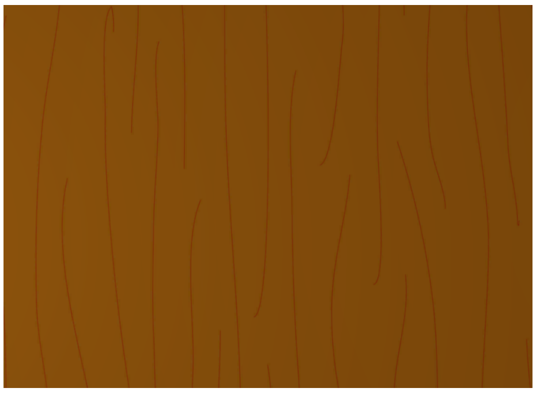

b) Wood Texture with Scale 2

Figure 5.4.: Wood Texture with different scales

```
half3 zDiff = tex2D (_DiffuseMap, zUV);

half3 blendWeights = pow (abs(IN.worldNormal), _TriplanarBlendSharpness);

blendWeights = blendWeights / (blendWeights.x + blendWeights.y +
blendWeights.z);

o.Albedo = (xDiff * blendWeights.x + yDiff * blendWeights.y +
zDiff * blendWeights.z) * _Color ;
```

The implementation was taken from [Pal] and visible in 5.1. The surface shader in Unity which implements the triplanar shading takes the following steps. At first, it finds the uv coordinates for each axis. They are divided by the texture scale so that the user can take the size of the objects into consideration. The difference in size can be seen in 5.4. After that, the textures are sampled from the previously acquired uv coordinates for each diffuse map.

The blend sharpness tries to increase the influence of the axis with the highest value. This increases the sharpness. The absolute value of the world normal is taken because it does not actually matter if you look at the positive or negative axis. They are both sampled from the same texture e.g. the side texture is used in the x axis . Finally, the texture is blended together based on the previously gained blend weights for each axis and their respective color from the sampling as well as tinting color.

With the help of this shading method, the need for uv mapping is gone. This comes at the cost of additional texture lookups in the shader which results in a small performance penalty. Overall the saved working time on uv maps was the deciding factor for the author to use this shading technique.

5.1.6. Example of a finished Wooden Figure

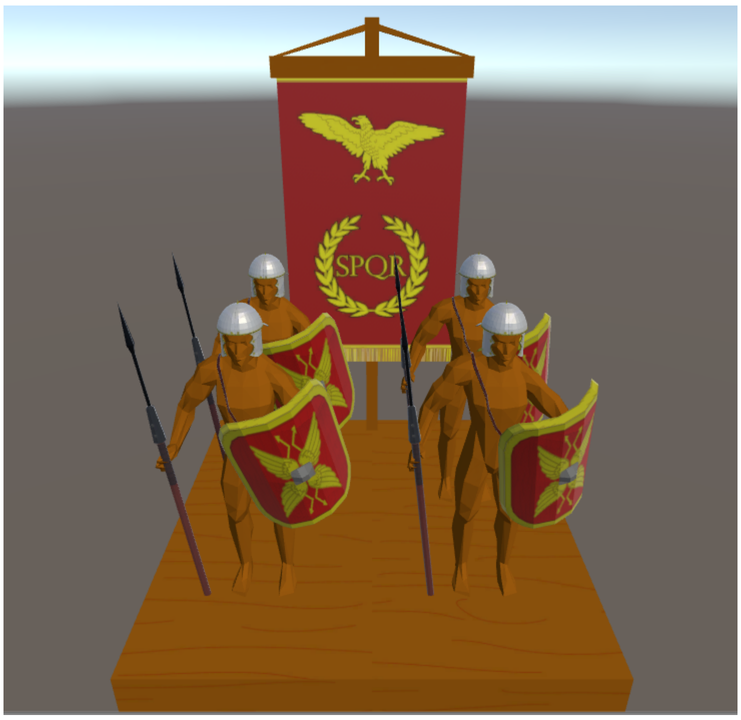


Figure 5.5.: Example of a finished wooden figure

Source: Selfmade

A example of a finished figure of a unit group can be seen in figure 5.5. After some experiments with just a single unit on a base, the author decided to use multiple units on the same base to represent the group structure better. It would be the best to just add the real amount of soldiers there, but then the user could not really recognize a single unit. Also, the equipment is way too small to be recognized as well. The number of 4 (2 if it is a cavalry unit) was chosen to allow the player to inspect the units and their equipment.

Each unit also has a banner at the back to show the side which it belongs to. It also works as a health bar to show the amount of still available man in this unit. The roman units often carried a standard during that time. It helped the unit to orientate towards the main part of their unit group. But this standards did not require a banner like in the game. The carthaginian soldiers did not carry standards or similar items into the battle but the author decided to add them to the carthaginian soldiers as well so that they can be used as an identifying feature.

Each unit carries some equipment. The visible hastati unit in figure 5.5 uses a montfort helmet, a sword belt with a gladius sword, a scutum and multiple pila. The user will connect these weapons with the respective soldier and learn more about their equipment this way.

5.1.7. Terrain Rendering

The terrain is, as discussed in the section in the previous analysis chapter, not exactly clear. The exact position can not be determined and the landscape has changed a lot in a timespan of approximately 2000 years. Daly states that one can safely assume that the height difference was only a mere 66 centimeter per 100 meter [Dal03, P.35] which means that the battle took place in flat terrain. This is helpful in multiple ways. It allows the author to use a flat map for the game which increases with the "general"-feeling and provides the user with an outlook over his battlefield. Additionally, this allows easier movement calculations, as the author can assume, due to the flat map, that only the y axis matters in the rotation.

A nearby river, called the Aufidus during that time frame, is mentioned in the sources. The water stream reduces the possible movement area and has to be visible on the map in the game. The rest of the terrain is not further discussed but it can be assumed that it was mostly grassland in order to provide no side with an advantage. In todays time the place which Kromayer mentions mostly consists of fields.

One way to approach the rendering of terrain is the replicative rendering of a seamless texture next to each other. A seamless texture is a texture which can be put together to a grid structure without adding visible borders between the single instances of the texture. This technique is often used in games and will be used in this thesis aswell.

But there is still the need to show other textures except grass like water for the river to generate a interesting terrain. This blending can be done in a two dimensional picture manipulation program like GIMP upfront, but the author decided that he wanted to use the terrain information, e.g. here is water and here is grass, to actually influence the movement of the units. This especially makes sense if you include more than just grass and water, because the movement is different on sand than on grass. The general idea is to use one map which carries the terrain information and various seamless textures to actually create the visuals of the map.

For this first test, it was decided to use the terrain types grass, sand, water and rocks. The blend map had to carry the information how strong the respective type at that

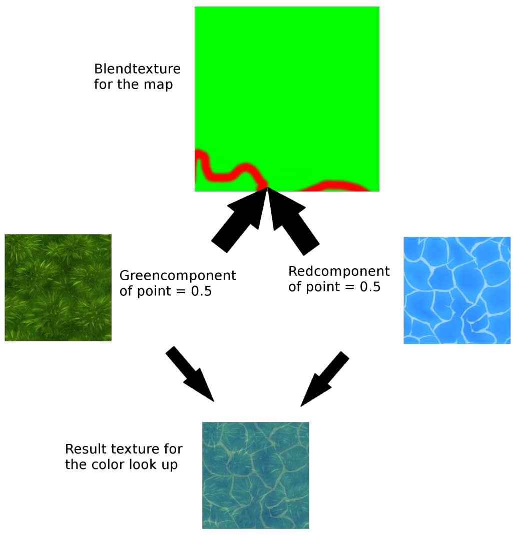


Figure 5.6.: Explanation Terrain Blending

Source: Selfmade

point was. The design dictates that the value of all terrain types added together should be not bigger than one. The blending itself is just a simple linear blending, e.g. if the a certain terrain point has a grass strength of 0.25 and a sand strength of 0.75, then the final color is determined by adding the color from the grass texture multiplied by 0.25 and the color from the sand texture multiplied by 0.75.

Listing 5.2: Triplanar Surface Shading

```
void surf (Input IN, inout SurfaceOutputStandard o) {
  fixed3 c = tex2D(_Blend, IN.uv_Blend);
  fixed3 b2 = tex2D(_Blend2, IN.uv_Blend2);
  fixed rest = 1.0;
  fixed3 result = c.r * tex2D(_Water, IN.uv_Water);
  rest = rest - c.r;
  fixed f = clamp(min(rest, b2.r), 0.0, 1.0f);
  result += f * tex2D(_Stone, IN.uv_Stone);
  rest = rest - b2.r;
  f = clamp(min(rest, c.b), 0.0, 1.0f);
  result += f * tex2D(_Sand, IN.uv_Sand);
  rest = rest - c.b;
  f = clamp(min(rest, c.g), 0.0, 1.0f);
  result += f * tex2D(_Grass, IN.uv_Grass);
  o.Albedo = result;
}
```

The actual implementation is visible above in the listing 5.2. An additive shading approach was used for the various layers. This requires an order of the various layers, so the author ordered them by importance. This order is a result of their influence on the unit movement. The order is as followed:

1. Water
2. Stone
3. Sand
4. Grass

Each layer is saved in one of the red, green or blue channels of the blendmap. The alpha channel of the blend map is not easily editable, so the author decided to introduce a second blend map. The rest is clamped to enforce the maximum of 1 of all added up values. Any values which add up to more than 1 are ignored by just using the values from the first layers. This results in maybe missing some information from the later layers but somewhere information have to be lost to correct the error. Another approach would be to normalize the blend values of all layers to one by dividing each value through the added result of all values, but you are losing information here as well, so the result is the same. Maybe the other approach would be better on the performance for graphical processing unit due to its lack of using clamp which must implement a conditional, but that has to be profiled to be determined.

This approach could be easily extended to feature more than the 4 existing layers. During later stages of the development, the sand layer was discarded because there are no sources which confirm the existence of sand on the battlefield. So, to not endanger the historic authenticity, the layer was removed from the shader. For the rocks, another approach was chosen as an implementation.

The rocks are now represented as three dimensional models on the actual map. This helps the player with the visualization of the of the rock and the removed movement of the units there. In blender, the author created some low polygon meshes for rocks. These were then imported into Unity. Some experiments like creating a high polygon variant of the same rock and then baking the normal map out of it on the low polygon rock were tested but not found suitable because they do not match the stylistic approach for the other models which all lack a normal map.

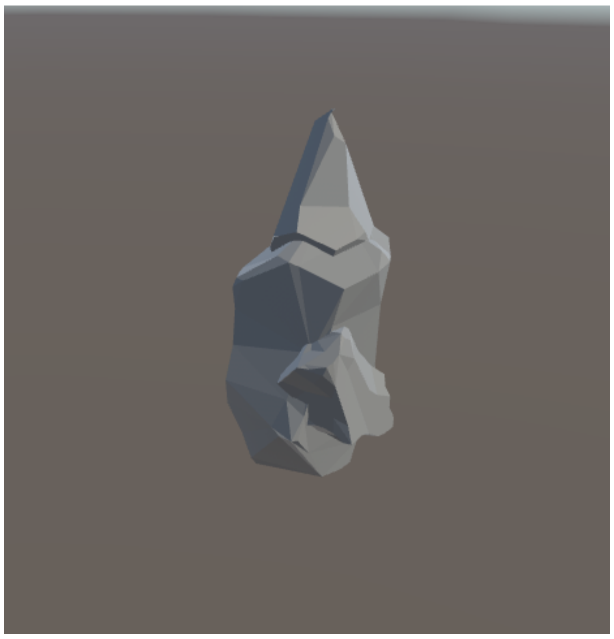


Figure 5.7.: Rock 3D Model

Source: Selfmade

5.2. Interaction

The use of the Vive Controllers requires that the developer changes the typical interaction methods and mechanics to new ones, because the old ones do not work in a Virtual Reality environment.

For this purpose, the author developed a control schema which is visible in figure 5.8. The diagram has five different types of components.

- Purple Rhombus: Shows a decision the user has to take
- Blue Rhombus: Decision due to the state of the game
- Blue Rectangle: A process in the game
- Yellow Parallelogram: One of multiple possible options for the user
- Purple Rectangle with rounded corners: Starting point

The components are connected with edges. Each edge can have a button like the trigger connected to it or be part of a yes/no decision or just a connector. The diagram

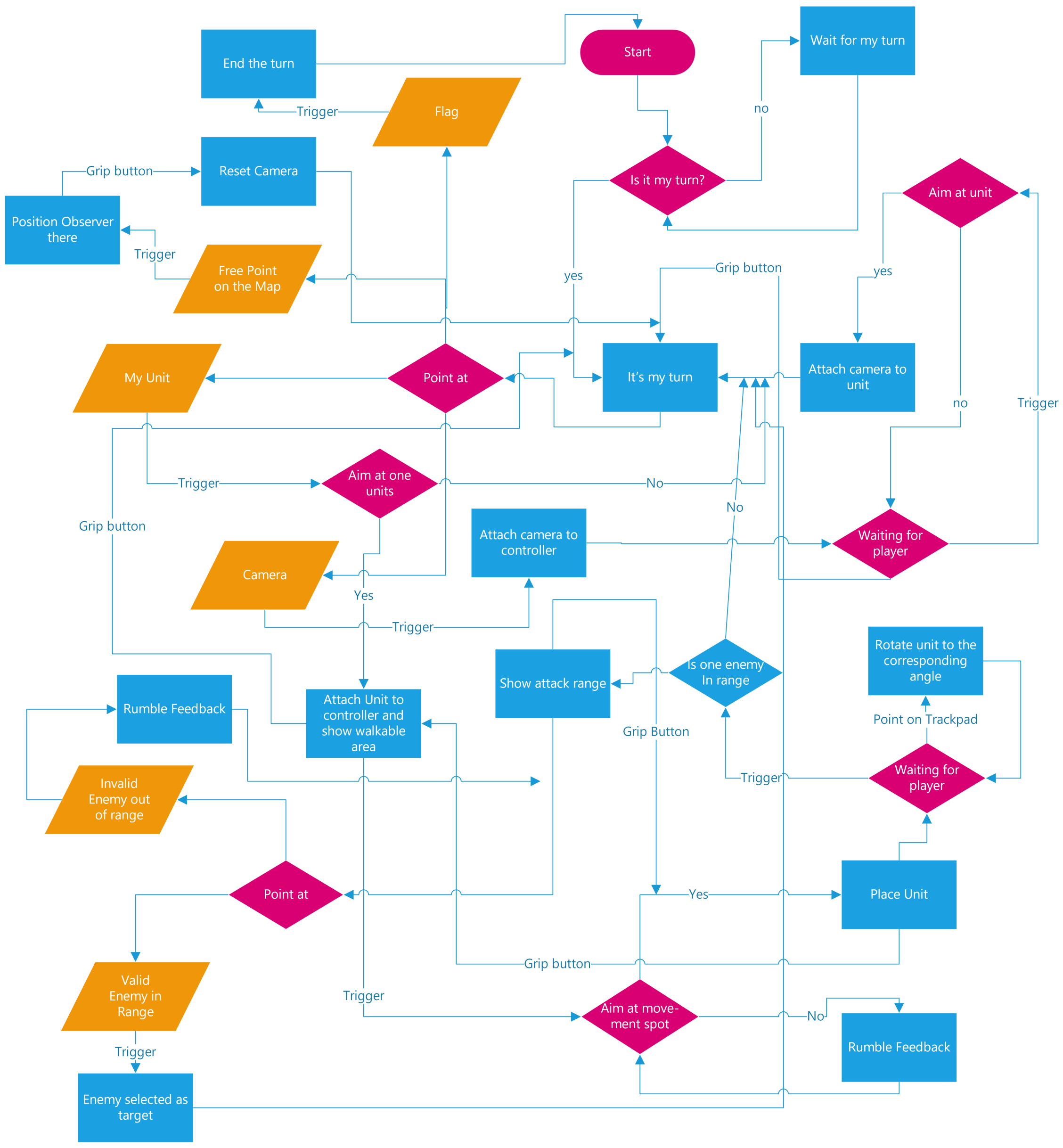


Figure 5.8.: Diagramm of the possible interactions in the game

Source: Selfmade

shows all possible interactions in the battle simulation part. Most of them are also available in replay mode but the unit control is not.

The controllers are used as pointers in the virtual environment. For this purpose, rays were attached to the front of them. With the help of these rays, the player is able to aim at objects and interact with them. These rays are interactive, they stop at objects which are interactable.

The general idea during the development was to introduce a two button system. You press the trigger to go forward in the process and one of the grip buttons on the side to go backward. The author tried to keep this principle in all interactions. Some interactions use special buttons like the touchpad.

Another goal in the development was to reduce the typical two dimensional user interface elements as much as possible and integrate these elements directly into the world. Also, elements which correspond to a real world action, will be preferred. An example for this would be the interaction to get to the main menu. The player has to aim at the door and press the trigger button. It is a well known idea to leave the room through a door to get to another room. The author decided to use this idea for the game as well as it fits into the environment. The interaction is not included in the diagram because it is already quite filled and just represents the commands for the game loop.

5.2.1. Unit Control

One of the most interesting parts of the control schema is the control of units. The control schema of a unit consists of the following four different phases.

- Select Unit
- Find Target Space
- Find Target Rotation
- Select Enemey

During the selection of an unit, the player has to mark the unit which he wants to control. There are two possible approaches for this. One approach would be proximity to the unit. When the controller is close to the unit the player presses the trigger and the unit is selected. But this approach does not work well in this game context because the controller is rather big compared to the units. The game can not determine which unit should be chosen if multiple units are close to the controller. Another approach is a selection with a ray. This method is similiar to a mouse selection which is known to most users. The user targets the unit with the ray which is attached to the controller and presses the trigger to confirm the selection. This works better because the user knows the procedure from the mouse selection and the outline from the user interface allows an easy selection.

After selecting a unit, the place where the unit should be placed has to be found. A possible approach for this problem is to take the position of the controller in the three dimensional space as the target position. This requires the user to be able to reach the position in the real world where he wants to place the units. This could end in immersion breaking movement like going into the table in the virtual environment which is not here in the real world. That kind of behaviour should be reduced as much as possible to increase immersion. The other approach for this problem is similiar to mouse selection by selecting a target position through ray selection from the controller. This allows the user to just point at the desired target position and press the trigger. If the user chooses an invalid position a short rumble feedback is triggered to show that the selection is invalid.

The target rotation is the second-to-last step for the unit control. Again, two approaches were explored. The first one was to use the pose of the controller. During testing, the use of this was not intuitive and it produced in some instances undesired behaviour when the user has to turn his hand around for 180°. The other approach is the selection of the rotation through the touchpad. For this purpose, the author used the input from the touchpad as a position on the unit circle. If the user did not touch the outer ring of the unit circle the position is projected onto the ring. The position on the unit circle is used to determine the desired angle. Over a small time frame, the unit then rotates to the desired rotation.

The last step is the selection of the enemy. This was handled similiar to the unit selection in the first step. The same arguments apply which led to the ray selection.

5.2.2. Perspective changes

The use of Virtual Reality allows the use of intuitive look around mechanics from different perspectives. It was considered interesting by the author to add the possibility to view the battle field from the perspective of a single soldier. For this purpose, you can follow a single unit during his movement. This is implemented by using a kind of real world interactive user interface element. A camera is placed next to the map. The outline highlight shows the interactivity of the camera. When the user selects the camera, a model is attached to the controller. After that, the player can attach the small model to a unit. The unit gets a small camera model attached to it to show the successful selection. When the player ends his turn, the rendering camera switches to a perspective which is positioned and scaled to match the head of the selected unit. This allows the player to experience the battle from the persepctive of a soldier instead of his general view.

Additionally, the player is able to select any point on the map and switch to an observer position at that place during his turn. During the movement selection and the observing of the environment, the positional tracking of the Vive is disabled so that the player stays at the same position but the rotational tracking is still here to enable the intuitive look around mechanic.

5.3. Replay

The Serious Game should contain not only a battle simulation part but also a replay part for the ancient battle. For this purpose a replay part was added which is reachable through the main menu.

This replay part should teach the player about the tactics which were used by the respective commanders. In this battle, especially the strategy of Hannibal is interesting as it shows the pincer movement by his army. The strategy can be best shown by just showing the unit movement. The resulting fights are not that important so the author decided to just show the movement of the units without the combat simulation of the battle simulation part.

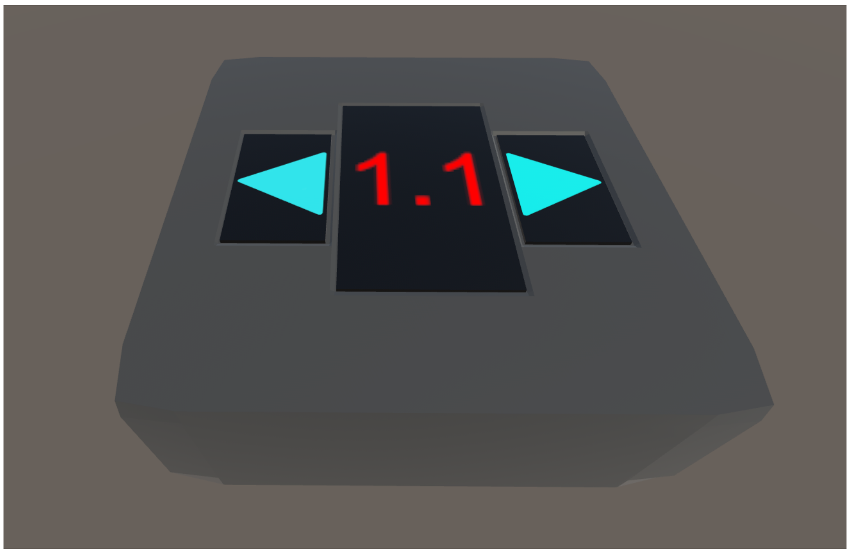


Figure 5.9.: Picture of the replay device

Source: Selfmade

The actual implementation is done by saving a set of seven steps for each unit. Each step contains a position, rotation and an action. The action can be movement to a specific position or death for dead units. This is used especially for the cavalry on the roman side which flees from the battle. The first six steps are taken from the book [Dal03, P.40-41]. The seventh step was added by the author to the show the end result of the tactic and generate a better ending point for the replay of the battle.

The replay mode is controlled by pressing the respective buttons for forward and backward on the replay device in the scene. Each step has three sub steps which are always the same. In the first sub step the unit stand still at their respective position and rotation for that step. After that, a path to the new position and rotation in the next step is shown in the next sub step. In the last sub step, the unit moves to the new position. The player can freely go forward and backward in the steps and sub steps and watch everything as often as he wants.

5.4. User Interface

This section discusses the user interface in the game. The author the thesis use the term user interface to describe additional elements which were added to the game to visualize essential information to the user. The respective control components are discussed in the section 5.2.

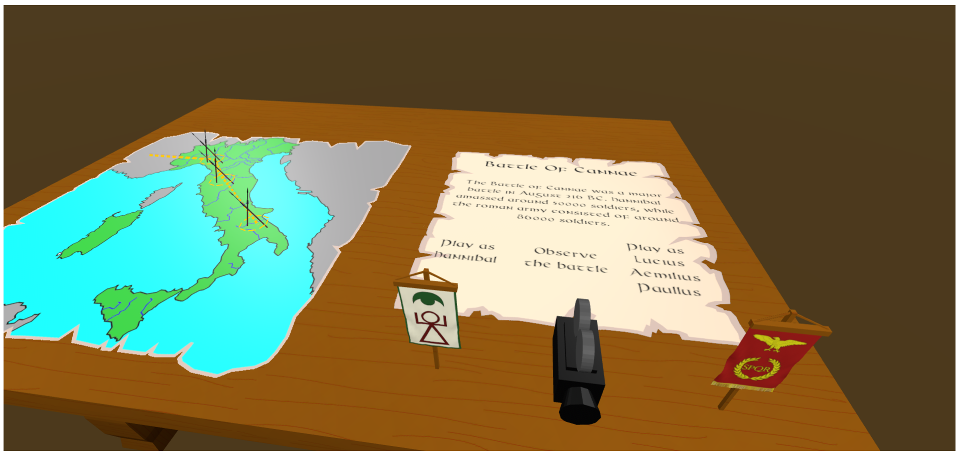
5.4.1. Main Menu

The main menu contains a selection of various battles. The path of Hannibal through Italy is shown as well as two other important victories of him. If the player selects the Battle of Cannae, he can choose between the carthaginian side, which is represented by the white banner, the roman side, which is represented by the red banner, and the replay mode, where he can watch the battle. The representation of one side by using the respective banner is first shown here and shortly explained by the small text above it.

The main menu already represents the first possibility to teach some information. The previous battles can be all selected and contain a small summary. The inclusion of details like this in the game world make it easier for the player to just explore the environment and learn something in the process instead of just reading and repeating a text. The visualization of the path helps to introduce the current state of Hannibal's army.

5.4.2. Generating meshes as an Information Overlay at runtime

If an unit is selected in the game, the possible movement area should be shown. The game does not use a grid as a base because the whole map is clickable and selectable


Battle Of Cannae

The Battle of Cannae was a major battle in August 216 BC. hannibal amassed around 50000 soldiers, while the roman army consisted of around 86000 soldiers.

Figure 5.10.: Screenshot from the Main Menu

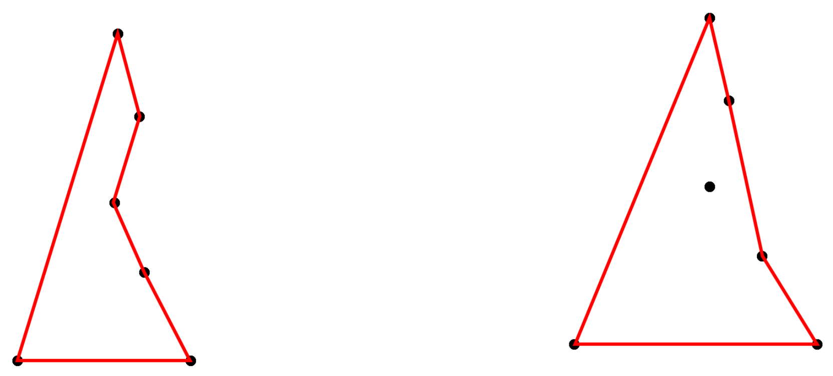
Source: Selfmade

as a target for movement. This makes it rather hard to show the area in general. Due to this restriction, the following algorithm was first tried:

1. Create a set of points in a grid like structure around the unit
2. Check for each point if he is reachable by the unit
3. Create the concave hull around the set of reachable points
4. Triangulate the concave hull

This approach has two major problems. The first one is related to performance. To test the the algorithm, the author used a concave hull generator taken from Github [Ali17]. This resulted in calculation time of multiple seconds even for small sets of points and even without the triangulation of the resulting hull. The resulting wait time was considered to big for the player.

The other problem is related to the hull being concave. The movement system was defined in a way that made it possible that the unit movement results in a concave form. Additionally, obstacles can lay in the way which can produce holes. A concave hull is not clearly defined which is visible in figure 5.11. The set of points on the left and on the right both create a concave hull based on the base set of points, but they differ in the number of points and in the precision. Only the convex hull clearly defined and could be used, but the geographic form of the unit movement does not have to be convex as shown above.



a) Valid concave hull of a set of points **b)** Another valid concave hull of a set of points

Figure 5.11.: Two concave hulls of the same set of points

Due to these problems, the author decided to switch to another approach for this problem. Now the rendering is done by the marching cubes algorithm[LC87] so the steps now look as follows:

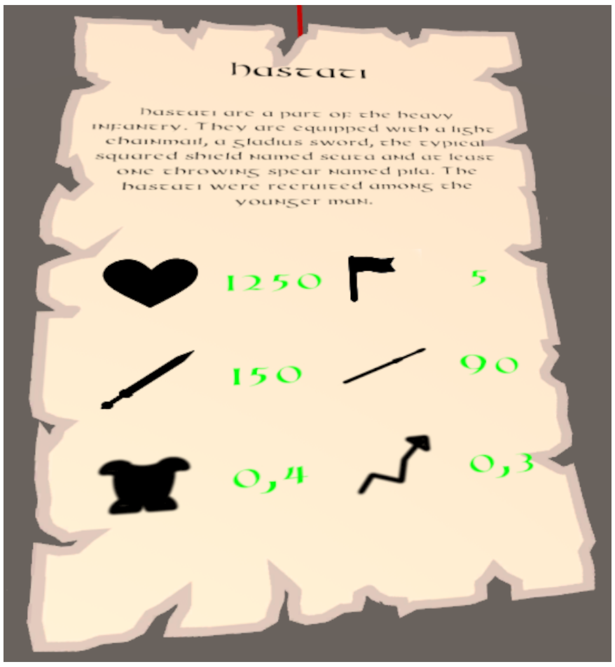
1. Create a set of points in a grid like structure around the unit
2. Check for each point if he is reachable by the unit
3. Create a mesh for Unity using the Marching Cubes algorithm

The Marching Cubes algorithm takes an up to three dimensional grid of density values. The algorithm now checks the density values for each voxel created by the grid and finds a fitting triangle for the voxel where all corners which are above a certain threshold are inside the resulting mesh. The implementation was still too slow for real time. So the multi threading approach which was used in the artificial intelligence was used here aswell. Only the steps 1 and 2 in the algorithm were added as workload for the second thread which was spawned for the unit selection because the third step requires the editing of the mesh component in unity which is not possible from another thread. With this solution, the player still has to wait for around 0.5 seconds, but this was considered good enough.

The same approach is also used to show the attack range of a unit. The attack range could be handled by scaling a circle according to the attack range but this approach is already implemented and works, so the author decided to pay the additional performance cost and use this.

5.4.3. Unit Information Paper

Each unit contains a variety of other values which need to be shown to the player, so that he can make an informed choice on his movement and attack. The typical implementation in a standard monitor environment would be attachment of two dimensional user interface elements to the screen. This does not work well in Virtual Reality because it is hard for our eyes to focus on something that close on the screen [Uni17].



hastati

hastati are a part of the heavy infantry. They are equipped with a light chainmail, a gladius sword, the typical squared shield named scuta and at least one throwing spear named pila. The hastati were recruited among the younger man.

Figure 5.12.: Example for the unit information paper on the second controller

Source: Selfmade

Another problem is the varying eyesight of the playing user. If a great variety of people play the game, most of them differ in eyesight. Some people can't use their glasses in Virtual Reality headsets due to their design. A user friendly approach would be to vary the size of text elements to allow the user to choose a size which he is comfortable in. But varying text size result in different sized user interface elements which poses a problem for consistency.

The author chose to implement the following approach. Only one unit is selected at any time. The needed information for gameplay like attack damage or health as well as a short historic description of the unit are written on a paper which lies next to the playing area. The paper is updated every time a new unit is selected.

In practice this posed some problems. One problem was the missing option to change size of the paper. It would be possible to add some kind of option menu, but then the whole sheet has to be rescaled. Another problem was that it was hard to notice that the values actually changed. There were no visual or audio cues which allow the player to notice the changes. Due to these problems the approach was changed slightly.

The author noticed that the second controller is mostly not used because the player uses one controller to actually give orders. But the controllers is most of the time still in the viewing area of the player. Due to this fact, the paper was attached to the second controller. This allows to change the closeness to the eyes to allow for varying eyesights. The paper is only visible for the player if the other controller has selected one unit. The appearance and disappearance due to selection is a hint to the player to check for the paper to see the changes on it.

5.4.4. Unit specific User Interface

Each unit also has some additional user interface elements which are attached directly to the wooden figure on the playing field. One example for this this kind of user element was the health bar in the banner for each unit. If the unit loses health, the length of the banner is reduced.

This is implemented in the game by using the transparency layer of the texture. After the author received the texture, he added a transparency gradient from full transparency to no transparency on top of the color values of the texture. The texture did not us the alpha channel, so no information was lost there. In the game, the material is set to alpha cutoff. This means that the color information of a point on the texture is discarded if the alpha value of the point is below a certain threshold which is defined in the material. This threshold is updated at runtime if the health of the unit changes. The code just uses the percentage of health remaining as the threshold for the alpha cutoff.

Another example for an user interface element which is directly attached to the figure is the chosen path. The path shows the route of the unit from his starting point to the selected endpoint. During the rotational part of the movement target selection the path

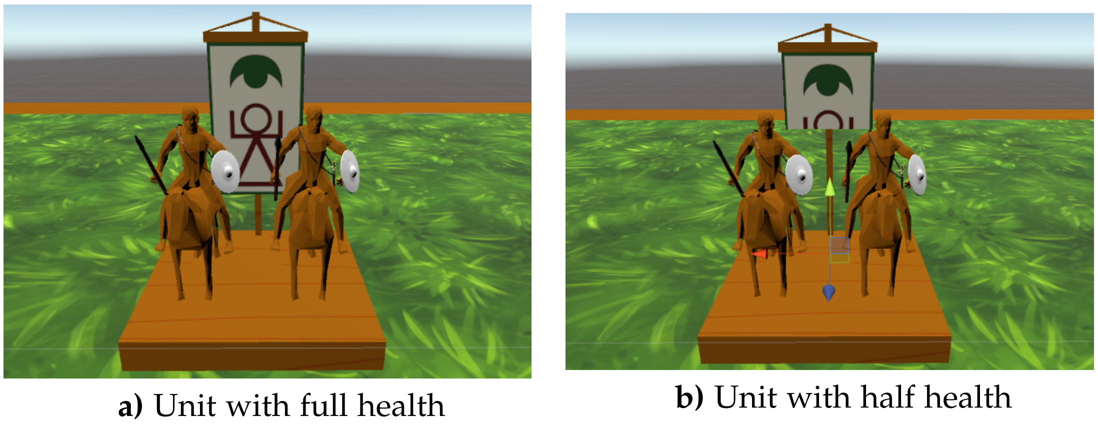

a) Unit with full health **b)** Unit with half health

Figure 5.13.: Shows the difference between 100% health and 50% health

as well as the unit show the validity of the path by being tinted red or green. This is a typical approach to show the validity and should universally be understood. After selecting a target rotation the path stays in place until the actual movement happens.

The path has additional functionality. During the movement calculation the movement cost from the terrain is determined at a maximum of eight points on the route. This restriction of eight points is due to Unity not allowing more keys on a gradient. Then the path is tinted according to the movement cost, so areas like water, where movement is slower, are colored red while grass is colored green. This gradient is automatically created at runtime for every path. The tinting of the path in various color helps to transfer the knowledge about the terrain to the player.

5.4.5. Outline

Some elements in the world are interactable but it is not always clear which elements are. To make it easier for the player to distinguish between interactable and non-interactable objects, an outline for objects was introduced.

The author used an implementation from github [Sto]. Another implementation from the Unity Asset Store was not compatible with the lab renderer from Valve. The implementation uses a standard solution for this problem. The model is cloned into another game object. Then the shader moves the vertices along a perpendicular axis to an edge and shades these vertices behind the actual game object. An example for this shader is visible in figure 5.15. The author chose this blue color as a signal for interactable objects.

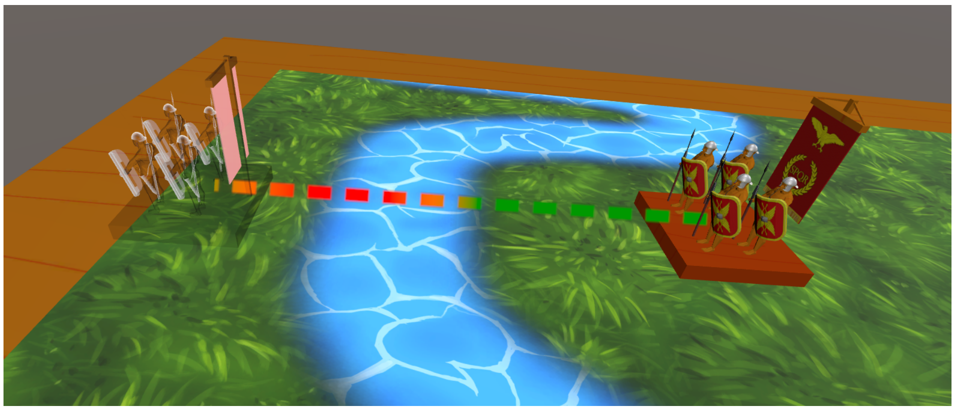


Figure 5.14.: Colorcoded path depending on the movement cost

Source: Selfmade

5.5. Gameplay

The gameplay is an interesting part in a Serious Game because the developer has to balance the gameplay components with the information density and authenticity. Thus, achieving a good balance in this respect can be rather hard.

5.5.1. Movement Calculation

Movement is a rather interesting part in the game environment. It should contain enough depth to make meaningful choices, but also be simple enough to be understood by every player. This game focuses on Ancient Battles, so the author decided to focus on a rather complicated movement system which better represents the movement in ancient battles.

The movement in said battles is rather hard to reproduce due to some missing information. But also finding a computational representation presented itself as a rather hard topic. The soldiers in a legion had to march lock-step which posed some problems for the unexperienced troops. Due to their movement as a group it is rather hard to turn towards one side. They need quite a lot of space for this purpose.

Straight movement from point a to point b with the inclusion of a rotation does not produce good looking results and does not represent a marching legion. Additionally, customizable restrictions are rather hard to implement. The restrictions need to be customizable to fit for different unit types, e. g. Velites are more mobile than Triarii.

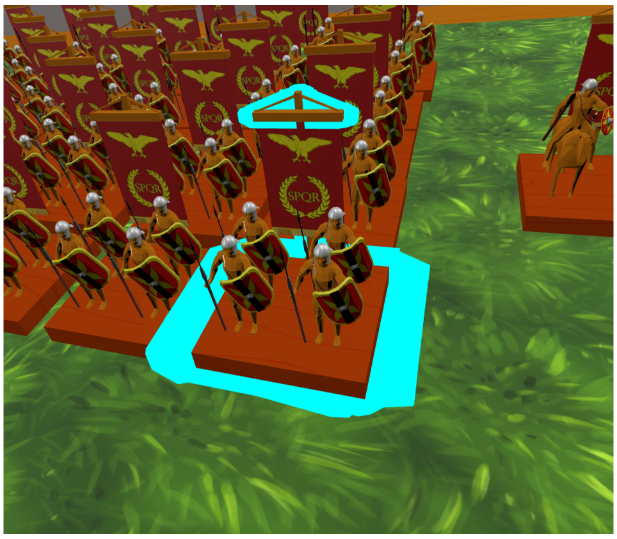


Figure 5.15.: Example for an unit with an outline

Source: Selfmade

$$B(t) = (1-t)^3 * P_0 + 3 * (1-t)^2 * t * P_1 + 3 * (1-t) * t^2 * P_2 + t^3 * P_3 \tag{5.1}$$

Bezier curves are parametric curves which produce better looking results. For the purpose of this thesis, the author will use quadratic Bezier curves. They are defined by the equation 5.1. Cubic Bezier curve represent a form of interpolation with four different points. An example for a cubic Bezier curve can be seen in figure 5.16.

The starting point P_0, the intermediate points P_1 and P_2 and the end point P_3 are defined in the beginning. As it is a form of interpolation, a interpolation variable also exists and is set to 0.5 in the figure. To actually calculate the value at the point $t = 0.5$, a linear interpolation for all three connecting lines between P_0 - P_3 is done and called P_{01}, P_{12} and P_{23}. These new points are now connected. On these newly created lines, another round of linear interpolation is done to create the points P_A and P_B. By performing a last linear interpolation on the line between P_A and P_B, the actual value of the cubic Bezier curve can be calculated.

If you use this approach on enough values for t, you can get a discrete representation of the curve in the game. With the knowledge about Bezier curves, we can use them to fit our needs. The starting point for each movement and the ending point are defined by the starting position and the chosen point by the player. The point P_1 is defined by using the local forward vector of the initial rotation and scaling with the result of the

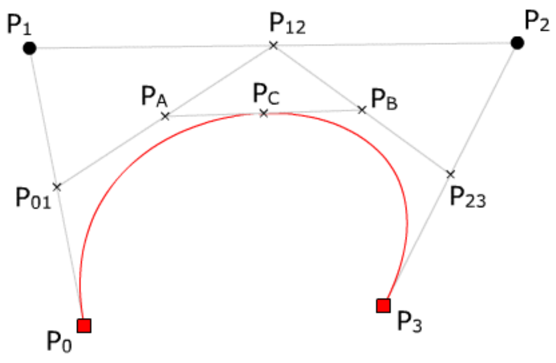


Figure 5.16.: Cubic Bezier Curve

Source: [Gro11]

division of the actual length of the path and the movement speed of the specific unit. This division represents basically the percentage of the possibly used movementspeed of the unit for this round and creates nicer looking curves. If the point is not scaled, the units always take longer routes even if the only move for a really small distance.

The bezier curve is evaluated at a number of points based on a step size. This step size is again determined by the percentage of the used movespeed in this turn. This stepsize is then clamped into an area between 0.143 and 0.33. The lower bound is due to the unity restrictions on they amount of keys on color gradients. The color gradient is used in the tinting of the visual path of the unit. The upper bound is set to require a minimum of three evaluation steps which are used for the movement calculation.

After evaluating the curve at the respective number of points based on the step size with the previously calculated points P_1 and P_2, the rotation quaternions for each point can be calculated. The start and the end rotation are determined by the starting position and the chosen rotation of the player. The steps in between are calculated by looking from each point to the next calculated point in the set.

$$Cost = ((angle/TurnRateUnit) - 0.5) * 2.0f) * Distance(previousPoint, currentPoint) \tag{5.2}$$

With the points and the rotation, the actual movement restriction calculations can take place. For this purpose, a cost variable is introduced. The algorithm now checks every pair of consecutive points. If one of the points is not on the map or if one of 20 interpolated points between them is not on the map, the movement is not possible. Additionally, if one of the points is too close to another unit so that they intersect, the movement is not possible. If both checks are successful, the movement cost of the second points scaled with the distance between the points is added as a cost. After checking the points, the rotations are checked aswell. Each unit has a turn rate as a unit specific value. If the rotations differ by more than the specified turn angle and the route is longer than ten percent of the possible movespeed, the movement is not possible. If the ratio of the angle between the two rotations and the allowed turn rate is bigger than 0.5, a cost factor is added according to equation 5.2. At the end, the method checks if the cost factor is smaller than the movespeed of the unit. If this is the case, the movement is allowed, otherwise it is denied.

Earlier in the description, a collision check with other units was mentioned. For this purpose, the author implemented a rotated rectangle collision check. The unity built-in function could not be used because Unity restricts the use of their functions to the main thread but the check was needed on the walkable area thread and the artificial intelligence thread. The collision check for rotated rectangle consists of 4 steps and works due to the separating axis theorem. In the first step four axis are determined where each is perpendicular on an edge of the rectangles. Only four are needed because two edges always share the same axis. After that, each point of the two rectangles is projected onto one axis. As a third step, a scalar value for each point is calculated. In the last step, the minimum and the maximum values of both rectangles are checked against each other to see if there is an overlap. An overlap means a collision on this axis. If every of the four axis has no overlap, the rectangles do not collide. For performance reasons, a simple sphere collision check was introduced as a first step before the rectangle collision check.

5.5.2. Combat System

Creating an interesting and historical correct combat simulation is rather difficult. Just using one attack value in every situation and one defense value is not enough. Imagine a group of cavalry charging into a phalanx from the front. This results in heavy casualties for the cavalry. But if the attack is instead from the back, the casualties are heavier on the phalanx side. Due to this reason, a simple value comparison is not enough.

The author choose to implement various factors which can modify the health damage of the attack. The first two factors relate to differences in angle. They are visible in figure 5.17 as the angles α and β. The angle α is calculated as the difference in the forward vector of the attacker (green arrow of the yellow capsule) and the defender (green arrow of the red capsule). The other angle is defined as the difference between the forward vector of the attacker and the forward vector of the defender if he is looking at the attacker (blue arrow of the red defender). They both use the fact that it is easy to attack someone from behind. This will cause heavy panic and casualties on the

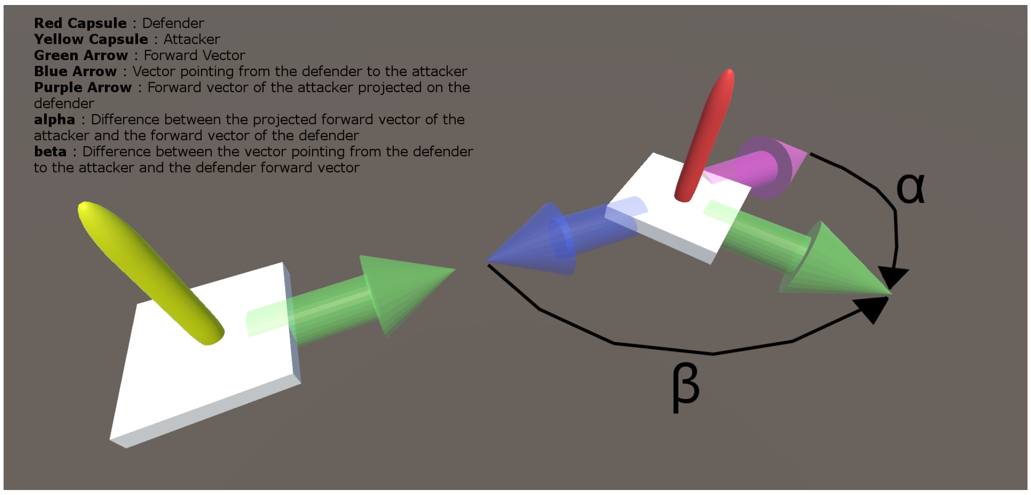

Red Capsule : Defender
Yellow Capsule : Attacker
Green Arrow : Forward Vector
Blue Arrow : Vector pointing from the defender to the attacker
Purple Arrow : Forward vector of the attacker projected on the defender
alpha : Difference between the projected forward vector of the attacker and the forward vector of the defender
beta : Difference between the vector pointing from the defender to the attacker and the defender forward vector
α
β

Figure 5.17.: Angle difference between normals

Source: Selfmade

defender side. They are split up to two factors to allow easier balancing. The values are normalized to the range 0.0 to 1.0 by dividing them by 180 which is the greatest possible angle and then clamped to 0.1 to 1.0 and scaled for balance purposes.

The distance is important for combats. For ranged weapons the distance is important because people can only throw for a certain distance and it is harder to hit an enemy which is further away from you. For melee range combat the distance is also important. Not only do weapon lengths differ greatly, e.g. spear versus sword combat, but it is also possible for more people to fight, if you are closer together in this game. This is a reduction from reality where you are limited by the length of your legion but in the game a fight takes place for a longer amount of time. If the troops are closer together, the fight starts earlier and more people will hit an enemy before the fight is over. After this considerations, a distance modifier curve was chosen which reduces the modifier if you reach 75% of the combat distance from 1 to about 0.5 ab 100% distance. The curve is visible in figure 5.18.

The health modifier is used to scale the damage accordingly to your health. Troops which have suffered casualties lack the soldiers to fight and their morales is not as good as the one of fresh troops. A direct linear curve was chosen for these.

The armor value of the defender reduces the damage which he takes. The armor value is between 0.0 - 1.0 and represents a reduction by the specified value, e.g. an armor value of 0.1 reduces the incoming damage for the defender by 10%.

The combat system is extremely punishing for the defender. The person who is

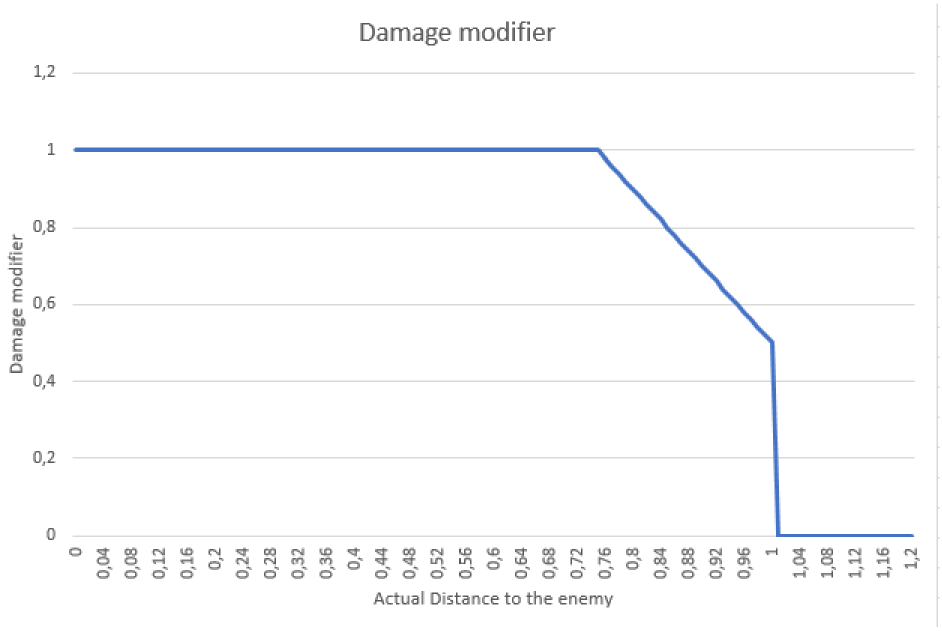


Figure 5.18.: Distance modifier compared to percentage distance to the enemy

Source: Selfmade

allowed to take the first turn is favored by the fact that he can attack first and the defender has to suffer the casualties from these attacks before attack back. To counteract this problem, the combat system actually fights two rounds each time. The first round is the specified unit versus the defender and the second round the roles are switched. This allows to better balance the game aspect and represents the reality better because the defender was also inflicting casualties on the attacker.

5.6. Artificial Intelligence

For the battle simulation part, an artificial intelligence is needed to move the units around. It should react to the player movement and use the same strategy as Hannibal in the battle.

5.6.1. General Idea

The author chose an simple and straightforward implementation. The idea of an utility based artificial intelligence is the rating of a finite set of existing possibilities based on their usefulness[Vid]. This approach can be easily used in the game. A set of points is chosen around the unit, comparable to the set which is used for the walkable area. For each point in this set, the unit first tries to calculate if it can go there and saves all possible rotations. After that, a huge list with possible positions and rotations is available. Each pair is rated based on some factors and in the end the artificial intelligence chooses the best option or one of the better options based on their utility.

There are three factors which the artificial intelligence takes into consideration. The first one is the distance to all enemies. The utility functions uses the inverse, so it tries to minimize the distance. This is useful, because the artificial intelligence stays in range of the player, so that he is engaged with the game and his turns do not feel meaningless. Also, it allows to attack other units more easily.

The second factor is the possibility for a fight. The artificial intelligence simulates each possible fight and rates them based on the inflicted health damage to the defender. This results in an artificial intelligence which actively tries to look for fights and uses the best possible position for a fight.

The last factor is a result of the requirement that the artificial intelligence should react like the respective commander. For the replay part of the game, a set of 7 steps for each unit was introduced. The artificial intelligence will try to match the steps because it gets an increased utility for closeness to these steps. The state of the artificial intelligence carries a counter with the current goal step which is increased if the error which is represented by the difference of each unit to the respective goal step is below a certain threshold. This goal step allows the artificial intelligence to take more than one turn for each step if it can not reach the required goals. The distance penalty for the utility for each step is rather high, so the artificial intelligence should favor the historic positions in the battle.

During the playtime, the game required a visual cue to show that the Artificial Intelligence is still calculating. For this purpose two rotating gears were introduced at the other side of the table at the position of the alleged other player.

5.6.2. Multithreading the Artificial Intelligence

The movement calculations are rather time consuming, so multithreading was needed to allow the game to go on during the calculations. Unity allows the use of coroutines but these do not actually run on other threads they just don't block. Additionally, Unity does not allow any Application Programming Interface calls from a function which does run on a non-mainthread. Due to this, most components which were needed for the calculations in the multhreading were reimplemented like the MultithreadedTransform class. The structure due to these limitations often required to just feed the input data into the other thread and wait for the result.

In a first approach the author used was the Task.Factory class from the C# library. This is the recommended way according to the Microsoft Documentation [rpe17]. After implementing this approach, a major problem occurred. In the game environment, especially in Virtual Reality, it is essential to maintain a consistent frame rate to not create any problems for the user. The Task approach creates tasks and distributes them on existing threads in the thread pool. This resulted in a rather large spike in the beginning. Additionally, the rendertime was increased because the threadpool took a lot of the available resources. After some testing, it was decided that this was not a possible solution to the multithreading problem.

The other way for task based parallelism in the C# library is the use of the Thread class. This approach worked really well, but is considerably slow because only one thread can be used to not disturb the calculations of the Unity threads. The required time for the artificial intelligence can be controlled by reducing or increasing the amount of checked points. The author decided that the maximum amount of time for the artificial intelligence calculations should be 30 seconds and tried to balance the settings around that. A higher amount of points is useful because the Artificial Intelligence has a bigger pool of possible targets for their movement.

6. Evaluation

After the actual implementation, the resulting game has to be tested to evaluate it. At first, the technical requirement of 90 frames per second will be tested and the results will be discussed. After that, a user study is performed. At first, the goals of the study are defined as three areas which need to be evaluated. After that, the results are discussed.

6.1. Performance

Unity allows three different kinds of performance measurements. The first one is through the stats window in the editor. This is not useful because the test should be performed on a built version of the game and not in the editor. The second variant is by attaching the unity profiler to the project. This is rather difficult because the game was built on another PC than the PC that the build is run on. So it is not possible to attach the profiler to the built version. The last way is to measure the time between two frames via script and save that time. This method was used to evaluate the performance of the game. During early tests, the author noticed partly extreme variance in the recorded frames per second between the different methods.

The following diagrams 6.1, 6.2 and 6.3 use render time instead of frames per second. The frames per second can be easily calculated by dividing 1 through the render time. This results e.g. in a render time of 0,0166 milliseconds for a frame of 60 per second. The axis are scaled differently on each diagram to make the difference more visible. The goal was to reach 90 frames per second at all times which corresponds to a render time of 0,011 seconds. The render time are always added up over a timespan of five seconds (called a timestep) where the game saves the lowest and the highest render time and also calculates the average render time.

The main menu figure 6.1 shows that the average render time is around 0,011 seconds. But there are still some frames which take way longer to reach the actual goal, especially visible in the timestep 2. But this render time is already not good, considering that the main menu only consists of a few objects and only some scripts.

The replay mode which is visible in figure 6.2 shows extreme spikes where the worst render time range up to 0.25 seconds. This corresponds to four frames per seconds and is really low so that even some users noticed the low frames per second. These

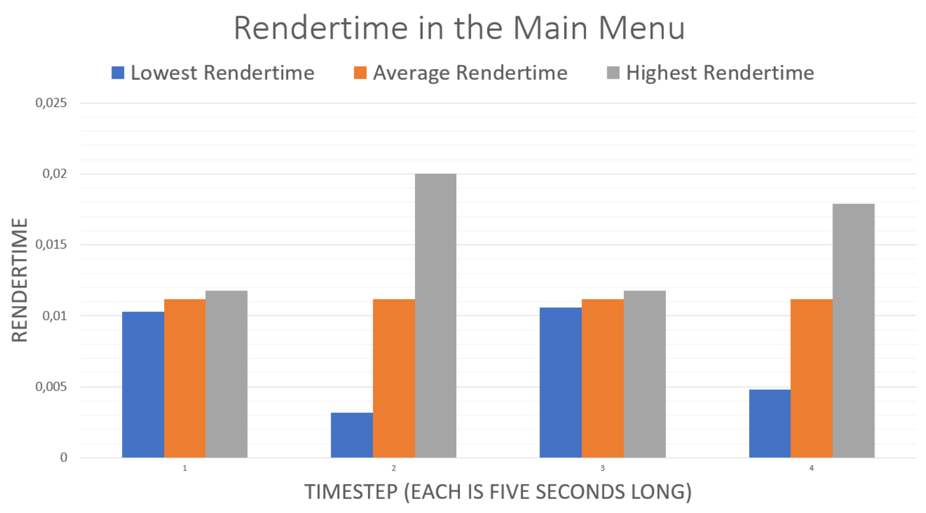


Figure 6.1.: Rendertime in the Main Menu

Source: Selfmade

spikes result from the spawning of the ghost prefabs. Instantiating is comparably slow in Unity, so an object pool would be a better solution. Another solution would be the reuse of the first spawned object. The other problem which results in low frames per seconds is the use of the transparency shader on the ghost prefabs. Transparency is due to its nature complicated because it requires blending with the existing background and against other transparent objects, so the use of approximately 100 transparent figures with multiple transparent objects results in high render time. One of the easier solutions would be to reduce the amount of units on the field.

In the ingame figure 6.3 the average render time is around 0,022 seconds which corresponds to around 45 frames per second. The goal was not reached and the game is not optimized enough. A solution would be to further optimize the existing scripts and reduce the amount of geometry like units in the scene. The spikes in the later stages are most likely from the artificial intelligence calculations in the background, but they do not really influence the average render time so they were considered acceptable.

6.2. Goals

Three areas are particularly interesting for an user study on the game.

The first and most important question is the usefulness as a Serious Game. Each Serious Game has a serious purpose which was in this case to teach the player about

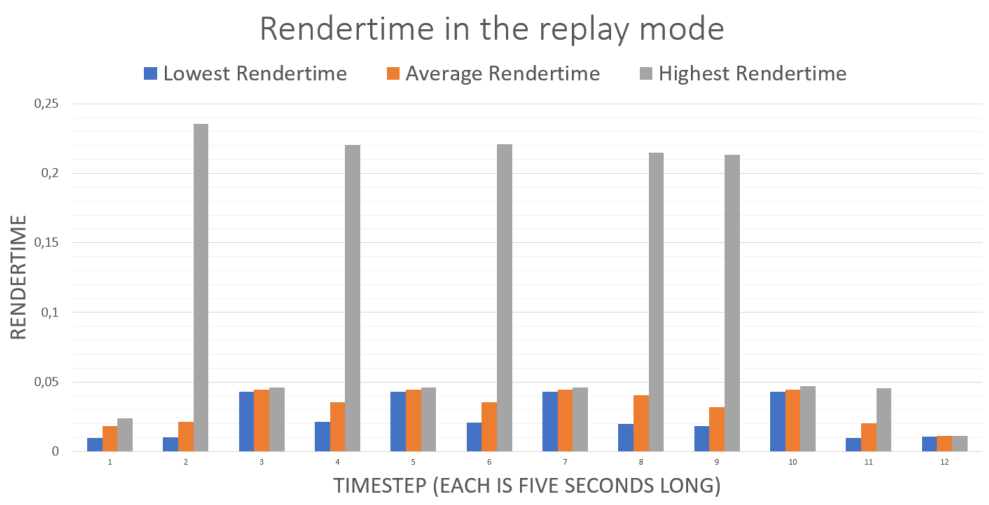


Figure 6.2.: Rendertime in the replay mode

Source: Selfmade

the tactics in Ancient Battles. The survey has to capture the knowledge about the tactic which was used by Hannibal after playing the game. But it also has to consider that the player possibly already knew the tactic beforehand. The knowledge about previous battles and unit types in the army which was put also into the game could be tested aswell.

Another interesting area is the implementation of the user interface in the game. Due to the shift to Virtual Reality, the user interface has to be adapted to the new needs. User interfaces are still an area of research so the take on this topic in this Serious Game should be evaluated. As stated in the related work session, the number of Serious Game in Virtual Reality is rather low so there is not a lot of existing work to compare the user interface to.

The last topic are the controls of the game with the HTC Vive controllers. These controllers are motion tracked by the base stations and offer new possibilities for the controls. But these new possibilities also pose some challenges which have to be solved. The control schema of this game should be evaluated to check if it fits the game and uses the newly offered possibilities.

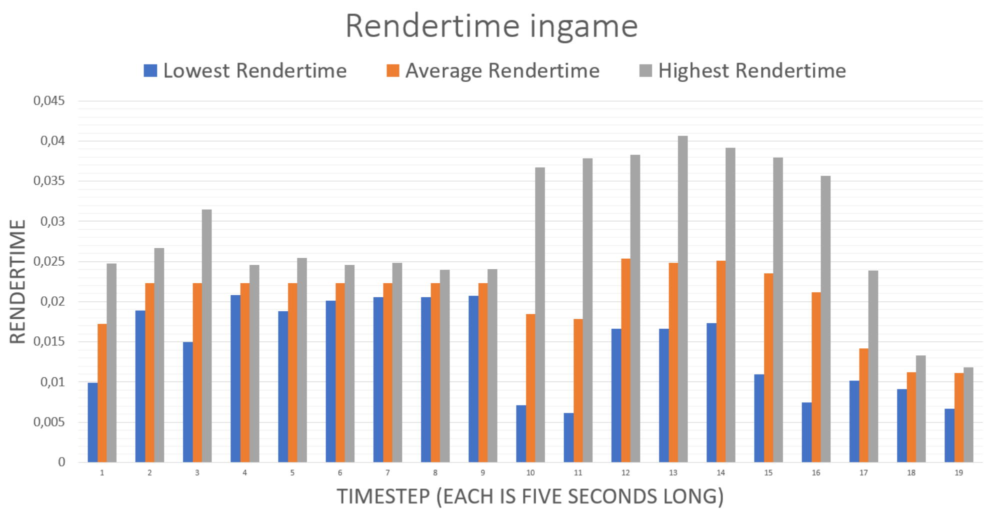


Figure 6.3.: Rendertime ingame

Source: Selfmade

6.3. Results

The survey itself was done as a two part survey before and after the gameplay test on a google formular. The results were added in Appendix A. The number of participants was eight.

The first two questions try to determine the demographic background of the users. One can clearly see that the users were mostly male and in the age range of 19-34. This leads to an increased acceptance of new technology and knowledge about existing technology and should be noted beforehand.

The next section's questions asks about existing knowledge. The user rate their existing knowledge about the Ancient Battles and especially about the Punic Wars as low, but were able to name some important Ancient Battles like the Battle of the Teutoburg Forest or the Battle of Gaugamela. The question was not directly related to the Punic Wars so that the user do not look actively for exactly this information later in the game. The design was chosen in this way to avoid that the user look for specific facts after believing to know what will be asked later on. The last question asks about the frequency of use of virtual reality headsets. The results are quite mixed but low on the high end and on the low end which should provide interesting results for the Virtual Reality related questions later on.

In the first question of the set of questions, which were posed after the gameplay test, the user is tasked to rate how fun the game is. The results were rather good with an average of 3.5 points on a 5 point scale. Due to the added serious purpose of this game, this result can be seen rather good, but there are still possibilities to further the improve the fun.

Almost all users said that they did not experience motion sickness except one. The one person gave a free text answer which shows that the motion sickness from the question was not clearly defined and the user had to define it for himself. Here the design of the study was flawed and a definition should have been provided to the user.

The questions 3.4.1 and 3.4.2 are about the user interface in the game. The interface has been rated fairly high where 87,5% of the user have rated it with 4 points on 5 point scale. This interface is fairly suitable for the purpose of the game, but there are some additional remarks which should be implemented as stated by some of the users. The battle phase should be redesigned, so that each battle is more visible and the user can recognize which unit took how much damage. At the moment the user waits for the health damage modifiers to disappear and then evaluates the situation after that. But it is not easy to redesign it without increasing the battle phase time too much, because there are often a lot of fights during the battle phase. Another solution has to be found for this problem. The ranged units were missing a counter for their ammunition. Also, units which were not used yet, are not marked, so the user sometimes has to guess if the unit was already moved or is waiting for a command. Due to the amount of units and the restricted place, each unit is rather small, which is another remark which can not easily be fixed without reducing the number of units.

The controls are the topic of the next two questions. The rating of the controls is ok with an average of 3.125 points but especially the two ratings with the value two show that there is a lot of room for improvement. The additional remarks show a lot of the problems the user had for this questions. The rotation feels imprecise for the user because it is not continuous. The user should not be required to actually click on the touch pad, the unit should rotate all the time if the user touches the touchpad. The game is missing an in place movement which is required for some turns. The user can not command his unit to stay in place and attack from there. This should be fixed by allowing the selection of an in place attack. Also the movement is restricted because the user can not select places where another unit is standing during his round. This restrictions should be removed to allow the user to use the complete tactical possibilities of the unit movement. Another problem is the missing possibility for multiselection of units. Due to the high number of units, it is rather exhausting for the user to give each unit its own command. If the user was able to give the same command to multiple units, he could move multiple units at once which helps with the exhausting round time. Additionally, an user suggested direct button control for the replay mode instead of the replay device.

Most users did prefer the Virtual Reality experience compared to the monitor experience. But this could also be explained by the excitement about the new technology which most people did not try very often.

Most user did not watch the battle in the replay mode. During the observation, the author noticed that the users wanted to actually give commands and be active instead of watching the replay which is more a film-like experience. This is not surprising and should be noted. The implementation for the complete replay was not suitable due to this fact. An one step or two step replay in the game if the user changed something compared to Hannibal would be better, but that option has to be explored in future work.

The last two questions tried to show if the user learned something about the battle. It is not known if the user knew the strategy beforehand, but due to them not being able to name the battle in question 2.3, one can assume that they did not actually knew the strategy beforehand. A high amount of users learned the strategy in their greatest abstraction (surrounding the enemy). No user did provide a more in-depth explanation why he was able to surround them. But another user provided an interesting answer where he explained that he did not know the strategy because he lost. This shows another flaw in the concept as the user was not guided to the replay if he lost and did not learn the actual serious purpose of this game. Only one person was able to name one unit with the correct historic name which was also available as a unit in the game. Some people noted that they played as Carthago. The unit selection and the available unit was also visible if you selected an enemy unit so this is not a valid reason to not know these names. But the absence of these names shows that the users did not use the unit information which was attached to the second controller. This could be related to the fact that it was not directly required to look at these information because the playing area provided most of the information like health bars and unit position.

During the play tests, the author noticed that the average play time was too long. Most player do not have the endurance to play for 45 minutes in a single battle. The author suggests a reduction in play time to around 20 minutes which seems to be a time frame the user wants to spend on a game. This can be achieved by removing some units, remove restrictions in the movement and allow multiselection, reducing the artificial intelligence calculation time and reducing the amount of units.

7. Conclusion and Future Work

This thesis shows the analysis of existing games and the requirements for a new Serious Game. The use of Virtual Reality is discussed and the design decisions, especially on the user interface and the controls, are highlighted. The implementation in Unity is explained. The evaluation showed that some goals were not reached, especially in the serious purpose of the game and the controls of the game. The controls needed more testing during the development, especially with other users which did not have experience with the control scheme. Possible improvements for this implementation were discussed in the evaluation chapter. The serious purpose, the transfer of knowledge about the tactic in the battle was reached, but the addition of other information was not successful. This could be related to the fact that the users were not forced to use the information for the game and were not paying attention to them for this reason. The author suggests that future attempts try to integrate this information even more into the game and require the use of them. This thesis showed that the implementation of a serious game is possible and that people can have fun while learning about a topic. This is interesting for future projects because it shows the potential of games for the purpose of education.

A. Results Study

Eight results were recorded.

A.1. Demographic questions

Question 1.1: Please state your gender:

- Male (6 = 75%)
- Female (2 = 25%)
- Other

Question 1.2: How old are you ?

- Under 12
- 13 - 18
- 19 - 24 (4 = 50 %)
- 25 - 34 (3 = 37,5%)
- 35 - 44
- 45 - 54 (1 = 12,5 %)
- 55 - 64
- Above 65

A.2. Previous Knowledge

Question 2.1: How much do you know about Ancient Battles ?

- 1 - Nothing (2 = 25 %)

- 2 (6 = 75 %)
- 3
- 4
- 5 - Almost everything

Question 2.2: How much do you know about the Punic Wars ?

- 1 - Nothing (4 = 50 %)
- 2 (3 = 37,5 %)
- 3 (1 = 12,5 %)
- 4
- 5 - Almost everything

Question 2.3: Name some important Ancient Battles (Freetext - Question)

- Schlacht im Teutoburger Wald, Termopylen
- Schlacht von Kartago
- Battle of Gaugamela (Alexander)
- Teuteburger Wald
- Schlacht im Teuteburger Wald, Belagerung von Troya

Question 2.4: Did you ever use a Virtual Reality Headset before ?

- No, I haven't used one before (1 = 12,5%)
- Yes, only once or twice (3 = 37,5%)
- Yes, from time to time (3 = 37,5%)
- Yes, all the time (1 = 12,5%)

A.3. After Gamplay Test Questions

Question 3.1: How fun was the game ?

- 1 - Not very fun
- 2 (1 = 12,5 %)
- 3 (3 = 37,5 %)
- 4 (3 = 37,5 %)
- 5 - A lot of fun (1 = 12,5 %)

Question 3.2: Did you experience motion sickness during the play time ?

- Yes
- No (7 = 87,5 %)
- Bin mir nicht sicher, komisch war das schon (Freetext answer) (1 = 12,5 %)

Question 3.3.1: Rate the User Interface (elements, which show additional Information to the player but are not real components of the 3D world. An example would be the path which shows the route of the units)!

- 1 - Badly designed
- 2
- 3 (1 = 12,5 %)
- 4 (7 = 87,5 %)
- 5 - Well designed

Question 3.3.2: Additional remarks for the user interface (Freetext question)

- Damage indicators after each other
- Please display the spear count for ranged units. Mark unused units.
- health damage not clear which unit
- alles ein bisschen klein

Question 3.4.1: Rate the controls of the Game!

- 1 - Does not work well
- 2 (2 = 25 %)
- 3 (3 = 37,5 %)
- 4 (3 = 37,5 %)
- 5 - Works very well

Question 3.4.2: Additional remarks for the controls (Freetext question)

- Movement feels a bit imprecise
- Add button control for replay mode, continous rotation while touch the pad
- no in place movement
- no multiselection

Question 3.5: Do you prefer a Virtual Reality experience compared to the typical monitor experience in this game context ?

- Yes (6 = 75%)
- No (2 = 25%)

Question 3.6: Did you watch the battle in the replay mode ?

- Yes (2 = 25%)
- No (6 = 75%)

Question 3.7: Please shortly describe the strategy which Hannibal used to beat the Roman Army (Freetext question)

- He surronded them
- attack from behind with the ponys?
- Encircle them.
- he surrounded them with his cavalry

- he encircled them with his units
- don’t know, I lost
- sorrounding, flanking with horses
- sie kamen von allen seiten

Question 3.8: Name some units in the Roman Army (Freetext question)

- I did not play the romans
- Legionär?
- Didn’t play the romans
- Hastati
- did play as carthago
- infantry, cavalry
- Kavallerie? Pilumwerfer?
- Reiter, Speerträger

List of Figures

List of Tables

Bibliography

[Abt87] C. C. Abt. *Serious games*. Reprinted. Lanham [etc]: University Press of America, 1987. ISBN: 0819161489.

[Ali17] A. Aliaga. *Concave Hull Generator*. 2017-10-17. URL: https://github.com/Liagson/ConcaveHullGenerator.

[Ant] Anthony Stonehouse. *User interface design in video games*. URL: https://www.gamasutra.com/blogs/AnthonyStonehouse/20140227/211823/User_interface_design_in_video_games.php (visited on 10/12/2017).

[BBC] BBC. *Episode 1, Time Commanders - BBC Four*. URL: http://www.bbc.co.uk/programmes/b084xym1 (visited on 10/12/2017).

[Bed05] R. Bedser. *Hannibal v Rome*. 2005.

[Bee] C. Beers. *Alexander Review*. URL: https://www.gamespot.com/reviews/alexander-review/1900-6114517/ (visited on 10/04/2017).

[Ber+15] M. Bernardes, F. Barros, M. Simoes, and M. Castelo-Branco. "A serious game with virtual reality for travel training with Autism Spectrum Disorder." In: *2015 International Conference on Virtual Rehabilitation (ICVR)*. Ed. by IEEE. IEEE, 2015, pp. 127–128. ISBN: 978-1-4799-8984-3. DOI: 10.1109/ICVR.2015.7358609.

[BG] D. Beeler and A. Gosalia. *Asynchronous Timewarp on Oculus Rift*. URL: https://developer.oculus.com/blog/asynchronous-timewarp-on-oculus-rift/ (visited on 10/12/2017).

[Bria] Brilliant Game Studios. *Brilliant Game Studios - Facebook*. URL: https://de-de.facebook.com/BrilliantGameStudios/ (visited on 10/04/2017).

[Brib] Brilliant Game Studios. *Ultimate Epic Battle Simulator bei Steam*. URL: http://store.steampowered.com/app/616560/Ultimate_Epic_Battle_Simulator/ (visited on 10/04/2017).

[But] S. Butts. *Alexander - IGN*. URL: http://www.ign.com/articles/2004/12/02/alexander (visited on 10/04/2017).

[Cre17] Creative Assembly. *Creative Assembly I Welcome to Creative Assembly*. 9.10.2017. URL: https://www.creative-assembly.com/ (visited on 10/12/2017).

[DAJ11] D. Djaouti, J. Alvarez, and J.-P. Jessel. *Classifying Serious Games: the G/P/S model*. 2011. URL: `http://www.ludoscience.com/EN/diffusion/537-Classifying-Serious-Games-The-GPS-Model.html` (visited on 09/26/2017).

[Dal03] G. Daly. *Cannae: The experience of battle in the Second Punic War*. Repr. London: Routledge, 2003. ISBN: 9780415261470.

[Edu17] EducatedGrizzlyBear. *Loading Screen Ultimate Epic Battle Simulator*. 6.06.2017. URL: `https://i.imgur.com/hpBtAyG.jpg` (visited on 10/04/2017).

[ETC] ETC-USC. *CES2016_HTCVive_Pre_Winters*. URL: `https://www.flickr.com/photos/92587836@N04/24177102722/` (visited on 10/04/2017).

[Fri] B. Frischer. *Rome Reborn*. URL: `http://www.romereborn.org/` (visited on 10/04/2017).

[Gar90] C. Garvey. *Play*. Enl. ed. The Developing child. Cambridge, Mass: Harvard University Press, 1990. ISBN: 9780674673656.

[Gra] Grand View Research. *Virtual Reality Market Size Worth $48.5 Billion By 2025 | CAGR: 46.7%*. URL: `http://www.grandviewresearch.com/press-release/global-virtual-reality-vr-market` (visited on 10/04/2017).

[Gro11] T. Groleau. *Approximating Cubic Bezier Curves in Flash MX*. 21.06.2011. URL: `http://www.timotheegroleau.com/Flash/articles/cubic_bezier_in_flash.htm` (visited on 10/19/2017).

[GSC] GSC Game World. *GSC Game World*. URL: `http://www.gsc-game.com/` (visited on 10/04/2017).

[Hig17] T. Higgins. *Human Horse Model*. 2017-10-17. URL: `https://www.assetstore.unity3d.com/en/#!/content/16687`.

[HTC] HTC. *Vive | Discover Virtual Reality Beyond Imagination*. URL: `https://www.vive.com/de/` (visited on 10/04/2017).

[LC87] W. E. Lorensen and H. E. Cline. "Marching cubes: A high resolution 3D surface construction algorithm." In: *ACM SIGGRAPH Computer Graphics* 21.4 (1987), pp. 163–169. ISSN: 00978930. DOI: `10.1145/37402.37422`.

[Liu+16] A. J. Liu, Z. Dong, M. Hašan, and S. Marschner. "Simulating the Structure and Texture of Solid Wood." In: *ACM Trans. Graph.* 35.6 (2016), 170:1–170:11. ISSN: 0730-0301. DOI: `10.1145/2980179.2980255`. URL: `http://doi.acm.org/10.1145/2980179.2980255,%20[Titel%20anhand%20dieser%20DOI%20in%20Citavi-Projekt%20%C3%BCbernehmen]`.

[Liu06] D. Liu. *Dead Space*. 2012-10-06. URL: `https://asura1234.files.wordpress.com/2012/10/fff48669_vbattach858671.jpg` (visited on 10/23/2017).

[M13] D. M. *Test: Total War – Rome 2*. 30.07.2013. URL: http://www.ingame.de/files/2013/08/Vorschau-Total-War-Rome-2-2.jpg (visited on 10/12/2017).

[mar] marketsandmarkets.com. *Serious Game Market by Vertical (Education, Corporate, Healthcare, Retail, Media and Advertising), Application (Training, Sales, Human Resource, Marketing), Platform, End-User (Enterprise, Consumer), and Region - Forecast to 2020*. URL: http://www.marketsandmarkets.com/Market-Reports/serious-game-market-67640395.html (visited on 10/04/2017).

[Mas] E. Massenberg. *Verschiedene Texturen für 3D Modelle*.

[Met] Metacritic. *Alexander*. URL: http://www.metacritic.com/game/pc/alexander (visited on 10/04/2017).

[Mic] Microsoft. *Microsoft Hololens*. URL: https://www.microsoft.com/de-de/hololens (visited on 10/04/2017).

[Mix23] Mixamo. *Auto-Rigger for 3D Models*. 2017-06-23. URL: https://www.mixamo.com/auto-rigger.

[Owe] B. Owens. *Use Tri-Planar Texture Mapping for Better Terrain*. URL: https://gamedevelopment.tutsplus.com/articles/use-tri-planar-texture-mapping-for-better-terrain--gamedev-13821 (visited on 10/13/2017).

[Pal] M. Palko. *Triplanar Mapping*. URL: http://www.martinpalko.com/triplanar-mapping/ (visited on 10/13/2017).

[Pro] O. Prof. Dr. Bendel. *Virtuelle Realität*. Ed. by Springer Gabler Verlag. URL: http://wirtschaftslexikon.gabler.de/Archiv/-2045879784/virtuelle-realitaet-v1.html (visited on 10/04/2017).

[rpe17] rpetrusha. *Task-based Asynchronous Programming*. 18.10.2017. URL: https://docs.microsoft.com/en-us/dotnet/standard/parallel-programming/task-based-asynchronous-programming (visited on 10/19/2017).

[Sam] Samsung. *Samsung GALAXY Note Gear VR - Funktionen*. URL: http://www.samsung.com/de/promotions/galaxynote4/feature/gearvr/ (visited on 10/04/2017).

[Sch] C. F. Schneider. *Total War: Rome 2 - 117 Fraktionen, 500 Einheiten, 183 Karten-Regionen - GameStar*. URL: http://www.gamestar.de/artikel/total-war-rome-2-117-fraktionen-500-einheiten-183-karten-regionen,3010829.html (visited on 10/04/2017).

[SEG] SEGA. *This is Total War*. URL: https://www.totalwar.com/ (visited on 10/04/2017).

[SM00] H. Schumann and W. Müller. *Visualisierung*. Berlin and Heidelberg: Springer, 2000. ISBN: 3-540-64944-1. DOI: 10.1007/978-3-642-57193-0. URL: http://dx.doi.org/10.1007/978-3-642-57193-0.

[SR08] B. Sawyer and D. Rejeski. *Executive Summary of Serious Games: Improving Public Policy Through Game-based Learning and Simulation*. 2017-11-08. URL: https://www.wilsoncenter.org/publication/executive-summary-serious-games-improving-public-policy-through-game-based-learning-and.

[Sto] StoneFox. *thestonefox/VRTK*. URL: https://github.com/thestonefox/VRTK (visited on 10/17/2017).

[stra] str8labs. *Allianz Global Learning*. URL: https://straightlabs.com/allianz-global-learning-2/ (visited on 10/04/2017).

[strb] str8labs. *Audi Virtual Training*. URL: https://straightlabs.com/audi-virtual-training/ (visited on 10/04/2017).

[The] The Department of History, United States Military Academy. *Battle of Cannae, 215 BC - Initial Roman attack*. URL: https://commons.wikimedia.org/wiki/File:Battle_of_Cannae,_215_BC_-_Initial_Roman_attack.png (visited on 10/23/2017).

[Tra] V. K. Tran. *Tileable Grass und Water Texture*.

[Uni17] Unity. *Unity - User Interfaces for VR*. 2017-10-17. URL: https://unity3d.com/de/learn/tutorials/topics/virtual-reality/user-interfaces-vr (visited on 10/17/2017).

[Vid] J. M. Vidal. "Fundamentals of Multiagent Systems." In: (). (Visited on 10/19/2017).

[Yia17] P. Yianni. *Male Human Model*. 2017-10-17. URL: https://opengameart.org/content/male-human-low-poly-base-under-1000-polys.