Environment re-creation methods for virtual heritage using a game engine with discernment of visual learning cues

David Svånå

Master of Science in Informatics
Submission date: November 2010
Supervisor: Torbjørn Hallgren, IDI
Co-supervisor: Jo Skjermo, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem description

The goal is to investigate visual presentation techniques and environmental elements that can help and guide users of interactive virtual worlds. This will be done with additional focus on new applications of these principles within virtual heritage and similar systems.

The techniques should be investigated and analyzed from a set of samples. Then, an interactive implementation of some techniques will be made. The implementation will consist of a virtual re-creation of a chosen real-world historic environment. This will be constructed using a modern graphics- and game-engine.

Start date: August 15th 2009
Supervisor: Torbjørn Hallgren, IDI
Co-supervisor: Jo Skjermo, IDI

Abstract

This thesis presents an analysis of visual cues and environmental hints gathered from computer games and cinematic theory. These cues can help users of interactive virtual worlds to navigate and understand them in a comprehensive context, in an integrated manner. This can be applied to most interactive virtual environments. It is also viewed in a perspective of *virtual heritage*; reconstructions of historical locations.

There is currently not much research documenting such cues. Here, the sampled cues are split into visual, environmental and interface categories. The techniques are analyzed both from a general standpoint and for potential use in virtual heritage. Most of the cues' analyses indicate that they could be very useful in virtual heritage or similar applications.

One such application, a high-fidelity re-creation of the medieval city of Nidaros, is made using the Unreal Engine 3 graphics- and game-engine. Construction of the environment mimics the needs of a comprehensive virtual heritage project, and provides an easily extensible test case. Many technical aspects of the construction are described in detail.

Selected cues and design techniques are successfully applied to the re-created interactive environment. Users of the program are able to walk freely in the city. A discussion of the results is provided, and many ideas for further expansion are suggested.

The results suggest that the presented combination of techniques constitute a new and promising perspective to any type of virtual environment. The use of a game engine could also help cut production costs and provide a fully interactive, high quality learning experience.

Acknowledgments

I would like to thank my supervisors, Torbjørn Hallgren and Jo Skjermo, for their guidance during this project. Their knowledge within virtual heritage has been invaluable, and their interest and support for this work have been very motivating.

The preliminary project was very important for the creation of this work and the virtual environment. I would like to thank my classmates for many entertaining days of work and inspiration, both on that project and during many other days at the lab.

Finally, I would like to specially thank my family. They have always supported me and provided their own guidance and encouragement during these years.

Contents

List of Figures

Nomenclature

Actor	Unreal Engine term; a derivate branch of the engine's main class. Most actors are directly placeable in the game world.
First-person	Gaming context; a perspective rendered from the viewpoint of the player character.
MMORPG	**M**assively **M**ultiplayer **O**nline **R**ole-**P**laying **G**ame
Nidaros	The medieval name of the present day city of Trondheim, Norway. The literal meaning is *the city by the mouth of the river Nid*.
RPG	**R**ole-**P**laying **G**ame
Third-person	Gaming context; a perspective rendered a distance from the player character. Not necessarily focused on the avatar.
UDK	**U**nreal **D**evelopment **K**it. A visualization and game development toolkit.
WIMP	Interface term; **W**indows, **I**cons, **M**ouse, **P**ull-down menu

To my friends and family.

Chapter 1

Introduction

1.1 Motivation and pragmatic significance

“Visualisation has been defined as ’to form a mental image of something incapable of being viewed or not at that moment visible”’[7]

What do you normally see when you read a book about lost civilizations or perhaps life in the medieval age? How do they convey their knowledge to you? There are nicely realized illustrations and facts about everyday life, tools and trade, architecture and so on. Oftentimes there are stories and guesswork of how it would have been. When you go to a museum you can often see these things as they were. However, they are still fragmented and disconnected; you go from one exhibition to another, and you might leave wondering where that tool or where that house had its place in society. Rarely or perhaps never do you get to see everything the archaeologists and historians know of a time or a place come together to form a comprehensive impression of it. It’s hard to remember everything you have read or been told – people have a memory that is spatial and association-based.

However, with a well-crafted and interactive 3D-world, the learning experience and entertainment value would be significantly augmented in almost every aspect. The interactivity in a real-time application can be remarkably better than with other solutions due to the visual feedback and real-time manipulation of the environment, and helps reinforce learning and lengthens the attention span you would normally achieve by presenting information passively[15, 24, 32, 39]. The 3D-environment places everything, which previously could only be studied separately, together to form a comprehensive experience.

The motivation for most virtual heritage projects in the first place is indeed to *"see the ancient world as the ancients did"*, and to create new understandings when you can see that world in context[54]. The goal should be to create an immersive world such as this. For an audience from the general public, it seems natural that a user friendly environment that triggers the audience's curiosity should follow hand-in-hand. This type of experience may be less suited for the most advanced or in-depth subjects, e.g. it is not suited to carefully study an item from a dig site or indeed to do any sort of research on individual pieces. Then again it can be argued that is rarely the case in ordinary museums either: the items that are there can be seen as an end-product, already studied, whose information is now being displayed. A point can also be made that a system of this type could with greater ease potentially suit all audience types better by layering, or at least pointing to, more advanced information.

There are several finished virtual environments and ongoing projects in the field of virtual heritage (some relevant examples are presented in chapter 2). However, the vast majority are static worlds that come with few, if any, comments or annotations about what you are seeing (or the comments are disconnected from the product, e.g. in a separate document). Many of them also have a very narrow focus on immersion, ignoring this is a world that the audience is supposed to believe. Their focus subject (commonly architecture) is often very rigorously created and accurate. But if it is to act as a virtual museum to people that are just interested in how things were like back then, it may miss its mark almost entirely. Virtual Heritage projects seldom lack focus on accuracy but they almost always lack meaningful content or entertaining immersion. This severely impedes its enjoyment – environments often do not include elements that can engage the general public[7]. This is partly due to the difficulty involved in convincing domain experts (i.e. archaeologists) that interactive 3D environments can be *"instructive – not simply eye-catching novelties"*[54]. Perhaps many are disinclined because there's relatively little research or so few examples of implementation of interactivity and other immersive characteristics in virtual heritage. This is what this thesis aims to remedy; to identify solid elements that can enhance the experience while being useful in and of themselves.

1.2 Intended audience

This thesis should appeal to people that are interested in, or work with, design and construction of interactive environments of almost any kind. In addition archaeologists, conservators, historians and museum workers should find interest in the motivation, ideas and results presented here. The non-technical chapters should also be good to follow for casual readers interested in these topics. The implementation chapters will require knowledge in computer visualization terms and technicalities. However, it is not overly complex and should not be hard to follow for people who have some experience working with computer graphics. For the reader looking to construct a similar environment him- or herself, some general programming knowledge is required. It should be noted that the implementation discussion will not include specific instructions for the development environment other than topics of select relevance.

1.3 Project goals

This thesis will present a high-fidelity interactive environment where examples of usage- and learning-enhancing elements are implemented. These elements should be selected after a careful identification and analysis of an initial, broader set of such elements. The focus will be the analysis and technical implementation of a virtual heritage environment using a few of these elements. The reader will receive motivation to help construct virtual heritage environments in this better and more audience-friendly way. The reader will gain the knowledge of what these elements are, where they are appropriate, and how to implement them in a practical way using a state-of-the-art graphics- and game-engine.

1.4 Outline

Chapter 2 presents examples of previous virtual heritage projects, predominantly interactive ones. Important relevant aspects of these projects (which contribute towards making virtual heritage or virtual environments in general more usable through the application of visual elements and design theory) are mentioned. The project that inspired this thesis is also mentioned.

Chapter 3 describes the test case, that is, the environment that will be constructed and its historical basis. Some aspects of the environment that will act as test subjects for the uncovered techniques are described here.

Chapter 4 provides background and pointers for where to search towards appropriate techniques, and then goes on to identify and analyze a selection of these in-depth.

Chapter 5 recounts the process and challenges had with the implementation of the virtual environment itself together with the elements discussed in chapter 4.

Chapter 6 discusses the results from chapter 4 and 5 and provides an analysis of the implementation's current state as well as shortcomings, and a comparison of the final result and project goals.

Chapter 7 summarizes the thesis, analyzes the discussions of the previous chapter in order to reach conclusions of the thesis' accomplishments, and points to the future work that can be carried out to build upon or branch out from these conclusions, as well as what can be added to the test case environment.

Chapter 2

Previous work

2.1 Interactive virtual heritage environments

Some significant projects in the field of virtual heritage are presented in this section. In addition to an impression of the work and history of virtual heritage, relevant elements to this thesis' research can be identified in many of the projects. Although methods of conveying information in an informative or entertaining way is not a focus of their research, many of them have utilized such elements in some way - both consciously and unconsciously. A majority of the implementations use a game engine as their choice of presentation method, and discuss their experiences using such tools.

2.1.1 Dudley castle virtual tour

An early computer visualization of Dudley Castle in the West Midlands of England as it appeared around 1550[41]. The system itself was opened by HM the Queen in June 1994 and was in use in the castle's Visitor Centre until 2005. Coining itself the first example of a Virtual Tour (and the use of that term), the system consisted of a series of still images that was entirely computer rendered but based on photographs of today's ruins and combined with the historical knowledge of the castle to create an impression of touring through the ruins as they might have appeared. The user could navigate through the ruins image by image using a few buttons on the system's stand in the Visitor Centre.

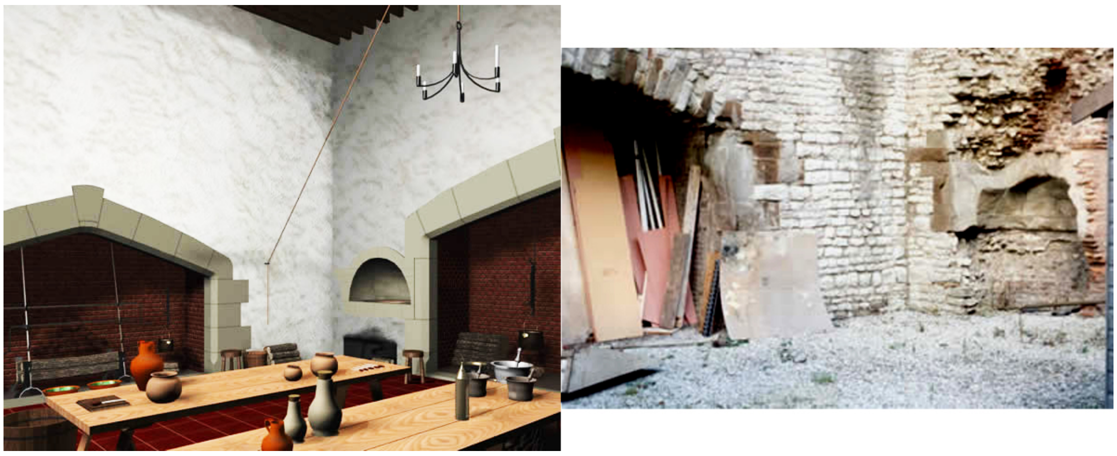


FIGURE 2.1: Dudley Castle Kitchen. The project's creator comments: "This [the picture on the right side] is what visitors see today. It takes some knowledge or imagination to build a picture in your minds [sic] eye as to what it must have been like when you are presented with this. A strong case for visualisation." From [42].

2.1.2 Virtual Notre Dame cathedral

One of the earliest examples of the use of a game engine for virtual heritage is the reconstruction of the Notre Dame cathedral from year 2000[16]. First, a preliminary project was realized called the Virtual Florida Everglades. The developers compared development environments and ended up with Epic Games' Unreal Engine, praising its then-cutting-edge technical capabilities and cost effectiveness. This engine is still being updated and is widely used today. The Virtual Florida Everglades was used to test various aspects of development such as AI, graphics and performance on a personal computer. The results from this project were then used to determine the system requirements and to confirm continued use of the engine for the main project: the virtual Notre Dame cathedral. The developers stressed the importance of a real 3D-modeled world with more impact - no stitched images or other tricks which is not really worthy of the label *virtual*.

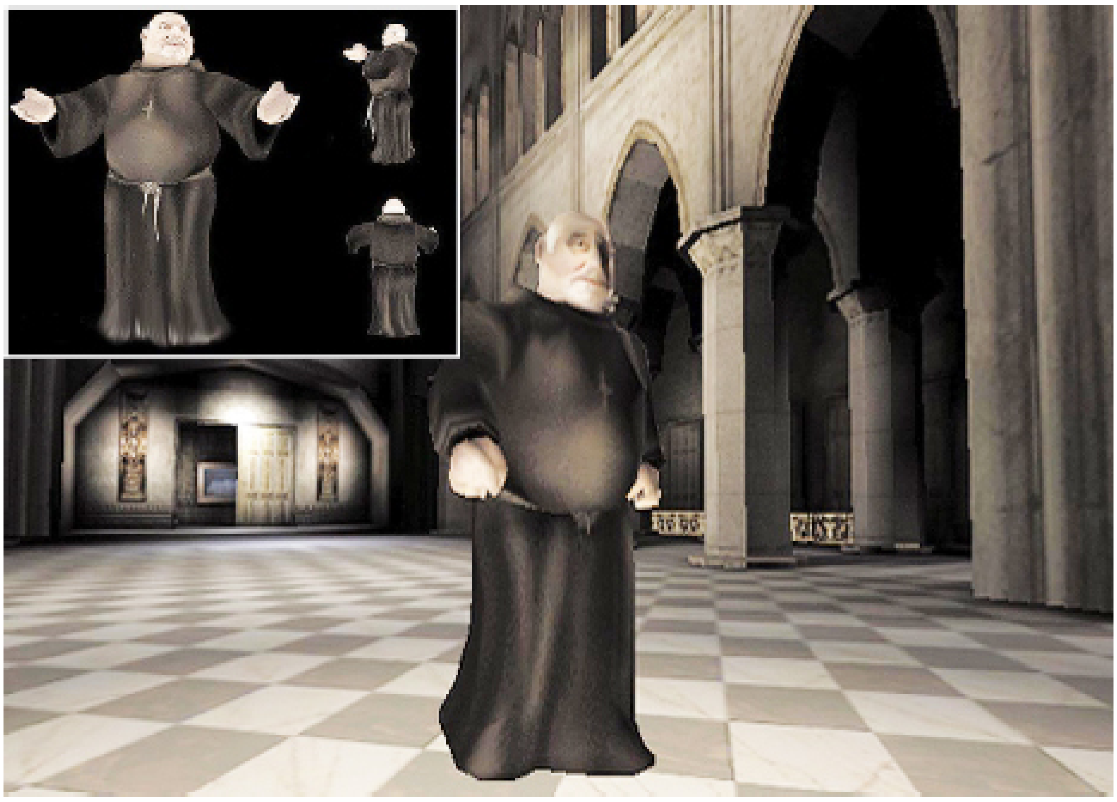


Figure 2.2: The re-created Notre Dame cathedral with the project's virtual tour guide. From [16].

The team employed several modelers and programmers concentrating on portions of the cathedral, with the modelers especially focusing on how to reduce the polygon count - an issue which is not as much of a problem in more modern engines which can process massive amounts of polygons. However, the result was very convincing at the time. In addition, the team made a *"virtual tour guide"* who could lead users around the cathedral and explain points of interest.

The team concluded that an impression of reality at this level leads the user into a state of immersion. The employment of less orthodox means such as a game engine makes it possible to arrive at an otherwise "impossible" goal and to stay within a limited budget. The VRND project has since been referred to many times in literature discussing virtual heritage.

2.1.3 Fatehpur Sikri

Also from year 2000, this was a reconstruction of the palace complex of Fatehpur Sikri which is a UNESCO World Heritage Site[30]. This was done as a case study when researching the general possibility and techniques of *"documentation of complex heritage structures"*. It was called an excellent challenge for modeling with easily available source material (especially orthographic projections of buildings). The visualization software utilized in the first stage of the project were AutoCAD and 3d Studio Max - which means this was not an interactive endeavor from the start. For the second phase of the project, a proper walk-through engine was developed.

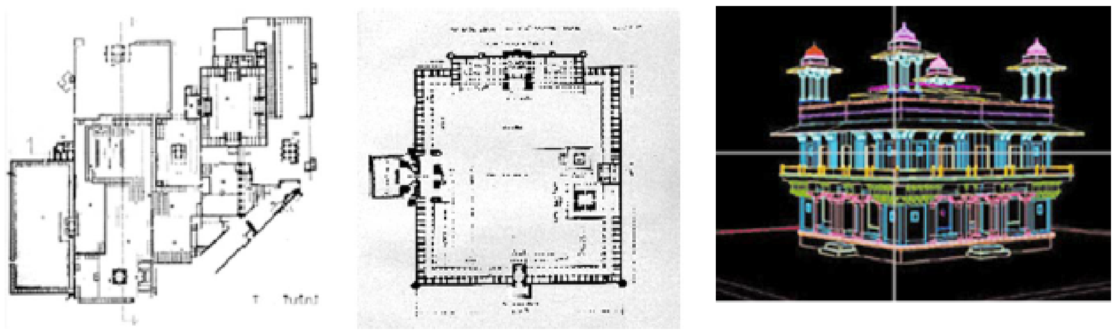


Figure 2.3: Fatehpur Sikri re-creation. 3D modeling of heritage structures using projections from archaeological surveys. From [30].

A major challenge was to create a fairly high level of detail in the environment with the limited computer processing power available. A particularly prominent problem that they found was that typical heritage re-creations are characterized by a high level of detail, i.e. a high number of polygons on single objects. Because of this, the environment was optimized and fragmented into smaller elements. The performance results that were achieved when running in real-time were deemed satisfactory - although the frame-rate would not be considered satisfactory by today's standards. This is a challenge the team noted as possible future work. It was concluded that overall, a construction of such a virtual environment was possible and would enable users to examine a heritage site without physically being there.

2.1.4 The virtual Pompeii project and the virtual theatre district of Pompeii

A reconstruction of the Theater District in the Roman city of Pompeii, the projects have seen some iterations[40, 67]. Originally developed in 1995, it was technically advanced for its time[52]. Later iterations have used both the Unreal Engine and most recently the Unity engine, both of which used some of the original material made in the 1995 project. The results are made publicly available on the Internet *"as an opportunity to explore how to to employ Immersive VR and desktop 3D media for education"*.

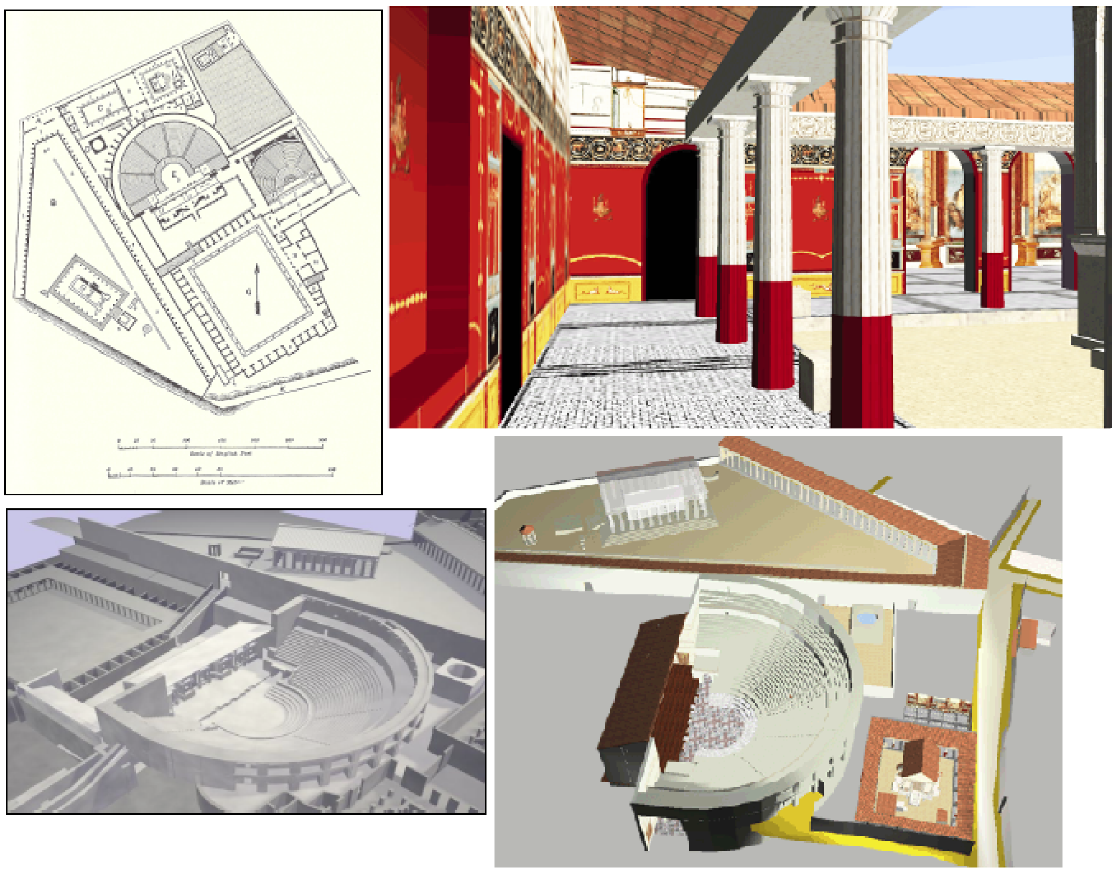


FIGURE 2.4: *Top left:* Plan of the Theatre District. *Bottom left:* The model in Unity format. *Top and Bottom right:* Images from the 1995 project. From [40, 67].

Indeed, the creators' goal is educational. Notably, they have not actively distinguished between objects from different time periods in the model due to uncertainty and continuous new discoveries. However, it is a *"representative collection of historical images [...] available for public use"*. On their website, the user can click on a model to be presented with information about it, enabling the user to focus on specific areas of the reconstruction and to compare that over several time periods. This has been a goal for some time, with the authors expressing the possibility of the creation of a game. In such a game, one proposed mechanic was to enable the user to use revealed information, gained by clicking or selecting a model, to be allowed to explore other areas.

The authors suggest other applications the project can be used in or built upon. It could be classroom activities including ones that actively involve construction and modification of the environment. Museums could benefit from presenting exhibits of physical objects that are also present in the virtual representation. The current model using Unity and the website in which the model will be presented is still under construction at the time of this writing.

2.1.5 The virtual egyptian temple

A project under continuous development, the Virtual Egyptian Temple does not originate from any particular site, but is a combination of key elements from a New Kingdom temple[39]. It is an example of game engines used actively in the construction of a virtual heritage environment. A 2005 version used the modifiable engine of *Unreal Tournament 2004* by Epic Games, citing its support of content creation and re-programmability as well as support for multiple users over the Internet as reasons for the choice.

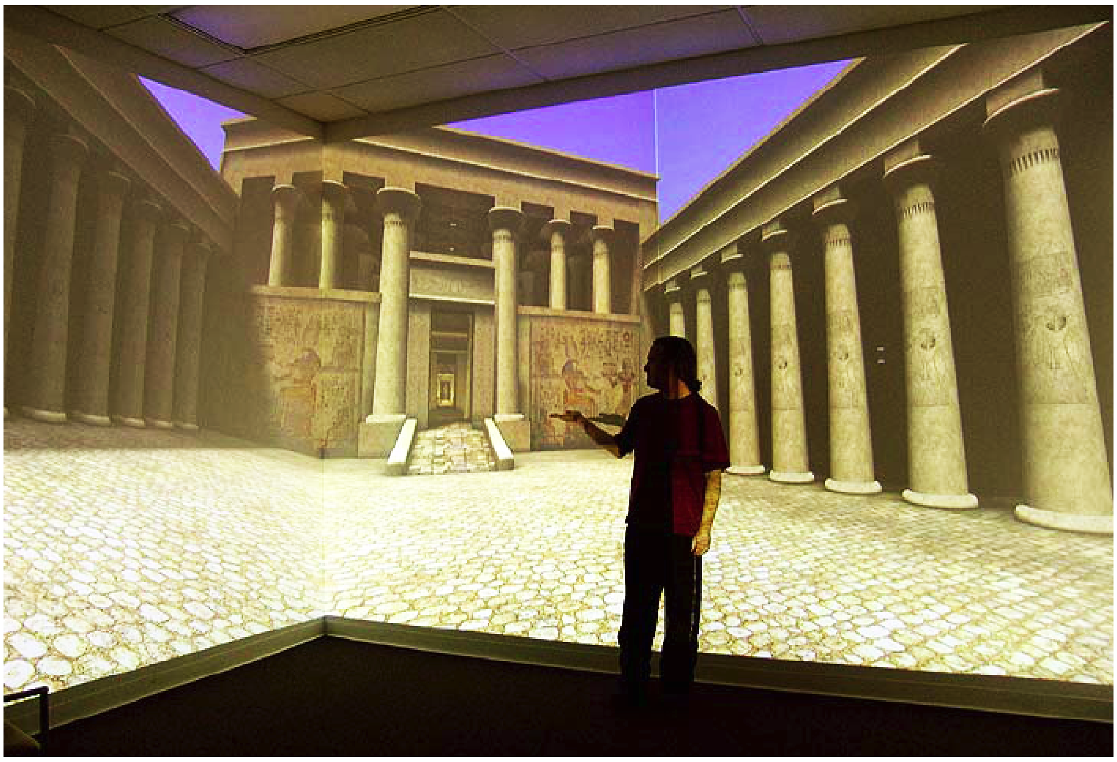


FIGURE 2.5: The Virtual Egyptian Temple as viewed in the Virtual Theater. From [39].

The idea of avatar interaction, such as a tour guide (in a similar fashion as the Virtual Notre Dame cathedral), are also mentioned as reasons for the choice of technology. The team utilized a modification of the game to adapt it to a large immersive display they call the Virtual Theater. The team argues in favor of this perspective:

> *"[...] immersive displays are particularly effective, because the student can see the temple in life-size, with approximately the same view as if the temple were real. Immersion also conveys a sense of presence or "being there" in the virtual environment. Handled properly, this can be used to focus students' attention on the subject matter."*[39]

As future work the team mention developing interactive verbal descriptions that can be activated by the user, arguing that text competes with the visual sense while sound does not. The development has lead all up to present day, and a newer incarnation is scheduled to be publicly available in 2011[38]. It uses another game engine, the Unity engine. It has added additional virtual objects and other immersive features such as ambient music.

2.1.6 Virtual heritage projects of Rome

The historically important city of Rome is the subject of multiple virtual heritage projects, some of which intertwine with each other. These projects have an unclear or limited level of actual interactivity, although many are still under development. The most ambitious project to date is the ongoing *Rome Reborn*, in which the entirety of *"urban development of ancient Rome from the first settlement in the late Bronze Age (ca. 1000 B.C.) to the depopulation of the city in the early Middle Ages (ca. A.D. 550)"* will be reconstructed over the course of several years[50]. The reconstruction effort will be receiving input from many sources and developers. The project started modeling the city at the year A.D. 320, when its population had reached its peak. The re-creation is more to model the look and feel of the city than to accurately represent every individual piece[26].

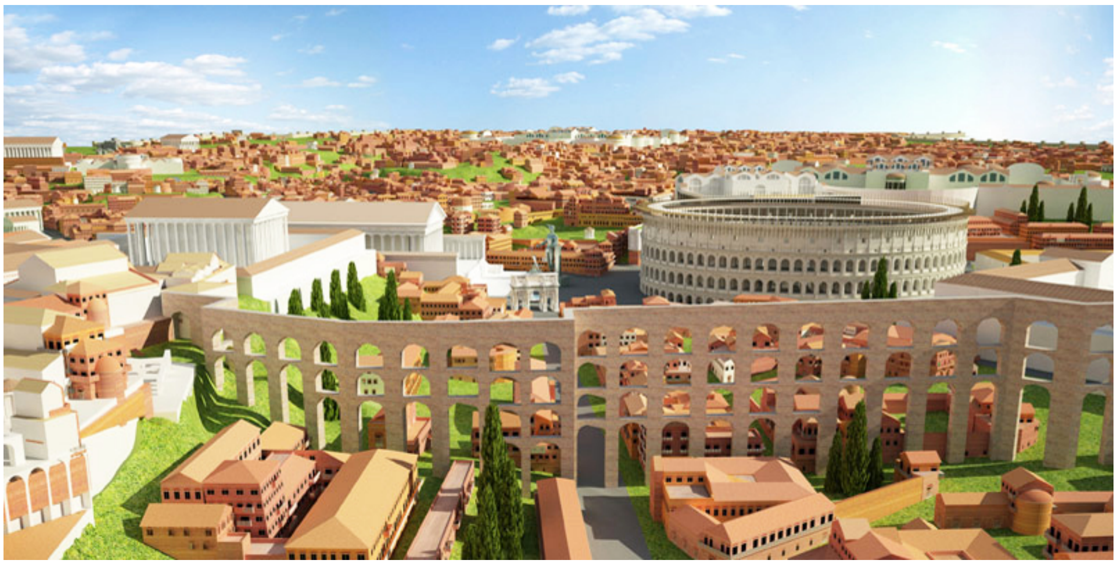


Figure 2.6: A screen shot from the 2.1 version of Rome Reborn, viewing the valley of the Flavian Amphitheater. From [50].

The project is supplied with material partly by sub-projects such as *The Digital Roman Forum Project* of the UCLA Cultural Virtual Reality Laboratory[26]. The team argues in favor of a digital model in the first place, as opposed to miniature models or engravings, by the possibility to derive the environment from accurate archaeological data which can be explored at will in three dimensions. They also mention the inclusion of two aids to make it easier for users to understand the model. The *Navigator* shows the user his or hers position in the environment by a red dot over a plan of the Forum. This plan then has several numbered features from which basic information and also meta data about the object can be accessed. Many different presentation methods are considered; the team's preferred method is using a large screen which can display the meta data at the same time as the models are navigated using the open-source 3D engine VR Juggler.

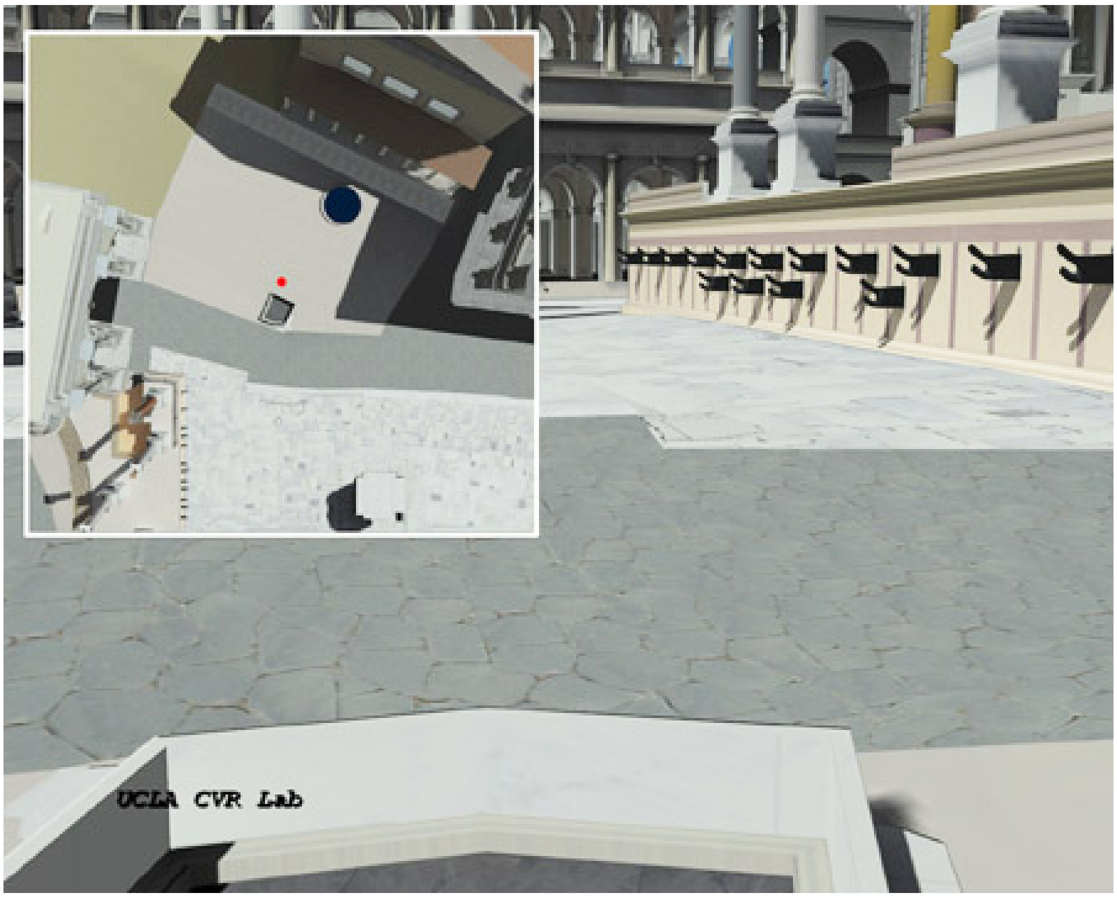


FIGURE 2.7: A view from The Digital Roman Forum Project. The Navigator window is displayed in-frame, with the user's position depicted as the red dot. From [25].

Another smaller project is VirtualRome, which is not directly related to the ones mentioned above, but is a user-navigated environment accessible directly from a browser[6]. The team discusses what good solutions for web integration would be, touching upon Global Earth Geo-viewers such as Google Earth. They ended up favoring OpenSceneGraph as a flexible general purpose system. It is noted that the web integration took relatively large amounts of time from the rest of the project. A *"framework for the integration of 3D realtime application [sic] within web browsers"* was then developed. It can be argued that this project is not in a complete state where it is useful, but the idea of browser integration is nonetheless worth closer research.

2.2 Preliminary project

The inspiration for this thesis was based on a cross-discipline local university project, *Experts in Teamwork* (EiT), where a challenge was given to construct a virtual heritage application or model[45]. The team consisted of a history student and three computer science or informatics students. Working in collaboration with archaeologists and other external domain experts, the goal was to make a re-creation of a profiled area in the medieval Norwegian city of Nidaros (today's Trondheim), more precisely around the year 1300. It was decided, for the first time in EiT's history, to use a modifiable computer game engine in order to realize the environment (as opposed to pre-rendered video). The area that was chosen was the so-called library site (*Bibliotekstomta*).

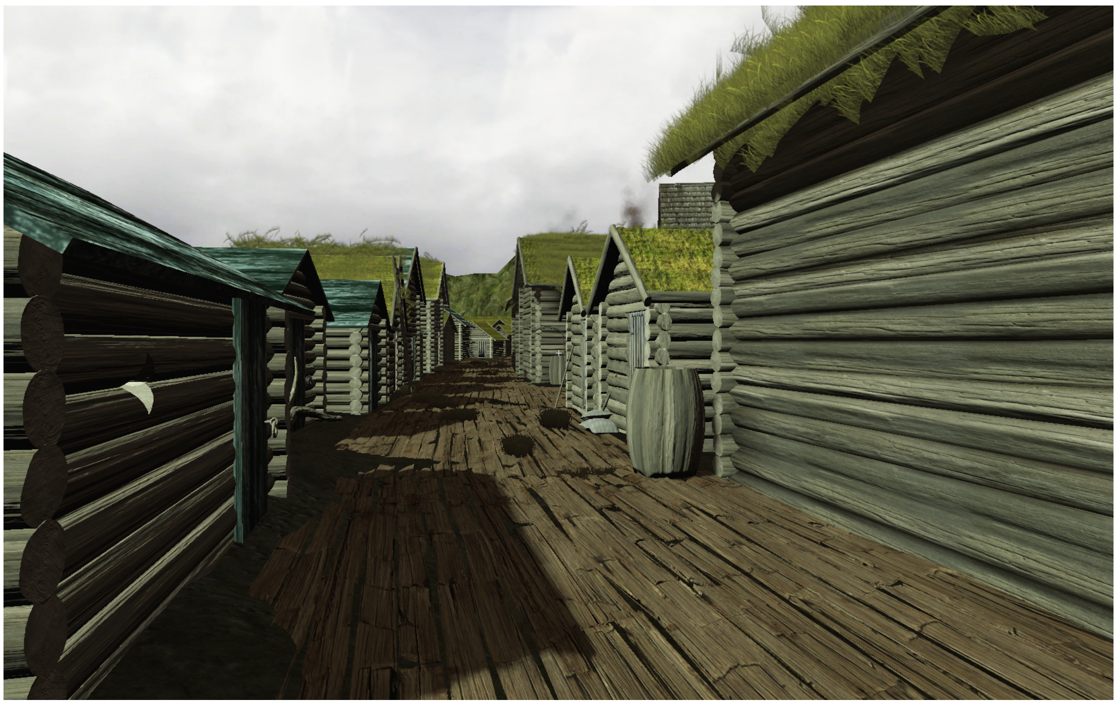


FIGURE 2.8: View from the user's (freely controlled) perspective down a street in the environment of the preliminary project. From [45].

The result received acclaim within the university, in part for its ability to create a time-travel like experience. However, it was still a static experience (with the exception of enabling the user to walk around freely in the environment) with no in-product annotations or other pointers provided to users. Rather, this information was detailed in the project report.

Albeit leaving a positive impact on audiences, the environment lacked the informational qualities necessary for it to be a viable candidate for use in a museum or to be shown to non-experts. From this, the idea that interactive virtual heritage environments need these qualities was formed. With the now demonstrated relative availability of constructing such environments, these ideas contributed to the motivation for the thesis you are reading now.

Chapter 3

Test case

3.1 Background material

The area which is created derives from the preliminary project described in chapter 2. The initial small area is extended to comprise the entire central area of Nidaros (the medieval name of Trondheim, Norway) about year 1300. The re-creation is not intended as a rigorous account of the area at that time, but is meant as test bed to implement the techniques discovered in chapter 4. Still, the historical and archaeological background will be important as means to create an authentic and believable environment which in a relevant way is able to simulate the needs of any virtual heritage project.

Most background and factual information will be extracted from the university's considerable amount of existing work in virtual heritage and medieval Trondheim. Resources such as 3D-models will be a combination of existing models from those mentioned resources as well as original 3D-modeling work.

3.1.1 Historical background

Presumably founded by the Viking king Olav Tryggvason in the year 997, the city of Nidaros was the capital of Norway until 1217. Central to its history is the Nidaros Cathedral (Norwegian: *Nidarosdomen*) on which construction spanned from 1070 and was finished around 1300. The cathedral was badly damaged in fire on multiple occasions, the first as soon as 1327. The city settlement itself suffered from many devastating fires throughout history due to the narrow streets and almost exclusively wooden houses. The entire city plan was changed to a suggestion by Johan Caspar von Cicignon after a fire in 1681[70].

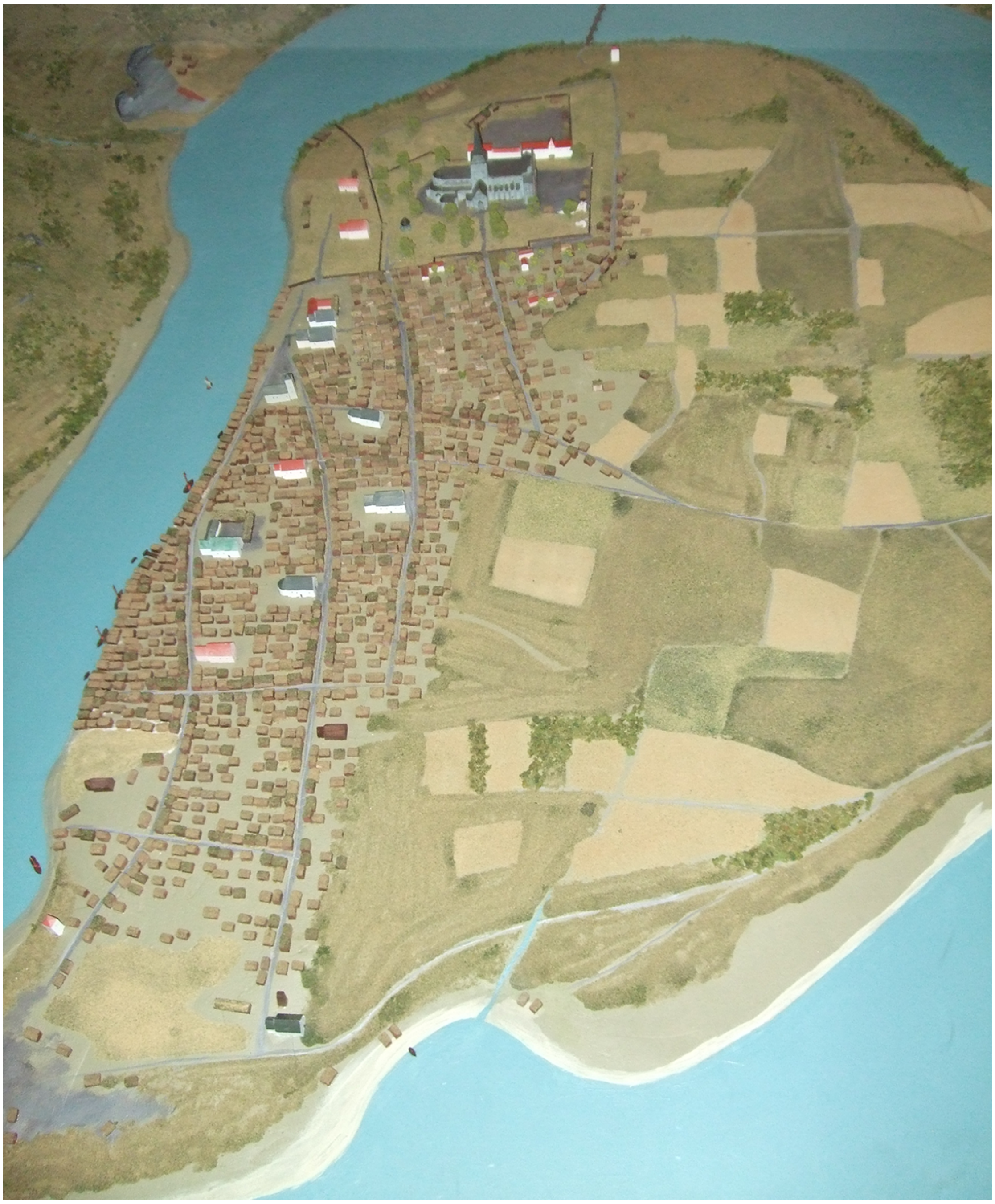


Figure 3.1: Miniature model of the city as it could have appeared around 1300, viewing the area from the northern shores. From The Museum of Natural History and Archaeology in Trondheim (The Medieval exhibition)[5, 33].

The city reached a population peak between 1300-1349. After this, the Black Death plague hit the area in full. Due to both the construction of the cathedral finishing and the population peak in this period, this time between the finalization of the cathedral and the drop in population was chosen for the re-creation.

3.1.2 Resources utilization example

Presented here is a particular example of the way external virtual assets were utilized in the virtual world created for the test case. They are too numerous, as well as being too much at the side of the thesis' scope to all be presented here.

A landmark church is *Vår Frue Kirke* which has followed the city throughout its history. It had been created for a virtual heritage animation project[4] in multiple versions which represented its history spanning the centuries. The world presented here use parts of this model as a basis for use in the game engine. A particular architectural element, a tympanum (decorate wall surface), is paid special attention to because it is used as a showcase for visual cues. This stone was located under one of the windows. The depicted scene is no longer visible, and it is replaced by a representative image. It likely was of great significance because of its protective stone arch and decorations with heads of a king and a queen on either side.

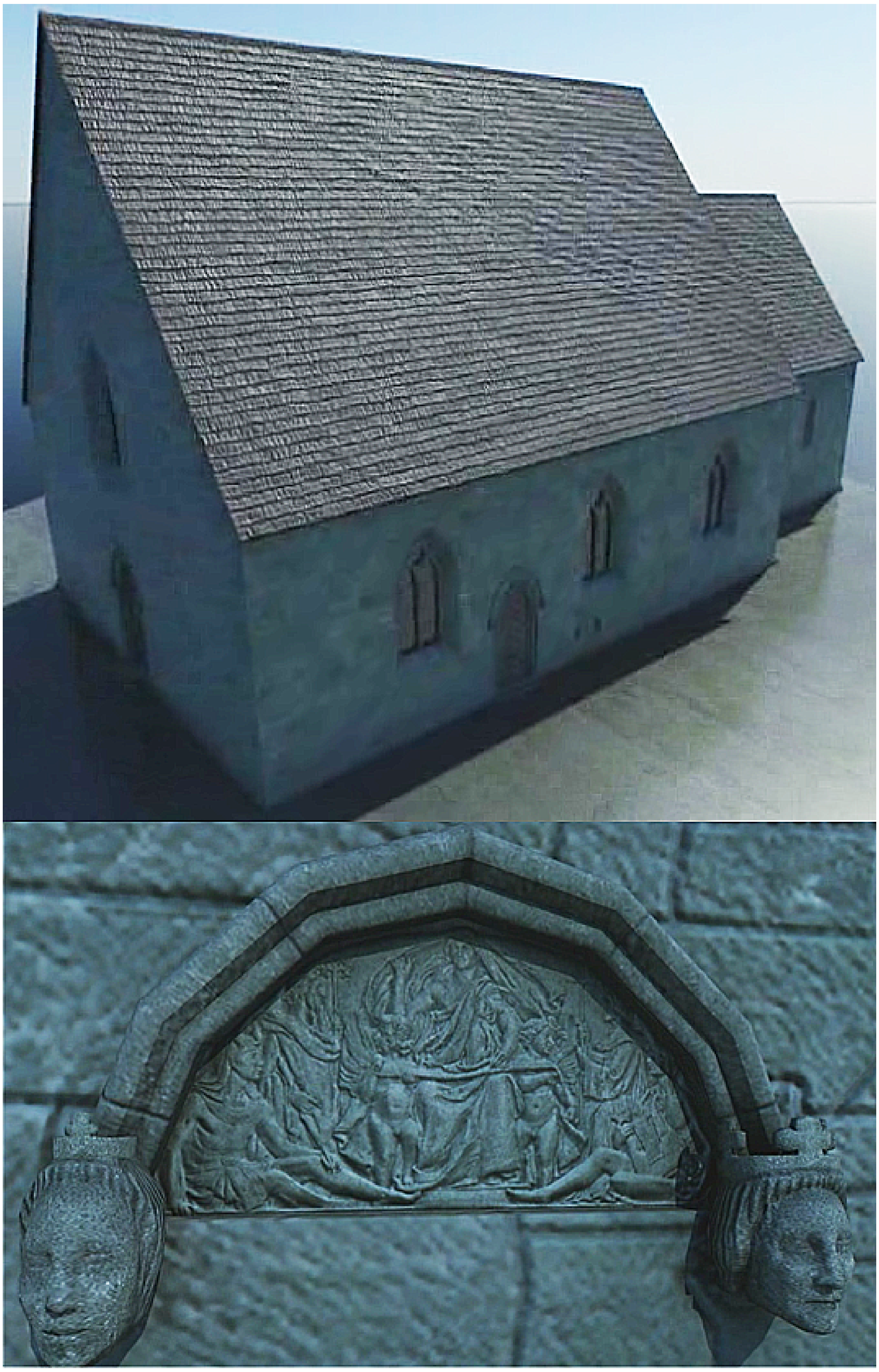


FIGURE 3.2: *Top:* A model of Vår Frue Kirke as it appeared in the 13th century. *Bottom:* The wall detail tympanum stone. From [4].

3.2 Test focus

In order to test the techniques that are identified, a route or path that the user should walk through needs to be identified. In fact, the goal of the construction should be to use the techniques to guide the user along this path. Along the way, objects of historical importance should be easily identifiable and information about them should be accessible in an integrated manner.

Because of the imbalance of available information in some areas relative to others, and to achieve a well-rounded test base, many details which would otherwise be considered significant in a fully realized re-creation will be left out or not mentioned in annotations. Specific objects and buildings or areas will be selected as convenient examples of implementation of the identified techniques.

It is natural to plan the route along the areas which we know the most about. There are large holes in the available knowledge of specific areas or buildings at the general location, but some areas have been studied more closely (or simply have more easily accessible historic resources about them). The areas of the arch bishop's residence and the cathedral have much information available about them. The busy Kaupmannastrete (literally *Merchant Street*), the street where most merchants were situated, ran from the areas of the cathedral along the river. Many churches are known to have been located along this road. Small harbors along the river bordered the street. Most populated areas stretched inwards the peninsula from here, and fields were situated outside.

This creates a natural choice of a route from the cathedral grounds north roughly along Kaupmannastrete, which would then terminate some distance before reaching the northern end of the settlements. For variety's sake, the route will go in a large circle covering much of the city.

Along this route, some interesting objects will be highlighted for closer inspection. This will require that the user actively looks around in areas close to the main route.

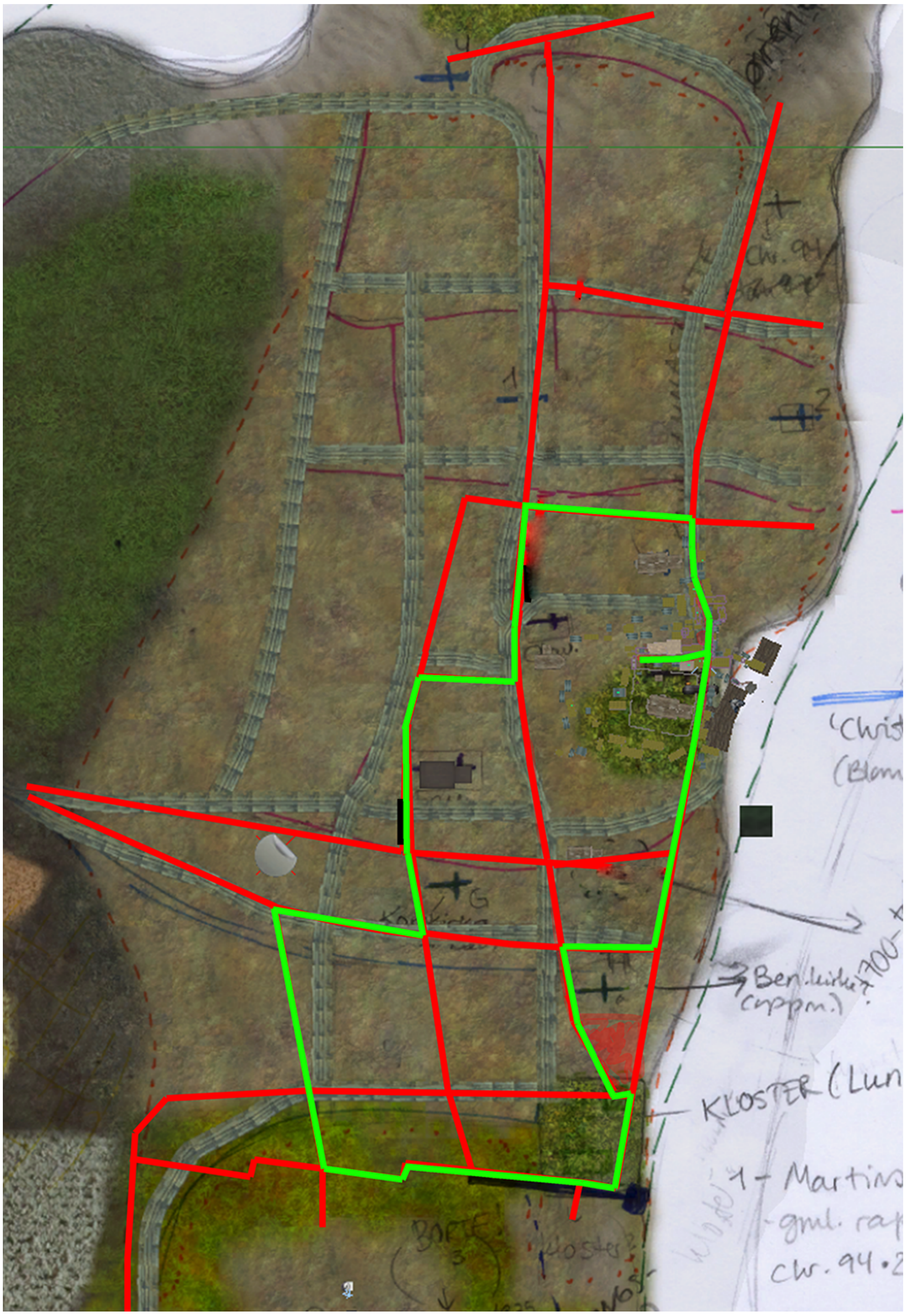


FIGURE 3.3: A rough sketch of the intended roads, partly from [5] and partly from the miniature model in figure 3.1. The green colored streets define the intended main route to walk along. Still, the user will be free to explore many of the red colored streets.

Chapter 4

Identification of techniques

4.1 Search scope and confinement

The techniques that are searched for should help solve two major problems posed by this thesis:

- Visual cues for learning and interactivity.
  The small hints and interactive elements that guides and explains to the user what they are looking at or even where to look. These are best suited to answer the motivational foundation of this thesis, and will constitute most of the research.
- Re-creation of an environment.
  How do you re-create an environment that is suited for an audience who are supposed to navigate and understand a place (and time) they have never seen before? Methods for proper immersion and world design that are not confusing but guides the user, while maintaining an authentic look, are needed.

We need techniques to construct a good world composition. Movies could be interesting to investigate: movie making is not all about the story, but also things like the composition of the shot and the construction of the setting. However, most techniques there would be tailored for a specific angle and interactivity is not taken into consideration at all. Theater scenes work in practically the same way. There are interesting and relevant elements in both of these areas, but most of the time they would not work if used directly in an interactive environment. They are mostly about the setting, not the cues we are also looking for. But why turn to movies and theater in the first place? A simple answer is that they have traditionally been the only places these ideas have been needed; that is not the case any longer.

Computer and video game production involve many of the techniques seen in both movies and theater, but these and many new techniques are built upon and developed specifically to keep the player involved, interested and not least knowing what to do (frustration kills interest and immersiveness, consequently alienating the audience from the game's intended presentation). Indeed, these are all elements that games for many years have had to construct in a well thought-out way.

Relevantly, the coupling of computer game design theory and virtual heritage is not something that is prevalent in completed virtual heritage projects to date. It is perhaps in part due to a notion that game design theory is about how to make the playing rules of a game, however, that is far from accurate. In the last few years in particular, there has been extensive attention towards the use of the interactive style of computer games for the purpose of education and training (often termed *serious games*) including its integration in cultural heritage[24]. Another contributing factor could simply be that the professional fields of cultural heritage and game design are so culturally divided that one part or the other think that they cannot, or should not, be combined. Whatever the reason, it is clear that there's a lot of ground to be covered in the attempt to bring both fields closer.

In game design theory, there are many subtle elements in play, all of which work in symphony to aid the complete experience. These are seemingly the forgotten heroes and are too numerous to mention. The high pertinence of all of these discussed reasons to the research matter leads to the conclusion that the identification and characterization of relevant elements used in games are going to be the search scope of this chapter. As described, they include both environmental design choices and techniques to aid interactivity and information flow.

> *"Rather than prescribing guidelines, it is more productive to supply exploratory questions to provide operational guidance for designers seeking to instantiate game design strategies and devices into various learning activities."*[18]

It should be made clear that the inclusion of traditional game challenges such as specific tasks given to the user or more involved gaming (such as agility or following a set of rules) is not the focus of this research. In fact, it has been argued that the presence of a typical game goal would reduce the likelihood of proper immersion in a virtual heritage environment, and even contribute to the user ignoring the environment itself and the actual content[8]. Observational ease is the first and foremost property that shall be investigated. Therefore, the case test will not be a traditional game with objectives, rewards and punishment; the elements taken from games will be there solely to promote the ease of recognizing and revealing information about the environment itself to the user audience.

4.2 Identification

An oft-cited paper states:

> *"Development of learning tools based on the adventure game could provide educators with a superior mechanism to entice learners into virtual environments where knowledge is acquired thought [sic] intrinsic motivation."*[3]

This is perhaps unintentionally fitting the subject of this thesis, because much literature then proceeds trying to define games with their rules and how to make them engaging[9, 18, 24]. It is probable that many researchers construct a problem for themselves thinking that they have to choose a genre or type of game and translate that more or less directly to virtual heritage. This thesis takes another route; it will directly extract concrete elements of games from any genre and analyze them. Then some will be used, where appropriate, in another context than their original gaming home.

Even so, it is natural to ask what genre of games could contribute the highest number of useful elements in this setting, simply in order to start the search with something that is likely to give results. The user of this particular virtual environment is going to walk around in a first-person perspective, but that does not necessarily mean that first-person games should be the sole or primary inspiration. Games such as adventure games that are played from various implementations of third-person perspectives are indeed very topical. This is because the point of interest here is not so much from which perspective the games are designed for; the interesting things are how the games convey hints and train the user in tasks or the story. Most role-playing games (RPGs) have many of these elements. Perhaps a good pointer of where to search is to analyze *popular* or critically acclaimed games because they are likely to use many successful implementations of these cues. Below is a list of many elements that a variety of games have used, and discussions whether they would fit in the Virtual Heritage setting.

All of the following definitions of categories of cues are suggestions for classification of the various types of elements.

4.2.1 Visual cues

Definition: *Visual* cues are distinguished from the other cues by being anything that only visually augments the whole, or parts, of an already present environment in order to show something to the user. This means that visual cues do not move objects around, change the landscape of the world or anything that could be considered affecting the world physically. Rather, it can be thought of as a post-process effect that is layered over either elements of the world or the entire world.

4.2.1.1 Item shimmer

Description This is the practice of creating a visually distinctive shader effect that can dynamically cover an object in order to communicate to the user that there is something special about that object. The effect is usually made in colors that are not used anywhere else in the environment. Often, it is animated and/or changes depending on which angle it is viewed from, which is usually even better to distinguish the object from the rest of the environment. It can often be described as having a glowing appearance. The effect is only visible when it needs to be, i.e. only when the user is able to interact with it. It should not be possible to distinguish the affected object from other identical objects when the effect is not active.

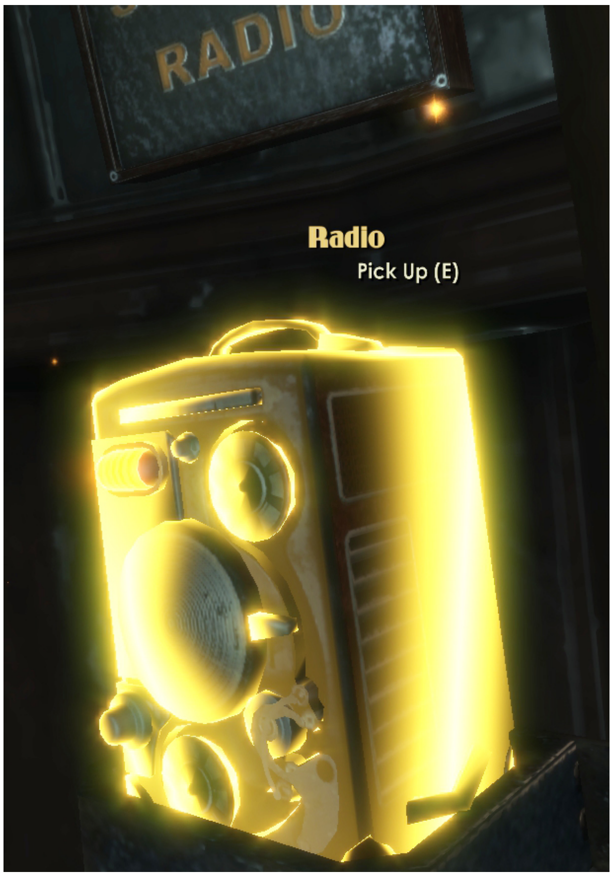


FIGURE 4.1: The current objective of the game is to pick up a radio, so a layered shimmer effect covers the entire object in question. An additional animated particle effect (a subtle version of the Sparkler, also described in this chapter) is also applied in this case. Viewed here from up close, but this effect makes the item stand out from afar together with decorative or non-interactive items. From *BioShock* by Irrational Games[28].

Common applications This is often used in games where a specific objective requires the player to interact (pick up, activate, move or similar) with an object. It can be a major game objective or common pick-up items that should be easy to distinguish from debris and other non-interactive elements.

Suitability for virtual heritage In a virtual heritage environment that does not contain an inherent objective such as the one discussed here, its use could at first glance seem limited. However, there is nothing blocking the use of this effect just to show the user something interesting. The basic reasoning for its use is that there are often many similar or even identical looking objects in an environment (or small and hard-to-see objects that could even be in dark areas) and the effect will make it stand out in contrast from the background. For example, if there are an abundance of tools in a re-created workshop, it would not make sense to enable the user to interact with every single (and often identical) tool. This would only add to confusion, especially if the result of the interaction was the same from every item. However, if the shimmer effect were applied to a few of the unique tools, it would signal to the user that it is possible to interact with it. An interaction can then result in a presentation of additional information or anything that would be appropriate in the situation, e.g. an explanatory video clip. If different and distinct consequences are planned to occur on interaction with objects, the shimmer effect can be color coded. This enables the user to quickly understand which interaction leads to which type of result (e.g. a yellow effect could result in a text box with facts about the object, and a red effect could result in a video clip).

4.2.1.2 Edge glow

Description Every object in a 3D-space can be thought of as being projected onto the screen. This projection can be viewed as a 2D-representation of the object for every frame. This image will thus have outer edges in relation to the camera. The effect uses this idea to create the edge glow, where said imagined edges are highlighted to create a canvas around an object of interest. The highlight itself is usually a line of distinctive style and color.

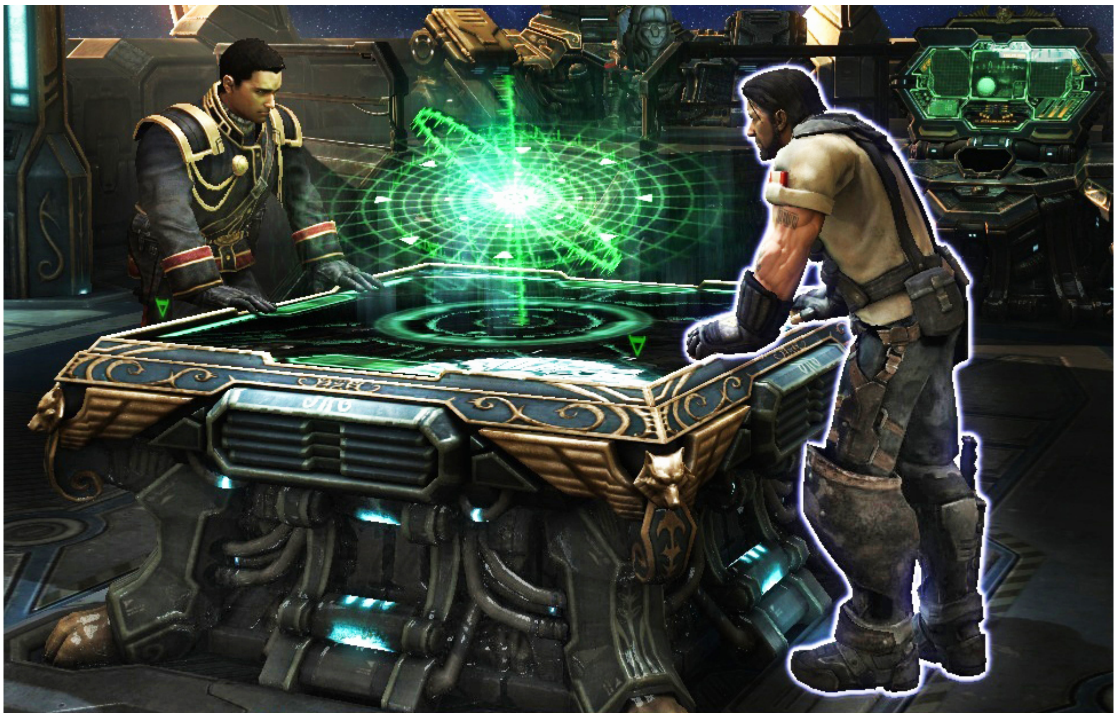


FIGURE 4.2: The character on the right appears to glow around his edges as the user hovers with the mouse pointer over him. From *Starcraft 2: Wings of Liberty* by Blizzard Entertainment[21].

Common applications Common in point-and-click adventure games, the user is presented with a static view of an area with the possibility to swipe the mouse pointer over any part of the view. Occasionally, as the user hovers the mouse pointer over an object it is suddenly highlighted with the edge glow effect. It is excellent to convey the extent of the object regardless of the lighting conditions and clutter of other objects. The technique can also be applied in order to make the user feel like he or she searched for something hidden. Some examples of games that use the technique are *Heroes of Might and Magic 3*[12], *Diablo II*[49] and, as shown in figure 4.2, *Starcraft 2: Wings of Liberty.*

Suitability for virtual heritage Almost exclusively seen together with a static camera view, the effect is best suited for relatively large overviews of a specific area with many interesting objects where the designer wants the original impression of the image to persist. Indeed, one of its strengths is that the object's original colors are always intact and not filled or cluttered as would be the case with the shimmer effect. However, it does require the user to actively search in order to activate it, thus it is not particularly helpful to get an overview if the search area is not clearly defined. For these reasons it is not as well suited as the shimmer effect in a first-person environment where the surroundings change all the time. If the author of the virtual heritage environment constructs a specific overview, e.g. a bird's perspective of a town not controlled by the user, this effect could prove very useful if the user was instructed to mouse over the different buildings. Bottom line: the technique is aesthetically pleasing and non-intrusive, but requires active involvement to be of any use.

4.2.1.3 Rim lighting

Description A shader effect which brighten and separate in-game character's rim diffuse color and specularity in order to make the character seemingly pop out of the environment. Its name can be somewhat deceiving in that the effect does not provide additional illumination of a model, but instead use a highlighting color as an emissive effect. This technique ensures easy and fast recognition and separation from the background elements. Commonly used in film, rim light is often used to separate a person from the background. The technique is usually fairly subtle meaning the audience is often not aware that the effect is being applied, so the object is still perceived as an integral part of the frame[31].

Common applications The fact that the effect is toned down makes it ideal for situations where a more obvious cue is not needed. This can be situations where active interaction is not necessarily required but where the object still should be easily distinguishable. Commonly utilized in first-person shooter games and other fast-paced environments so that players can more easily identify and pick targets. The effect becomes most important at mid-range, because at a distance the effect gets noisy and is normally toned down based on the distance to the camera[31]. At close range, objects should be obvious regardless of any rim highlighting.

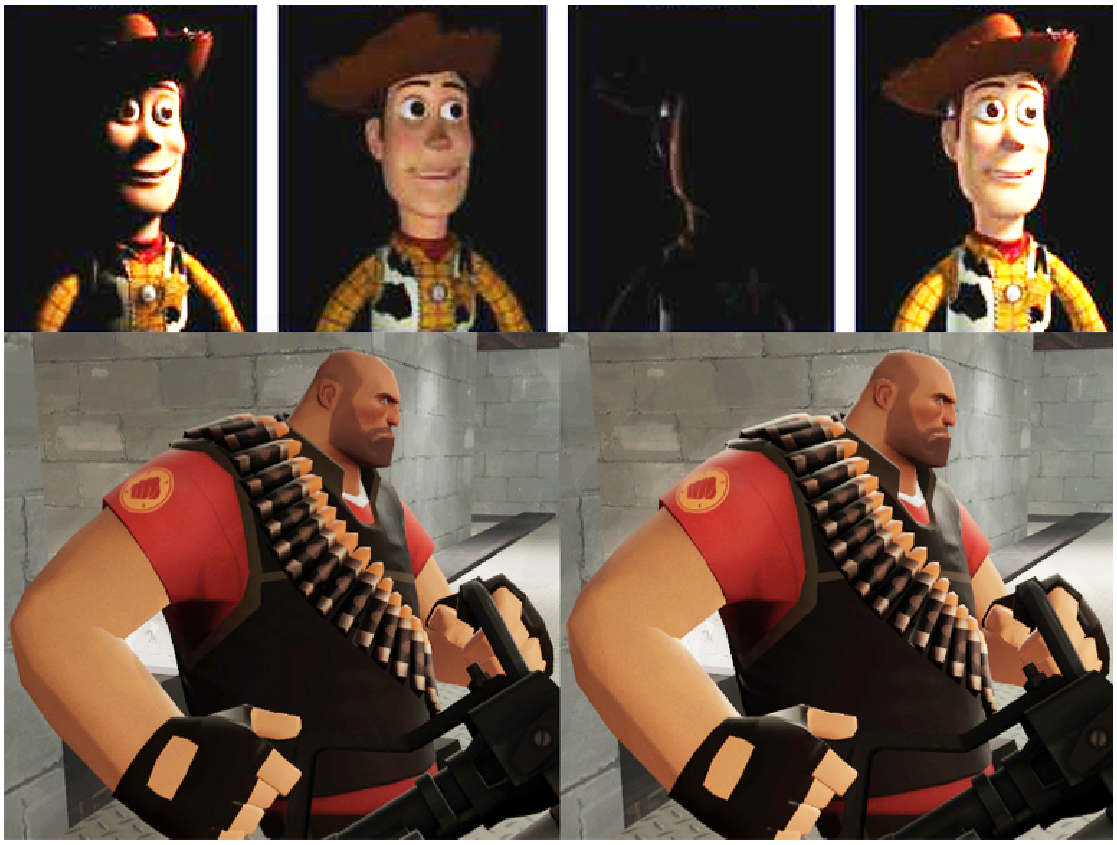


Figure 4.3: *Top from left to right:* Key light, fill light, rim light, all lights. From [64]. *Bottom left:* Game character without rim lighting. *Bottom right:* Game character with rim lighting. From [14].

Suitability for virtual heritage Due to its subtlety, rim lighting could always be activated without users even noticing. Because the effect has been almost exclusively used on characters in games and because other cues such as item shimmer are better for more geometrically shaped objects, rim lighting should also be used for characters in virtual heritage. Unless there is a multi user environment implemented, this applies to non-player characters - i.e. computer controlled characters, interactive or not. The effect will need to be tuned to the environment's lighting conditions, but there should be few reasons to not include this effect except to reach very specific goals. For instance, the creators may wish to visualize a night time environment to illustrate how hard it was to spot other people in the streets. However, most concerns about historical realism are not very applicable in this case, much due to the effect's flexibility to subtly adapt to the background.

4.2.1.4 Sparklers and "bread-crumb" trails

Description This is a particle effect that looks like its real-world firework counterpart from which it has lent its name. It does not modify the appearance of objects at all, but creates an emissive particle stream, or sparks, from or leading to an object or even an area of arbitrary size.

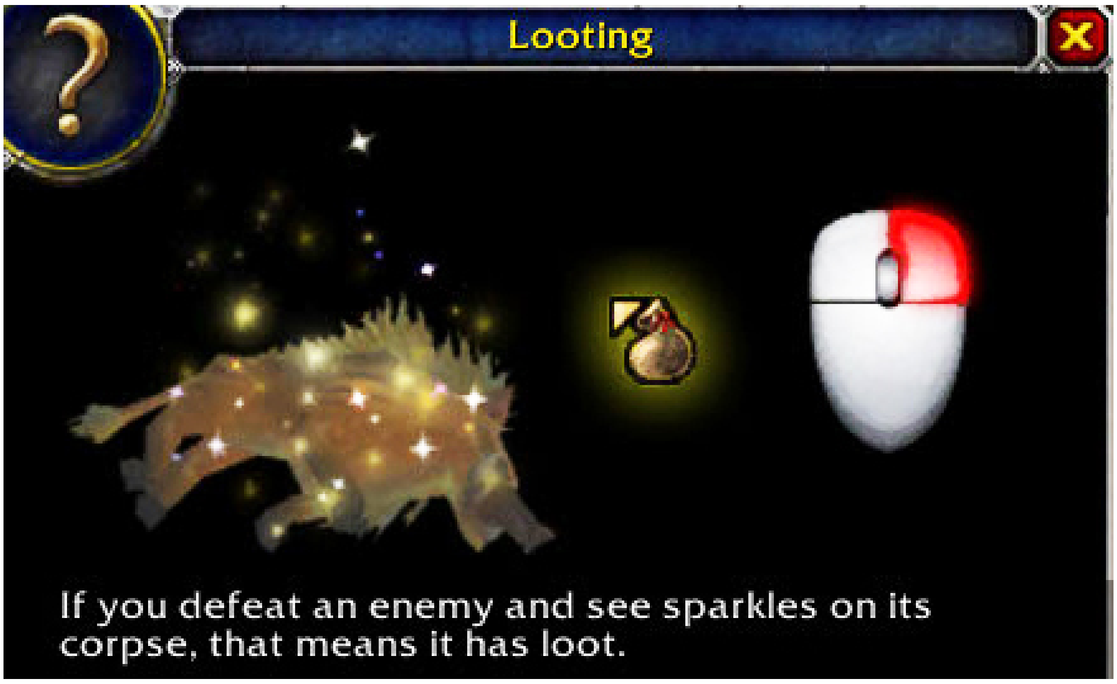

Looting
If you defeat an enemy and see sparkles on its corpse, that means it has loot.

FIGURE 4.4: Sparklers are in this case applied to a defeated enemy, indicating an interaction opportunity which would here result in searching the body for useful items. From the help system of *World of Warcraft* by Blizzard Entertainment[20].

Common applications Because of its independence of any particular object, this method can easily scale accordingly for multiple applications, e.g. to draw attention to a single object or to mark a path based on where the user is in relation to the objective. An area can be circled by this effect, indicating something would happen if the user steps into that area.

Popularized by a few games in the RPG genre, the method is especially suitable for open-ended environments where it can indicate a possibility (but not necessarily a mandatory objective) that the user has an opportunity to do something if the hint is followed. This application is less intrusive than many other techniques and is suited to these games because of its obvious appearance while it still has less immersion-breaking characteristics.

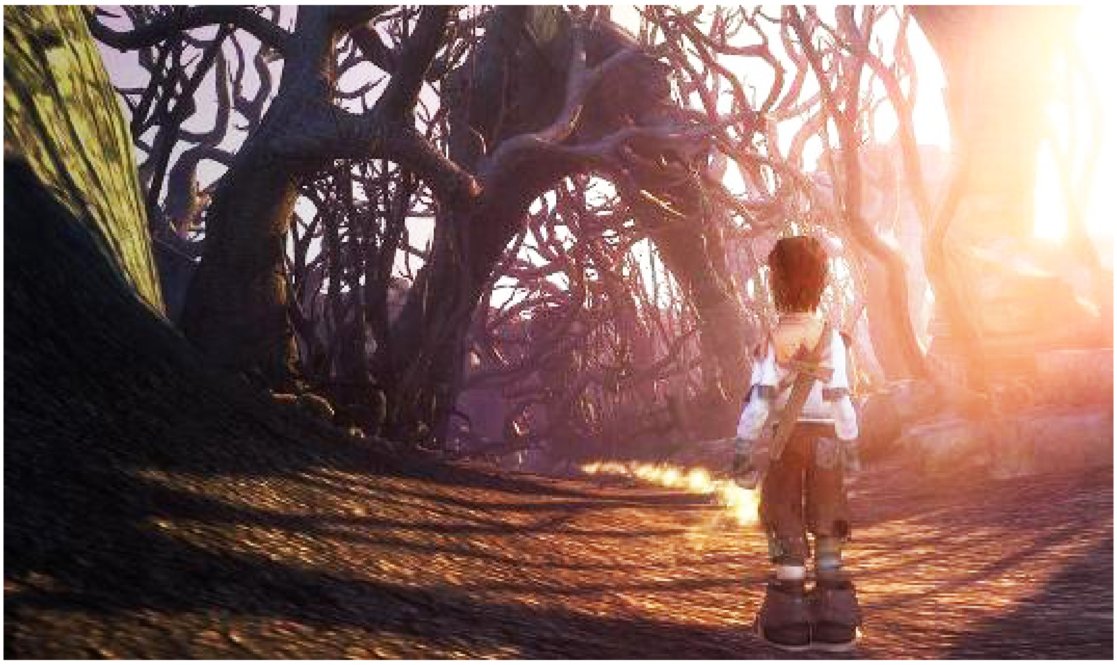


Figure 4.5: The avatar has a path laid out before him in the form of a sparkle effect, specifically dubbed the “bread-crumb trail”[69, 71], which is dynamically generated and indicates the most appropriate route by road from the player to the current main objective of the game. If the player moves in another direction, the game will detect that and update the effect to create a new path. From *Fable II* by Lionhead Studios[63]. Image from [22].

Suitability for virtual heritage As we have seen, there are two major uses for this technique. To not directly affect the looks of an object at all is often a good thing and a better choice than other methods, regardless of which implementation of the technique the designers choose to use. Although usage of the trail, or “bread crumb”, is more intrusive than is the case if the effect is used on a single object, the design considerations should be similar. As with some other visual cues, differing colors can be applied to indicate various types of interaction consequences.

The dynamic nature of the effect facilitates even less development effort than with other effects, because existing models or shaders does not have to be modified. Rather, a minor addition in programming could add the opportunity to place this effect in whichever area deemed appropriate. Due to this, the method is well suited for constantly developing projects, which are prevalent in heritage environments where new discoveries are made even though the re-creation effort has started.

Creators should be careful to ensure that it is perceived as a visual cue and not as an effect of the environment. This is a danger when the effect is as disconnected to any specific object as this effect is. In practice, this means that the implementation should look distinctive enough both in color and animation to not be confused with natural environmental effects, e.g. fire or actual sparks.

The trail, intended as a way to tell a straying game player to get back on track[71], can distract and deceive the user into thinking that the trail is mandatory to follow. Such mandatory requirements are not present in most heritage environments. The audience should be assumed to be inexperienced in interpreting cues and not able to determine whether it is a mandatory objective. For these users it is best used in rare cases where there is something especially important to show the user that can not wait, in an area that consists of very linear movement, or if a user has actively requested directions to an area.

4.2.1.5 Filtered vision

Description A variety of implementations related by the property that they change the rendering of the environment itself, often completely. It is still a visual effect only and does not actually alter the environment. A real-world analogy which has also been used in many games is night-vision goggles or heat-sensitive cameras. Elements that are not important are often filtered and rendered in such a way that they all blend together in a single color, so that the important elements are very easy to distinguish. The user could be described as entering a simulated state of clear-sightedness.

Common applications Except for the use of the real-world equipment mentioned, effects such as this that are utilized to identify objectives or the like are not very common to see in games. A notable example of its usage is the “Eagle Vision” effect in *Assassin’s Creed II*[46]. When the effect is toggled by the player, the otherwise colorful world is toned to a monotonous gray while only important people and objects (even invisible writings on a few walls) are shaded in bright blue, red or gold colors. The user interface is also hidden while in this mode. In a city with crowds consisting of hundreds of people on busy streets, it allows the player to immediately identify all important elements. The player is still free to walk around and explore while in this mode.

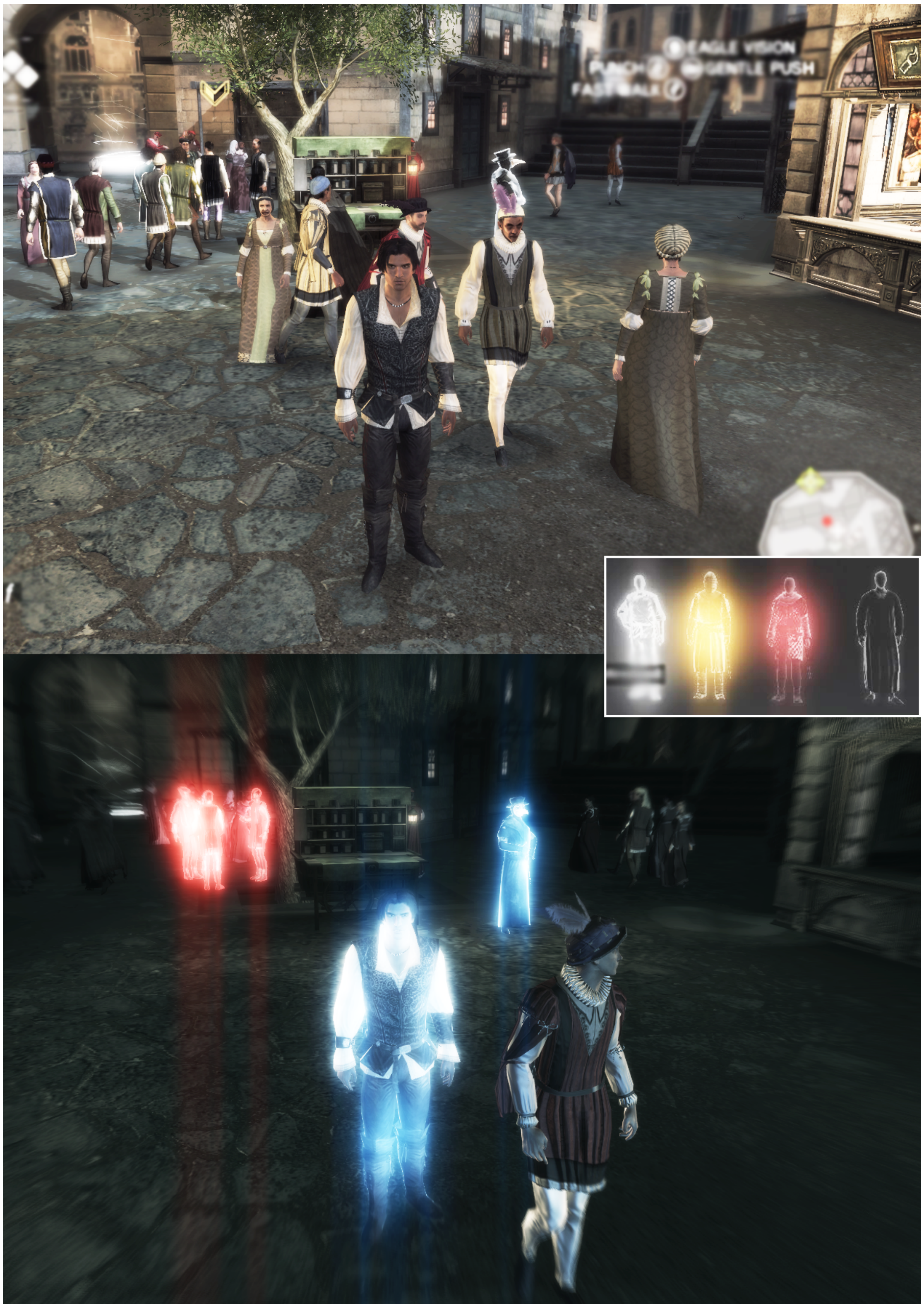


FIGURE 4.6: The “Eagle Vision” effect. *Top:* A normal view from the game; this is how things look like the majority of the time. *Bottom:* The effect is applied, coloring allies blue and hostiles red. The rest of the frame is rendered highly desaturated. *Middle Frame:* Examples of potential colors indicating various roles. This can be applied to items as well as people. From *Assassin's Creed II* by Ubisoft Montreal[46].

Suitability for virtual heritage While the advantage of complete clarity is obvious, the effect's impact on immersion is very dramatic: the user is completely removed from the known environment and thrown into something that could resemble an alternate dimension. The method was made for help in confusing environments, such as the one mentioned above where there are crowds of people present to confuse the user. Heritage settings are normally calm and even if there are many characters on screen, there is not any particular threat or time constraint present that would encourage a need for such a fast identification. The effect could be applied to items or anything else as well, but it should be considered whether such a dramatic effect is needed for this purpose. The fact that it has such a large impact on the presentation would likely enable the audience to understand that they are no longer looking at the actual reconstruction: it should be evident that this is a cue mode. However, the immersion is still broken and there is also a concern that users will continue to keep the effect enabled while walking around, which would result in complete disregard of the environment.

4.2.2 Environmental cues

Definition: *Environmental* cues comprise all the consideration that has gone into where to place objects in the world and how the world design itself passively communicates to the user opportunities in spacial navigation. A fundamental example is obstacles positioned in such a way that the user will understand that some area is uninteresting, prompting the user to the other way instead.

4.2.2.1 Fencing and gates

Concept and execution Fencing is a metaphorical label and comprises the idea of keeping the user inside a predetermined area in the virtual environment and the techniques involved to make the user understand which areas are passable (preferable before hitting an invisible wall). Most environments are not infinite, or the designer will in most cases want to confine the user inside the area of interest. However, an invisible wall that the user bumps into seemingly at random is very poor design and leaves the user very frustrated. Inconsistency in fencing is another potential design flaw. For example, there should not be a door that lets the user through and an identical door that does not (i.e. is only a static property).

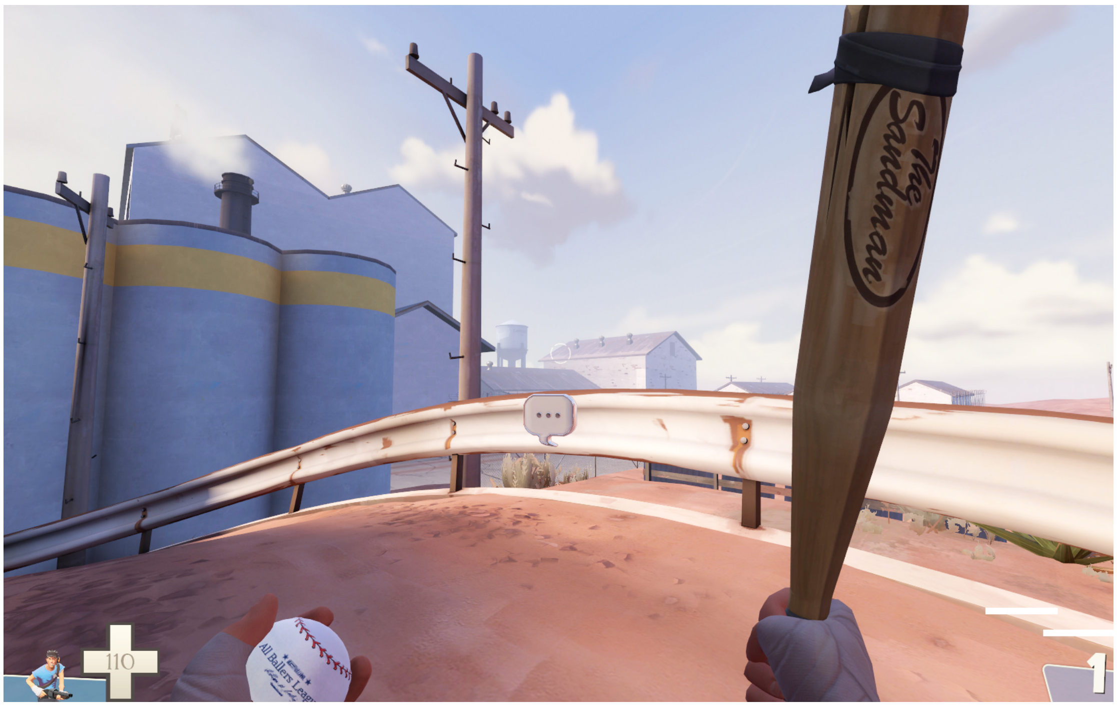


Figure 4.7: Fencing in its literal form. There are only bland, uninteresting elements behind the obstruction. The design visually leads the player into a more interesting direction. In a less stylized environment, this lack of excuses to block the area behind the obstruction could be considered a rather poor solution. From *Team Fortress 2* by Valve Corporation[14].

This is valid for anything that is used to block the user. Terrain should have a consistent angle of incline that is impassable, or a certain denseness of trees in a forest could indicate the same. Invisible walls can in some cases be forgiven, such as when blocking a wasteland or an unswimmable ocean. However, this needs to be clearly marked by different texturing and obvious visual clues to the user as to where he or she should not go. Neither should there be anything interesting on the other side of the fence, unless it is obviously unattainable or non-interactive. Dead-ends should be avoided; the fencing should only be placed so that the user will not need to backtrack significant distances[23].

Suitability for virtual heritage A virtual heritage environment of the type discussed here would be free-roam area where the user him- or herself are able to walk around without many barriers. Fencing should for the most part be used in the outer regions of the area of interest just to keep the user from getting lost in a terrain with no buildings or otherwise valuable content.

There are a few exceptions to this. Invisible collision walls should absolutely be used to avoid getting the user stuck on small nooks and crannies in the environment; the user should be able to move around easily[23]. There could also be specific areas where the designer would want the user to experience a linear scripted event. In that case temporary obstructions should be added to make sure that the player experiences the event properly or does not walk away from the event entirely. This is called a *gate*: a forced constriction of freedom of movement to assert that the user experiences something.

Many level designers go out of their way to shape the environment with the goal of avoiding obvious fences and dead-ends, particularly in fast-paced games. In a slow-paced exploratory environment such as virtual heritage, this rule should only be followed in the most pressing cases or in cases where the area is designed by guesswork, mostly because historical authenticity should have the highest priority.

4.2.2.2 Landmarks

Concept and execution The user needs many ways to facilitate navigation of the environment. The virtual (re)construction of an area can tend to include many similar-looking locations, and it often becomes easy to get lost and lose track of where you are. The simple addition of a few landmarks into the virtual world can greatly help alleviate this problem.

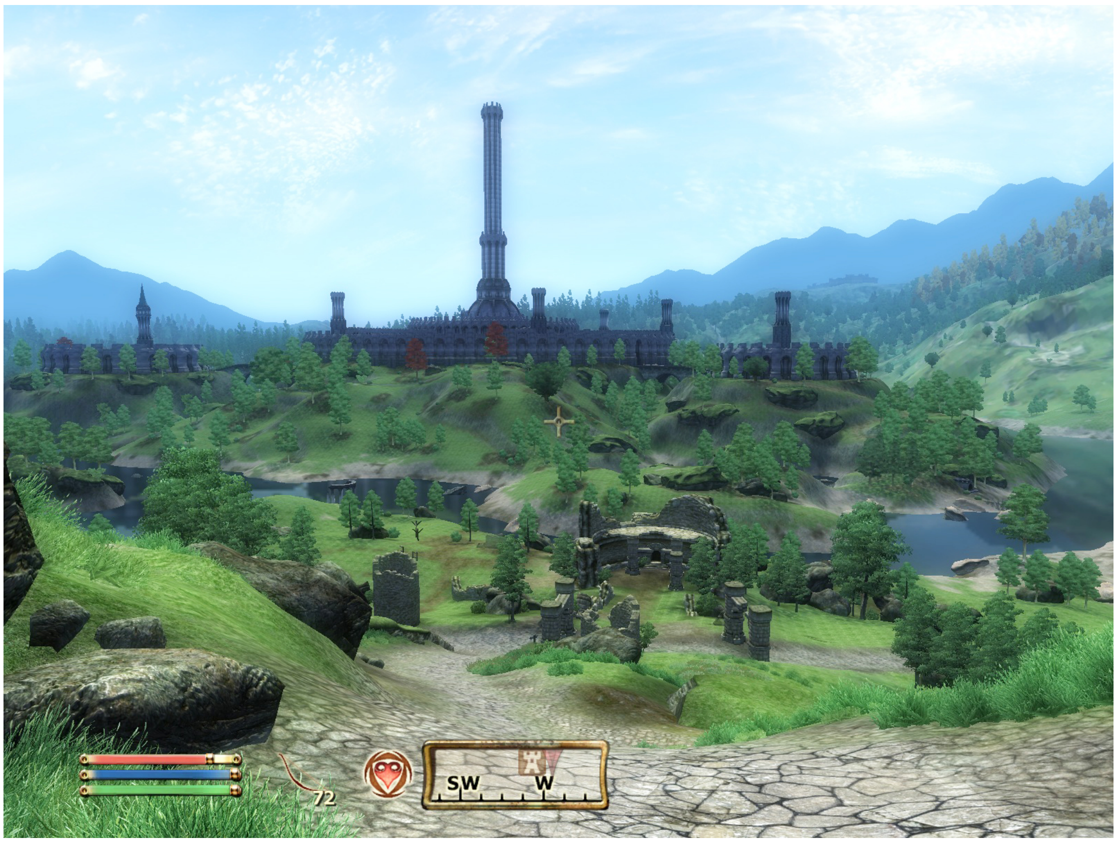


Figure 4.8: The depicted environment is huge with the forest often lacking in contrast, but a tall tower in a location which is introduced from the start of the game helps the player orient him- or herself from almost anywhere within the game. From *The Elder Scrolls IV: Oblivion* by Bethesda Game Studios[61].

They can help people remember locations and grounds their spatial awareness. They can be of any size or form - the key is that they stand out in some way[23]. In large environments you could encounter situations where the general scale of the world itself could be ambiguous. Here, landmarks can help users compare the rest of the world to a known scale and enable them to create a better virtual travel plan for themselves[43].

Suitability for virtual heritage Large landmarks should only be used in a heritage environment if it is highly necessary and the navigation problems can't be solved in another way. The reasoning for this is that virtual heritage environments are reconstructions of real locations, and the addition of landmarks could greatly impact the historical correctness of the environment. Even if the landmarks are plausible additions to the environment and are added in a non-intrusive way, the immersive experience of the user will be at risk. A considerable challenge in the first place is the time and effort involved in finding and modeling landmarks that does fit the environment appropriately. Due to this complexity, this would first and foremost be natural elements that nobody could argue whether or not existed. These could be elements such as a great tree, a special rock formation or even irregularities on the architecture itself such as (historically correct) graffiti on a wall.

Instead of attempts to add additional objects which have no solid historical basis of being included (or things that are not present in the original plan), an alternative is the addition of completely synthetic landmarks. Users would arguably react better towards an object that is obviously not part of the world (much in a similar fashion as the artificial visual cues discussed in this chapter) as opposed to elements that could induce doubt as to whether or not they are part of the intended historical representation. In a virtual heritage setting, this indeed seems like the better solution. However, it is as before equally important to not over-use them. As with other visual cues the designer should provide a method of disabling them temporarily (as discussed in chapter 4.3).

Ultimately, it is a design question where every situation has to be looked upon individually to make sure that the final decision fits the situation of the system and environment in question. In historically less rigorous situations, looks and immersion can be favored more than is otherwise recommended because of historical constrictions.

4.2.2.3 Composition and lighting perception

Concept and execution

> *"The composition in an area should shape the area in order to show the player the path forward."*[43]

Composition is quite a generic term, but it can be separated from other more specific cues simply by that generic property. The composition of an environment should guide the user's eyes toward important paths, objects and non-player characters. As opposed to cinematography, the creator does not have control over the camera angle or position from which the user will see the world through; the designer has to be concerned about the entire game space[31]. So unless required by design, the risk that the user will miss those important elements should be avoided[43].

Lighting and architectural composition go hand-in-hand and should always complement each other. For instance, animated geometry will always be good at drawing the user's attention. This could be a windmill in the distance, a banner blowing in the wind or any other believable element. At the same time, the notion of animated objects can also include animated lights[43], such as the pronounced flickering of a candle emanating out from a window. Think of moths being drawn to a light bulb at night - the same principle is valid for humans who are lost or searching for a path: they will probably elect to move towards the light. It is important to implement a sense of direction and continuity, as well as adding reference points, with lights[31]. When an exit is hidden in darkness, but a nearby unimportant area grabs the player's attention with bright lighting, the user is likely to miss that exit. Likewise, if the entire area is interesting there should be no large dark gaps left that will unbalance the composition (this is true for the coloring of the light as well)[43]. Thus, in addition to the traditional application of light as a mood-changer in more dramatized environments, lighting can reduce general frustration by making objects or areas easier to spot and will have great implication on usability and user engagement[19].

If possible, the designers of the environment should also try to incorporate a direction of composition. If you want the user to move in a certain direction, the scene should contain perceived horizontal lines that gather in that direction. Comparatively, if the user should look up or down, the composition should contain vertical lines in the appropriate direction. These lines can consist of anything, indeed they can and should be a part of the environment itself.

Suitability for virtual heritage There is a problem with implementing many architectural composition guidelines in a heritage setting. The environment is, for the most part, already planned and there is little freedom to change that. It will depend on how accurate the depiction has to be.

In cases where the designer does not have a lot of freedom regarding the look of the environment itself, another more dependable route is to concentrate the development efforts more towards the lighting composition. It is more situationally flexible and can even be used in environments that already have a high degree of pre-planning behind them. The lighting composition simply offers more freedom to the developer. However, in a virtual heritage setting there are some additional concerns that need to be addressed before lights are implemented.

Currently, many virtual heritage environments show the world with completely arbitrary lighting. Often, this results in lighting conditions as if the environment was lit by a bright and steady electrical source - not how it might have looked like at the time in question[17]. The designer should always take into consideration the color given off by the burning fuel of a torch or candle, also when considering using light as a tool to guide the user as is the case of some learning cues - this means that the added lighting due to learning purposes should be in style with the original lighting. These light sources could have, especially in the area examined in this thesis, been made from animal fat or tallow. In terms of an interactive system relating to this thesis' purposes, these effects are not necessary to recreate in a rigorous manner. However, they should be approximated as best as possible to ensure an authentic look. For instance, environments in the medieval period, both inside and outside, were far from as brightly lit as they are today. Smaller windows and often candles as the only light sources were common, resulting in dim interiors[17]. A common design flaw in any virtual environment is to create light without an actual visualization of the light source, so when creating the lighting composition care needs to be taken to provide a believable light source as opposed to having the light appear to emanate from nothing.

4.2.2.4 Audio cues

Description If skillfully applied, audio can both accompany other visual and interface cues as well as being used entirely on their own as clues to the user. Short confirmatory sounds which communicates, in addition to the visual feedback, that an interaction request from the user has been received are typical examples. These include small clicks, fanfares or single drums - anything that can lessen the strain on the user's eyes by utilizing another sensory channel such as hearing[39]. Alternatively, ambient sounds integral to the environment can be balanced and adjusted in relation to each other where appropriate to attract the user's attention - or be disabled entirely for a short duration to accentuate a single sound[43]. Music is also part of this consideration - e.g. small pieces of music may be played when an area of special significance is entered to communicate something to the user.

Common applications Audible feedback from interface interaction is almost universally used in all types and genres of games. When a player picks up something or receives a new objective, there is normally an accompanying audible cue to make sure that the player becomes aware of this. Dominant sounds that originate somewhere in the environment itself are essentially used as bait to draw the user to any specific area.

Suitability for virtual heritage The first thing that the creators need to take into consideration is whether the environment should use sound at all - both for cues and for other purposes. Using appropriate sound for the environment tremendously augments the immersive experience[23]. However, depending on the real-world circumstances of the presentation, it can be difficult to properly use sound to any effect. This holds especially true if the sounds are supposed to be perceived as coming from a particular direction in the virtual world. For example, if the presentation is running in a kiosk along several other identical or different presentations it would only serve to make the experience more distracting. Consider the number of people who are watching the environment at a time (even if there is only one person who controls it). Often, the best case is when the environment is designed for and used by one person at a time. If the presentation is on a personal computer, the designer has more incentive to spend time implementing proper audible cues although there should not be any situations depending entirely on audible cues. Some users may not have the proper equipment, or suffer from impaired hearing.

Ambient audio from the environment (i.e. non-cue sounds) should never be overused or be too present in the overall soundtrack. This is not only about loudness - ambient sounds need to be consistent but not repetitive[23]. Even without cue sounds, this will become an annoyance to the user after a very short time. Some variety can be achieved without too much development cost by using subtle tricks such as using the same sound sample in multiple locations but changing the pitch of the sound for each instance[43]. When cues are involved it is important that users are able to clearly distinguish them from ambient sounds, so they must be designed by the principle that people unconsciously prioritize unexpected sounds[23]. For example, if the environment wants to showcase an aqueduct, a distinct dripping sound from a nearby pillar is a good cue sound which will attract attention - while the constant flow of water would be the subtle ambient sound.

This includes music, which if not run all the time as ambient music is always very noticeable and can instill hints of revelation (or danger if such a thing is included).

In a heritage environment, period-appropriate sounds should be used even for interface cues. This means to avoid any use of very artificial or synthetic sounds that would completely break the illusion of an ancient world. Every little bit of consideration such as this helps the immersion and overall quality of the presentation; it is more easily noticed than missed.

4.2.2.5 Guide character

Description A non-player character placed in the environment, which optionally follows the user around. The guide can explain what the user is seeing, or could be asked questions from the user. A guide may provide simple directions or a more intricate dialog system.

Common applications The Virtual Notre Dame project featured a guide, as seen in chapter 2. One thesis have argued in favor of virtual guides, stating that a virtual guide *"provides a potential source of essential cultural information to the user immersed in it."*[56]. Another has explored the importance of behavior generation and animation quality[59] in guide characters. In games, the implementation depends largely on the genre. In games with a linear story progression, it is usually implemented as a side-kick character that will comment on the environments. They will sometimes slip hints and participate in the story or even help the player in situations that require cooperation. In RPGs, there are common central hubs and characters which serve to point the player in the right direction. In the MMORPG *World of Warcraft*[20], guards serve as guides. The player can ask them a pre-set selection of questions which the guard will answer with a description as well as mark the mini-map with a special symbol at the location that was asked for.

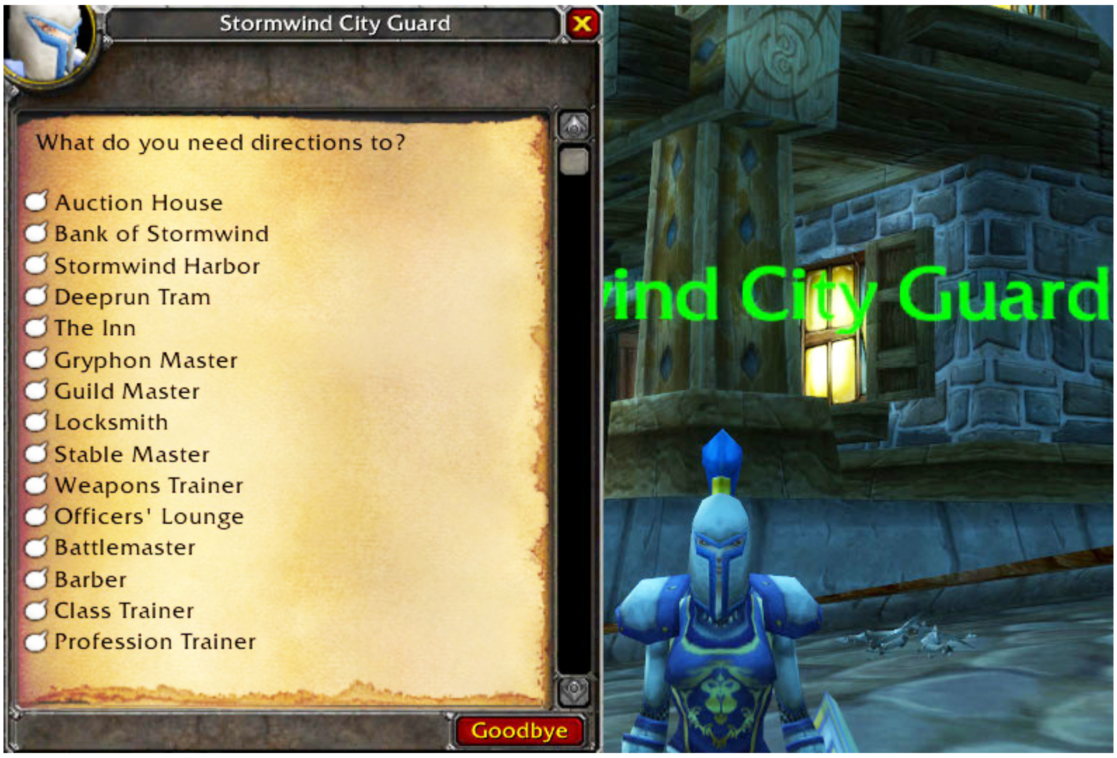


Figure 4.9: This selection of pre-made questions appears by interacting with a guard in one of the hub areas in the game. The guide will provide directions and mark the location on the player's mini-map. From *World of Warcraft* by Blizzard Entertainment[20].

Suitability for virtual heritage It has previously been suggested to use an agent-actor dialog mechanism, where users ask questions themselves to the guide, as a way to enhance virtual heritage environments[8]. If the creators are willing to *"sacrifice some historical accuracy for accessibility"*, a guide character can be used[2]. The character can explain things in another way than information panels with text are able to. The user would feel more involved - but there is a careful balance to maintain to make sure that the feature does not resemble a game or a game's mechanics too much.

Because the user is likely to not know anything about the environment that is explored, it can be difficult to ask the right questions. Even if the task is only to select a question from a set of pre-made ones, the user does not have anything to base his or her decision on. In an environment where users should be presented information without having to do uneducated guesswork, this would arguably resemble a simple gimmick with little actual value. The guide should therefore be talkative, acting on contextual information such as current location without the need for users to ask questions. The best case for user-induced guiding is in cases where users know what they are looking for and only want to know where it is. This should largely result in directions to the important landmarks; the in-depth information should be presented at the locations themselves.

4.2.3 Interface cues

Definition: *Interface* cues are all of the elements that do not use objects included directly in the world, or events that remove the user entirely from controlling the avatar. It should be noted that objects that are placed as a physical part of the world in order to trigger a part of the interface are only bridges to the interface, not part of the interface itself.

4.2.3.1 Maps and mini-maps

Description This could be an overlay of a drawn or projected top-down view map of the area layered over the entire screen space. A mini-map is a variation in which a small portion of the map is displayed. Typically, it is located in a corner of the screen. The map display updates according to the user's position, staying centered over that position. The area displayed is a chosen (and often modifiable) radius around the user's position. Both maps and mini-maps can be annotated with arrows pointing to interesting places nearby the user's current position or icons connected to specific areas.

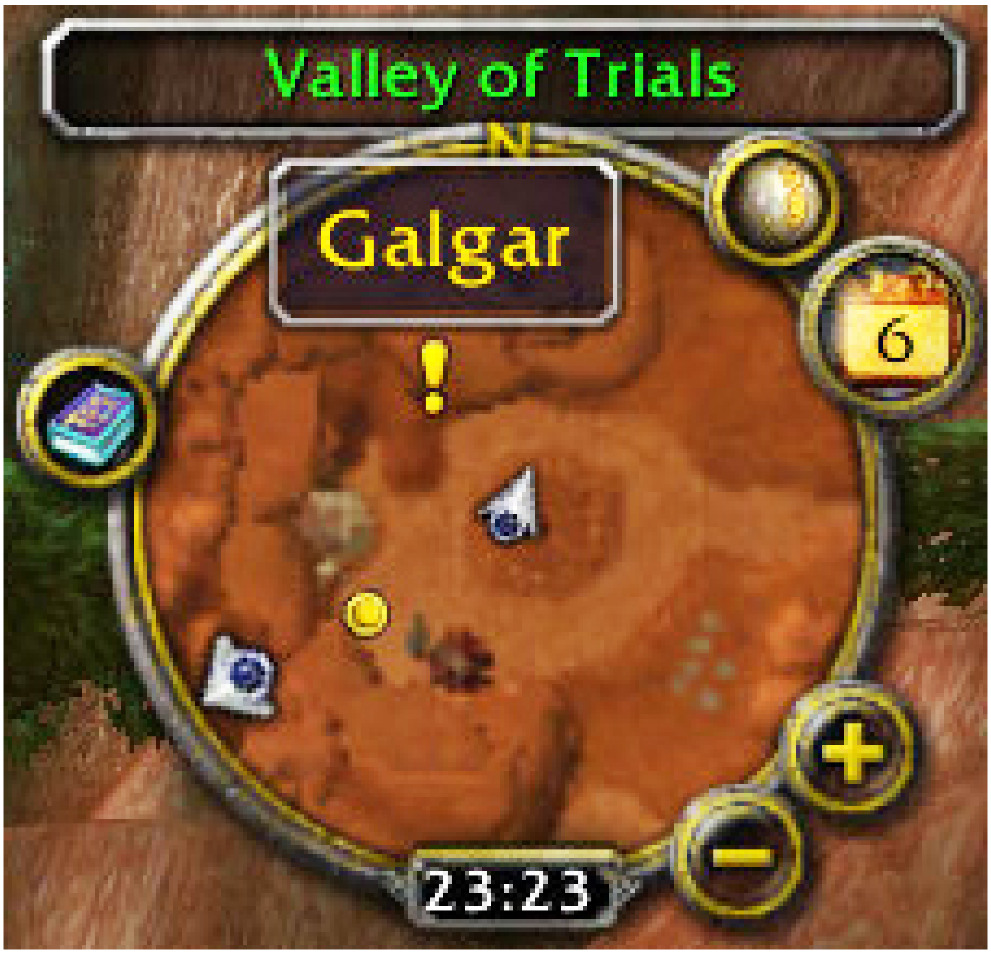


Figure 4.10: An implementation of a mini-map with an orthographic top-down view. The map only uses a fraction of the screen space. The player's current location and orientation can be seen together with nearby interesting elements. If the user positions the mouse pointer over those elements, short descriptions about them are displayed in the tooltip. From *World of Warcraft* by Blizzard Entertainment[20].

Common applications The concept of using maps may seem obvious. However, in games there are many implementations which make maps more effective and relevant to use. If only a full-screen implementation of a map is used, a common nuisance to the user is the need to frequently activate that map to check something. This essentially removes the user from being able to experience the actual environment in a comprehensive manner. With a mini-map, the user can glance at the side of his or her screen in order to confirm location or path without stopping. Maps are very common in slower-paced games featuring large environments such as RPGs and their multi-player variants.

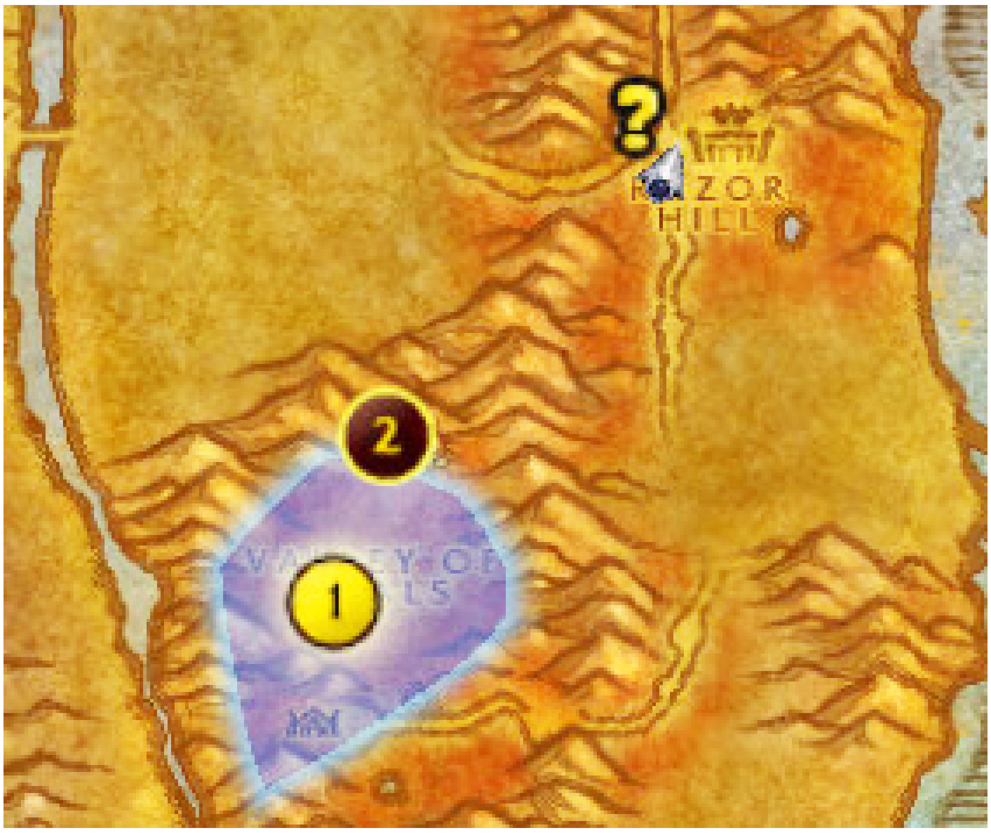


FIGURE 4.11: Portion of a full-screen overview map. Important objectives are indicated, along with the player's position. A single objective can be selected to bring up a hint (blue-shaded area) of where to search for the items required to complete it. From *World of Warcraft* by Blizzard Entertainment[20].

Suitability for virtual heritage Traditionally used with game objectives in mind, maps in some form have seen some attention in virtual heritage, e.g. as seen in [26] described in chapter 2. It has been suggested to use a *"memento map"* that completes itself with additional icons etc. as the user discovers more of the environment[8]. For the purpose of a less goal-oriented environment, the icons and annotations should not be hidden in this way but be present at all times because there is no longer a reason to hide them. In a static world, unless otherwise obvious by implementation, the map should stay the same no matter what the user does in the environment.

Mini-maps are less common. A question that needs to be asked when implementing a mini-map is if the mini-map and the full-screen map should depict the same information and be the same type of map. The default choice should be "yes", where the larger map is simply that - larger - so that it is easier to get a comprehensive overview. The most important thing in an un-linear environment, such as a free walk-through of a heritage location, is to avoid user frustration of getting lost[23]. In general, there is no need for maps if the area is below a certain size or is very open so that it is easy to get an overview simply by looking around. A map in such an area would only clutter the interface and is unnecessary.

The icons and annotations' tool-tips should not consist of more than a short sentence or a title. The map is not meant to be used as a source of information, but only as a tool to point to that information. The design should encourage users to view the exhibitions in first-person which, after all, is the reason for creating the environment.

4.2.3.2 Certifications or checkpoints

Description This is a visual confirmation (or even point system) that confirms to the user what/where he or she has done/been. It can often involve special objectives. It may be a simple one time only confirmation, or a record that is stored which the user can look back into. There are small, specific requirements to obtain them.

Common applications In games such as racing games it is normal to include checkpoints that the player needs to go through, often in order, so that the game can know if the player has completed a circuit or similar objective.

More interesting are *achievements*, which can be earned by players. These often reward points, which may or may not be exchanged for virtual items or other tangible benefits. Moreover, they serve as a checklist for players who wish to make sure they have experienced all of the challenges or areas the game has to offer. Achievements can often be compared to other player's achievements, as an at-a-glance method of measuring who has progressed further than the other.

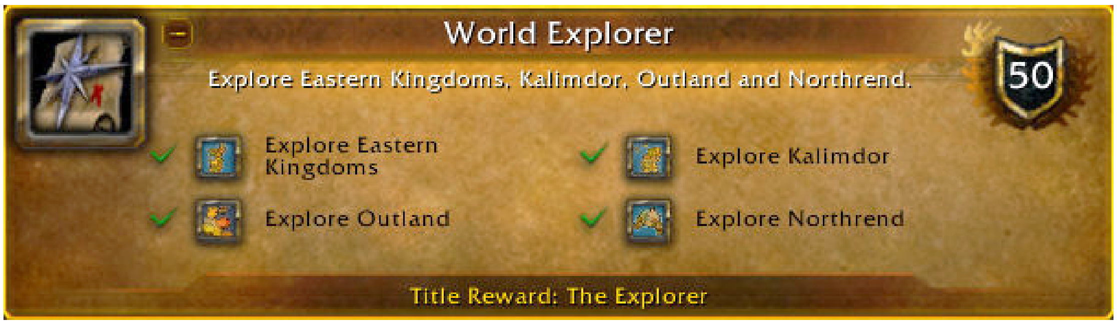


FIGURE 4.12: Example of an achievement. Here, progress from dozens of other sub-achievements involving exploration of the world are tracked, which adds up to this final confirmation of complete world exploration. From *World of Warcraft* by Blizzard Entertainment[20].

Suitability for virtual heritage In addition to a mere checklist, achievements may be used as incentives and encouragement to explore the world. Although this borders to the realm of actual game mechanics, they pose no direct challenge to users and can solely be regarded as an exploration aid. Regardless of whether the user earns points, it is a great reference list for users to gain an impression of the amount of content that can be experienced in the virtual world.

4.2.3.3 Controlling narrative and the camera

Description The program or game may take over control of the camera in order to show something to the user, such as an important event or an introduction to a new area. This may be implemented as controlling and moving the avatar itself, or by complete detachment of the camera. For narrative purposes, an omni-present voice (optionally subtitled) can explain and describe the surroundings as the user enters a new area or when an event happens.

Common applications Games often take pride in avoiding both a narrative voice and taking control from the player. It is believed that this severely affects the immersive factor and is unfair to players who wish to explore and act as they would have themselves[1]. Some games take pride in never detaching the camera or taking avatar control from the player, e.g. *Half-Life* by Valve Corporation[13]. There, the control always remains in the player's hands even during important events, and the player's avatar itself never speaks.

A common substitute for narration is non-player characters communicating directly to the player which is depicted as using radio communiqués or similar devices.

Suitability for virtual heritage It is in many cases not necessary to take control of the camera from the user. In some cases, more subtle tricks such as vignette effects or slight render alterations can be used[31]. For example, when the user looks at something or enters a specific area, creating a border around the screen edges while changing the brightness in the center may indicate a change in focus and is a hint to player to be extra aware of what he or she is looking at in that moment. It would be ideal to provide any special camera or narration as another mode or in an entirely separate (from the interactive) portion of the program to avoid those moments that break the immersion. With a separate presentation, the narrative could arguably be done better because it is allowed to act on its own terms.

If an omni-present voice is going to be narrating an area, it is an advantage if the user is introduced to this from the beginning, although this is not always convenient if the environment is presented at a public stand where people will start in the same state as the previous person left it.

4.2.3.4 Compass and arrows

Description These non-interactive interface elements will provide directional awareness at all times. A compass on-screen can provide more information than simple direction, i.e. it may display elements of importance near and far. Various implementations of arrows which always point to a single target also belong here.

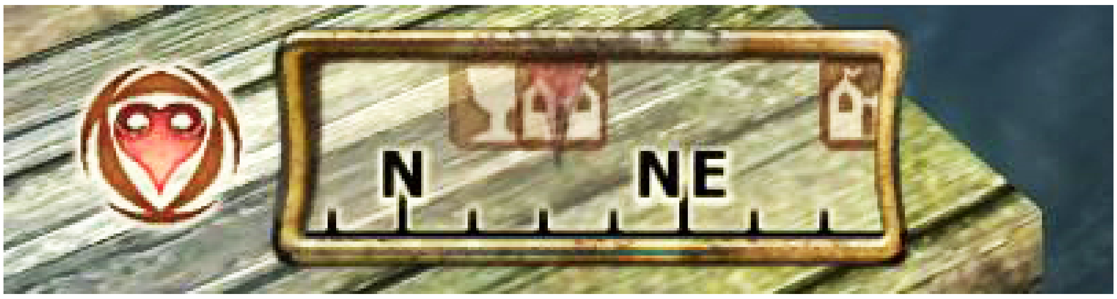


Figure 4.13: Compass which detects nearby important elements and provides a general direction to those as well as the current main objective. It updates based on the direction the player faces. From *The Elder Scrolls IV: Oblivion* by Bethesda Game Studios[61].

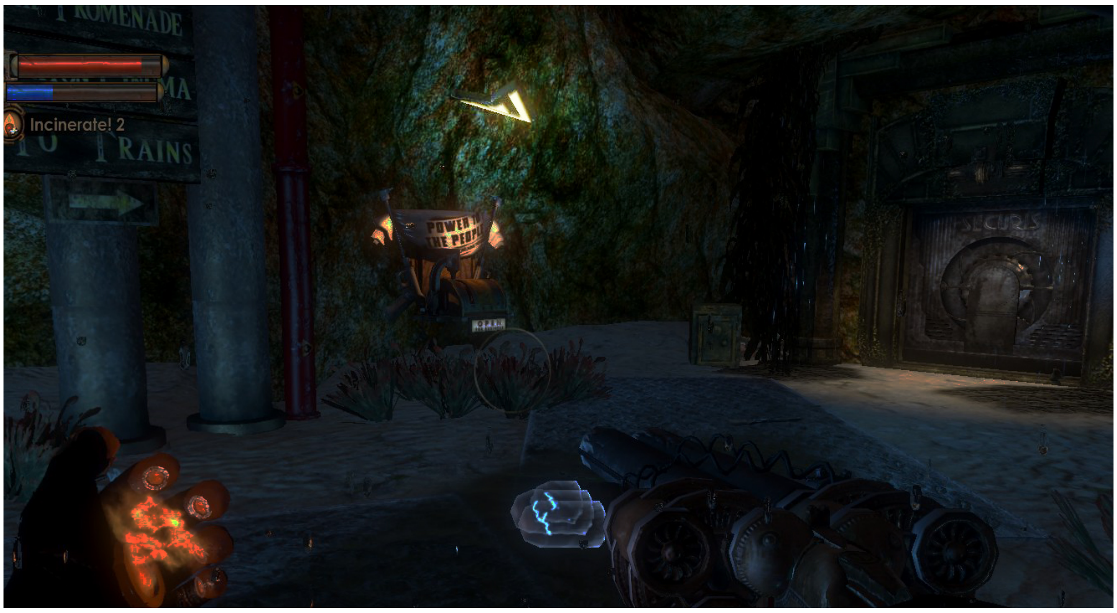


Figure 4.14: The 3D quest arrow (brightened) rotates, always pointing in the direction of the objective relative to the player. From *BioShock 2* by 2K Games[27].

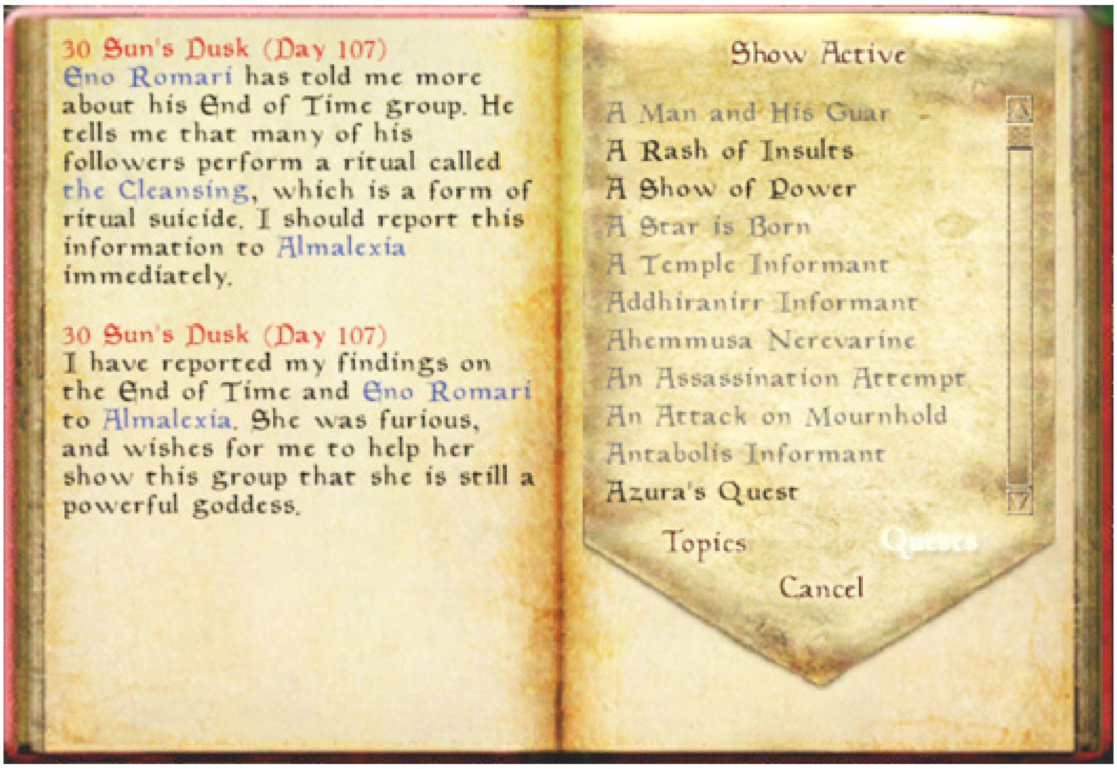
Suitability for virtual heritage A compass is a small addition to a program but results in a large gain in user-friendliness. It takes a tiny amount of space, leaving a small footprint in terms of remaining screen real-estate. If inspiration is taken from the example mentioned in figure 4.13, the compass could replace other interface elements which occupy more screen space. The user will need to be educated about its features if the compass is to feature elements other than directional awareness. Many people have a preference to discover things on their own[22, 71], and a compass leads to a more exploratory feeling by only giving the general direction as opposed to the path or exact location right away.

The 3D arrow can only point to one target at a time. As a consequence, in an environment such as this where there is no main objective, the arrow should only be present if it is specifically desired and requested by the user through other mechanisms. The arrow is therefore an on-demand feature and should not appear without context, as this may leave the user confused as to whether it is a mandatory suggestion.

4.2.3.5 Journal

Description A journal which can be displayed to the user by pressing a designated button (on-screen or otherwise). This may be implemented as an encyclopedia of information about the environment, or it can be made to closer resemble a log of events that have happened or as descriptions of places that have been visited. In either case, it acts as a reference book with all facts (or facts only discovered thus far while playing) that are relevant.

Common applications Most games that features multiple objectives, or quests, implement this. The quest logs may be updated dynamically based on the player's progress. Some adventure games have diaries which will sum up the events of the game up until the present. This may be written as a detailed recount and in a less matter-of-fact way than in games that are less tuned towards the experience of a story.



30 Sun's Dusk (Day 107)
Eno Romari has told me more about his End of Time group. He tells me that many of his followers perform a ritual called the Cleansing, which is a form of ritual suicide. I should report this information to Almalexia immediately.

30 Sun's Dusk (Day 107)
I have reported my findings on the End of Time and Eno Romari to Almalexia. She was furious, and wishes for me to help her show this group that she is still a powerful goddess.

FIGURE 4.15: A journal in the form of a dynamic personal diary of in-game events which updates whenever a significant event has occurred. From *The Elder Scrolls III: Morrowind* by Bethesda Game Studios[60].

Suitability for virtual heritage As opposed to most games' implementation of this, the virtual heritage variant would retain all information of an object or objective even after the area in question has been visited. Games routinely hide or remove completed content in an effort to avoid distraction from new objectives. This functionality would not be ideal in a heritage environment, where the presentation of information *is* the objective. Methods which can sort and effectively facilitate search through the journal should be utilized in order to avoid frustration, i.e. if the user simply wants to do a quick fact check or another similar situations. Ideally, if an object in the world is interacted with, the journal should be coupled with that to provide direct access to the journal's information. Analogously, the journal should provide directions to the location of the object in the world. As seen in chapter 2, meta data could also be provided - although it is clear that the creation of such a dynamic system for every project would involve a considerable effort.

4.3 Evaluation

4.3.1 Considerations of restraint

In order to keep environmental immersion, original aesthetics and historical believability, most of the visual and interface cues presented here would need the ability to be toggled on and off to create a *realistic mode* and one or several *cue modes*. Such temporal solutions have been deemed very appropriate in heritage environments if carefully applied using proper interaction design[39]. The possibility to create several different modes could create confusion in and of itself, so the creators should be careful not to make anything that would take more than a short time to learn to use (i.e. a couple of minutes as a maximum) for a user from the general public.

Due to the technical possibilities in a virtually designed world, it should be considered (within a project's scope) to include all of the time periods known about the area. Evaluation of the importance of elements from various periods should not be entirely dependent on the designer[10]. This could either be implemented simply as a full change of the area between different eras, or as a user-controlled view of the evolution of select objects within the world - which can be from the smallest tool to large architectural complexes. This is an excellent case for the use of visual cues as well, since it would not otherwise be clear which objects that could be viewed in this manner.

> *"Ultimately, it is the instructional design of any learning activity which will determine its success, not the technology it employs* per se.*"*[39]

The question remains though: which cues are the most appropriate? It has been discussed in this chapter how each method could be applied to virtual heritage environments in particular. From this discussion, it is easier to narrow the selection to the most promising techniques. However, it is not clear whether specific cues would function well together in practice and what the best way to go about making them work together would be. The presentation could easily risk clogging the scene, and the world's immersive characteristics, if each and all of the cues were used indiscriminately and without concern of each other. This would clearly work against the very motivation of using learning cues in the first place, and is thus very important to address.

There are cases where cues can be dropped altogether. For instance, where a simple recognition of a special object or character is desired it is often far more effective to consider the silhouette of the model at creation time. This is far more important than any other detail the designers can apply to the object itself (e.g. texture work)[14].

In general, the environment gains accessibility and lowers user confusion by containing only a limited number of cue types - even if there are many more unused that would be appropriate in and of themselves. The designer should strive to utilize the chosen pool of learning tools to their maximum potential. Self-imposed restrictions in general often result in better and more effective or creative solutions. So to begin, creators should pick a selection of learning cues appropriate to their situation. Then another design iteration would follow, where questions are asked whether any of the cues' intended function can be replaced by other cues without the environment losing any significant accessibility.

A long ongoing discussion which has recently seen increased interest is whether games are getting "dumbed down" precisely because of all the cues and guidance that a player receives[22]. The trend since games started to appear commercially has always been an increase in this sort of hand-holding or as some would say, spoon-feeding of the player. There is controversy whether this is a good or a bad thing regarding game design and if the ultimate enjoyment of games becomes worse when players receive less actual challenges. Overly obvious or intrusive cues can potentially insult the user because they may be perceived as an indication that the developer assumes the audience lacks common powers of reasoning[53].

> *"One of the greatest features of early PC adventures was their ability to make the player feel as though he or she were really exploring a new landscape."*[22]

What can be gathered from this discussion is that all these cues are very effective at what they do - showing things to the player. Indeed they seem to be too effective in some cases as this controversy shows. A virtual heritage environment or similar environments which is not guided by challenges or objectives as seen in traditional games, can avoid concern about these controversies because the goal is not to challenge the user but purely to present information. Focus lies elsewhere in games, where mechanisms to aid exploration may diminish immersion[71]. If the creators still wish to implement a proper game (or any challenge to the users such as exploration) in the environment, the utilization of cues will have to be examined from a whole other perspective to ensure an interesting experience, which is not in the scope of this thesis.

4.3.2 Additional recommendations

From the previous section it is clear that it is not possible to recommend a universal selection of learning cues that would always work well within any virtual heritage application. In the identification discussions in this chapter, unique qualities have been identified about each of the cues and their potential usage in virtual heritage. However, it can still be difficult to gain a clear understanding of situations where one of them is more appropriate than the other. Rather, recommendations can be passed on which cues that are most likely to function well together in these environments by identifying their role in relation to each other, as well as some combinations that definitely should be avoided (i.e. not used at the same time).

There are several of the identified cues whose purpose may seem to overlap. In particular, the *visual* cues are all variations of showing or accentuating things using shader or post-processing effects.

Edge glow differs from rim light because it defines the entire shape of the object, and it is also not dependent on distance as the rim light tend to be. It is a much less subtle effect. Edge glow works best with one fixed camera angle while the user is searching the picture frame with the mouse cursor. Rim lighting (in computer graphics) is independent of the camera angle because the light always comes from behind the object or character. This means that both of these do not conflict with each other. Although in an environment the user navigates on his or her own, the rim lighting seems like the better choice for characters. Edge glow best belongs where it has been proved to be a great asset, which is in fixed-camera menu-like screens where the user only controls the cursor.

While rim lighting is good for characters, medium sized objects and smaller items are better suited with techniques such as item shimmer. It is important to be aware that with the shimmer effect, the smallest items will tend to appear only as either glowing light or too dark, depending on the tuning of the shader effect. For these smallest items, the best way to gain attention from the audience is the sparklers. This is because they ensure that the item is seen as intended in its entirety while the sparkle effect itself may very well be larger than the object.

Related to the sparklers effect on single objects is the “bread-crumb” trail. It may be argued that mini-maps or compasses could remove the need for this trail effect. However, there are some weaknesses in using a 2D representation, e.g. situations such as when multiple floors in a large building contain many objects which would overlap on the mini-map. Here, trails are excellent at guiding the user in three dimensions over different elevations where mini-map icons would block each other from being properly identified. Similarly, compasses only show the general direction to an object and while that may be a desired functionality as previously discussed, trails leave no ambiguity at all.

A compass would, on the other hand, also reduce the need for mini-map. If a compass is used with all the recommended features, it should be considered to only implement a full-screen map because the function of the mini-map is in part overlapped by the compass.

In order to avoid unnecessary confusion, many visual cues that are applied to a model can be divided further into separate, interesting areas on the model. The individual areas could be colored differently if an interaction were to have different effects, as previously mentioned. An example of such division of the model can be seen in figure 4.16.

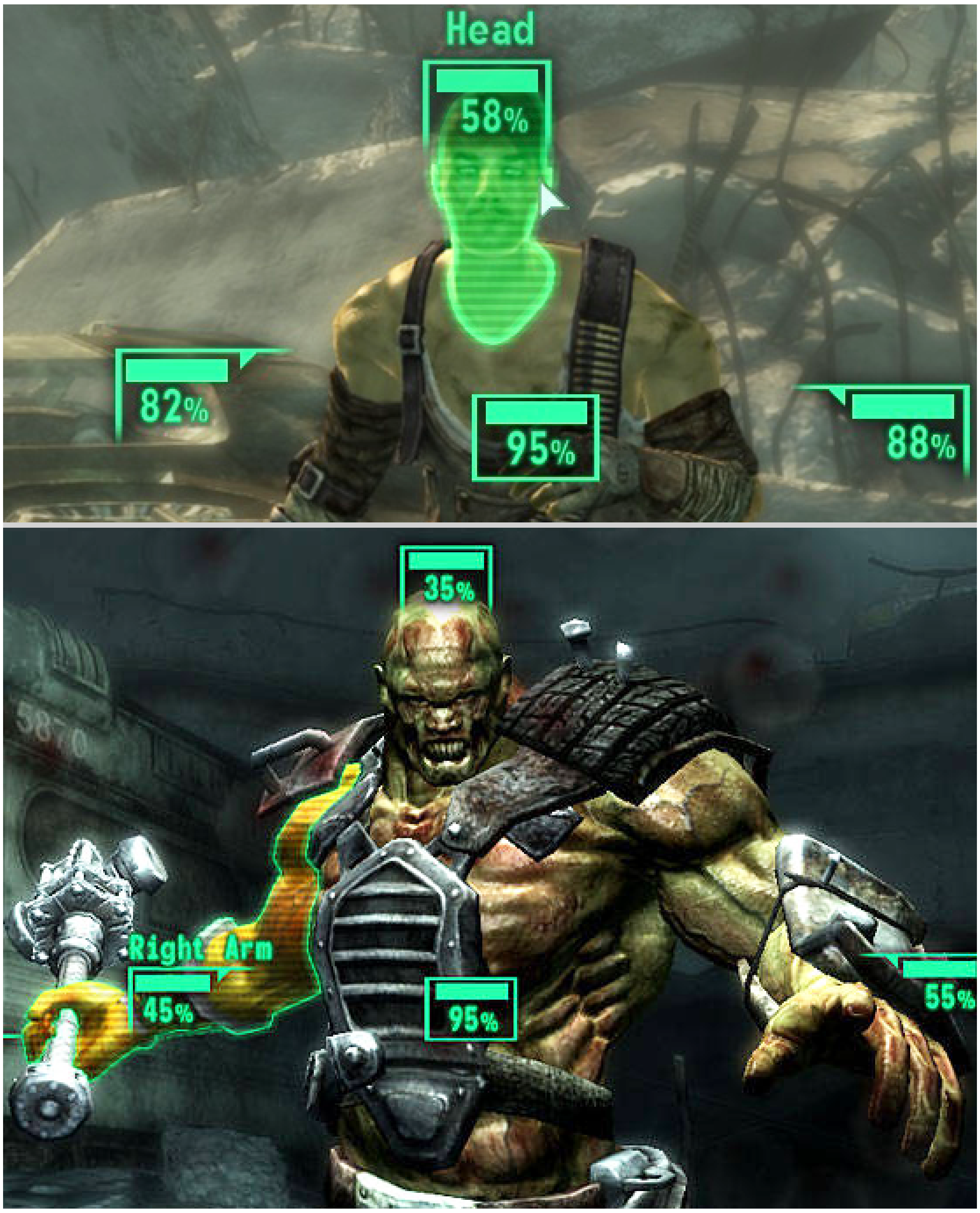


Figure 4.16: In these cases, the action of the game is paused and the player can use the mouse cursor to highlight differing parts of the target. An edge glow mechanism is applied while the game is in this state, functioning on each individual part of the model as the player explores the frame. From *Fallout 3* by Bethesda Game Studios[62].

Arrows and annotated icons on the mini-map can remove the need for excessive use of landmarks, interactive guides or other elements which disturb the impression of the environment more directly. It is necessary to find a balance between interface elements and elements that are part of the environment. A point will be reached where a large portion of the screen space is used by the interface. In that case, it should be considered to use smaller interface elements as replacement for others, e.g. a compass may replace the mini-map as previously discussed. It will become necessary to employ creativity in the attempt to reach this balance. One particular idea is to directly integrate the interface in the environment, as exemplified in figure 4.17, although this may affect usability towards novice users as it potentially becomes harder to navigate the user interface.

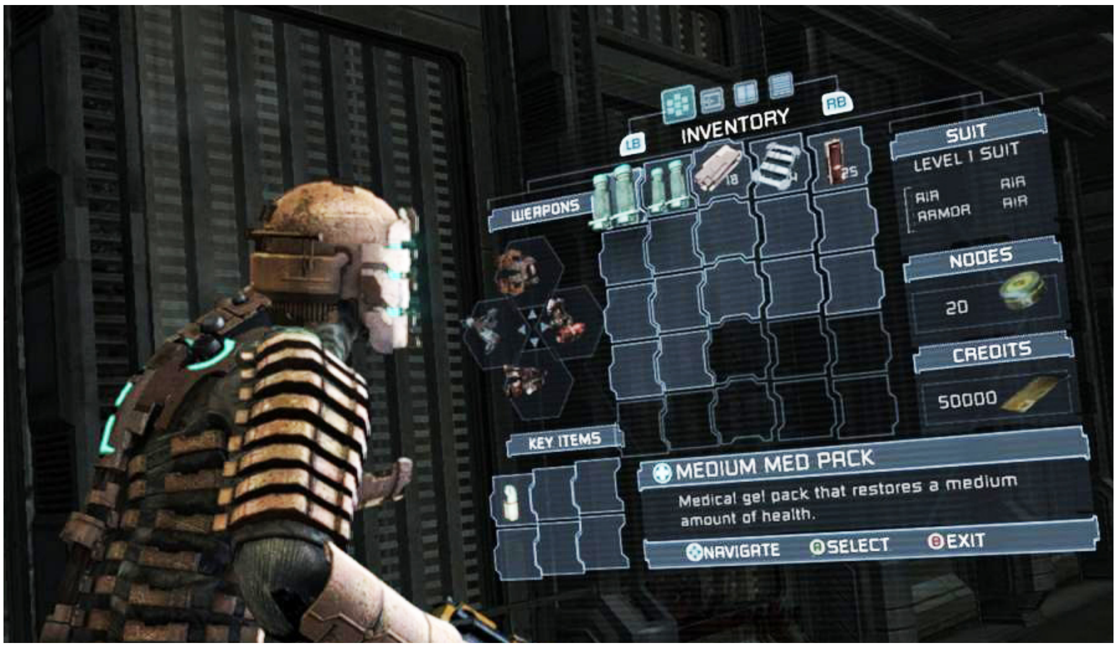


FIGURE 4.17: The interface is integrated into the environment. The player's health indicator is displayed on the back of the character model. Other functions are displayed as a holographic projection originating from the avatar's suit. The projections can be toggled on and off, so they are only visible when needed. From *Dead Space* by EA Redwood Shores[57].

Creativity in general is important. For example, if a fence or a gate mechanism is implemented, thick fog which encroach upon the player could act as a temporary fence (also leaving an opportunity to alter parts of the environment which the user is now blocked from viewing). If the environment is linear, the user could uncover more of the environment as a new area (or goal) is reached[9]. To implement this in practice, a suggestion is to use lighting composition guidelines to keep some areas completely dark and illuminate them when they are meant to be uncovered, with ignition of torches or similar light sources. In both of these examples, with fog and lighting acting as gate keeper mechanisms, invisible collision walls should be applied at the borders as discussed in section 4.2.2.1.

As discussed, the filtered vision techniques are impressive at first glance, but may be unnecessary or simply too intrusive in a heritage application. The other visual cues mentioned in this chapter will provide adequate replacement in most situations as long as the environment is not excessively filled with visual clutter.

Finally, elements from traditional interfaces such as WIMP interfaces can and should be used in conjunction with the other specialized elements discussed here. These are various interface features which are ubiquitous in most games and office applications alike. Examples are elements such as sliders and drop-down menus used within an option dialog box or control pane. Many guidelines exist towards how these sorts of elements work on screen, e.g. the importance of consistent use of colors, feedback to interaction and analyses of each type of element's strengths and weaknesses[37].

Chapter 5

Implementation

5.1 Technology description and choice

Modern real-time interactive environments can possess nearly photo realistic qualities. However, geometrical precision is still limited. This calls for the use of solutions that simulate this complexity, such as normal mapping and other features that can help represent a high level of detail despite the lack of very complex models[24]. To create a system from the bottom-up is not necessary because there are many alternatives in terms of toolkits that can be used to create modern graphics. One such toolkit is OpenSceneGraph which is popular due to its complete open-source nature and cross-platform support based on the API of OpenGL[51]. However, the creation of a rich and interactive world can be hard to justify within limited production means, i.e. money, staff resources, problem scope and so on. In many cases, the advantages gained from an open-source base are not necessary for carrying out the creation of the virtual environment; the development process is only hindered by extra work "re-inventing the wheel". There are other alternatives that facilitate the interactivity and many other commonly needed features in virtual worlds, both graphical and other technical solutions. Game development toolkits are very powerful and feature-rich, preparing the development process for most eventualities and present a ready framework of tools. There are some very apparent technologies that games can contribute especially well to Virtual Heritage projects. Perhaps the most obvious is real-time graphics rendering of high quality. Avatar technology is another example; this is the technology and design choices behind the concept that a user is (often directly) controlling a character that exists in the world. Together with countless other features which facilitate the discussed needs, it is clear that a modern and powerful, but fairly accessible game engine is needed for this particular purpose.

As explored in chapter 2, the most prevalent engines utilized for interactive environments in virtual heritage are the Unreal Engine[35] and the Unity engine[65]. They operate with different licensing terms. The engines have been used for architectural visualizations in general with successful and impressive results.

Unity is very popular among independent developers - it offers a free version, and a “pro” version with many additional features in graphics and other technologies, such as full-screen post-processing effects, render-to-texture, audio filters and many others. It has won several honors in the last years, and won the Wall Street Journal 2010 Technology Innovation Award in the software category. It includes an integrated development environment, and supports all major platforms including browser integration.

The Unreal Engine is one of the leading industry game engines. In the form of the Unreal Development Kit (UDK)[34], it includes all advanced features (including those mentioned about Unity) in a sales-based license (i.e. it is free if the project is not used for commercial purposes, as well as for educational use). Until recently, the engine was only licensed to non-profit projects through the developer’s games; all other licenses were expensive full-source licenses. However, recently the UDK was released, focusing on a standalone version of the engine with its integrated development environment, the Unreal Editor. It has been in use for well over a decade, with new versions being released on a steady basis. The UDK is updated monthly at present, adding new graphics technology and other features to the engine and editor. Its high visual standard and other advanced features such as physics and animation systems, combined with re-programmability and a large community around it, are among the things that have been argued in its favor in virtual heritage[39].

Both engines’ generality and relatively open nature should do well with most interactive virtual heritage projects. With the use of integrated scripting languages like UnrealScript or JavaScript, the designer can control the engines’ behavior almost completely, which is crucial for virtual heritage and many other types of visualizations: it does not need to act like a game at all. To choose, a few considerations weighed heavily. The implementation of the preliminary project used the Unreal Engine, and access to resources from other local projects utilizing the UDK was better. The considerable time of previous working experience with the UDK would result in shortened development time and leave more time to focus on the subject research. Access to all possible advanced features from the beginning without paying a license was deemed important so that most unforeseen feature requirements could be implemented. For these reasons, the Unreal Engine became the choice of the implementation part of this thesis.

A Virtual Heritage project will much more likely than not need its own assets and art resources, such as textures and 3D-models. These should in a professional product be based on knowledge from skilled personnel of how things looked like in the area in question. In order to keep this requirement manageable, any project using this engine should make extensive use of the editor's material creation possibilities, in which advanced shaders can be visually constructed from base art textures and procedural information. This not only reduces the number of unique art assets needed, but it also provides a great deal of flexibility to the constructors, where they can adjust materials exactly to their liking (e.g. the wood used in a house, the type of metal in a piece of art or the fabric of a cloth). This method of creating and using shaders are explored in order to provide many of the visual cues discussed in this thesis. Visual cues aside, as opposed to basic texturing, material shaders are a very available method to add a rich layer of realism to the environment and authors would do themselves a favor not to underestimate such possibilities.

5.2 Visual effects

5.2.1 Shimmer implementation

Shader description and use The shimmer effect implementation needed to accomplish two major goals:

- Be visually distinctive and pleasing to look at.
- Not require too much additional time to implement per additional instance it is needed (i.e. it should be easy to add new objects with this effect at any time).

As discussed in chapter 4, an animated appearance is desirable to ensure that the effect draws the user's attention. To solve this, a utility texture was created in an image editing software where a random cloud render was applied one time for each color channel (R,G,B). This is a good practice to base many random-looking effects upon, and is thus not only limited to this effect's use but can be reused for many different applications. In the shader, this was then animated (panned in an arbitrary direction) and desaturated, before a parameterized color was added to the result, so that the effect's overall color can be changed completely in any instance of its use by the use of a script or any premade trigger. This created a rough, windy overall look to the final material.

To add the recommended feature that the effect should change depending on the angle that it is viewed from in the environment, a Fresnel-expression was multiplied with the color creating a dependency on the camera viewing angle to the surface's world-space normal vector. The angle of zero-value, i.e. when the color is completely removed, was initially set to approximately 70 degrees so that the color would only be visible on any surface with the glancing angle of 70 to 90 degrees.

The resultant color values was then multiplied by a constant in order to create a *bloom* effect which creates the appearance of color radiating off the surface, adding extra visual impact in contrast with the rest of the environment.

In testing, the Fresnel expression appeared to remove too much of the effect. A wider zero-value angle was tested, but this immediately resulted in the original surface color being covered too much by color and was not pleasing to behold. In order to alleviate this, an idea was to only partially offset the angle of the Fresnel expression by modifying the expression's normal calculations. This should be procedurally generated, i.e. not use another texture as a basis, in order to avoid unnecessary memory use. To generate this effect, the engine's rotator and panner expressions were utilized. The UV coordinates generated from these can be treated as x and y components of a tangent space normal, so a z component was derived from these coordinates. This process was repeated and, in short, the result could be added together with any object's material's original normal map. The normals are then finally used to offset the Fresnel expression's default normal vectors.

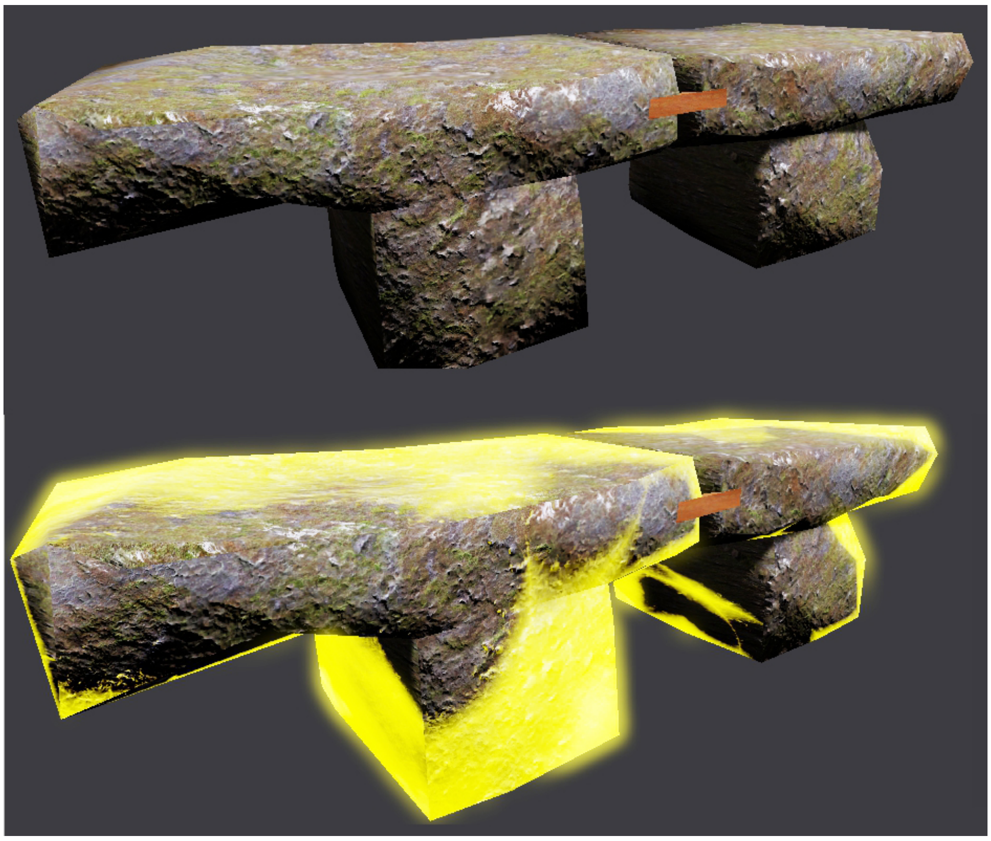


FIGURE 5.1: *Top:* A model of a stone bench as it normally appears in the environment. *Bottom:* The shimmer effect is applied to the model. The effect is animated and the appearance of the bench changes both with time and the angle it is viewed from.

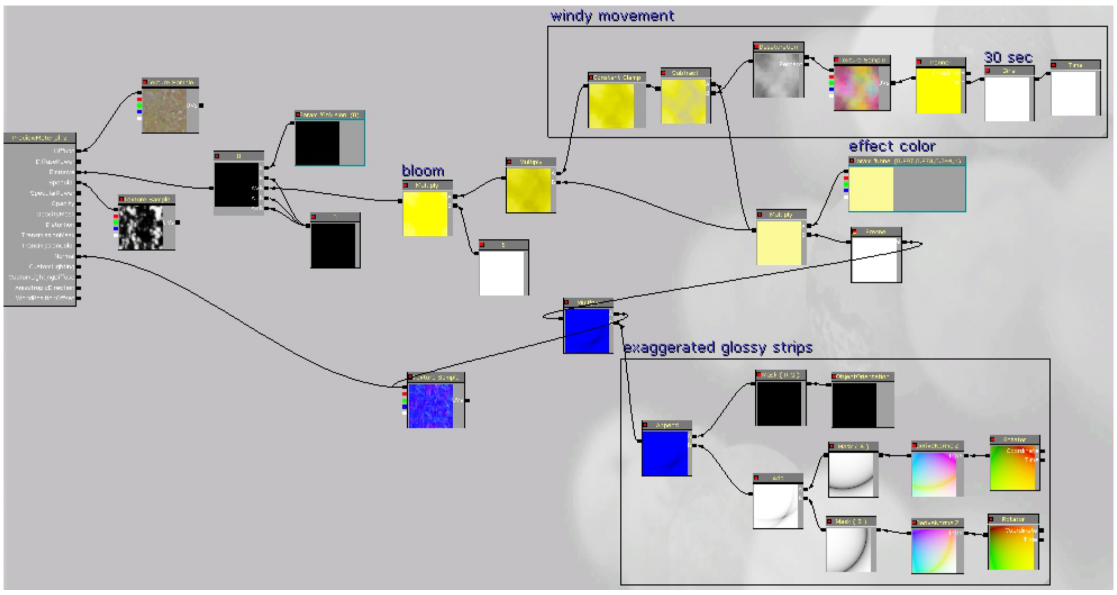


FIGURE 5.2: Overview for an impression of the entire material shader created for the stone bench. Most of the instructions are reusable; the stone appearance itself only constitutes a couple of instructions.

Fortunately, regarding the second goal, the creation of this shader does not need to be customized for every object's material. The instructions can easily be copied and essentially plugged into any material shader without the need to worry about the new object's UV coordinate setup or any other technicality, thus allowing effective reuse of the effect for any new model that is added in the future. A small exception is for very flat surfaces, where the Fresnel value may need to be adjusted. This is done in the case of the tympanum stone mentioned in chapter 3.

The effect is toggled through program code, which is described in section 5.4.

5.2.2 Sparklers implementation

The sparklers were implemented through the use of particle systems. The combination of several particle system configurations are combined into a single emitter actor which is placed appropriately near interesting objects in the environment. The particle configurations consist of initial distributed values as well as values that change using a curve which works over each particle's lifetime. An example of this is varying alpha (transparency) values for each particle. This in itself required some material configurations to mask the particle textures correctly. The spawn locations are distributed over a limited area. All other variables such as particle life time, size, velocity and so on are randomly distributed within pre-determined ranges. The overall look of the effect has been tweaked to be very distinguishable and not look like a natural phenomenon such as fire or real sparks, according to the results in chapter 4.

A final control layer was added, implementing the PhysX system of the UDK to apply negative gravity to the particles. This modifies their velocity so that they always push up into the air after some time, regardless of the emitter's initial orientation.

The sparklers are, as with the shimmer effect, toggled through program code. This is described in section 5.4.

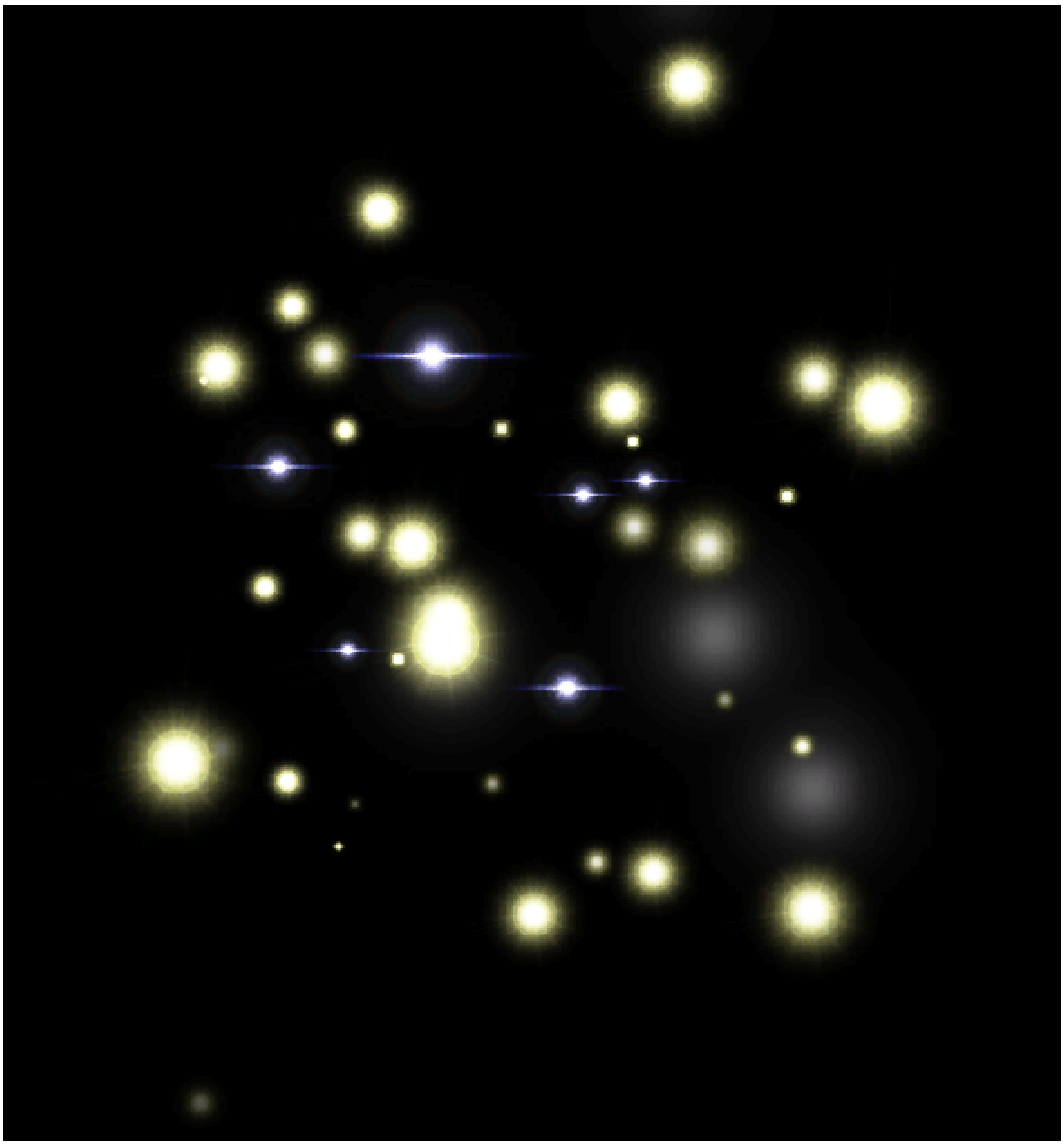


FIGURE 5.3: A preview image of the sparklers on black background. When animated, the particles move and blink in and out as a way to increase the user's awareness of the effect.

5.2.3 Trail

The bread crumb trail described in chapter 4 was found to fit situations where a single objective is present. However, without any such pointers and even with the aid of a map, the environment was deemed too confusing due to the architectural similarity and narrow streets which provide too little overview. It was decided to add a synthetic trail which marks the route identified in chapter 3. In addition, in situations where the user is outside of the main route, indications were added that lead back to the trail using the shortest path.

In order to avoid the path cluttering the frame more than necessary, the effect's visibility is based on the distance to the user's current position. In other words, it is not visible from that distance outward.

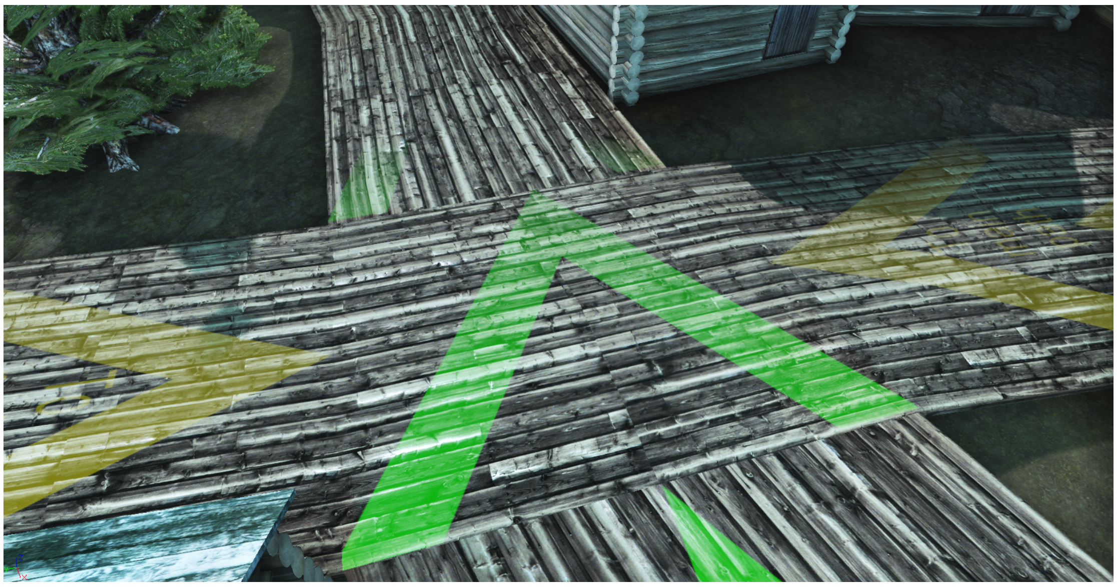


FIGURE 5.4: The trail at an intersection. The main route is indicated in green. The yellow arrows are from other streets and always lead to the main route. The effect is translucent and becomes invisible from a certain distance from the user's viewpoint.

The trail is also completely toggleable like the previous cues, described in section 5.4.

5.2.4 Collision indication

In keeping with the recommendations from chapter 4, where invisible walls are difficult to avoid there should be some sort of indication that the collision is actually there. Most of the time, this is done naturally with the buildings and other environmental elements. Still, there are some areas where the addition of blocking elements would not look right. Alternatively, persistent users may find ways to climb buildings or in similar ways avoid those elements to gain access to areas that are not intended for users to be in. This is solved by blocking off the world with invisible collision walls that do not interact with the lighting system. When the user gets too close, a color indication was implemented based on the distance to the user's viewpoint. The closer the user get to the walls, the more opaque the effect will become. This leaves no doubt of where the intended area of play is, while ensuring not to frustrate users unnecessarily by bumping into completely invisible walls.

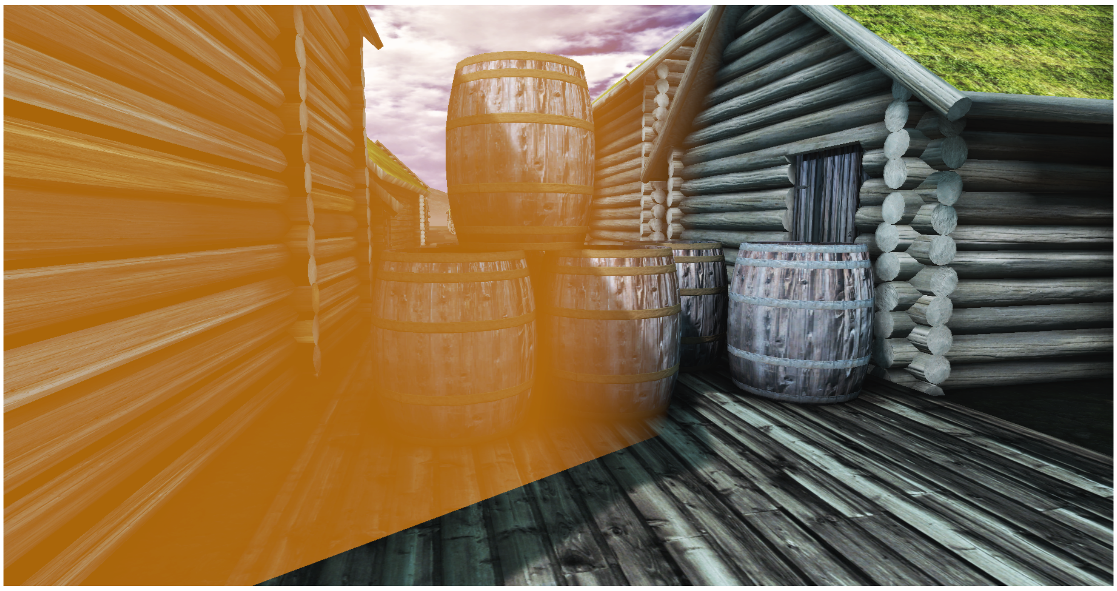


Figure 5.5: The user is too close to an unintended area. In addition to environmental elements blocking the path such as these barrels, the user also gets a visual feedback that the area is off-limits with the help of this collision indication.

5.3 Environment

5.3.1 Landscape

During the research regarding the environment surrounding the city, an initial problem was posed: How much terrain is necessary to implement? For performance reasons you would want to avoid large amounts of detailed ground that the user will never see anyway, but at any rate it has to be at a believable scale. Additionally, what is the necessary level-of-detail as the distance from the play field increases?

To answer these questions, an analysis was performed to reveal what would be visible from the area of the city the user would walk around in. For this, elevation data of said area is needed. An assumption is made that the elevation of the landscape is visually the same as it was in the 14th century. The university (NTNU) has high-precision elevation data available. However, for this rough analysis of visibility in a radius of many kilometers, a free source of medium-accuracy elevation data was used[1]. This is is well suited for the kind of precision needed in this case.

Among many different elevation data types, .DEM (Digital Elevation Model) and SRTM data (.HGT) are widely used. A few freeware programs also exist that handle these formats, which is well suited for visualization purposes (i.e. there should usually be no need to utilize expensive programs just for this purpose, thus cutting project costs). To read and confine the initial elevation data files, 3DEM[2] was utilized and then MICRODEM[3] primarily for its view shed and panorama functions. A point was specified roughly at the (real-world) coordinates N63.434 E10.404, at a height of about 25 meters above ground level. A visualization was achieved of the line of sight in all directions from this point.

[1]`http://www.viewfinderpanoramas.org/dem3.html` (P32)

[2]Discontinued, downloaded from `http://freegeographytools.com/2009/3dem-website-is-gone-but-3dem-still-available-here`

[3]`http://www.usna.edu/Users/oceano/pguth/website/microdem/microdem.htm`

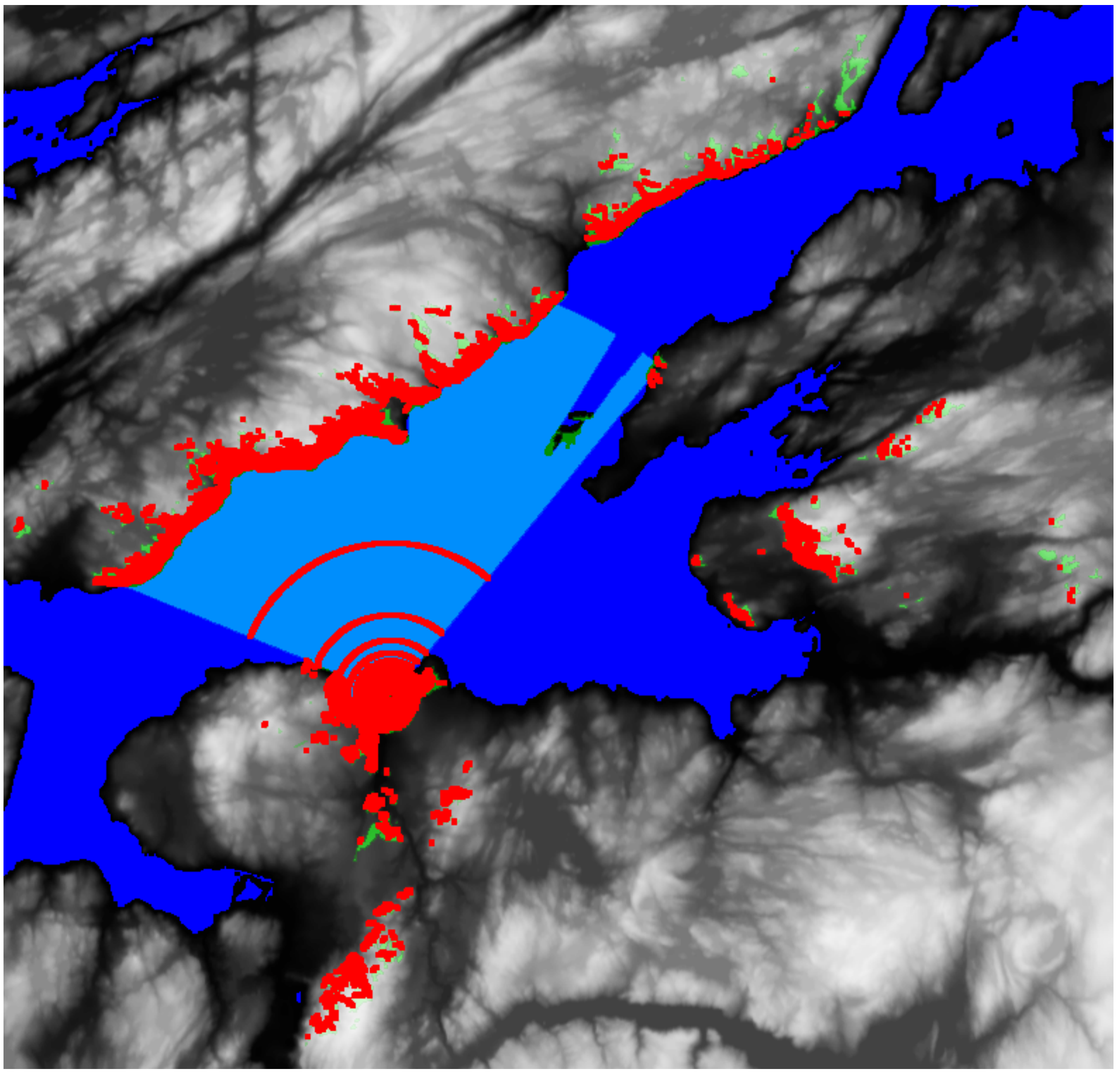


FIGURE 5.6: Terrain view shed and panorama visibility. The red points are locations visible from the specified point.

The majority of the points constitute a 70-kilometer square of terrain. Four choices appear (in descending order from most to least realistic):

1. Re-create the entire terrain in a 1-to-1 scale.
2. Use the entire terrain, but scale it down some to facilitate better performance; it would still maintain the same panoramic view.
3. Create a fake backdrop effect with a very small terrain.
4. Create a large, flat panoramic surface (the world environment's sky dome would already be present, so an option would be to add the panorama to it).

Option 1 has every benefit except it may result in problems performance-wise. This should be attempted in a test environment; if there are no noteworthy performance problems, it should be utilized in favor of the other options. The most significant advantages to this are that it would result in the most exact visualization and it would facilitate easy future expansion of the environment or application as a whole.

In any event, some tests of the second option were performed. An approximation of the full-scale terrain is possible in the engine. A height map of 16bit accuracy gray scale bitmap was made with 3DEM and Terragen Classic[4] to make a terrain representation in the editor. In order to calculate the correct scaling in the engine, the following formula[5] is used for every axis:

$$NumPatches_{axis} \cdot (DrawScale_{axis} \cdot DrawScale3D) = UnrealUnits_{axis}$$

where $1\, UnrealUnit = 2\, cm$. The test terrain was scaled down in all dimensions to about 1.080.000 uu which translates to 21.600 meters; this did indeed result in the same visual backdrop effect as a full-scale terrain. But the arguments for a full-scale terrain only make this viable if the large terrain does not induce a major performance hit. At any rate, the small-scale terrain model is useful for a few tests. The engine utilizes automatic view occlusion, but for such an organic shape, the results are difficult to predict. Preferably, the results should be similar to the view shed analysis described earlier - that is, everything should be occluded except terrain which is visible from the play field. A more thorough test was appropriate to check if the imported terrain data is occluded well enough to achieve good performance. Thus, using two editor viewports, one as the primary render and the other as an occlusion child in top-down orthogonal view, a clear impression of what is being rendered is achieved. The rendered patches of terrain did indeed prove, for the most part, to coincide with the line of sight analysis described earlier. The following is a composite image of the re-created terrain actually rendered from the vantage points the user would have access to in the play field.

[4]`http://www.planetside.co.uk/content/view/16/28/`

[5]Formulas according to `http://udn.epicgames.com/Three/TerrainDesign.html`, also where an in-depth discussion and recommendations of terrain creation in the current version of the engine can be found.

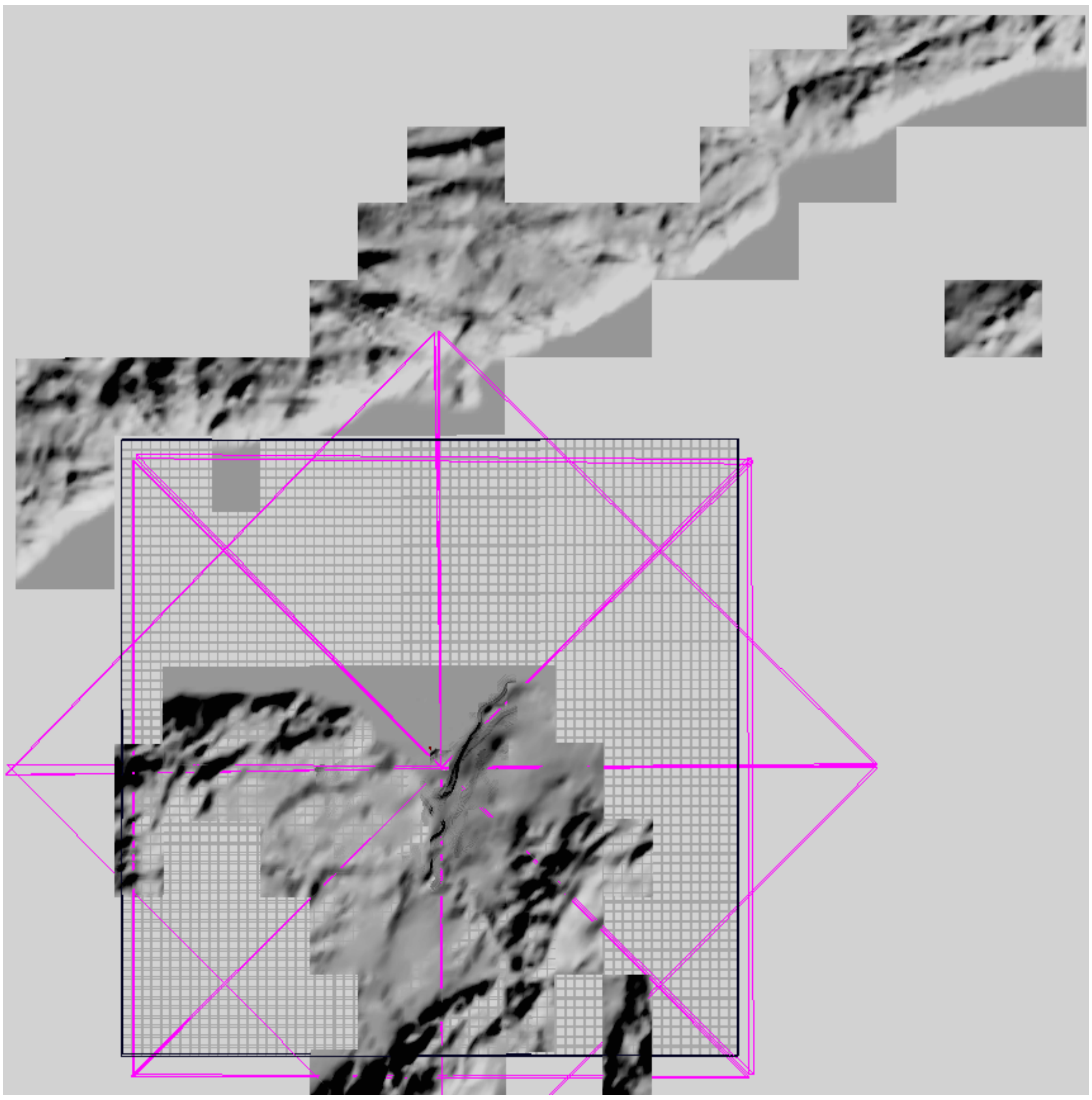


FIGURE 5.7: The viewport origin is located at the same point as the view shed analysis origin, and a picture is collected every 45 degrees to capture all occlusion the engine performs, which forms this composite image. Note the similarity in location of the final rendered terrain and the real-world analysis in figure 5.6.

With these preliminary tests leading to such a promising result, a full-scale 70km terrain was considered possible. It was implemented as a 512 by 512 patch terrain from a height map with a similar number of pixels[6] and with a draw scale in both X and Y directions of 6835.9375. This took a very long time to translate and scale because the operations needed to work over huge distances. But once it was in its final location within the engine, there was no notable performance hit (compared to the downscaled terrain tested earlier).

[6]The engine requires the number of terrain patches to be $HeightmapPixels - 1$.

Due to the system that stores shadow and light information is relying on light maps, some may be concerned about the size of the light maps on the huge terrain and consequently the file size. However, the light maps are sized according to the number of terrain patches, making the draw scale irrelevant. This means that even though the terrain spans 70km, the light map will not be particularly large at all.

However, the backdrop terrain was too large-scale to represent the local play field terrain; because of the huge draw scale, it caused too large terrain patches resulting in a jagged view when up close. To solve this, i.e. produce a smoother terrain, an additional smaller area but higher LoD terrain with patches that are placed relatively close to each other needed to be made. This covers the entire playable area with a fairly large margin. It needed to be merged or, more precisely, positioned together with the backdrop terrain in such a way that the transition is invisible to the user. A 257 by 257 pixel gray scale height map of the central area was acquired for this use.

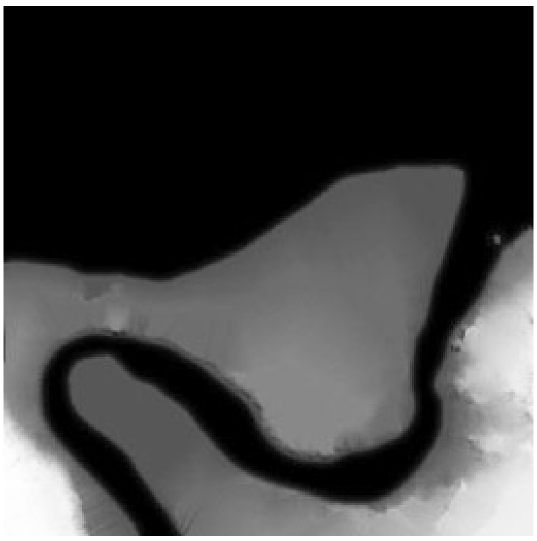


FIGURE 5.8: The height map of the central area visualized in gray scale. From [44].

The area is, in real-world scale, a square with sides approximately 2400 meters. This means that 2400 meters corresponds to 120000uu, which when using 256 terrain patches (required to match the height map pixel count), yields a draw scale of 468.75 in X and Y directions, where DrawScale3D is 1.

The height map of the central area resembles the state of the landscape of the present. The most important variations in the riverbank from the medieval model (described in chapter 3) were altered with the help of an orthographic projection of a sketch of the model. During this additional landscape modeling, this projection was scaled to cover the entire area making it fairly easy to identify the exact location of riverbank alterations relative to one another.

As mentioned, the two terrains were positioned together with an initially indistinguishable transition from one to the other. Due to the huge scale of the largest terrain, some additional LoD measures had to be taken, which threatened this transition. A dynamic morphing affecting tessellation based on distance to each vertex on this terrain was activated, resulting in some visible “islands” of the terrain penetrating the smaller terrain from a distance. This was alleviated by raising the minimum limit of distance before the morphing takes place, as well as directly lowering some parts of the terrain that should not be seen anyway.

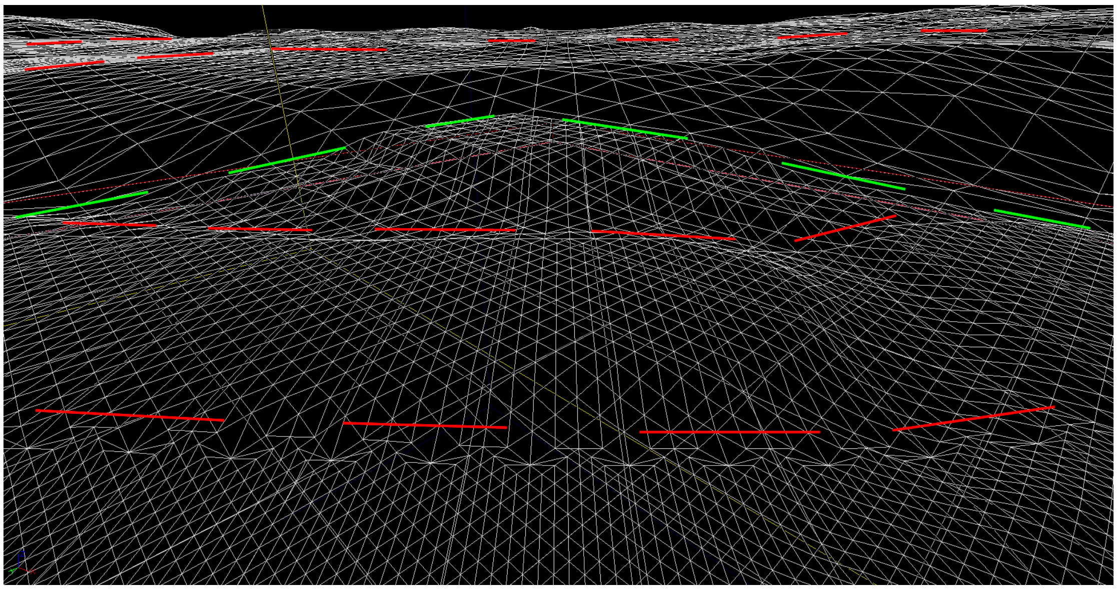


Figure 5.9: Here viewed in wire frame is the tessellation of the terrain patches based on distance. The tessellation shifts have been indicated by red lines, and the transition from the small central terrain to the larger has been indicated by green lines.

The terrain was textured using a mix of a procedural method and manual texture painting. The procedural method is based on variables that control how much of a material layer to let through based on the absolute height as well as the slope of the terrain. Some random noise is added to this. The manual paint was added to mix in additional variety, especially to create a worn, muddy look at areas that would have had high traffic.

5.3.2 Sea and river

The creation of water included both the fjord and the river Nid which runs along the entire settlement. For both, the height maps previously utilized to create the terrain features were now used as a reference while modeling the two-dimensional water bodies. The same material shader was originally intended for both, although this had to be abandoned in favor of making a more accurate representation of the moving water in the river. Technically, they still are the same shader but they are different instances of each other. A variable in the parent shader is set to control the UV tiling and panning movement of the texture samples, so that the river appears to flow in a single direction while the sea has a more undetermined direction. The UV texture coordinates had to be modified (bent) in modeling software to accommodate the appearance of a water flow following the shape of the river. The shader uses numerous processes to create a sufficiently organic-looking appearance. Real-time reflection is applied using a dedicated render to a texture via a scene capture actor parallel to the sea level, and normal maps are used to perturb the reflection image. The water's opacity is changed according to a pixel's distance to the shore. A similar measurement is also used to blend water foam into the surface near the shore in order to more clearly distinguish where the water line begins. In addition to other effects such as distortions of surfaces behind the shader, post-process and physics modifiers were created in case the user or camera should move below the water's surface. The use of the height maps and the interesting special effects is a great-looking feature which adds much believability to the impression of the virtual world as a whole.

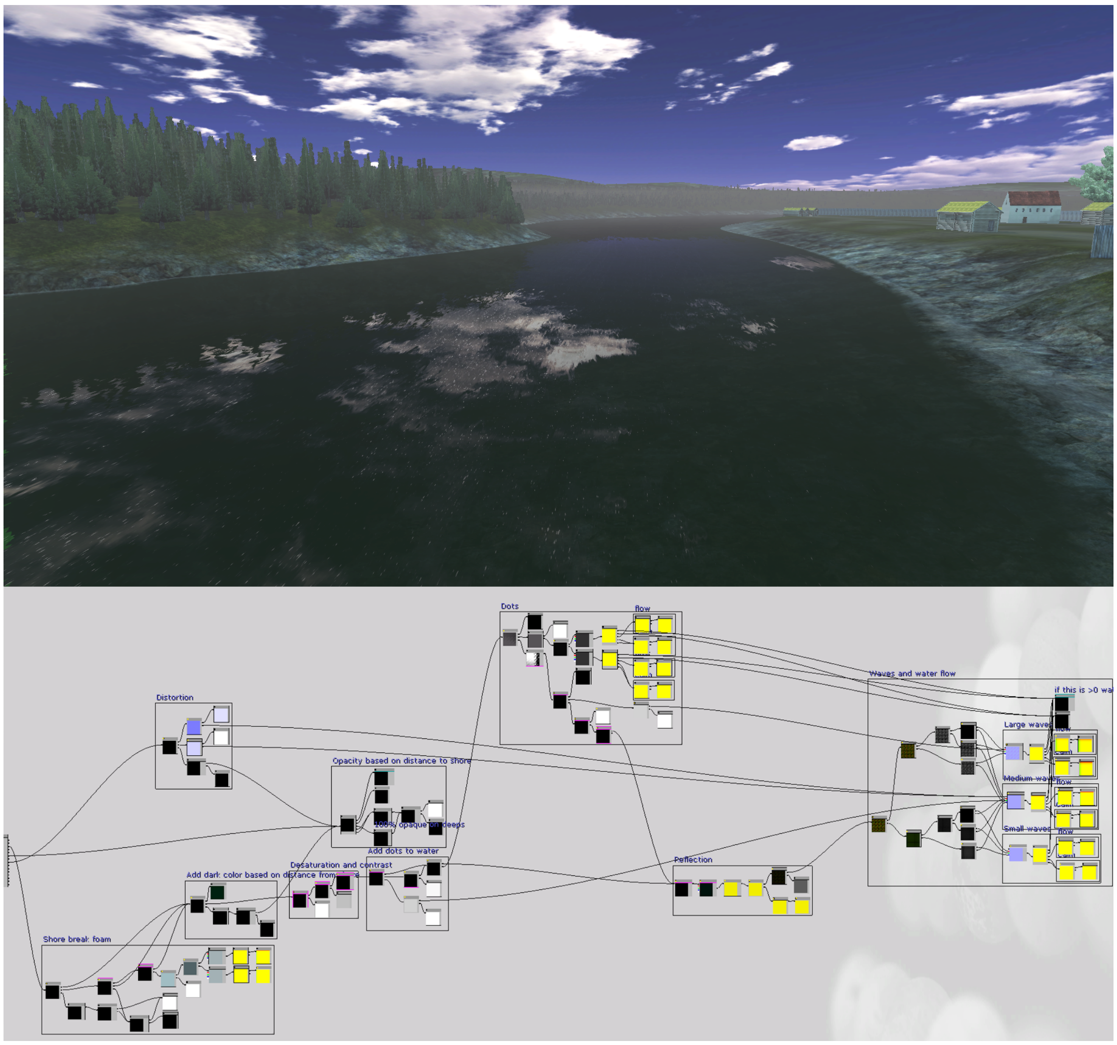


FIGURE 5.10: *Top:* The finalized look of the water, with real time reflections (as opposed to a cube map or other pre-calculated diffuse color). As the shader is highly procedurally animated based on real-time reflections, it may be difficult to gain a representative impression of it by looking at a single frame on paper. *Bottom:* An impression of the amount of logic created for the shader.

5.3.3 Sky

The sky consists of a sky dome model, scaled to encompass the entire landscape. This is placed like a lid over the viewable area. A panoramic unlit surface material depicting the sky is applied and panned around the scene at a very slow rate to create the appearance of clouds moving. The sun is emulated through light shafts (or light beams) which interact with the white clouds as well as geometry on the ground to create a blinding light effect. This effect share the same actor as the directional light (discussed in section 5.3.5, see for depiction), and is thus emanating from the correct position in the sky (the light's trace distance had to be extended to accommodate the large distances in this particular environment). Finally, a lens-flare effect was added for additional visual impact.

5.3.4 Flora

The flora implemented is largely trees. They are created with the SpeedTree Modeler[36]. The models can be imported into the UDK. They can appear to react to the wind and, if desired, other physical variables. The leaves or conifers may be represented as “billboard” surfaces (always facing the camera), or as real geometry. In this implementation, both alternatives are used. To create models of the Norwegian spruce, the needles were created using only a reference image and projected onto frond-shaped meshes attached to the branches of the trees. A mixture of randomization and explicit placement resulted in the final look. In total, 12 different trees were created. To ensure a randomized look, they are scaled and some foliage materials are interchanged on the models.

The fully detailed trees are used in the areas of focus. This means they are located along the streets and in backyards. However, a distant forest needed to be created. Because the tree models are relatively expensive in terms of rendering time, the forest could not consist of the fully detailed trees. A variety of alternative approaches needed to be tested in order to identify which method could provide the best visual appearance possible while still keeping the processing cost at acceptable levels.

SpeedTree provide some level-of-detail functionality. At the farthest distance, it generates billboards for the entire tree and displays them based on the view angle. While a realistic alternative in terms of rendering cost, the practical placement of the trees had to be done on a per-tree basis. It was quickly determined that this was not a viable method when operating at the scale that this environment does.

Another method was to use *deco layers*, which are able to distribute SpeedTrees or static meshes on the terrain on any desired location. However, it became apparent that this only operates on single quads of the terrain, resulting in a too uniform look, as well as too few trees to create a convincing look of a forest.

Yet another alternative was foliage volumes. A volume can be created which distribute static meshes on any surface according to many variables. But it is a clumsy method because the shape of the distribution area is not very modifiable, causing the edge of the forest to appear completely straight. In addition, no control was present for controlling the rotation of the trees based on the ground slope angle.

Finally, it was decided to include a terrain layer with the same material as the grass material already existing, however this was modified to include foliage meshes where ever that layer was painted. The variable controls were also better through the terrain material. SpeedTrees can not be used as foliage, so simple static meshes were created by making several planes interconnecting at the middle of the tree, each plane manually mapped to the previously created billboard textures from the SpeedTree generation. Many meshes were created for this purpose. However, during testing it was found that due to performance reasons only about two unique models could be used at a time in the foliage layer without unacceptable frame rate loss. Thus, there are several unused assets. At a distance, this nevertheless appears as a believable forest.

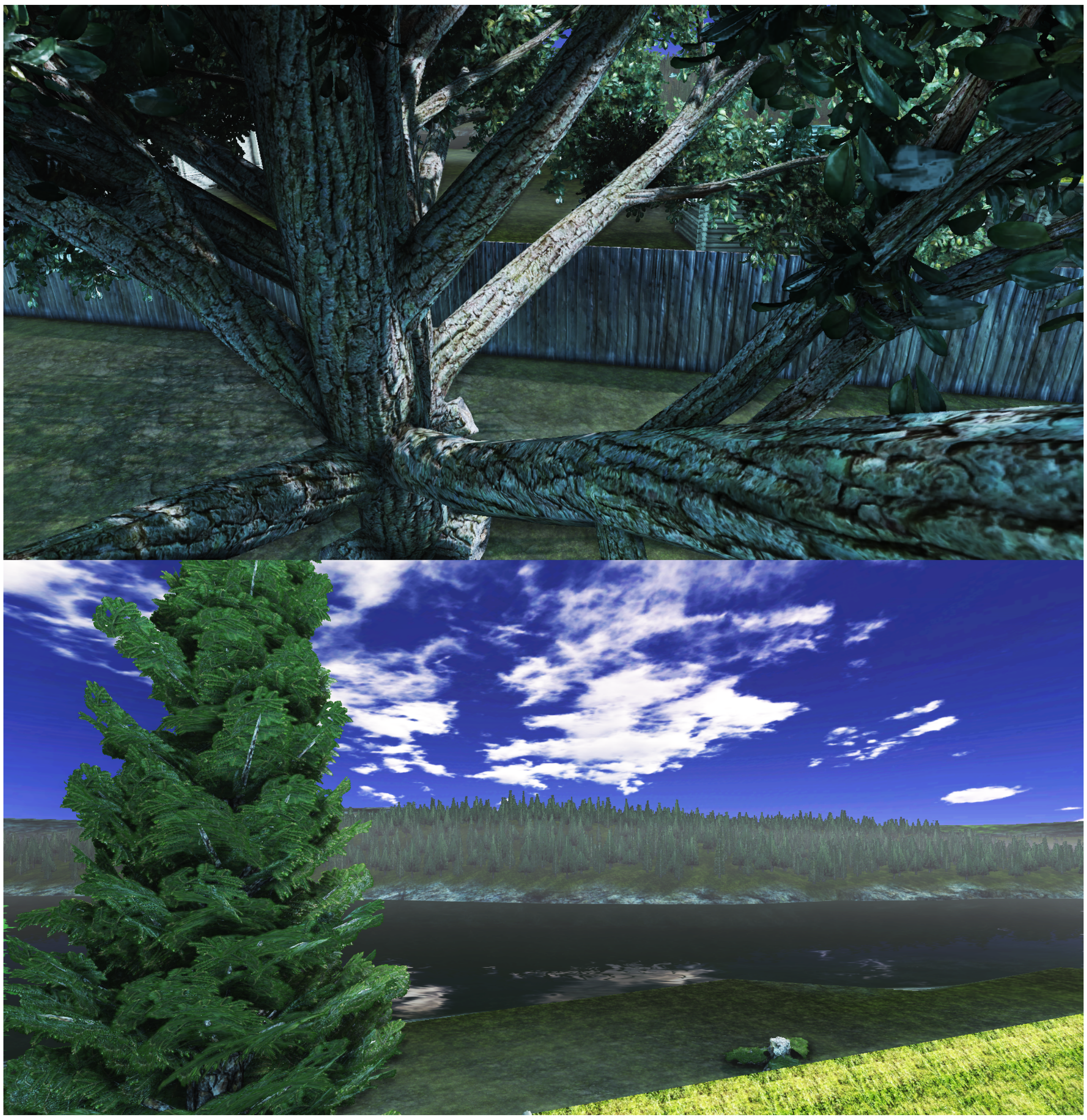


FIGURE 5.11: *Top:* Highly detailed and animated trees with alpha channel masked lighting are possible with the integration of SpeedTree models. *Bottom:* Norwegian spruce seen in the foreground with the distant forest on the other side of the river.

5.3.5 Lighting

As discussed in lighting perception in chapter 4, it is important to implement proper lighting in virtual heritage environments. This meant that the world could not simply be lit by seemingly random lights only present to illuminate the area. One of the more important elements in this regard is to design the light in a fashion accurate for the age and location in question, especially when learning cues related to placing lighting are used. Data was therefore acquired which provided accurate color values of the burning fuels that could have been used. This provided a reference on which the non-natural lighting was subsequently based upon. Originally, a night-time representation was also planned. However, in the final environment there is only day-time. Because of this, the non-natural lighting is used in the one location available; inside a house that the user can walk into. Still, this should provide an easily extensible platform to use as a base in future projects.

Material	Red	Green	Blue
Animal Fat:	0.759	0.24	0
Cod Liver Oil:	0.772	0.228	0
Light Bulb:	0.542	0.345	0.131

FIGURE 5.12: The RGB color values produced by burning relevant fuels. Also presented for comparison are the color values of a modern light bulb. From [17].

The natural lighting, i.e. ambient lighting and light from the sun, also presented challenges. The angle of the sun needed to be considered. The point specified for the view shed analysis presented previously, N63.434 E10.404, was used to calculate the sun's correct position at the date July 1st, year 1300 at three in the afternoon[48]. This yielded a solar azimuth of 152.52 degrees (clockwise from the north) and a solar elevation of 47.42 degrees (up from the horizon). These values were used to implement the engine's *dominant directional light* which can simulate the parallel light rays of the sun. The sun's color was approximated as a high noon sun with some adjustments[29] to fit the sky texture utilized which has a large amount of clear blue atmosphere.

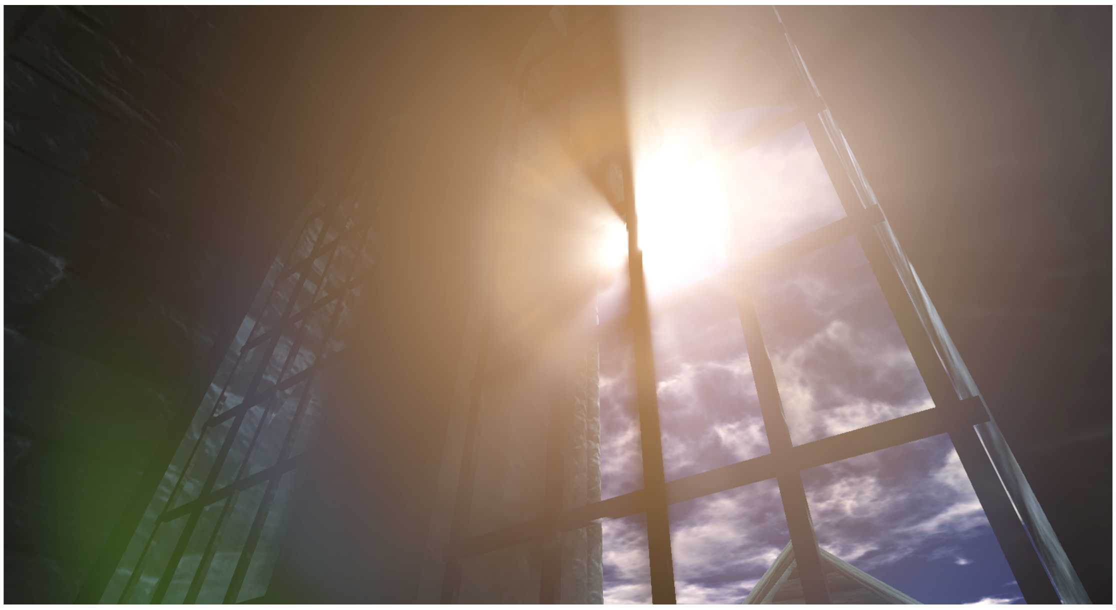


FIGURE 5.13: The sun, viewed with the blinding effect discussed in section 5.3.3. The position of the clouds in relation to the sun is important due to the amount of blending and thus the apparent position of the sun, because it has no actual geometric representation (this is somewhat alleviated by the lens-flare effect). Empty space in shadow from the sun will modulate the color of the scene when it is seen through.

The engine utilizes light maps to store static lighting information. This information is not, unless specified, calculated in run time in order to save processing time. It includes bleeding colors and indirect bounce lighting ray tracing. The light maps need to be included with every model as a separate (from the material) UV channel. There are some challenges that were encountered with this. As the construction used many 3D models created from other projects, the light maps were in most cases not present. The models that come from external projects or animation-only projects tend to not have light maps because it is not required in a non real time environment. This means that even though some models are received seemingly complete to any virtual heritage project, a lot of work may still go towards the conversion process, such as creating light maps for every single model. This can very easily lead to laborious work during the process of preparing the models for use in the game engine. Some of this work may be done semi-automatically with the combination of several automated mapping processes. However, there are many cases where it has to be done manually using a UVW-unwrapping procedure in order to look good or simply to be error-free.

An important requirement for light maps is that any surface on the model should not overlap on the UV map. The surfaces should also not be too close to each other depending on the resolution of the light map, because the lighting information of one surface could bleed over to the other surface on the light map, creating artifacts in the final representation. This means that the texel padding between the UV charts should be larger than the lowest light map resolution used on a mesh. In practice, this has been shown to be a minimum of 4 texels between UV charts - e.g. if using a light map resolution of 32, the texel padding should be 12.5% of the entire UV space[47].

In an effort to make sure that the general system requirements for the world are not higher than they needed to be, some memory optimizations were applied with great success. Except for the aforementioned padding in UV space, all other space between UV charts is a waste of memory: the memory is reserved in the light maps but the empty areas of the map are never utilized. Large models still need to light map every surface which can lead to high light map resolution requirements because of the needed padding. The memory requirements created by those large resolution light maps can be lowered by splitting the large models into smaller parts and light map those parts properly in smaller resolution light maps to waste less texel space. This smart utilization of the UV space will also result in better lighting quality because more texels can be used to store lighting information. This not only lowered memory requirements and increased the general lighting quality, but also decreased build times by almost an order of magnitude, as shown in figure 5.14.

Model	% lighting time	Total texel memory (kB)	% wasted texels
Large:	82.092	5461.33	87.2
Split:	15.564	165.28	31.56 (average)

FIGURE 5.14: The build times and unmapped texel memory cost were dramatically decreased by splitting larger models into smaller ones and light mapping the individual pieces properly for smaller light map resolutions. This also increased overall lighting quality. The remaining 2.344% build time is the test base BSP surface area.

For particularly large projects or projects that undergo many production iterations, the engine's parallel network build processing program (the Swarm Agent) could be utilized to great effect. Network build processing was not used for this project, because of its relatively low frequency of new builds.

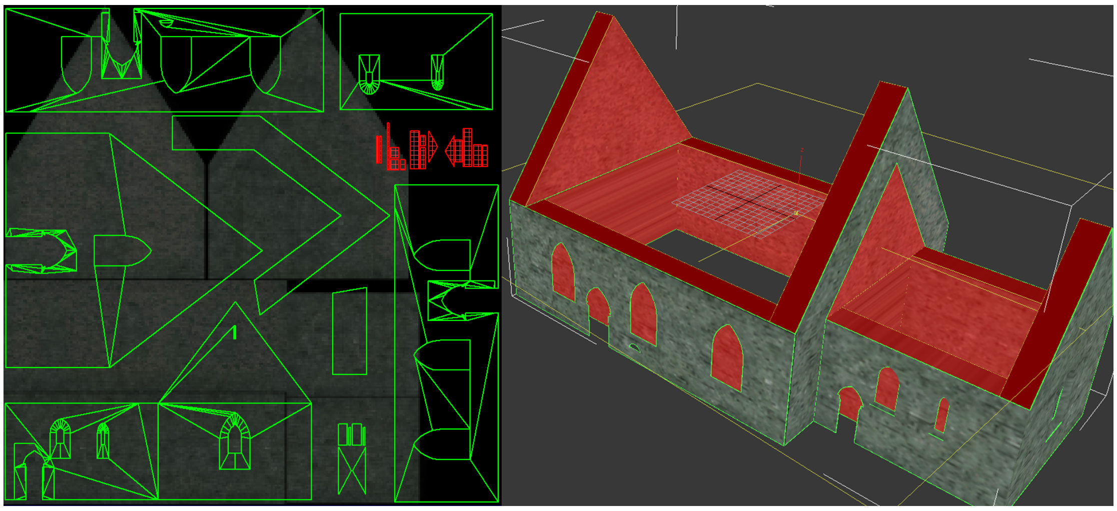


FIGURE 5.15: This large building contains surfaces that would never be seen by the user, indicated by the red selection. These surfaces do not need any light map quality at all, and can thus be shrunk relative to the other surfaces to leave more lighting information for those. This creates much better lighting quality overall (especially with large objects such as this) with the exact same memory requirement as before. The light map operates on a separate UV channel than the texture map which is seen in the background.

There are also some subtle post-process effects applied including depth-of-field, height based fog and a relatively noticeable ambient occlusion. The ambient occlusion helps alleviate some rendering artifacts due to light map resolutions, especially on the terrain itself where it darkens everything inside crevices created by other geometry. There is no ambient lighting at all - this was decided against because the engine's Lightmass system which calculate bounce lights does a good job of lighting indirectly lit surfaces. The one exception is for particularly large buildings such as the Nidaros cathedral, where a dedicated lighting channel which only affects that model was utilized to illuminate the side facing away from the sun, using a directional light pointing in the opposite direction of the sun which also only operates on that lighting channel (it does not affect any other model). Except for the case of the cathedral, this system presumably saved many days' work of simply adding proper ambient lighting. A global ambient lighting would also affect the directly lit areas and was not an option from a realistic lighting standpoint.

5.3.6 Additional notes

5.3.6.1 Texture and material creation

The source texture art and images used in this implementation are some of UDK's own, but largely from free resources on the web[66]. These textures have been modified in image editing software to display correctly when tiled and repeated on a 3D model surface. In many cases, derivative texture samples were created for normal maps, displacement maps (for parallax mapping), specular and ambient occlusion. This was done using special software[11].

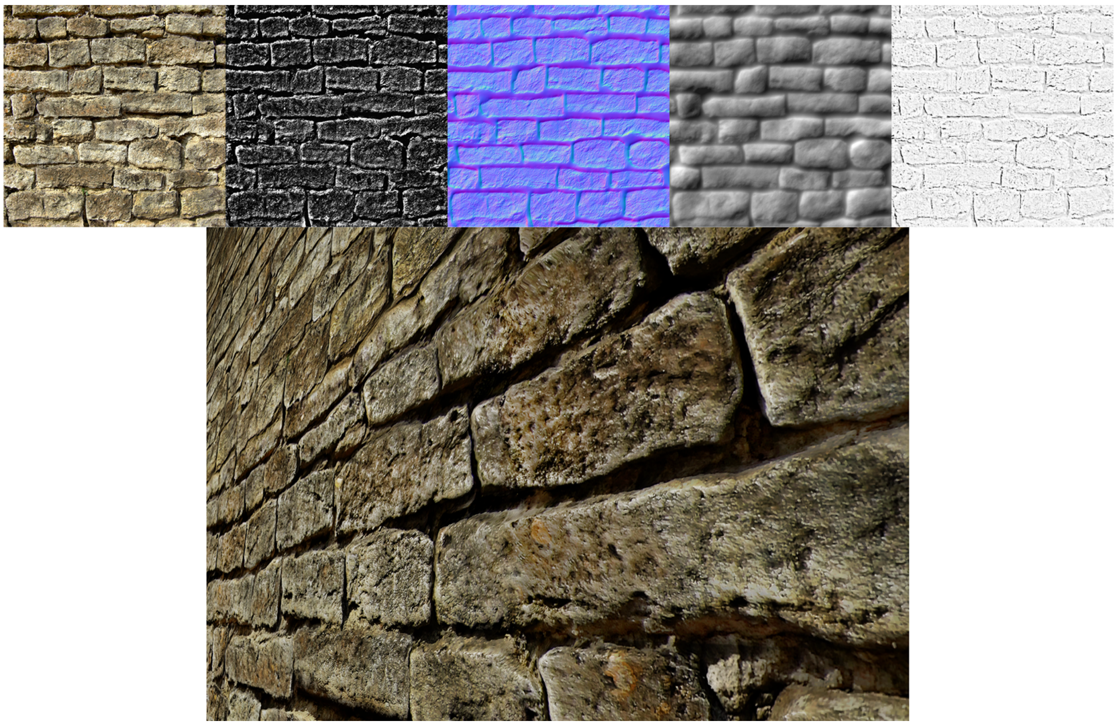


FIGURE 5.16: Material creation work-flow. *Top left to right:* The original texture sample was edited to lessen shadows from the sun from when the picture was taken. Specular, normal, displacement and ambient occlusion maps were derived from the edited texture sample. *Bottom:* The complete material shader applied to a surface with lighting.

5.3.6.2 Field of view considerations

The field of view (the angle at which the environment is projected on the screen) plays a large role in the user's interpretation and perception of the environment. It can affect a person's ability to orient and navigate interactively in 3D-space. There have been reports of nausea resulting from playing games with a too wide (and in some cases too narrow) field of view, however, the sense of presence in the environment is higher with broader angles (depending on the display)[55]. A broader field of view will distort the side of the view, while a narrow field of view will cause a zoomed-in appearance. A wider angle is closer to the real world experience of almost 180 degrees, however, the field of view active in the implementation is set to 90 degrees. This large discrepancy is not noticeable when viewing a flat screen, and provides a good balance between the immersive, navigational and presentational concerns.

5.4 Program

This section discusses the program code and logic, realized with UnrealScript as well as the visual scripting tool Kismet.

The script system relies on inheritance from a common class, Object. All other script classes have this as a common ancestor. The system communicates with C++ code, but practically all logic and environment elements are done in the script. Thus, the most effective method of creating a new type of simulation is to extend already existing classes and override, or create new, functions within them. The Kismet tool uses sequence objects created in the script to facilitate visual scripting within the editor. This consists of connecting objects from the environment to the sequence objects and route the logic in the desired way. This project make use of both regular script and the visual script.

5.4.1 Defining the environment properties and player pawn

In order to define a new environment, GameInfo is extended to Tidsvandring. Here, various properties such as file name conventions, pointers to related classes and the PostBeginPlay() function are defined. The PostBeginPlay() function is called every time a new game is started. In this case, actors which need to be referenced later due to cue visibility toggling are searched for and pointers to them are stored here. Because of this, the search does not have to be executed every time the cues are toggled on or off. Because of the intended lack of objectives or other game rules, further logic is omitted from this class.

```
class Tidsvandring extends GameInfo;

var InterpActor trail;
var array<Emitter> sparklers;

// Called when a level has been loaded
event PostBeginPlay()
{
  local InterpActor foundInterp;
  local Emitter foundEmitter;

  //---------find trail--------------

  foreach AllActors( class 'InterpActor', foundInterp )
  {
    if(foundInterp.Tag == 'trail_arrows')
    {
      trail = foundInterp;
    }
  }

  //---------find sparklers--------------
  //Empty the list on each level load
  sparklers.Length = 0;

  foreach AllActors( class 'Emitter', foundEmitter )
  {
    //All emitters whose Tag is Sparkler
    //alternative: if(Left(foundEmitter.Tag, 8) == "Sparkler")
    if(foundEmitter.Tag == 'Sparkler')
    {
      sparklers.AddItem( foundEmitter );
    }
  }
}
```

FIGURE 5.17: The PostBeginPlay() function of the Tidsvandring class. To avoid searching every time the specific actors are needed, they are stored here at the beginning of play. This optimizes the performance of the system and ensures a more reliable response when these variables are changed.

In early development, the MyPawn class was extended from the general Pawn class. However, because of much common logic with the UTPawn (the bundled example game) related to physical materials for the playback of sounds from footsteps on different materials, the current implementation simply extends from UTPawn. It overrides just a few functions to control walk speed and removing effects from UTGame. However, the now-unused code are retained in comments in MyPawn.uc.

The MyPlayerController references MyInput, which controls keyboard input and in turn the actual visibility toggling of the visual cues. This is discussed below.

5.4.2 Controlling the visual cues

To secure a normal appearance of any object when cue effects are not needed, the visual cues are set to be toggled on and off using the *R* key. The key itself is bound within the file DefaultInput.ini. The various cues use different ideas in their implementation, and are discussed individually below.

5.4.2.1 Shimmer

A parameter was added to the shader of objects that implements shimmer. This parameter determines whether or not to pass the shimmer effect color information through to the final render. Through program code, controls were added to change this parameter, enabling the user to activate or deactivate the effect at will. The parameter is made visible to the script through an instance of the material instance constant class, which is derived from the compiled material itself. See Figure 5.18 for the script implementation.

```
/**
 * Various functions called via keybinds in DefaultInput.ini
 */
class MyInput extends PlayerInput;

exec function POIVisionToggle()
{
  local MaterialInstanceConstant matInstance[3];
  local float instanceValue;
  local int i;

  // Toggle GoVision parameter on all specified materialinstanceconstants

  matInstance[0] =
    MaterialInstanceConstant'Daves_utilities.Materials.Wall_Changing_MIC';
  matInstance[1] =
    MaterialInstanceConstant'Models.Materials.RockMossy_shimmer_INST';
  matInstance[2] =
    MaterialInstanceConstant'Models.Materials.Tympanon_INST';

  for( i=0; i<3; i++ )
  {
    matInstance[i].GetScalarParameterValue('GoVision', instanceValue);

    if(instanceValue > 0)
    {
      matInstance[i].SetScalarParameterValue('GoVision',0); //Turn off
    }
    else
    {
      matInstance[i].SetScalarParameterValue('GoVision',1); //Turn on
    }
  }
}
```

FIGURE 5.18: This is the part of the POIVisionToggle function inside MyInput.uc that controls whether the shimmer color information passes through in the shaders which implement it. The material instance constants are directly referenced from the packages they are saved to. Then, the code grabs the GoVision parameter which should be implemented in all of these shaders and toggles between them. This is thus easily extensible if more detailed control on each cue from several keys is desired. Code for other cues is omitted in this figure.

5.4.2.2 Sparklers and trail

The Emitters which are cues (and not smoke, fire or other environmental effect also implemented) are each tagged in the environment as “Sparkler”. Thus, no change in actual code is needed if additional sparklers are added later in development. The pointers previously saved in PostBeginPlay() are referenced and the emitters (and trail) are toggled on and off. The actual toggling is only a change in visibility through the bHidden flag which most placeable actors have in common.

```
class MyInput extends PlayerInput;

exec function POIVisionToggle()
{
        //Toggle sparkler emitters

        foreach Tidsvandring(WorldInfo.Game).sparklers(sparkler)
        {
                if(sparkler.bHidden == false)
                        sparkler.SetHidden(true);
                else
                        sparkler.SetHidden(false);
        }
}
```

Figure 5.19: The part of the toggling code responsible for a simple change in visibility of the sparklers. The code for the trail is almost identical and is therefore omitted here.

5.4.2.3 Map

A simple map was added that points out the most important landmarks in the environment. The intention was also to transform and draw the user’s current location on top of this map. After some trial and error, it was determined that this would require the addition of a heads-up display system. While possible, the complexity this would add to the program as a whole was not considered worth it, especially because the head-up display needs to be called every simulation tick. This would add quite a bit of performance overhead.

In later versions of the UDK this will likely be easier to do using the upcoming Scaleform interface system, where interface components themselves are able to dynamically draw on the screen.

5.4.3 Menu system

The menu system consists of a small extension to the UIScene class of the UDK to intercept input before the default code handles it, so that the player can not escape the menu but needs to press a certain choice on the menu. The rest of the logic is handled with Kismet. There are some small Kismet sequences which are contained within the specific button widgets on each menu page. These send external events to the Kismet sequence of the *persistent level* (the main level). The events are handled further there, including accounting for a number of special cases. Some are linked to start matinee sequences such as a demo flight loop which flies through most of the environment.

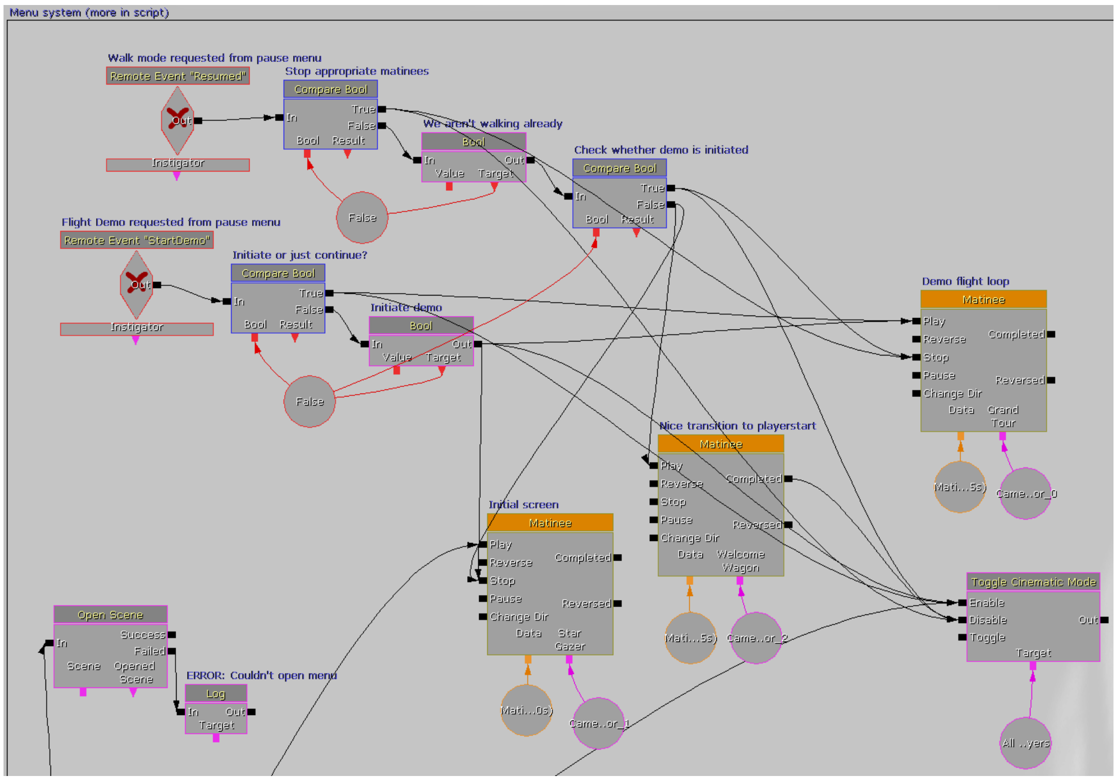


FIGURE 5.20: This is a part of the menu system done with visual scripting. The sequence receives external events and handles special cases, e.g. depending on if the user is already walking around or if the demo flight has been started. Smaller sequences of the menu system, not shown here, are contained within the UI widgets.

5.4.4 Additional logic

5.4.4.1 Matinee

The matinee sequences control the camera. The sequences themselves are controlled by the menu system. To create a demo flight tour of the environment, a comprehensive matinee object was created. It consists of nearly 100 key frames, which interpolate between each others' values to create a smooth motion. The cues can still be toggled while the program is in the demo flight mode.

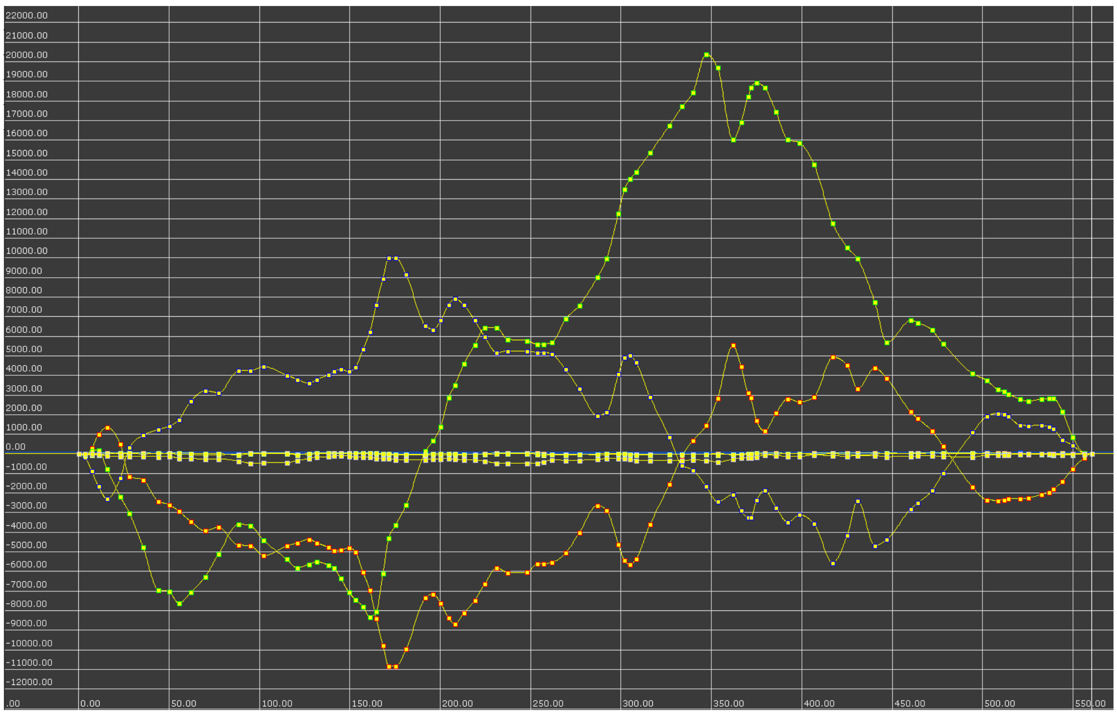


Figure 5.21: These are curves which have been created inside the matinee system. The individual values represent location and rotation of the viewport camera for each axis. Speed and acceleration are also controlled here.

There is also a sequence which does not control the camera but controls movable environment objects such as boats moving in the river. In order to extend the system, additional sequences would be quick to add.

5.4.4.2 Level streaming

The environment was initially made with a single persistent level. However, it became apparent that the large amount of models such as houses would cause the frame rate to go below acceptable levels (in total, there are over 1700 static mesh models in the environment). To solve this, the environment is split into several smalled blocks. These are then controlled individually either by direct control in Kismet, or with the use of level streaming volumes shaped for each partial level's requirements. If the user is outside a streaming volume, the associated level will unload. Because of some problems with delayed loading that could result in the user getting stuck in the environment, the volumes are set to only control rendering and not actual memory streaming. This increases the memory requirements because the levels are always resident. However, it results in much quicker loading of the levels because this is now reduced to a change in visibility. At any rate, the frame rate was significantly increased using this technique. It may still be rather low in some central areas. Further splitting of the partial levels could help here, but the performance was deemed acceptable for this particular implementation. It is worth mentioning that a specific level only contains the fjord water plane, and it is streamed out when the user is in the central areas (which is most of the time). This alone increased the frame rate by almost 100% in some locations. Some distance based culling is also employed on top of all this to stop rendering the smallest objects based on distance; this should not be noticeable at all from the user's point of view.

Finally, for many meshes additional level of detail instances were generated. This more than halved the triangle count rendered at a distance. Initially, the switch between the different level of details cause the instances to overlap when one of the meshes is culled and the other is made visible. This creates a brief flash because of this overlap during the transition. To solve this, the materials used on the meshes in question were instructed to use a *screen door fade* effect. This causes a dissolve effect instead of an instant switch between the instances, and is much less noticeable. It adds a few instructions to each shader. However, due to the amount of meshes benefiting from the level of detail functionality, the overall result was a gain of a few frames per second.

5.4.4.3 Localization

Menus and information pop ups, i.e. practically most text in the program, are read from easily modifiable configuration files. This has two main benefits: first, the text becomes very easy to change or correct. Second, a single setting can change the localization file which is used, changing the entire language of the program.

Chapter 6

Discussion

6.1 Research analysis

In chapter 4, many of the most prevalent techniques for visual cues and environmental re-creation guidelines were presented and analyzed in the context of virtual heritage reconstructions. There are some circumstances that may hinder a user's full utilization of these techniques. Hand-eye coordination is a skill which is important when controlling a virtual avatar while looking at the representation on a screen. Because the techniques are intended for all audiences, training in this basic skill may not be present in a large proportion of the user base. This represents a problem which is not handled in more detail in this thesis; visual cues facilitate an excellent guidance system, but the user may still require even more basic training to navigate a virtual world in a first-person perspective. However, cues can be used to solve this problem, at least in part, using a tutorial system with small-step lessons (this ought to be optional for users who feel comfortable using the control scheme).

The chapter ends with an evaluation that analyzes many of the cues from a more comprehensive perspective. An even more rigorous analysis of particular pros and cons, e.g. in table form, may seem natural. An argument against this is the fact that there are too many variables and circumstances that depend on each other to construct any meaningful analysis on such an atomic scale. Due to this kind of uncertainty, it is in this case deemed better to let readers use the knowledge as it stands to form it to their own special case requirements and design philosophy.

It may also seem natural to perform usability tests of techniques in this new setting on a live audience. However, it can also be argued that most of the techniques are already tested thoroughly, either in form of a previous research project or more often simply by their application in many of the highest quality computer and video games on the market. It is not a coincidence that the best games, in general, are also the most user friendly and easy to learn (although they may be difficult to master). Even though the virtual heritage setting is a new one for these techniques, their usability value is clear regardless of environment.

Cues may also be used intelligently to solve technical difficulties. A concrete example of this is an otherwise inconspicuous visual element such as fog which is normally used to provide a better overall immersion and visual impression as well as an environmental cue (e.g. a gate keeper mechanism, as discussed in chapter 4). In addition to these things it can be used to help the program's performance by occluding background elements which are invisible through the (dense) fog anyway[23].

Interestingly, the majority of virtual heritage projects does not actively or even not at all pay much attention to how they convey information. Many are only interested in the information itself and its implementation. There are some projects[39] that may be characterized as exceptions to this tendency, but there is a fine line between immersion and actual improvements in the way information is presented.

There has been a substantial amount of research towards providing better and more appropriate or intelligent lighting in virtual environments. For presentational purposes, many use techniques from cinema and other established fields. This research has in some cases been applied and tested in a virtual setting, typically with pre-rendered case environments or in a computer game engine. Some work has been done to try to incorporate a cinematic experience in games, especially useful in situations where the scene can not be entirely predicted at design time. For instance, an *Expressive Lighting Engine* (ELE) has been developed[19]. Based on cinematic theory, this was able to (automatically and in real-time) select the desired perceptual goals. This directed the players' attention to important scene elements in addition to changing the mood of the environment. For the testing purposes this was interfaced with the Unreal 2 Engine (from which the ELE could be tuned by any variable inside the engine) with successful results. While this is not directly related to the concept of cues, such real-time modification of environments based on the situation at any given time could certainly be useful in customizing the cues to adapt to the environment. Notwithstanding the potential improvement in user immersion, this is something that should be explored further due to the importance that cues should not be equally present at all times, as discussed in chapter 4.

In this project's goals, detailed in chapter 1, it is stated that the reader would gain knowledge of "usage- and learning-enhancing elements" and appropriate usage. In this thesis, they were dubbed "cues" and defined in various categories such as visual, environmental and interface cues. In the confinements set in chapter 4, the search space was limited to games and in some degree cinematic theory. There may still be other venues where other learning-enhancing elements can be found. This thesis has covered the most probable area where they might be found, but does not claim that area contains every answer. Still, every one of the techniques that were identified are major contributions to this young science. The discussions of new ways to utilize this knowledge encourage readers and researchers to "think outside of the box" and to seek new applications for previously unrelated topics.

6.2 Implementation analysis

In the introduction to chapter 5, it is argued in favor of the use of a game engine to create most kinds of virtual worlds without having to reinvent the wheel. It should be emphasized that most of the development work that a game engine does for you is the rendering system. The logic (depending on the environment) and practically all of the actual content will need to be created from the ground up. One should not think that using a game engine is particularly easy; it is no "game in a box" but it simply provides the most basic functions that most real-time virtual environments have in common. That being said, the basic functions will take a lot of resources and time to make properly, so the recommendation stands.

An implementation of a representative virtual heritage world was presented in the implementation. Some of the results obtained in chapter 4 were used in that implementation. They should be considered a proof of concept, and they show the feasibility of developing such elements with relatively few resources. In cases such as the trail, they can be made to be much more advanced or dynamic than presented. The game engine used here is also capable of these much more complicated implementations, so the presented environment should not be considered the limit of possibilities in that regard. The environment is easily extensible in this regard.

Because of the streaming system, there is no limit to the size of environments. The routes and general navigational design needs to be carefully considered to allow for seamless transitions. In the implementation, the loading locations are only approximated to achieve an acceptable performance. They are not fully optimized, which it should be for a commercial/finalized application. This is mostly a question of time spent optimizing the loading locations as well as splitting the content into smaller blocks as mentioned in section 5.4.4.2. In particularly demanding loading situations, techniques such as portals or gates (as discussed in section 4.2.2.1) can be utilized.

In section 5.3.5, it was mentioned that the use of assets from other projects will probably result in more conversion work than foreseen. This does not only concern light maps, but also other challenges such as the way meshes themselves are built (they require a special format for materials and sub-elements). This can result in the conversion process actually being slower than if the meshes were made from the ground up in the first place. Developers should be careful not to get stuck in a vicious cycle where external assets are being fixed and patched and it becomes too late to begin construction of higher quality content. This tendency indeed affected this environment, especially performance-wise because many resources from other projects were inappropriately optimized. It is recommended that many assets should be remodeled with better performance in mind if further development of this environment is considered.

Due to the relatively large re-use of the generic house models in the environment, many of the locations can look too similar to each other. This does not entirely follow the environmental recommendations from chapter 4, even though it does accentuate the effect of the visual cues that are implemented. A higher architectural variety is desired here, though this was not highly prioritized considering the scope of the test case. Still, dozens of large and small 3D models were created for this project in addition to the ones lent from other resources.

Overall, the original goals for the environment have been followed closely. It is a high-fidelity construction, and it includes numerous examples of visual cues. All of this created with limited resources, indicating that large budgets are not necessarily needed to make quality environments. The creation of this environment should increase the motivation of project decision makers to go ahead with a similar, professional endeavor.

Chapter 7

Conclusion and future work

7.1 Summary

The following is a short summary of the chapters in this thesis.

The introductory sections provided motivation and set the frame for the rest of the thesis. The motivation was based upon observations that most historic representations to date are not presented in a comprehensive way, and if they are, the representation lacks annotations or methods to help users (aside from domain experts) to navigate and understand the virtual world.

On previous work, examples were presented of past and on-going works in the field of virtual heritage. Where applicable, those projects' usage of visual cues (conscious or not) were discussed. It also acts as a representative selection of how virtual heritage has progressed and how the situation looked like at the time of this writing.

In the description of the test case, the major elements behind the re-creation of the implementation chapter were presented and given a more historic context. The central areas of the medieval city of Nidaros were set as a test bed for this implementation.

In identifying the actual techniques and accomplishing the research goal of the thesis, computer- and video-games were selected as the most promising source of material. Then commenced the definition and classification of *visual*, *environmental* and *interface* cues. These were analyzed, first with a general look on every one from their original context in games, then to their potential utilization in virtual heritage applications. Most of them were found to have a great potential within that kind of application. A final analysis set the cues in larger perspective and looked at how they can be used together in a meaningful way.

The implementation used the pre-determined test case elements to form a complete re-creation of a medieval environment. This placed them together in a larger context, using a few of the cues to represent information about locations in an inline fashion without pulling the user out of the experience. Care was taken to simulate an actual virtual heritage project's needs with a relatively detailed focus on how to effectively implement common environmental challenges such as landscape. Descriptions went in detail about the work within a game engine that an implementation of this kind of environment entails.

A discussion was provided which analyzed both the theoretical and practical results, tying up loose ends. Some additional observations were described, some of which went more in-depth about results obtained in the previous chapters. Potential or perceived shortcomings were discussed, more clearly showing the intention and background thought of decisions made during this work. It became clear that the project's original intentions have largely been fulfilled.

Finally, conclusions and suggestions for future work are presented in the following sections.

7.2 Conclusion

This thesis has presented a sizable number of concrete suggestions to present information in a better and more integrated way. The results suggest that use of common visual game design elements is an advantage when actively used in most presentations that wish to be accessible to a broad audience. The techniques can dramatically lessen frustration and increase the actual enjoyment users have when they experience the virtual world. The collection of techniques that are presented is practically unique, representing a never-before-seen focus on these kinds of usability and accessibility helpers, especially in a context seen outside of games.

The use of solutions such as a graphics- and game-engine shows the possibility of how it can greatly help the construction of virtual heritage environments which use these techniques. These results, and the raised awareness of the solutions they represent, indicate that their implementation would be very beneficial for any interactive virtual world project that aims for visualization and information representation rather than game objectives and challenges.

7.3 Future work

The following are thoughts on how to further improve and build upon the results presented in this thesis.

There are without a doubt still many other techniques that can be presented. Authors should be conscious about which elements that constitute *cues*, and which elements that should be classified otherwise - e.g. immersive, pure design considerations, game rules and objectives etc. As discussed in chapter 4, how to use cues when there is a game element involved (e.g. with the “dumbed down” discussion as a starting point) is a possible derivative topic. Also, if the virtual world has more dynamic elements (e.g. crowds of people walking around) than the one presented here, the effectiveness and appropriate usage of cues in such cases could be analyzed closer.

As discussed in chapter 6, a rigorous analysis of exact pros and cons may be unfeasible, but if a proper methodology can be identified this could be carried out to some effect.

New methods of combining cues should be identified. For example, a specific idea is to combine filtered vision with transparency of objects, an “x-ray” functionality. Here, objects from afar or in an inaccessible location could be selected and allow another cue, e.g. a trail, to show the way to the object. Another idea could be to use virtual overlays[2] as usable blueprints inside a game environment. Similarly ghost-images used in conjunction with a gesture functionality[68] could be a partial solution to the controller problem mentioned earlier.

Sounds, both ambient sounds and other sounds have not been prioritized in this project. Aside from contributing to the general immersive qualities of the product, sounds can also be used as cues. It should be considered an important part of any extension to this environment.

Evaluating users' experience using cues, i.e. usability testing, should be investigated further. This can be prone to error[58], so a good selection of test-subjects and -methods is important. It could be especially interesting to look more closely at differences in various age groups, and if cues extracted from games aimed at certain age groups are more effective towards their respective users. Differences between people who are computer proficient and people inexperienced with computers or games could also be significant.

Also as mentioned in chapter 6, there are potential problems with controlling from a first-person perspective - i.e. hand-eye coordination issues in untrained users. This is not investigated in this thesis, and should be looked at more closely. For example, console controllers may be easier for some users than the mouse-and-keyboard setup presented in this solution. Controller-less schemes where a camera analyze the physical movements of the users to control the program, e.g. as recently advertised Kinect for Xbox 360, Wii MotionPlus or PlayStation Move, are possible alternatives to look into whether they would fit this kind of environment.

There are several technologies which have not been utilized for this environment, but they are readily available. Virtual characters, i.e. people, can be controlled by using the UDK's crowd system. This should make the appearance of a busy city with people walking in the streets easier to achieve. There are skeletal animation systems in place, which combined with scripting can make detailed and believable characters (and animals). Procedural construction and placement of buildings are also supported in the UDK, where the designer creates rule sets which can use building parts to automatically construct entire cityscapes. As mentioned, memory streaming techniques can enable environments of virtually unlimited size. Memory streaming may also be used creatively, where for instance the entire area can change dynamically or based on input from the user, such as mentioned in section 4.3. One idea for this is to implement a time line slider with different years, where the user could see the city build and change as he or she interacts with the time line.

Cross-discipline collaboration efforts will be important in future development. Historians and archaeologists, in addition to the technical environment constructor, can continue work on implementation challenges to ensure a realistic and historically correct representation. The list of additional professions that are able to participate to make it even better is as long as it is in game development. These are not necessarily required, though they will all contribute to a higher quality project. Visual artists, animators, composers, story- and dialog-writers, programmers and sound engineers are some examples of this.

Bibliography

[1] Ernest Adams. *Fundamentals of Game Design*, page 163. New Riders Publishing, Thousand Oaks, CA, USA, 2009. ISBN-10: 0321643372, ISBN-13: 9780321643377.

[2] A.C. Addison. Emerging trends in virtual heritage. *Multimedia, IEEE*, 7(2):22 –25, apr. 2000.

[3] A. Amory, K. Naicker, J. Vincent, and C. Adams. The use of computer games as an educational tool: identification of appropriate game types and game elements. In *British Journal of Educational Technology, 30, 4*, pages 311–321, 1999.

[4] Anna Aresheva, Erlend Berne, Lars Andreas Dal, Fredrik Fossum, Ole Kristian Braut Grashei, and May Lill Morseth Løhre. Vår Frue Kirke, 2010. "Experts in Teamwork". Norwegian University of Science and Technology.

[5] Kristin Krogh Arnesen, Per Øyvind Solvang, Rolf-Henning Ulstein Klungsøyr, Sverre Simen Vadholm, and Øystein Kjærnet. Nidarneset i middelalderen, 2009. "Experts in Teamwork". Norwegian University of Science and Technology.

[6] Luigi Calori, Carlo Camporesi, and Sofia Pescarin. Virtual rome: a foss approach to web3d. In *Web3D '09: Proceedings of the 14th International Conference on 3D Web Technology*, pages 177–180, New York, NY, USA, 2009. ACM.

[7] Erik Champion. What is virtual heritage? `http://blogs.nyu.edu/projects/materialworld/2009/08/what_is_virtual_heritage.html`. Last checked September 5th 2010.

[8] Erik Champion. Applying game design theory to virtual heritage environments. In *GRAPHITE '03: Proceedings of the 1st international conference*

on Computer graphics and interactive techniques in Australasia and South East Asia, pages 273–274, New York, NY, USA, 2003. ACM.

[9] Erik Champion. Meaningful interaction in virtual learning environments. In *IE 2005: Proceedings of the second Australasian conference on Interactive entertainment*, pages 41–44, Sydney, Australia, Australia, 2005. Creativity & Cognition Studios Press.

[10] The Venice Charter. International charter for the conservation and restoration of monuments and sites. `http://www.international.icomos.org/e_venice.htm`. Last checked September 26th 2010.

[11] Ryan Clark. CrazyBump. `http://www.crazybump.com/`.

[12] New World Computing. Heroes of Might and Magic III: The Restoration of Erathia, 1999.

[13] Valve Corporation. Half-Life, 1998.

[14] Valve Corporation. Team Fortress 2, 2007. Developer's commentary.

[15] Miguel de Aguilera and Alfonso Mendiz. Video games and education: (education in the face of a "parallel school"). *Comput. Entertain.*, 2003.

[16] Victor DeLeon and Robert Berry. Bringing vr to the desktop: Are you game? *IEEE MultiMedia*, 7(2):68–72, 2000.

[17] Kate Devlin, Alan Chalmers, and Duncan Brown. Predictive lighting and perception in archaeological representations. In *UNESCO "World Heritage in the Digital Age" 30th Anniversary Digital Congress*. UNESCO World Heritage Centre, October 2002. Thanks to Alan Chalmers and Jassim Happa for providing the actual RGB flame color values.

[18] Michele D. Dickey. Engaging by design: How engagement strategies in popular computer and video games can inform instructional design. In *Educational Technology Research & Development*, volume 53, pages 67–83, 2005.

[19] Magy Seif El-Nasr, Joseph Zupko, and Keith Miron. Intelligent lighting for a better gaming experience. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems*, pages 1140–1141, New York, NY, USA, 2005. ACM.

[20] Blizzard Entertainment. World of Warcraft, 2005-2010.

[21] Blizzard Entertainment. Starcraft 2: Wings of Liberty, 2010.

[22] Kris Erickson. Editorial: Are games getting dumber? PS3 Informer - `http://www.ps3informer.com/playstation-3/games/editorial-are-games-getting-dumber-009365.php`, 2008. Last checked October 6th 2010.

[23] John Harold Feil and Marc Scattergood. *Beginning Game Level Design (Premier Press Game Development).* Course Technology PTR, 2005. ISBN-13: 978-1592004348.

[24] Alessandro E. Foni, George Papagiannakis, and Nadia Magnenat-Thalmann. A taxonomy of visualization strategies for cultural heritage applications. *J. Comput. Cult. Herit.*, 3(1), 2010.

[25] Bernard Frischer. The digital roman forum project of the cultural virtual reality laboratory: Remediating the traditions of roman topography. `http://www.frischerconsulting.com/frischer/FrischerWorkshopPaperIllustratedWeb_test.html`. Last checked October 2nd 2010.

[26] Bernard Frischer, Diane Favro, Dean Abernathy, and Monica De Simone. The digital roman forum project of the UCLA Cultural Virtual Reality Laboratory. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 34(5/W10), 2003.

[27] 2K Games. BioShock 2, 2010. Image from http://gamingthreshold.wordpress.com/2010/02/22/bioshock-2-power-to-the-people-weapon-upgrades/. Screen shot retouched to improve clarity.

[28] Irrational Games. BioShock, 2007.

[29] James Hastings-Trew. Reproducing real world light. `http://planetpixelemporium.com/tutorialpages/light.html`. Last checked September 24th 2010.

[30] N. Haval. Three-dimensional documentation of complex heritage structures. *Multimedia, IEEE*, 7(2):52 –55, apr. 2000.

[31] Evan Hirsch, Rick Stringfellow, Paul Amer, Brien Goodrich, Jamie Marshall, and L. Cliff Brett. Crossing the line: moving from film to games and possibly back. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, New York, NY, USA, 2007. ACM.

[32] Leah Hoffmann. Learning through games. *Commun. ACM*, 52(8):21–22, 2009.

[33] Vitenskapsmuseet i Trondheim (Middelalderutstillingen). Miniature model of Nidaros.

[34] Epic Games Inc. Unreal Development Kit. `http://www.udk.com/features`. Last checked October 2nd 2010.

[35] Epic Games Inc. Unreal Technology. `http://www.unreal.com/technology.php`. Last checked October 2nd 2010.

[36] Interactive Data Visualization Inc. SpeedTree. `http://www.speedtree.com/`.

[37] IT3402. Screen layout, gui-elements, navigations structure. NTNU course - User Interface Design.

[38] Jeffrey Jacobson, Lowry Burgess, Michael Darnell, Kerry Handron, Robyn Gillam, Lynn Holden, and Vashti Germaine. Virtual egyptian temple. `http://publicvr.org/html/pro_egypt.html`. Last checked September 26th 2010.

[39] Jeffrey Jacobson, School Of Information Sciences, and Lynn Holden. The virtual egyptian temple. In *World Conference on Educational Media, Hypermedia & Telecommunications (ED-MEDIA*, 2005.

[40] Jeffrey Jacobson and Jane Vadnal. The virtual pompeii project. In Griff Richards, editor, *Proceedings of World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2005*, pages 1644–1649, E-Learn 2005–World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education, October 2005. AACE.

[41] Colin Johnson. Computer visualisation of dudley castle c1550 - a virtual tour by royal appointment. `http://www.exrenda.net/dudley/`. Last checked September 19th 2010.

[42] Colin Johnson. The kitchen - dudley castle c1550. `http://www.exrenda.net/dudley/kitchen.htm`. Last checked September 22nd 2010.

[43] Sjoerd "Hourences" De Jong. The hows and whys of level design - second edition, 2008.

[44] Statens Kartverk. N1-map trondheim (equidistance 0.5 m) converted to gray scale height map. Map package NDData437 - map 32_1601Hoyde. UTM zone 32 based on EUREF89/WGS84.

[45] Andreas Larsen, David Svånå, Georgy Ushakov, and Ole Nordland. Prosjektrapport for bibliotektomta - middelalderens nidaros i virtuell virkelighet, 2009. "Experts in Teamwork" (only Norwegian version available). Norwegian University of Science and Technology.

[46] Ubisoft Montreal. Assassin's Creed II, 2009.

[47] Unreal Developer Network. Unwrapping meshes for light-maps. Last checked October 18th 2010.

[48] NOAA. Solar position calculator. `http://www.srrb.noaa.gov/highlights/sunrise/azel.html`. Last checked September 24th 2010.

[49] Blizzard North. Diablo II, 2000.

[50] Virtual World Heritage Laboratory University of Virginia. Rome Reborn. `http://www.romereborn.virginia.edu/`. Image from `http://www.romereborn.virginia.edu/gallery-current.php`. Last checked October 2nd 2010.

[51] OpenSceneGraph open-source contributors. Openscenegraph. `http://www.openscenegraph.org/projects/osg/wiki/About`. Last checked October 22nd 2010.

[52] PublicVR. Virtual theater district of pompeii. `http://publicvr.org/html/pro_pompeii.html`. Last checked september 26th 2010.

[53] Nayan Ramachandran. Opinion: Boss design - trial & punishment. `http://www.gamasutra.com/view/news/19005/Opinion_Boss_Design__Trial__Punishment.php`. Last checked October 24th 2010.

[54] Donald H. Sanders. Why do virtual heritage? `http://www.archaeology.org/online/features/virtualheritage/`. Last checked September 19th 2010.

[55] A. Fleming Seay, David M. Krum, Larry Hodges, and William Ribarsky. Simulator sickness and presence in a high field-of-view virtual environment. In *CHI '02: CHI '02 extended abstracts on Human factors in computing systems*, pages 784–785, New York, NY, USA, 2002. ACM.

[56] Jorge Ordóñez Serrano. Virtual guide for a virtual heritage environment. Master's thesis, Norwegian University of Science and Technology and Universidad Politécnica de Madrid, 2004.

[57] EA Redwood Shores. Dead Space, 2008. Screen shot from `http://jm.monkk.com/2009/02/15/game-dead-space-first-thoughts/`.

[58] Mel Slater. Measuring presence: A response to the witmer and singer presence questionnaire, 1999.

[59] Michael James Stokes. Multimodal behaviour generation frameworks in virtual heritage applications. Master's thesis, Norwegian University of Science and Technology, 2009.

[60] Bethesda Game Studios. The Elder Scrolls III: Morrowind, 2002. Image from `http://www.rpgfan.com/reviews/tribunal/Tribunal.html`.

[61] Bethesda Game Studios. The Elder Scrolls IV: Oblivion, 2006.

[62] Bethesda Game Studios. Fallout 3, 2008.

[63] Lionhead Studios. Fable II, 2008.

[64] Pixar Animation Studios. Visual storytelling through lighting. `http://www.frankwbaker.com/visual_storytelling_through_lighting.htm`. Last checked September 28th 2010.

[65] Unity Technologies. Unity 3 engine. `http://unity3d.com/unity/engine/`. Last checked October 2nd 2010.

[66] Marcel Vijfwinkel and site contributors. CGTextures. `http://www.cgtextures.com/`. Accessed throughout 2009-2010.

[67] A. Weis, J. Jacobson, and M. Darnell. The virtual theater district of pompeii. In *Computer Applications in Archaeology (CAA)*, April 2010.

[68] Sean White, Levi Lister, and Steven Feiner. Visual hints for tangible gestures in augmented reality. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, ISMAR '07, pages 1–4, Washington, DC, USA, 2007. IEEE Computer Society.

[69] Wikia. Fable 2 features. `http://fable.wikia.com/wiki/Fable_II`.

[70] Wikipedia. Multiple articles regarding the history of nidaros. `http://en.wikipedia.org/wiki/Nidaros`, `http://en.wikipedia.org/wiki/Trondheim`, `http://en.wikipedia.org/wiki/Nidaros_Cathedral`. Last checked October 13th 2010.

[71] David Wildgoose. The bread crumb trial: Why fable 2 should be wary of half-baked design choices. http://unifiedammo.wordpress.com/2008/05/26/the-bread-crumb-trial-why-fable-2-should-avoid-half-baked-design-choices/, 2008. Blog: Unified Ammo - Mostly about videogames. Last checked October 6th 2010.

Appendix A

Program instructions

A.1 System requirements

The implementation was developed and tested on a computer of the following specifications:

Windows 7 64-bit
Intel Core i7 CPU 920 @ 2.67GHz (8 CPUs), 2.8GHz
12 GB system RAM
NVIDIA GeForce GTX 285

To experience the implementation's standalone application as intended, the minimum system requirements is approximately (*not tested*):

Windows XP SP2
2.0+ GHz processor
4 GB system RAM
NVIDIA GeForce GTX 285 or similar performing Shader Model 3 compatible graphics card

A.2 Installation instructions

A.2.1 Executable program

The final compiled program is contained in an installation file, called "UDKInstall-Tidsvandring.exe". Simply execute this file to install the program. You get a choice of where to install to. A shortcut for starting the program will be generated. IMPORTANT: Configure the appropriate screen resolution for your system before starting (see section A.3 for how to do this).

A.2.2 Installation to UDK (optional)

All uncompiled files are stored in the \source directory of the attachment to this thesis. This is further split into two categories. The first, in \assets, is the original files such as 3ds max model files and texture image files which can useful to future developers. The second, in \udk, is the uncompiled packages as well as script source.

To install into to the UDK in order to make a fully functional development installation as it appeared before compiling , only the files in the \udk folder are needed. Install the UDK September 2010 version first (the September 2010 UDK installation is also provided in the files accompanying this thesis). Open the editor once to ensure it starts, then close it completely. Copy every folder from the \udk folder in the thesis files to the installation directory (default: C:\UDK\UDK-2010-09). The files is arranged correctly and should merge with the installation. If asked to replace any files, accept.

Now open the Unreal FrontEnd program with administrator rights. Go to the cooking tab and enter "TIDSV-Main" (without the quotes) into "Maps to cook". Click the small arrow next to the "Make" button, then choose "Full Recompile". When this is completed, in order to open the environment, start the editor and open "TIDSV-Main.udk". TIDSV-Main is the persistent level, i.e. the one controlling the other streaming levels. This is always the one which should be opened regarding this environment; it will automatically load the other referenced levels.

Note that on first load only, the engine may need to compile shaders. This may cause several minutes without responsiveness from the computer.

See section A.4 if you are a developer and plan to use another version of the UDK to continue development.

A.3 Configuration

The screen resolution can and should be configured to accommodate different systems. The default is full screen mode with 1920x1080 screen resolution. To change this in the installed program, first exit the program if it is open. Then go to the folder where the program is installed. The default directory is C:\UDK\Medieval Nidaros 1300-1350. From there, navigate to \UDKGam\Config\UDKEngine.ini and open it in a text editor such as Notepad. Search for "ResX" to go directly to the relevant setting (it is located under the [SystemSettings] category). Here, change ResX and ResY to your system's ideal screen resolution (preferably wide screen format). Full screen mode can also be disabled to make the program run in windowed mode. To do this, change the Fullscreen variable (also under [SystemSettings]) to False (default is True). Save the file and restart the program.

An emergency console command may be used go allow for walking through walls and fly. Press TAB then type *ghost*, then press enter. You can now fly, with collision disabled. To return to a normal state, do the same except this time type *walk*. This is obviously intended for debugging purposes only.

The timescale may be changed, allowing things such as very fast (or very slow) movement or to speed up matinee sequences. Press TAB then type *slomo NUMBER*, where NUMBER is a floating point indicating the time scale. When NUMBER = 1, the time flows as intended. When NUMBER = 1.5, the flow of time is increased by 50%.

The field of view may be changed from its default 90 degrees. Press TAB then type *fov NUMBER*, where NUMBER is an integer between 1 and 170. This may help on very large or very small displays but, as discussed, may incur symptoms of nausea if exaggerated. It is best left at the default in almost all circumstances.

A.4 Known problems

Using the Alt-TAB program switch functionality, or the “windows key”, will often cause major visual artifacts when returning to the program. It is recommended to not use this function.

If the program is run in anything else than a wide screen format, part of the interface (menus) may appear clipped depending on the screen resolution. To avoid this if a wide screen display is not available, consider running in windowed mode using a wide screen aspect ratio. See section A.3 for how to change this.

In an earlier version of the environment, the streaming system unloaded partial levels when the viewpoint was outside of their associated streaming volume. During the demo flight, the viewpoint will likely go outside of the streaming volumes that the user is located. This means that if the demo was interrupted when this is the case, the level would not be there for a few seconds after the user had regained control. If the user moved during this time, there was a chance that he or she may have gotten stuck inside a building with no way to get out. This is probably not the case any more because the streaming volumes were set to not unload from memory, but merely to hide and unhide from the renderer. In rare cases, users may still encounter the problem, although this was not a problem under final testing.

The streaming problem may also come into play if the user moves excessively fast through the environment using the shift key modifier; a partial level that has not yet been loaded may be reached and this situation may result in the same stuck-problem mentioned above. This is probably even more unlikely to encounter in the current version, however, there may still be a slight possibility.

If the system is expanded upon, developers should be aware that the UIScene (menu system) currently employed for this environment is not supported by UDK versions after the September 2010 version. The Scaleform UI system has replaced this. So, before transitioning to a more recent version of the UDK, the UIScene menu system should be removed. It will then need to be re-created in the new version.

Also if the system is ported to a newer version of the UDK, care should be taken to manually merge the relevant .ini files with the new ones. Many settings change between each version, and an automatic merge or an overwrite will cause problems (even if not immediately apparent).

Appendix B

Pictures

B.1 Screen shots

The following figures are a few screen shots of the finalized environment. Additionally, movie clips of the environment recorded while running the program are among the files attached to this thesis.

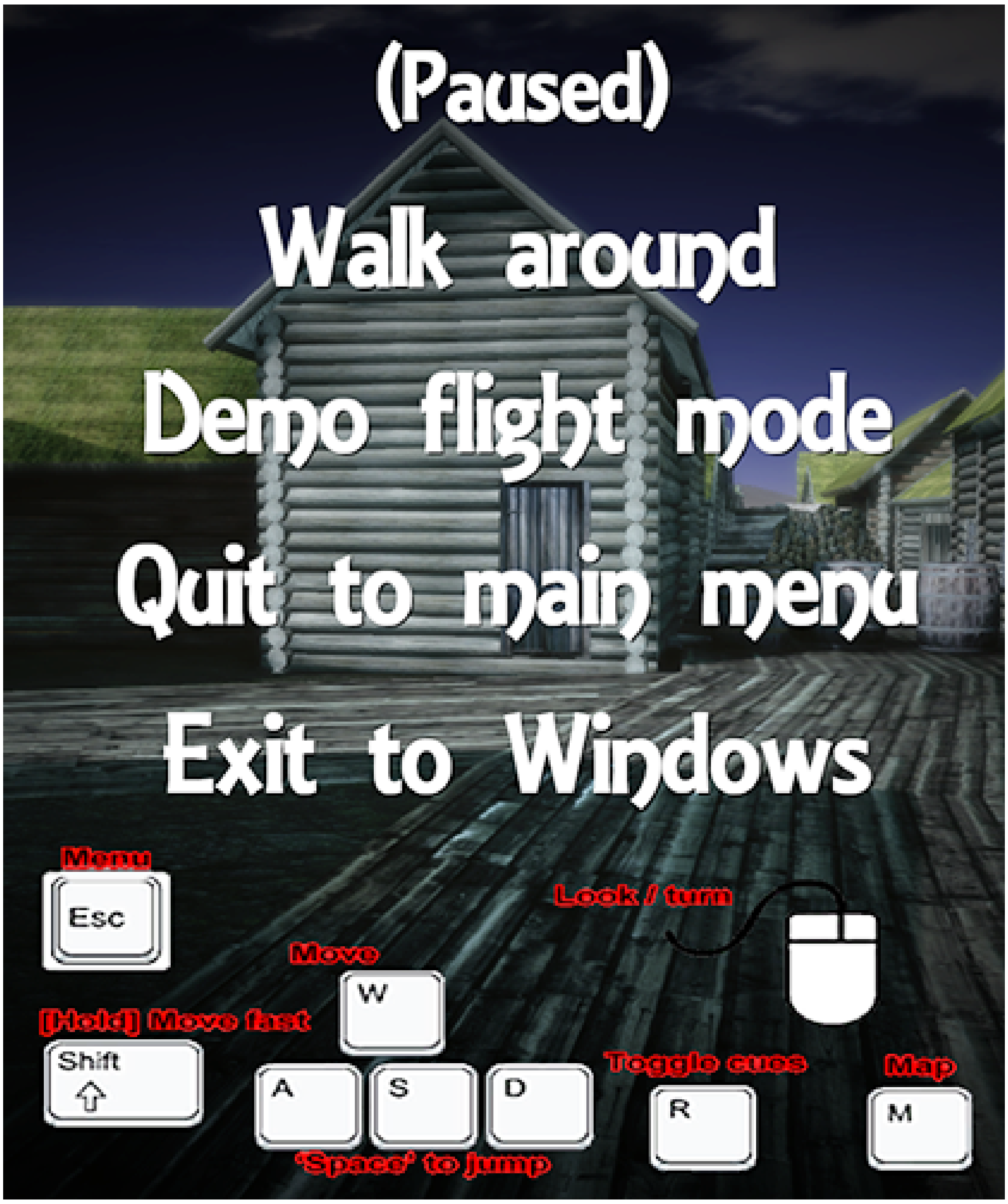


FIGURE B.1: The pause menu. A short explanation of the keyboard and mouse controls is displayed here.

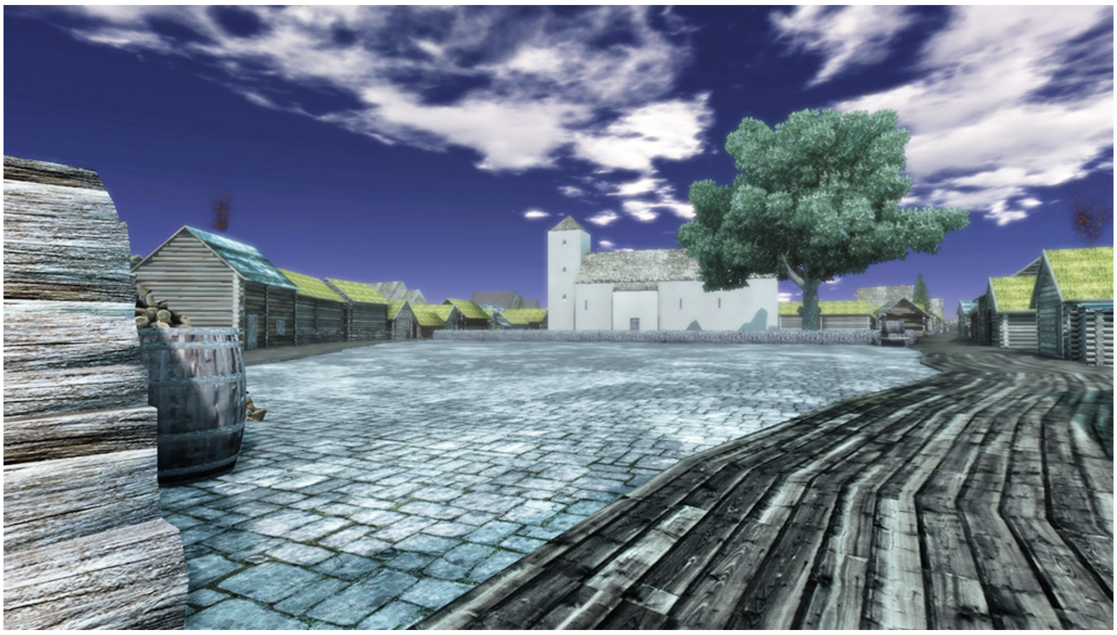


Figure B.2: Viewing north at the beginning of Kaupmannastreti, across an open public space.

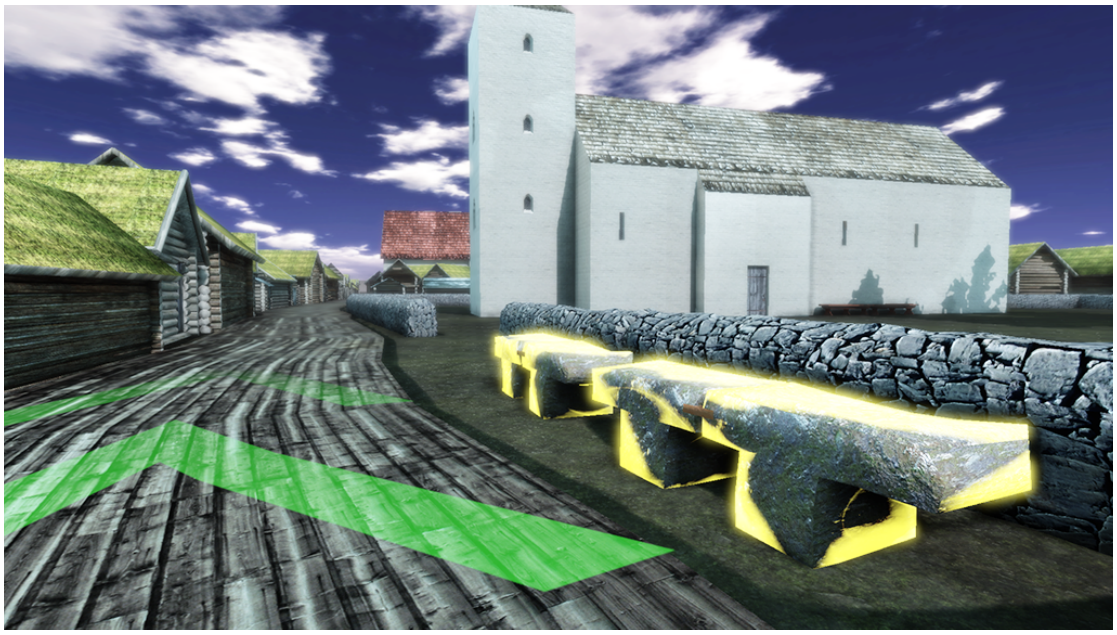


Figure B.3: Here, the trail can be seen pointing along the suggested route. Also, the shimmer cue can be seen on a couple of nearby benches.

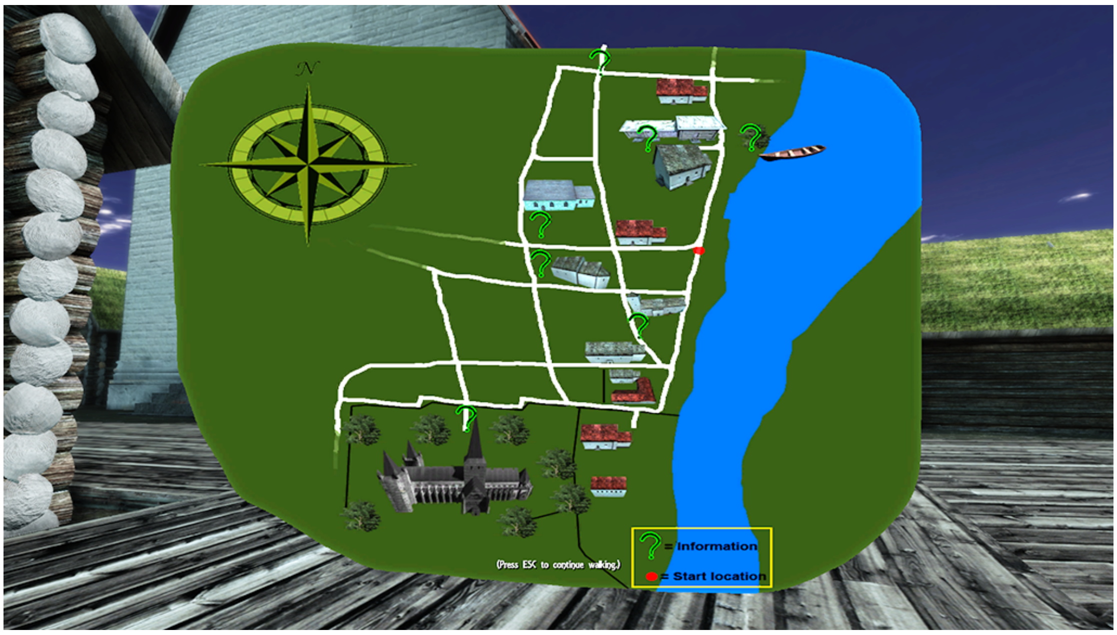


Figure B.4: This simple map can be displayed at any time. It points out the largest landmark buildings and the streets.

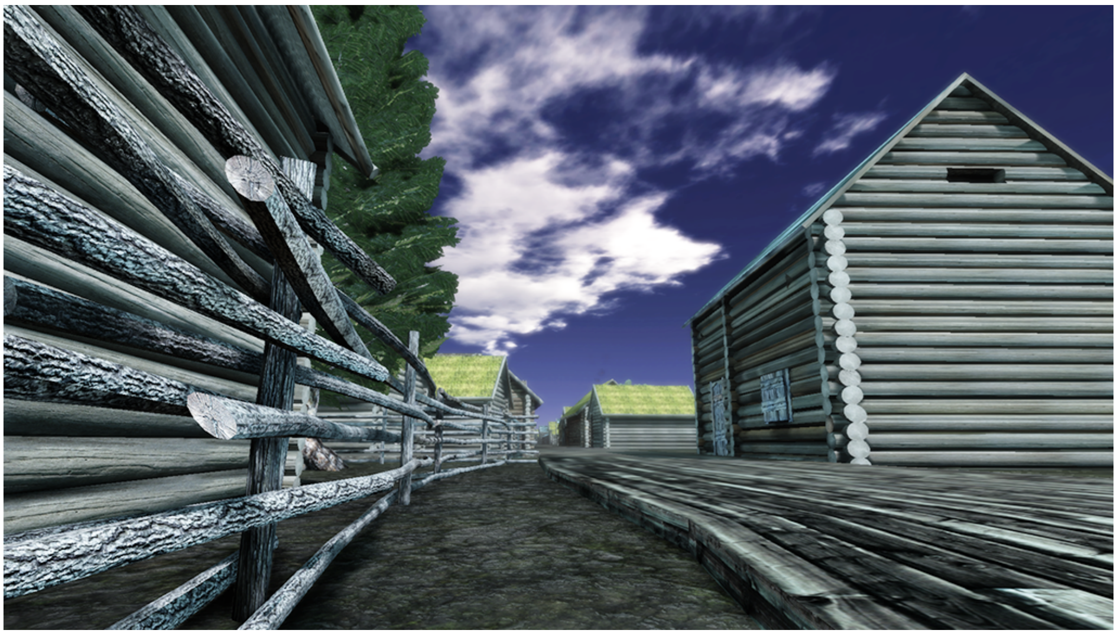


Figure B.5: A wooden fence that runs along a street.

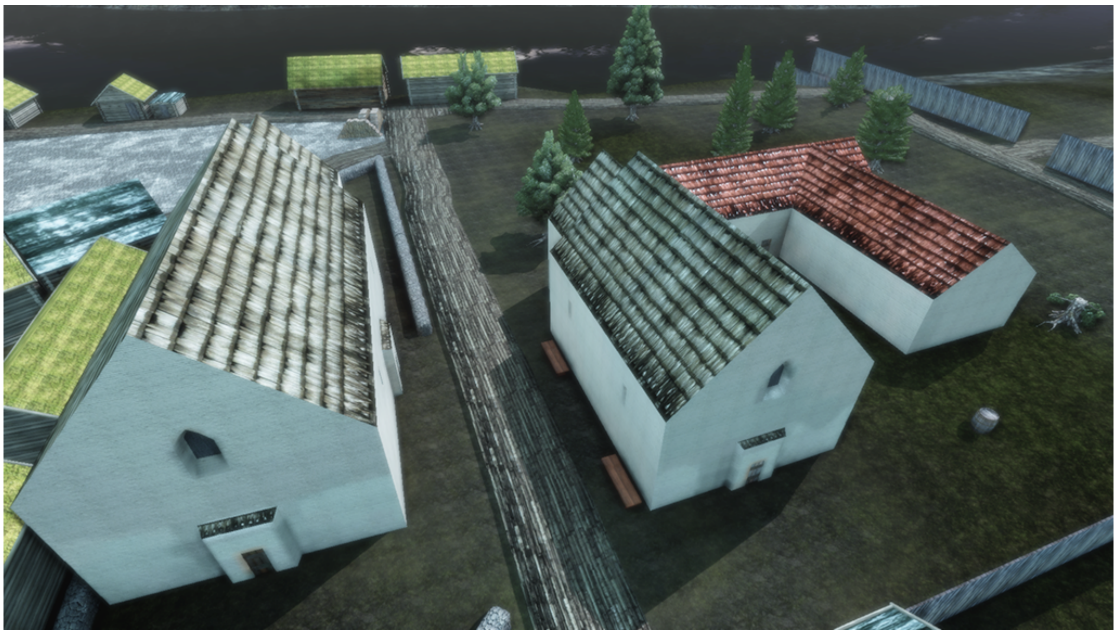


FIGURE B.6: Churches with attached building complex.

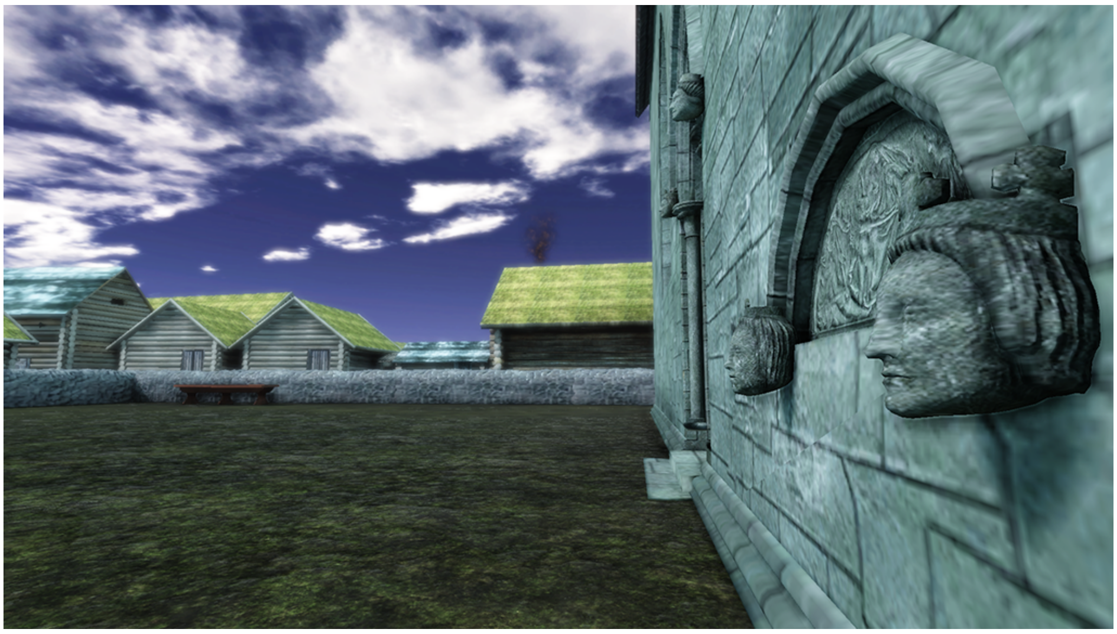


FIGURE B.7: The tympanum stone on the side wall of Vår Frue Kirke.

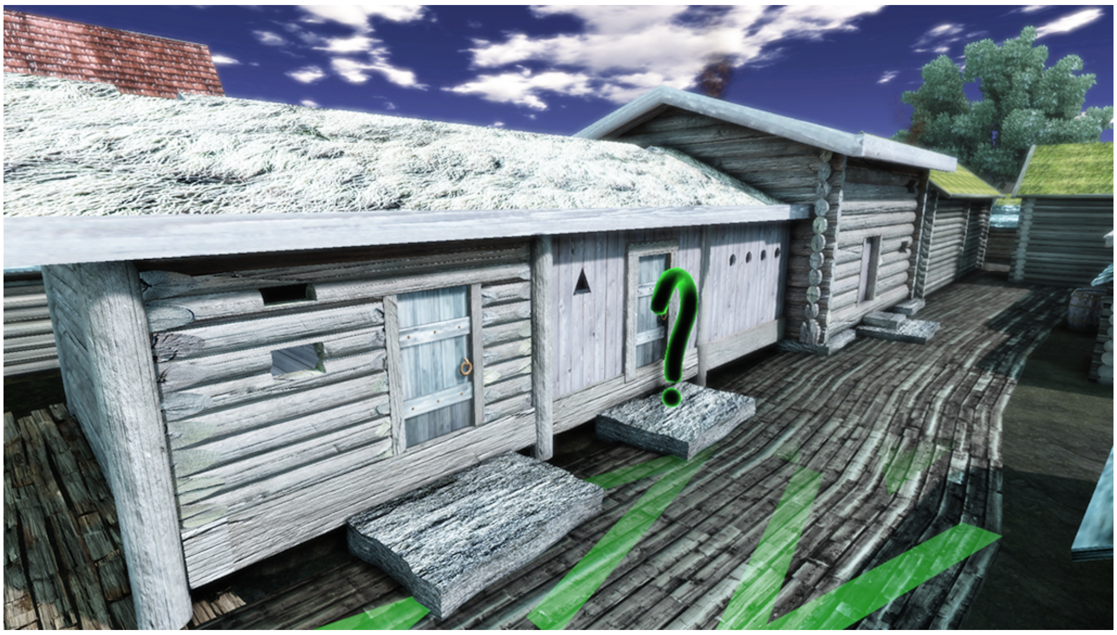


Figure B.8: The question mark will, when touched by the user, display information about this special house. The house is from the preliminary project.

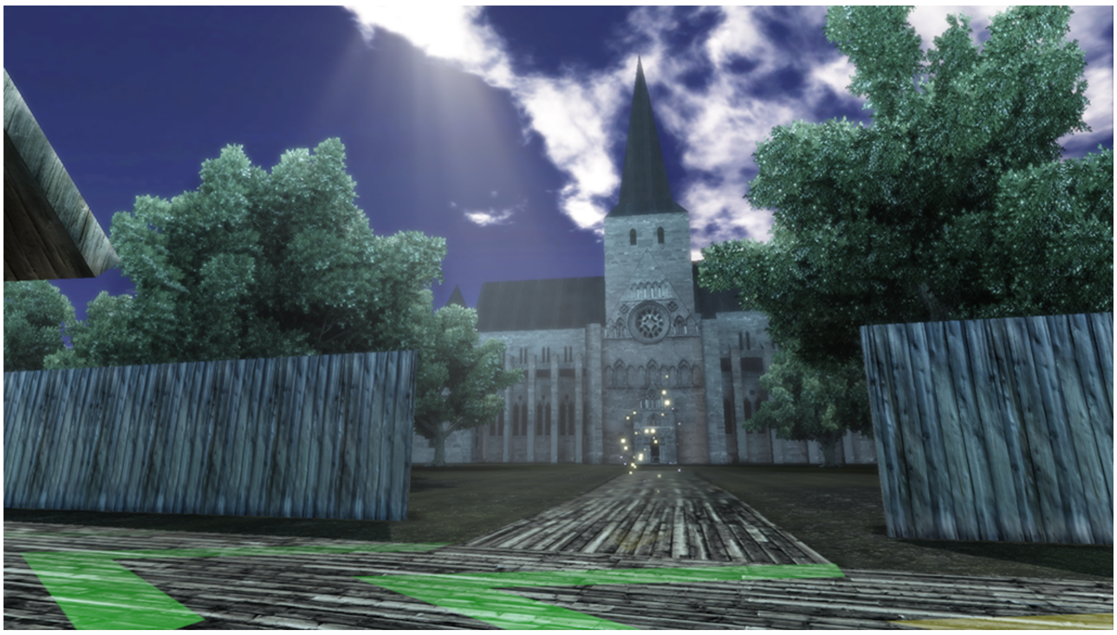


Figure B.9: Looking at the entrance to the Nidaros cathedral. The sparkler cue can be seen in front of it.

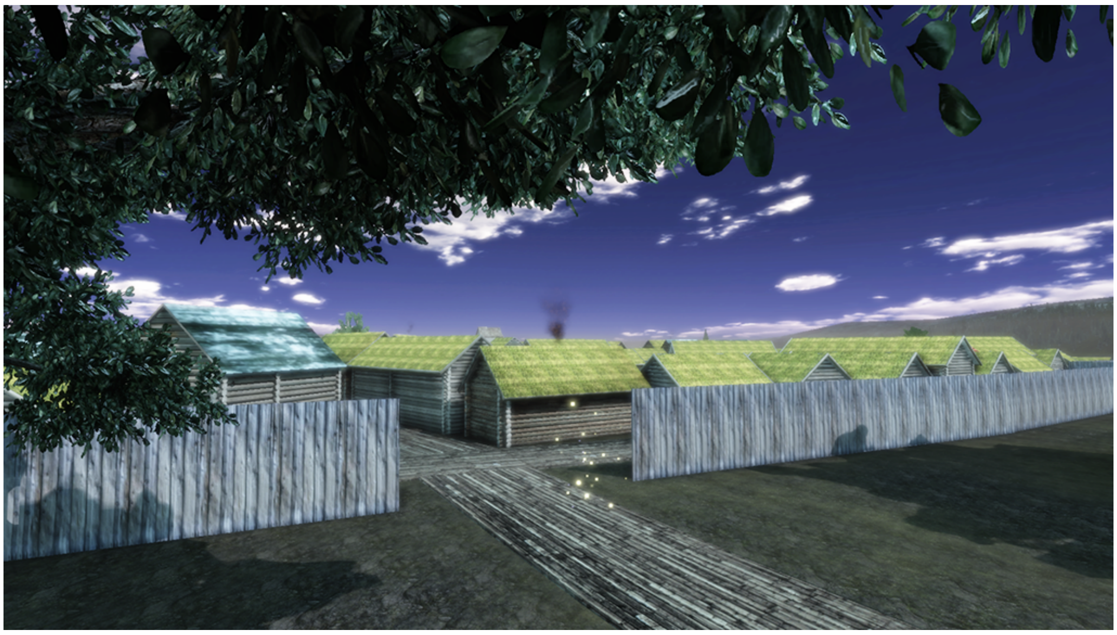


FIGURE B.10: A perspective of the cathedral entrance seen from under one of the trees.

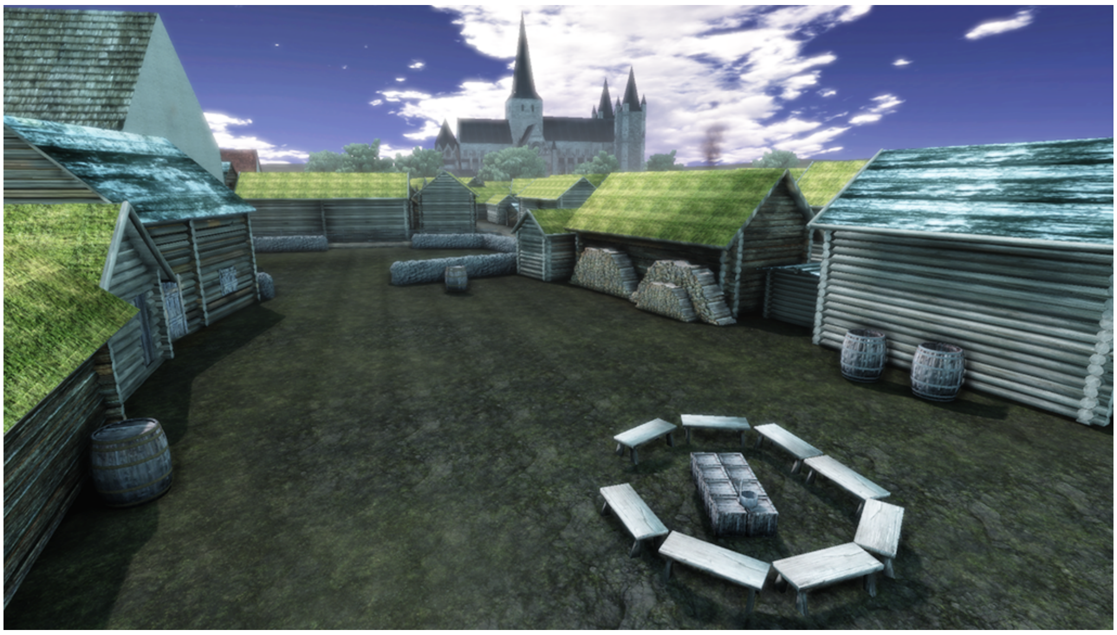


FIGURE B.11: A common area overlooking the cathedral in the background.

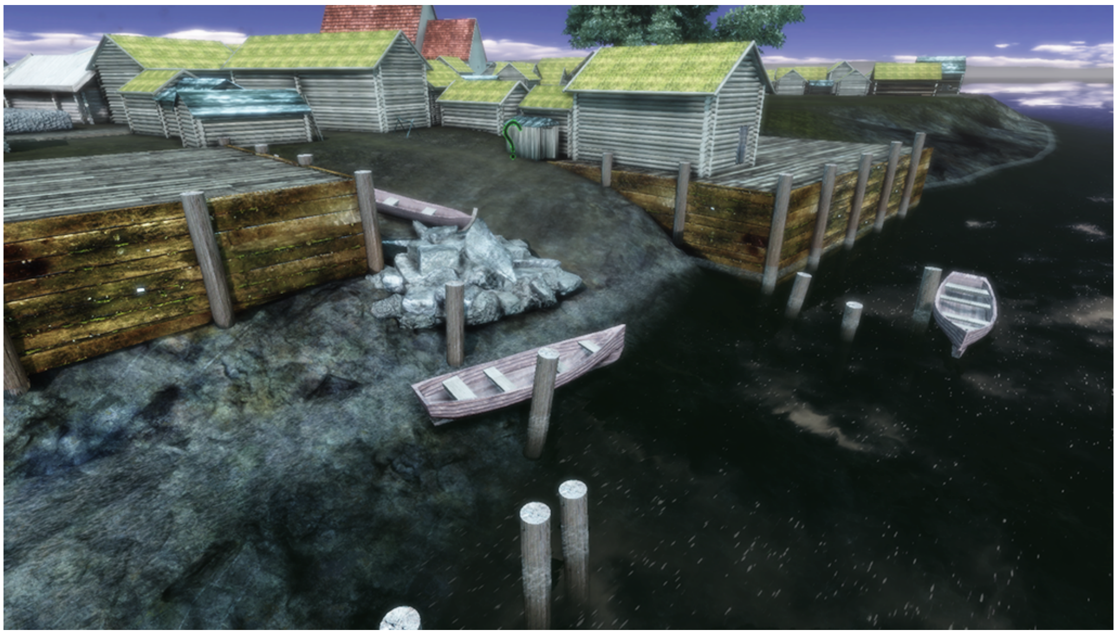


Figure B.12: A harbor with a couple of boats secured to some river poles.

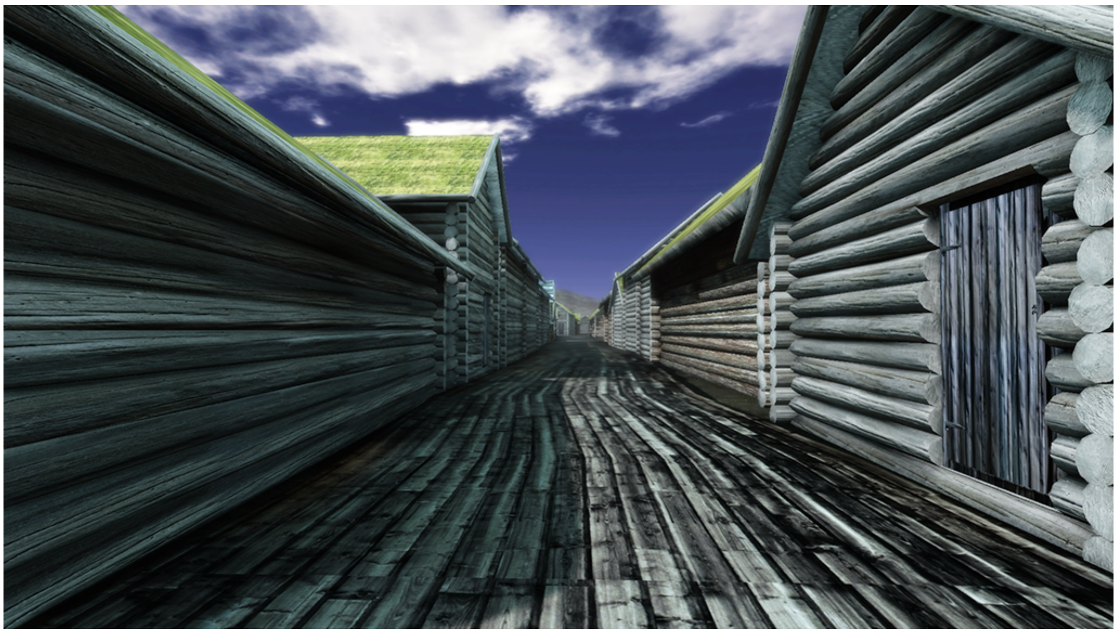


Figure B.13: View down an arbitrarily chosen street.

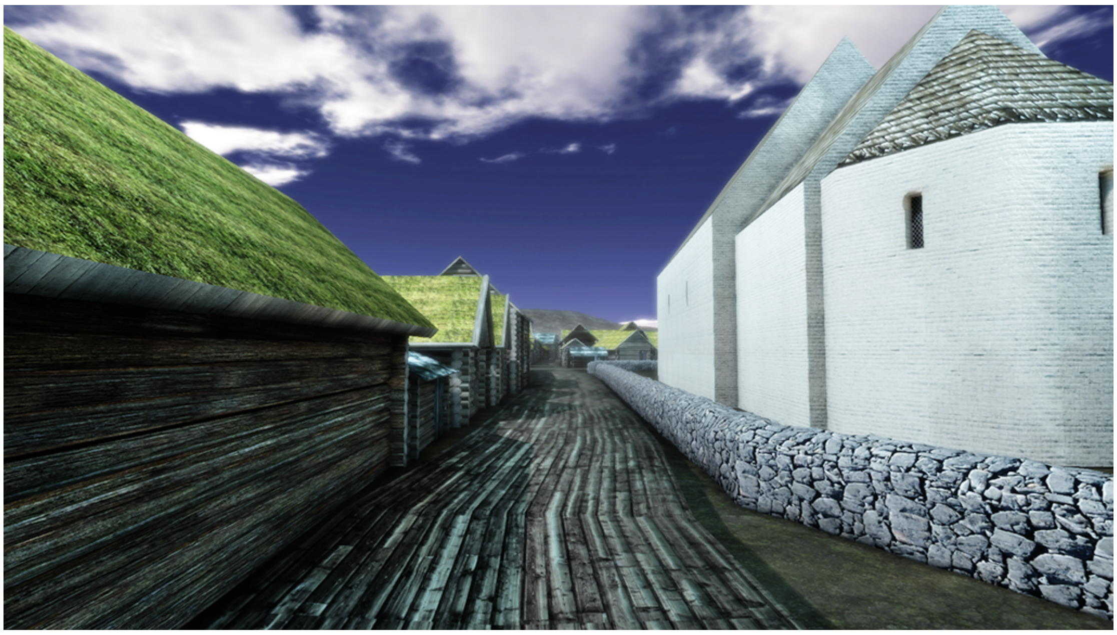


Figure B.14: View down a street, with one of the many churches on the right side.

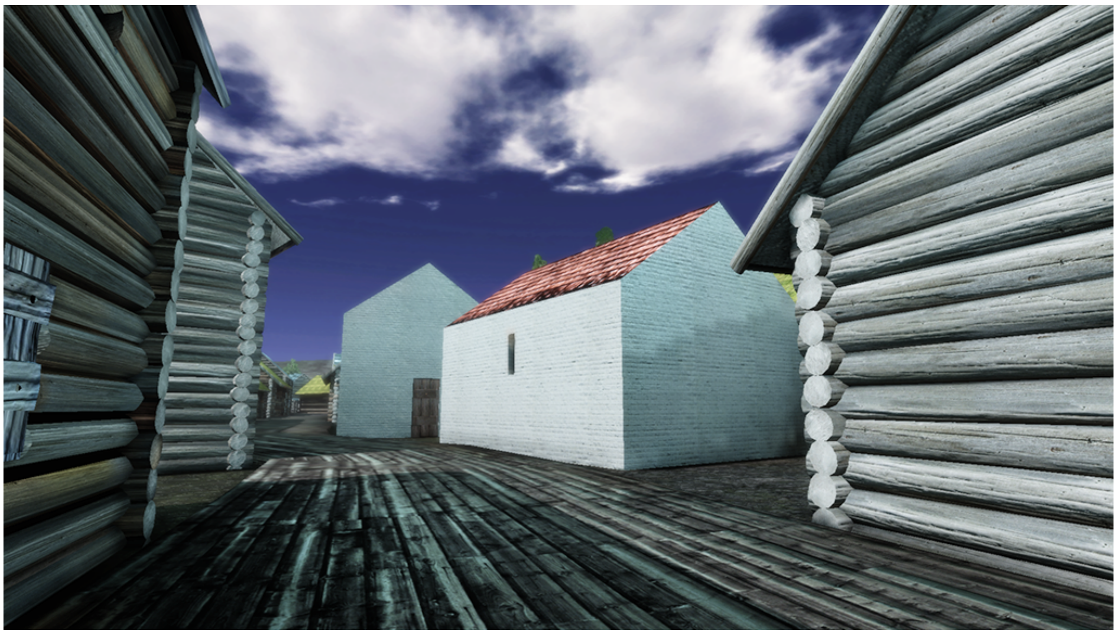


Figure B.15: A street with some unusual-looking houses.

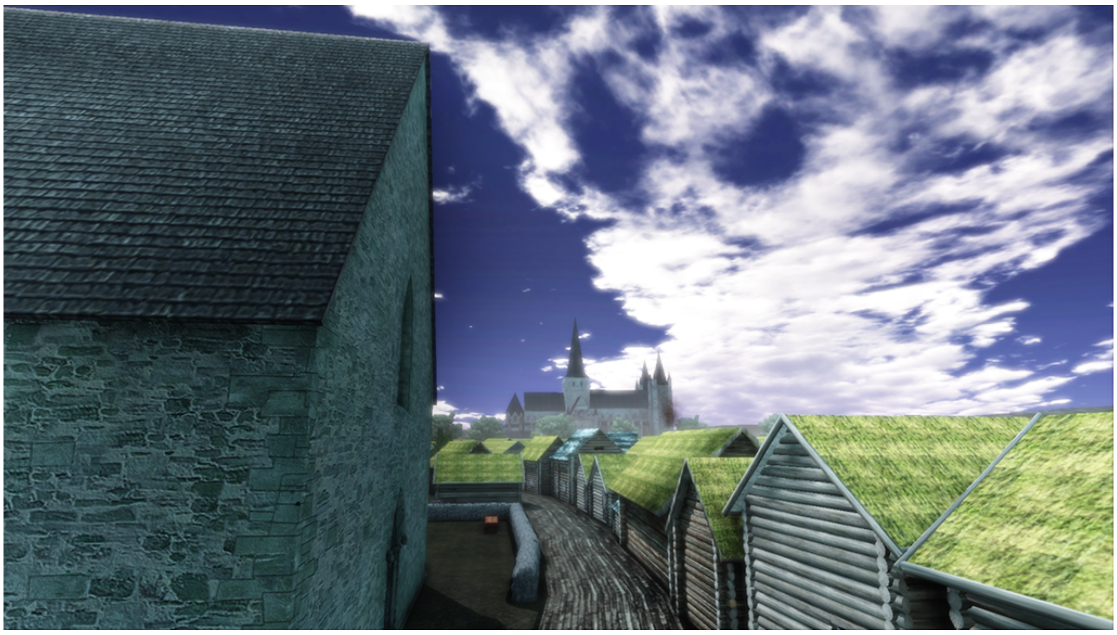


FIGURE B.16: View from Vår Frue Kirke towards the Nidaros cathedral.

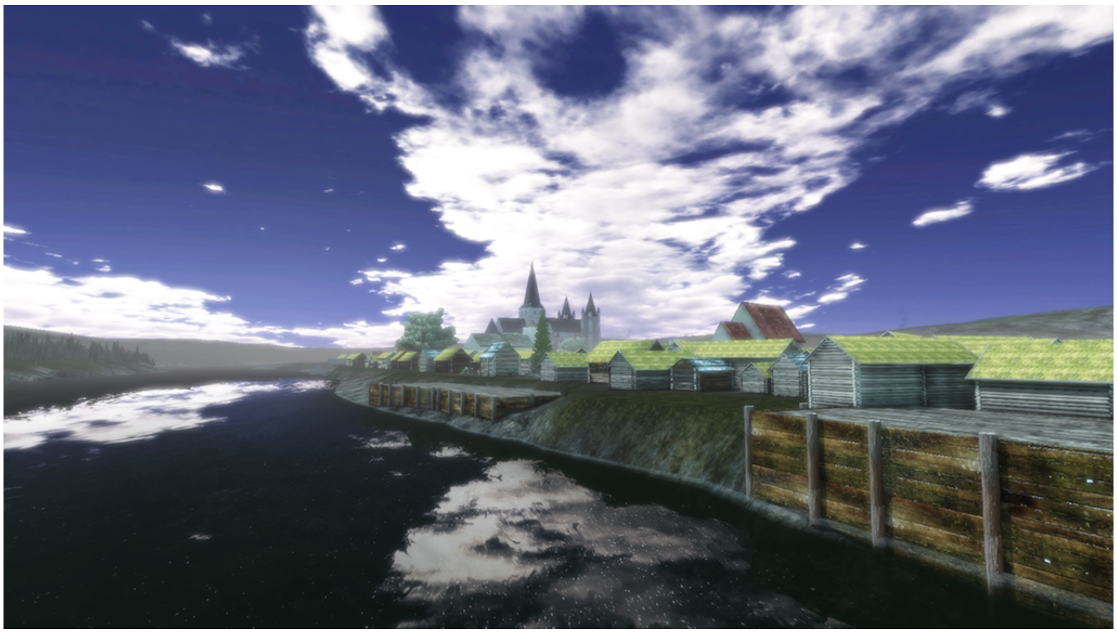


FIGURE B.17: An overview from the river.