Spike Island - An Immersive VR Experience

A thesis submitted to the National University of Ireland, Cork
in partial fulfilment of the requirements for the degree of
Master of Science in Interactive Media
in the Faculty of Science

October 2022

Karthika Cheeniyil
Department of Computer Science

Contents

List of Tables

List of Figures

Abstract

This thesis presents the design and creation of a 3D virtual environment of Spike Island, a former fort and notorious prison in Cork harbor, which is often described as "Ireland's Alcatraz" due to its dark and dreadful history.

The project's significant aspect is its use of web technologies to create a VR application, with a particular emphasis on employing a web-based gaming engine to implement the experience on the Web.

The Mitchel Fort and the structures inside it are the experience's main emphasis, and the two cells within it have been designed and developed with a few interactions to give the user a sense of being in the environment.

There have been numerous virtual heritage projects carried out in the past, most utilizing industry standard game engines such as Unity or Unreal Engine. With that said, this project explores the possibility of creating a VR experience of a heritage site utilizing a web-based platform that could enable anyone, without any prior training, to visit the site virtually from anywhere. The study attempts to meet the above requirement by using Babylon.js, which not only has backward compatibility but is also developing rapidly to be on par with native engines.

The study begins by analysing what virtual heritage applications have been developed in the form of virtual environments, and what elements were used in their creation. These elements are then applied to the design and creation of the virtual environment of Spike Island Prison.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institution of learning.

Signed:
Karthika Cheeniyil

Acknowledgements

I would like to express my gratitude to my supervisor, Prof. David Murphy, for his guidance and support throughout the course of the project. His knowledge and constant motivation have been invaluable to the development of this master's thesis project.

During this past year, my fellow classmates have also been incredibly helpful and supportive, and I am thankful to all of them.

Finally, I want to thank my family and friends back in India. I am extremely grateful to you for helping me survive all of the stress during the past year. I also appreciate your encouragement to keep moving forward during the last four months of intense project work. I could not have done it without you.

Chapter 1

Introduction

With scenes and elements that seem real, a virtual reality (VR) environment gives the user the feeling of complete immersion in their surroundings. This can be quite useful when physical presence in a certain place is not possible. Virtual reality in the tourism industry has been a popular topic for a while now. Being able to capture tourist locations in such a memorable and immersive way is not only a strong marketing tool but also allows individuals to have an experience of digitally exploring that area when they are unable to physically visit that place due to unavoidable circumstances. Additionally, virtual reality has experienced rapid growth in the gaming industry over the past few years. The adoption of advanced VR hardware and accessories among gamers is responsible for the growing demand for virtual reality in the sector. "According to a gaming expert from Solitaire Bliss, Virtual Reality is introduced as a significant change for gaming industries and is known to offer an improved version of simple user experience."

There has always been a focus on avatars, weapons, fights, and wars in the video game industry, but recently, a new division of this industry has been introduced for non-gamers. Xbox partnered with Rough Guides, to create the first travel guide to the gaming worlds, merging the travel and gaming markets to attract a new audience who are not attracted to guns and violence.

1.1 Thesis Statement

Interaction with 3D content has evolved dramatically in the gaming sectors, presenting the users with the most advanced virtual reality technologies. As a result of this, numerous expectations have been developed about what all can be achieved when a user is exploring a heritage site within a virtual reality setting. By developing virtual reality environments that utilizes real spatial data enabling the users to virtually connect with the location and the artifacts, the domain of virtual heritage is thereby attempting to address the growing need for interactive 3D contents of heritage sites (Wessels, 2015).

The proposed project for the master's thesis was undertaken to create an immersive virtual environment / virtual world of a historical tourism site wherein any virtual user around the

world can completely immerse in the virtual setting and interact with various forms of spatial data within the site without any prior site-specific knowledge. Spike Island, often known as Ireland's Alcatraz, was chosen for this project.

1.2 Spike Island - Ireland's Alcatraz

Spike Island is an island of 104-acres (about twice the area of The Vatican) in Cork's natural harbor, Ireland. The island's story is both fascinating and dark. This Island has been used for several purposes and has a recorded history that goes back to the 7th century. It was a Monastery, Fortress, Prison, and island residence before becoming a well-known tourist destination dominated by a star fort. Today the island has several museums and exhibitions on the social military penal and monastic past and in addition to the breathtaking scenery found along the 5 km long Ring of Spike, visitors can also join Ireland's best storytellers for a captivating look into the personal tales of this unique region of the Ancient East.

1.2.1 Historical overview of the prison

The island had formed part of Cork harbor's military defenses for almost eight decades. The site incorporated a pentagonal fort that was built at the east side of the island but soon with the French invasions of Ireland in 1796 and 1798 it was decided to replace the first Fort Westmoreland with a much larger fortress capable of holding 2000 – 3000 men.

This new fortress consisted of six bastions connected by ramparts and surrounded by a dry moat (McCarthy and O'Donnabhain, 2016). Spike Island's size, its location, and its new fort infrastructure convinced government officials that it would make an ideal depot where the criminals could be held temporarily and in 1847 it first became a convict depot. The prison was originally intended as a temporary detention center for no more than a few hundred convicts for a short amount of time, but it stayed open as a prison for 36 years with around 2,500 inmates by the early 1850's (Urbanus, 2020).

Spike Island's harsh conditions, poor medical care, and overcrowding led to an almost astronomical death rate, especially in the years during the famine and it was not until penal reforms in the 1850s reduced Spike Island's prisoner population from 2,500 to 900 that the numbers of dead finally began to drop. The island remained in use as a garrison and prison through the Irish War of Independence when IRA prisoners were held there in 1921. The whole prison was finally closed in 2004.

1.2.2 Spike Island Tourism Development

A government decision in July 2009 to transfer Spike Island from the Department of Justice, Equality, and Law reform to Cork County council allowed the council to collect and reinterpret its social and historical significance for Cork harbor and to explore the tourism potential of the island. Since then, the island has gained a reputation as a heritage tourist attraction, drawing thousands of visitors every year. As part of its redevelopment plans, the Council has been constantly working towards providing visitors with an experience reflecting the Island's history

within the harbor and its connection to the Irish diaspora (Laura, 2013). The former prison site owned by the County Council won the 2017 prestigious World Travel Awards as 'Europe's Leading Tourist Attraction'. There are several main attractions on the island, including an 1850s punishment block, a former prison where unruly inmates were punished, a 1985 cells and riot exhibition, six-inch guns guarding the harbor, and other attractions such as the gun park, which holds dozens of artilleries and coastal defense cannons and Mitchell Hall, which hosts temporary exhibitions of art.

1.3 Project Description

The project sets out to create an immersive VR experience of Spike Island using a web-based game engine incorporating WebXR. The focus will be more on the interior part of the fortress with all the buildings rather than the whole island with the aim of developing it further in the future. A previously done similar project incorporates several real data existing in relation to Spike Island such as historical plans, photographs, CAD drawings and LiDAR data which can be reused as assets for the current project. 3D Modelling and developing a virtual experience using web-based technologies are the two major parts of this project for which appropriate software, and platform will be discussed in the coming chapters. For the current project, it is intended that the user will be able interact with some elements inside a cell which should give the user sense of being there and further it is envisaged that in the future other parts of the prison will be developed as well. The completed project should then be assessed for system performance while running the scene on different operating systems and user experience when interacting with the scene.

Chapter 2

Background Research and Analysis

This chapter focuses on the background research and analysis that was carried out in relation to the project. Relevant literature review will be given followed by a brief analysis on possible technologies. An overview of some of the previous work done in the area is presented at the end of the chapter.

2.1 Literature Review

The following section will provide a literature review on the topic.

2.1.1 Virtual Heritage

Virtual heritage can be described as the combination of virtual reality and cultural heritage (Granström, 2013). As the name suggests, virtual heritage involves the process of recording, preserving or recreating historical, cultural artifacts, sites etc. with computer based interactive technologies. It aims to provide formative educational experiences through electronic manipulations of time and space to a global audience (Stone and Ojika, 2000). When discussing the visualization and representation of heritage and archaeology, two points the are frequently raised in the research and study are the authenticity or how valid is the gathered information and how crucial accuracy is in how this information is presented.

The validity of the information that has been gathered and the importance of accuracy in how it is presented are the two key points that are usually brought up when discussing the visualization and representation of heritage and archaeology. “To virtualize heritage means to actualize it digitally, to simulate it using computer graphics technology” (Roussou, 2002).

The day is not far off when we will no longer need to travel across the globe to see historic sites. We will simply be able to choose them from any online site and enter the virtual space from anywhere in the world (DeLeon and Berry, 2000). There has been successful use of virtual reality technology in cultural heritage for a wide range of purposes including excavation

documentation, education, museum displays, serious games and tourism (Sanders, 2014).

2.1.2 Virtual Heritage and Video games

In the past virtual heritage applications have often been criticized for their lack of interactivity, and those that do typically limit users to using spatial navigation to explore through a virtual world (Roussou et al., 2006). The lack of intangible elements and low level of interactivity in virtual heritage applications has continually forced those working in this field to look for inspiration and solutions in other related fields. Among such fields is the area of video games, which, on the technical side, is similar to virtual heritage based on virtual reality technology.

Using a highly interactive and dynamic approach, digital games appear to be capable of reproducing both tangible and intangible cultural artifacts. Their fun and play-based interfaces provide an engaging and powerful experience for users of all ages (Prensky, 2001).

Although there are certain similarities between virtual heritage applications and video games, the development of the two areas has largely taken independently. The target audience for video games has essentially always been the general public, whereas virtual reality systems and virtual heritage applications have often been restricted to usage in research and industry labs (Roussou, 2002). "Developing serious games for cultural heritage: a state-of-the-art review" by Anderson et al. explores the concept of serious games for virtual heritage.

The term serious games can be defined as "Computer games that are not limited to the aim of providing entertainment". Furthermore, serious game applications can be characterized as featuring high levels of communication, visual representation, collaboration, interactivity, and entertainment. (Anderson et al., 2010). Despite being a relatively new phenomenon, serious games are already being used for educational purposes in a wide range of applications.

In addition to being able to recreate and simulate cultural heritage sites in more detail than the traditional virtual heritage applications, modern entertainment 3D video games / serious games have been rapidly advancing in the area of interactivity and immersion while still running on relatively low-performance computers (Anderson et al., 2010; Champion, 2006).

2.1.3 Ideal Virtual world for heritage site

The phrase "virtual world" or "virtual environment" will be used in this essay to refer to a 3D fully immersive virtual space that allows user interaction and free roaming navigation, as opposed to online videos, animations, panorama tours, which restrict the user to a predefined path or tour and are frequently referred to as virtual tours (Sanders, 2001). An ideal virtual world is one which can make you believe that you are really there. It offers a "better than real life" or "better than being there experience" (Roussou, 2002)). Another description of an ideal virtual world was given by Wessels et al. 2014 "the virtual heritage much consist of high quality data, be simple to navigate and be interesting and informative to explore. Access to the virtual world should be easy as possible and thus should be made freely available on a web browser to allow an international audience to use the virtual world".

2.1.4 "The Feeling of Being There" - Presence

In addition to the technical requirements of generating a Virtual Heritage application, another intriguing issue is that these virtual environments should be able to provide the user with a sense of location and perhaps even emotional simulation than just delivering a static digitally created replica of the area of interest. The degree of presence experienced by the users in virtual environments has frequently been related to the efficacy of those settings.

In an article named "Measuring Presence in Virtual Environments: A Presence Questionnaire" by Witmer and Singer, presence is defined as:

"The subjective experience of being in one place or environment, even when one is physically situated in another. We believe that presence is a normal awareness phenomenon that requires directed attention and is based in the interaction between sensory stimulation, environmental factors that encourage involvement and enable immersion, and internal tendencies to become involved"(Witmer and Singer, 1998).

An experimental study was conducted to quantify the degree of presence felt by a user in immersive virtual environments. Based on the study, a logistic regression analysis showed that the subjective reporting of presence was significantly strongly associated with the visual and kinaesthetic representation systems and negatively associated with the auditory system (Slater et al., 1994).

Authenticity is a crucial component of cultural heritage. However, as virtual content simply depicts recreations or representations of the original cultural heritage material, it is impossible to infuse true authenticity into virtual heritage. This is why it's crucial to ensure that elements in virtual heritage are accurate and realistic because they can communicate a sense of authenticity even if they aren't real. Stone had a similar conviction, when in 2011 he wrote "the design of appropriate and effective content is a fundamental aspect in the development of an affective virtual environment."

This, according to Stone, is by far the most important factor in ensuring a sense of involvement or immersion. Today's highly effective game engines and toolkits, however, enable developers to not only create realistic and natural-looking virtual environments but also to include dynamic natural processes such as tree motion in response to simulated wind, dynamic lighting, and weather simulations like rain and fog (Depledge et al., 2011).

2.2 Methods of creating virtual heritage environments

Users can now explore sophisticated 3D environments, thanks to the advancements in real-time 3D graphics technologies, high bandwidth Internet connections, and contemporary web browsers. For building and visualizing a virtual environment on a desktop computer, there are numerous software programs available, both as independent downloadable applications and online platforms. The Unreal Engine, Unity, and VRML/X3D are the most widely used programs. Additionally, platforms for bringing 3D content online include O3D, OpenSpace3D, and TurnTool.

2.2.1 VRML and X3D

An international standard for describing and visualizing virtual reality 3D content and environments on the internet, called Virtual Reality Modelling Language (VRML), was initially created in 1994 (Champion, 2014). Web browsers can execute VRML environments once a plug-in is installed. Since its debut, the VRML standard has undergone numerous revisions. VRML has evolved and been replaced by Extensible 3D (X3D), which is controlled by Web3D (Web3D Consortium, 2012). The International Organization of Standards (ISO) has endorsed both VRML and X3D (Koutsoudis et al., 2012). Among the early virtual heritage applications "The Gebel Barkal Temple B300 (learning site)" is one which was developed using VRML. Champions (2014), however, criticizes them for being slow and buggy. With X3D, 3D content can be made accessible via the web using the latest web coding language XML5 (Koutsoudis et al., 2012). However, using X3D has its own drawbacks. It is very work intensive and often, the amount of work and time dedicated to create a virtual environment isn't reflected in the final product.

2.2.2 Second Life

The three-dimensional virtual environment Second Life is undoubtedly the most well-known hyperreality. The user of Second Life, often referred to as a "resident," enters the virtual world via a downloadable client program that creates a unique avatar. Although a voice-chat feature was launched in August 2007, most interactions between avatars still take place in written form via chat or instant messaging (Kaplan and Haenlein, 2009).

The underlying architecture of Second Life is built on a client-server model where the graphical user interface runs locally, while the 3D virtualisation is provided partly by the Havok physics code running on servers owned by Linden Lab (Warburton, 2009). According to Edwards, 2007, Second Life is the most popular Internet-based virtual world in terms of number subscribers and money exchange and are largely adopted for academic, social and business purposes. As opposed to other gaming engines where assets like monsters and weapons are stored locally, the visual experience in this engine is rendered in real time. Using this working approach, the creation of user-content is made possible in an unprecedented manner. Second Life, however, puts undue strain on users as they need a specific bandwidth capability and a specific graphics card to participate. If users attempt to access Second Life without the appropriate technological requirements, it can compromise critical elements of the end-user experience, particularly the frame rate. In this case, there is a possibility of 'lag' where heavy loads caused by a lot of objects in a single location slow down the experience and cause the elements on the scene to rez (resurrected) slowly or perhaps never rez at all (Warburton, 2009).

Cost should also be taken into account while choosing Second Life as the project's platform. There are equally compelling free alternatives, therefore this is not the best choice for a master's thesis project.

2.2.3 Game Engines

Modern interactive virtual environments are usually implemented using game engines, which provide the core technology for the creation and control of the virtual world. A game engine is an open, extendable software system on which a computer game or a similar application can be built (Anderson et al., 2010).

They typically offer a rendering engine, physics engine, sound, scripting, animation, AI, networking and other functionality that would be used in the creation of a video game. Game engines offer the Virtual heritage community great opportunities in creating interactive and engaging virtual environments of heritage sites without having to create software from scratch. As noted by DeLeon and Berry, 2000 video game companies gave made great strides in the area of presenting 3D virtual environments.

Unreal Engine

The Unreal Engine was developed by Epic games with the first game, Unreal, released in 1998. The engine was developed mainly for first person shooter (FPS) games. In 2009, Epic released the unreal Engine 3 Software Development Kit called Unreal Development Kit (UDK), available for free to create non-commercial games and applications. There is a large developer community that creates publicly accessible content for the Unreal engine, including virtual locations, characters, and objects (Jacobson and Holden, 2005).

Unity

Another widely used game engine is Unity, a cross platform game engine developed by Unity Technologies in 2005. Unity is a "game development ecosystem" commonly used to create games for web-plugins, desktop platforms, consoles and mobile devices. With the release of UDK other developers including Unity introduced a non-pro version of the game engine which is provided for free to keep up with the competition. The engine can also generate downloadable standalone programs and offers a plug-in for running projects on various web browsers (Koutsoudis et al., 2012).

The game engine for the web – Babylon.js

There are many 3D JavaScript engines available for use by developers. Popular examples include Three.js and Babylon.js (Mu, 2019). Babylon.js is a JavaScript framework that is built on top of Web Graphics Library (WebGL). WebGL API is used for rendering graphics within a web browser. Babylon.js takes away a lot of the complexity involved in WebGL programming. This library already supports the development of Web VR and Web XR. Feature-wise Babylon.js engine provides a plethora of them, most being amazingly well on par with the Unity engine, considering the development environment, with one of the main weights being on relative ease-of-use, especially once the user has gained some summary experience regarding the use of the engine (Järvilä, 2021). The primary objective of this project is to make the application accessible to as many users as possible, regardless of the device on which they are using it, so

a WebGL engine is a much better solution than using a native 3D engine such as Unreal or Unity.

2.3 Case Studies

This section presents significant case studies of projects in the area of virtual heritage in chronological order of the projects' release dates. In order to develop this virtual heritage project, research into some previously completed projects in the field was required to comprehend the key concepts that would be used in the project design.

2.3.1 Virtual Notre Dame Cathedral

In 1999, the Virtual Notre Dame (VRND) project was initiated. It was one of the first applications for virtual heritage to employ a game engine. After comparing various development environments, the team chose Epic Games' Unreal Engine, highlighting its then-advanced technological capabilities and affordability (Svånå, 2010). The developers aimed at creating a fully immersive 3D virtual environment of Notre Dame Cathedral as stated by DeLeon and Berry (2000) – "No tricks this time - virtual meant virtual, not a series of stitched images or panoramic, but an actual 3D model with more "oomph" than Virtual Reality Modelling Language (VRML) could deliver." The entire 3D model was created manually using low polygon counts and then to account for the details inside the simplified models, appropriate textures were used.

Figure 2.1: 3D Model of the Notre Dame Cathedral. From (DeLeon and Berry, 2000).

The completed 3D model along with a fully animated character model for the purpose a virtual tour guide were then imported into the Unreal Engine. The project's virtual tour guide is a fully animated character with basic AI that is implemented by scripting behaviours into the guide's character (DeLeon and Berry, 2000).

Useful Design and Data components

Considering the time of its release in the year 2000, this project has an vast amount of functionality including (Wessels, 2015) -

- Two navigability modes – walk and fly
- Virtual tour guide offering explanations of many points of interest via a small text box.
- Torch for dark areas
- Lighting effects for a better sense of presence.
- Multiplayer mode
- Virtual elements

Figure 2.2: The virtual tour guide of VRND. From (DeLeon and Berry, 2000).

2.3.2 Egypt Virtual Temple and Gates of Horus

The Virtual Egyptian Temple is a public VR project that simulates the construction of a traditional Egyptian temple using materials from a New Kingdom temple and serves as an example of how game engines are actively used in the creation of virtual heritage environments (Jacobson and Holden, 2005). The early versions of the project used VRML and later on Unreal Engine to create temple tour. The Unity engine is used in a subsequent version, which also includes virtual objects and ambient music (Svånå, 2010). In the Gates of Horus, the user takes on the role of a young priest being instructed about ancient Egyptian customs by a virtual high priest. The user is free to explore the temple with a first person view. The user can interact with certain objects having a golden glow highlight to them which then starts an audio narration by the high priest on a particular historical practice or ritual.

Figure 2.3: The virtual Egyptian Temple and its priest. From (Jacobson and Holden, 2005)

Useful Design and Data components

- Models with high resolution textures giving the scene a realistic look
- Audio narration
- Ambient audio for creating a sense of presence
- Interactable elements with glowing effects.

2.3.3 Rome Reborn

Rome Reborn is an international initiative to create 3D computer models of the entire city of Rome as it stood in 320AD. The project has been running since 1997 and is the world's largest digitisation project. Version 1.0 of the project was presented online in 2007 using static images and video flythroughs. Rome Reborn 1.0 was published in Google Earth as the "Ancient Rome 3D" layer in 2008. Given that Google Earth is the most widely used and best supported virtual globe application and that it makes it simple to create and deploy accessible, low-resolution 3D models, the possibility of offering the Rome Reborn project far wider exposure was increased. There were, however, some issues with this layer, such as the low-resolution 3D buildings with flat textures, which meant that many details of Rome Reborn 1.0 would be obscured (Wells et al., 2009). As of the current version 3.0, the model is optimized to be used through augmented reality in real time.

Figure 2.4: The valley of the Flavian Amphitheatre, Version 2.1. From (Svånå, 2010).

2.4 Chapter Summary

In this chapter an overview of the Virtual Heritage was given followed by a literature review on the topic. The second section of the chapter focused on some of the technologies available for creating a virtual heritage application accompanied by some case studies of previous projects in the field, noting that many of the aforementioned design components can be used in the current project as long as they produce the required results. Reviewing the literature and examining previous research and projects has revealed that game engines are the preferred platform for the majority of implementations.

Chapter 3

Project Design and Web Technologies used

The focus of this chapter is on the identification of techniques employed in the creation of a Virtual environment that enables a user to explore an environment virtually, in an immersive and engaging manner. The useful design and data components identified in the previous chapter from case studies of Virtual heritage applications are explored here and how it is intended to be implemented will be discussed.

3.1 Game engines: a platform for the VR environment

Following on from the background research and analysis carried out in the previous chapter into suitable platforms for the project, it was apparent that game engines offer the functionality of creating virtual environments and supply all the tools needed to realise the goal of an interactive virtual world with real-time, high quality graphics rendering capabilities. It was thus an obvious decision to adopt a Game Engine as a platform to develop the project.

The idea of employing a game engine for non-game applications is not new. 3D game engines have found their way to a number of non-game applications, ranging from CAVE installations to simulating context-aware services. Game technology has managed to develop systems that,on the one hand, exploit computers' potential to the maximum and, on the other, offer the user a sophisticated, interactive environment with 3D graphics and, in some cases, immersion capabilities that can be enjoyed, even on a moderate home computer. This decision is reflected in the article by (Lepouras and Vassilakis, 2004) where the conclusion was made that the most practical approach in the implementation of virtual reality with a majority target audience at an economical cost was to use game engines.

Many game engines support displaying 3D models and allowing interactivity between the user and the model, which is one of the fundamental aspects of the project. After an initial look at the multiple game engines available, the research and investigation was narrowed down to two choices; a native game engine Unity and a WebGL engine Babylon.js.

3.1.1 Unity3D and Babylon.js – A comparison

Comparing a native engine with a web based engine is not a fair comparison. There are limitations on WebGL engine that are not an issue for a native engine, but on the other hand the accessibility for a WebGL experience is much broader than a native app.

Unity is by far the most widely used game engine today and is adopted by various industries such as film, automotive, architecture, engineering and construction apart from gaming industry. Unity supports the C# programming language natively which is an industry standard language similar to Java or C++.

Babylon.js is comparatively new in the gaming industry and was initially released in 2003 under Microsoft Public License. The source code is written in TypeScript and then compiled into a JavaScript version. The JavaScript version is available to end users via NPM or CDN who can then code their projects in JavaScript accessing the engine's API. The Babylon.js 3D engine and user code is natively interpreted by all the web browser supporting the HTML5 standard and WebGL to undertake the 3D rendering. Babylon.js also uses a node material editor and the interaction set on the editor is very similar to how it is set by other tools in the industry such as Unity, Unreal, Substance Designer, Houdini etc.

Category	**Unity**	**Babylon.js**
Price	Non-pro version is free	Free
Primary programming language	C++	JavaScript, TypeScript
Scripting	C#, Visual Scripting (Bolt)	JavaScript, TypeScript
Cross-platform	Yes	Yes
2D/3D oriented	2D, 2.5D, 3D	3D
Integrated Physic	Yes	Yes
WebXr	Yes	Yes
License	Proprietary	Apache License 2.0
User support	Big community of users. Great support from the Forums and Unity answers.	Relatively small but a growing community of helpful developers. It's easy to find help on their forum.

Table 3.1: Comparison of Unity and Babylon.js

3.1.2 A Brief overview of WebGL

WebGL (Web Graphics Library) is a JavaScript API that allows the creation of sophisticated 3D graphics inside web browsers, without plug-ins. WebGL makes it possible to build a new generation of 3D web games, user interfaces and information visualization solutions that will run on any standard web browser, and on PCs, smartphones, tables, game console or other devices (Matsuda and Lea, 2013).

It uses JavaScript language in the HTML5 Canvas tag to render and display 3D graphics. And it accesses different graphics drivers using OpenGL ES API so as to inter-cross the platform limitation. Moreover, it uses GPU to realize the acceleration of 3D rendering in the hardware level. WebGL enables web browsers to show 3D scene and models more smoothly with the help of system's graphics card. There is no system platform or programming language restrictions. So it can complete a lot of graphical calculation and real-time 3D rendering. By using WebGL, the efficient graphics rendering no longer depends on the local 3D software. Since the first official release of WebGL in March 2011, the popularity of WebGL has dramatically increased, with an active community developing a number of tools and library, such as Three.js, Babylon.js or Unity.

3.1.3 WebXR

WebXR is the grouping of standards responsible for supporting rendered 3D scene in virtual and augmented reality, both experiential realms known together as mixed reality (XR). Virtual reality (VR), which presents a fully immersive world whose physical elements are entirely drawn by a device, differs considerably from augmented reality (AR), which instead superimposes graphical elements onto real-world surroundings. Simply put WebXR is an api on web browsers, that allows the XR hardware (such as headsets and mobiles devices) to work on websites through the use of WebGL. In practical terms, this means that when you visit a webpage – you will be able to interact with the 3D content – regardless of what device you are on.

Traditionally XR has been limited to native apps that must be downloaded onto a user's device, this has been due to hardware and software limitations. WebXR has huge potential, for the same reason that web apps have advantages over native apps – they are just easier and quicker to set up on your device and use. Several open-source JavaScript frameworks are available to interact with WebGL and WebXR, namely Three.js and Babylon.js. Unity also has emerging support for cross-compiling to WebXR, using a plugin but the resulting files are much larger than coding WebXR directly in JavaScript. Moreover, Unity does not yet support single pass VR rendering when using WebGL and hence Materials and shaders will have an impact on the performance of the application.

3.1.4 Final decision

The platform for the project had to be chosen that is available at no cost, requires only a beginner level of programming skills and makes it possible to incorporate all the identified Virtual environment components. Important criteria in the choice of the game engine were the ability to create an interactive WebXR incorporated virtual environment with user control and

navigation capabilities. Also the Virtual environment had to be available on a web browser. After considering all these factors, it was finally decided that for my project and thesis its best to go with the WebGL engine Babylon.js.

Even though its relatively new compared to Unity and other popular game engines out there with advance features, it satisfies the primary goal of this project which is to reach as many users as possible regardless of the device they are using. Whereas if the project had been targeted to a specific piece of hardware, adopting a native engine such as Unity would have given a lot more flexibility.

Another major reason to choose Babylon.js over Unity is that Babylon.js versions are backward compatible, meaning upgrading to a new version is easy, however there never has been any officially guaranteed backward compatibility between any versions of Unity, which could be an issue when considering a long term project. Focusing on the design elements of the project, the Babylon.js framework provides a lot of useful features that facilitates the development of a 3D scene. It strives to minimize the amount of work to do to set up a 3D environment, which will allow us to focus on the projects core features that is implementing WebXR.

3.2 Architecture of the project

The scripting for the project will be completely done in JavaScript. The script file and CSS file will be set separately as external files which will be then linked to the main .html file, this is to ensure a better workflow. Before diving into core aspects of the project, a basic architecture of the project shall be discussed. This is necessary to achieve good scalability and maintainability for any project.

The users will be presented with a Final Project directory as soon as they enter the provided URL which should take them directly to the index.html file, the entry point to the Virtual environment. Inside the directory, apart from the html file will be the scene.js which will hold the entire script for the project, a CSS file and an assets folder containing all the 3D models in .glb format and other media files.

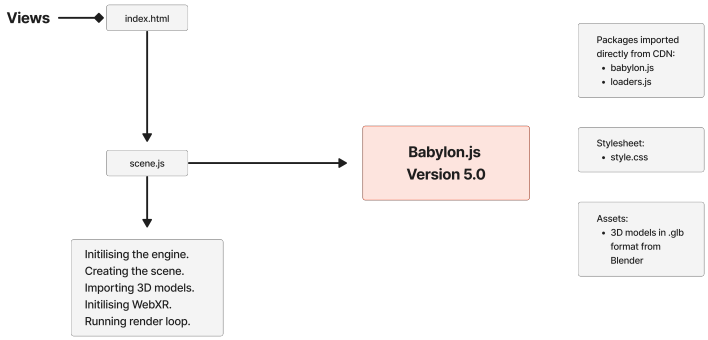

Figure 3.1: A graphical representation of the project architecture

3.3 Modelling Software – Blender

When building 3D applications, an important aspect is being able to import high quality 3D assets. Babylon.js comes with a set of plugins dedicated to importing 3D content onto the scene with a single line code. There are currently three file types supported by the Babylon namely gLTF , Wavefront OBJ and Babylon.js file format. For this project it was decided to use Blender for the modelling and texturing purpose. Blender is a free and open-source 3D computer graphics software tool set and it supports gLTF file format, hence it was best to export all the 3D modelled objects from blender in a .glb file format which is a binary version of gLTF to the Babylon scene with all the materials intact such as textures and animations, thus eliminating the need to reassemble the elements within the game engine.

3.4 Spitting up the Virtual environment into scenes

For a better workflow the decision was made to divide the environment into 3 separate scenes. As a result, different JavaScript files were created for each scene. This was so that if any changes had to be made, it would be easier to work with the appropriate JavaScript file without complicating the workflow. Finally, the files can be merged into a single file to ensure better performance as multiple JavaScript files on the same page can cause low performance.

Scene 1 – Fort Mitchel

The user will be presented with this scene as soon as they enter the VR session. Fort Mitchel, the star shaped fortress is the space that surrounds all the blocks inside. The user will be able to navigate through the space within this fort where all the models of the fort building and parade ground will be present. There will be no interaction in the scene.

Scene 2 – The Confinement Cell

This scene will present the user with some interaction usability's. The punishment block consisted of twenty eight solitary confinement cells and housed the most dangerous prisoners. With that keeping in mind, this level will focus on a modelled confinement cell inside the punishment block with some interactive elements. This scene is entered from the punishment block through an animated door.

Scene 3 – Modern Prison Cell

This scene consists of one of the Class C prison cells or what is known as modern cells in the C Block. The scene also incorporates interactive elements for an engaging experience.

3.5 Usability

This project is intended to be widely distributable and also has a wide target audience therefore this needs to be reflected in the user interface and input devices. Assumptions need to be made with regards to the technology available to the user and also the skill level of the user in order to fulfil these requirements.

3.5.1 Navigation

Users should have access to the same navigation abilities in the Virtual environment as they would in real life. Thus the user controls must enable the user to move forward, turn and view in a full 360-degree motion. Since the main aim of this project is WebXR, teleportation was chosen as a mean for the users to move around. Teleportation is completely under the control of the user, hence it is important to decide beforehand if the users are allowed to teleport anywhere within the visual range or are they are teleported to certain hotspots only within the space. The navigation aspects of the project will be further discussed in detail in the coming chapter.

3.5.2 Interaction

For an immersive experience, there should be some sort of interaction within the virtual space, and requires the incorporation of some form of input device. For a VR session Babylon.js provides sufficient support for incorporating pointer selection and grabbing abilities along with the teleportation features. The input methods to be used in the project should be widely recognised and familiar to most users.

3.5.3 Start Screen

An introductory start screen should be available for the user to access some basic information about the Virtual environment including instruction on how to navigate, interact with the elements, enter the VR session and also how to exit the VR session. This introductory screen will be short and concise in order to prevent confusion. Once the user is ready to explore the Virtual environment, they can enter the immersive session by clicking on the XR GUI on the start scene. The decision to keep a simple start screen before entering the VR session was made to immerse the user in the Virtual environment without any unnatural interfaces that could cause distraction.

3.5.4 Audio

Audio plays an important role in setting the mood of a scene. Audio can capture the ambiance of a place and creates a feeling of presence. The unique sounds that are inherent to a heritage site offer an alternative sensory input into the virtual world other than sight. Hence to make a more immersive experience it was decided to use a ambient sound in the scene with some positional 3D sounds attached to specific meshes that can only be triggered by user interactions.

3.6 Chapter Summary

Following the discussion and comparison of possible platforms for the project, the use of web technologies for the project was briefly discussed. This led to the decision to implement the project using Babylon.js. It was important to establish the project's architecture before beginning the design phase in order to ensure a seamless workflow. Afterwards, the choice of modelling software was discussed as well as how the virtual environment would be divided into three separate scenes with respective script files. Finally, the chapter discussed the usability aspects of the project, including navigation, user interfaces, audio, etc. As previously mentioned, this project is meant to be widely distributable and have a broad target audience. Therefore, this had a significant impact on the design and technical choices to be made for the virtual environment.

Chapter 4

Implementation

This chapter discusses how the project was developed and how the design elements discussed in the previous chapter were implemented. All the resources and techniques that were used will be described along with the problems that were encountered during the process.

4.1 Site Visit

For a better overview on the dimensions of the buildings and the overall space, it was decided to make a site visit before starting modelling. Due to the VR focus of this project, it was important to build the space with as much accuracy as possible in terms of its true real world dimensions since it is one of the crucial factor which affects the level of immersion inside a virtual environment. In the survey of the spaces some overall dimensions were recorded by the means of photographs, videos and also by talking with the tour guide who further suggested the book 'Too beautiful for thieves and pickpockets: a history of the Victorian convict prison on Spike Island' by (McCarthy and O'Donnabhain, 2016) which greatly helped in determining the dimensions of the cells and the overall buildings. Along with these, actually being present there gave me a sense of an approximation of the heights of the spaces and also door and window openings which was a big plus before committing to the modelling step.

4.2 Scale

Once the primary and secondary resources were collected, the next step was to determine a scale for the project. In order to avoid needing to scale each and every 3D model on each import from Blender to Babylon.js, it was crucial to get the platforms' scales to match up from the beginning, and making sure the unit setup was proper on each platform.

4.2.1 Babylon.js units

The Babylon.js documentation and forums revealed that the engine works on whatever scale is requested by the user, thus whatever scale is used in blender will be rendered exactly by the engine without any modifications however anything other than meters with a WebXr camera

could cause some issues as most of the time in Babylon.js scenes, the world origin is placed in the centre of the scene with 1 scene unit representing 1 meter in real life. This was an advantage of working with Babylon.js instead of a native game engine such as Unreal where there is no fixed relation between the unreal units and the real world units and the engine picks a scale to work with depending on the game type which could cause a toll on time deciding on scale and units also their conversion.

The default unit system in blender is metric system with a default unit scale of 1.0, hence it was easy to setup the scale in blender as 1 metre for every model with the approximation of the dimensions which were collected beforehand and export them as it is to be used in Babylon.js

4.3 Modelling and Texturing with Blender

The modelling of Mitchel Hall, Block A and Block B was done by re-using the assets from a previous project by Laura Mellet who have done an impressive job by creating exact replica of the Spike Island buildings by using real world data such as CAD data and drawings. All the assets from her project were retrieved in .fbx format as individual meshes which were then imported into blender for further manipulations for this project. The Punishment Block, 1915 Building, Solitary Confinement Cell, the Modern Prison and the Fort Mitchel were all built from scratch.

4.3.1 Modelling: Polycounts and Optimization

As mentioned in the introduction games have always been focused on weapons, fight, avatars etc. with an end goal but the current project has been gamified with no such intensions but only an end goal of providing the user with an immersive experience. One common thing between the traditional games and this project is that both are interactive in one way or another. This means no matter how much planning is done and how much time is spent, we will never be able to predict every possible thing a user can do at any given moment. This is where the concept of Real-Time Rendering comes into play. 3D graphics rendering is very costly in terms of processing time, and particular attention must be paid to optimization when modelling. The number of vertices required for a model in particular is very important (Silverman, 2013).

The assets from Laura's project consisted of simple and regular geometry which then helped me to generate simple meshes with low polygon counts. Similarly all the other 3D models were created with minimal polycounts and wherever possible repeating meshes were duplicated, which in blender is done with a shortcut key Alt D that stands for Duplicate linked or Deep Link. This was useful for creating linked objects, that means duplicate instances of similar meshes were created with attributes being linked, hence less toll on RAM usage.

Another helpful way to keep the polycount down was to remove unseen polygons from the models. A common technique used by 3D modellers is to completely ignore parts of an object that is impossible for the user to see, hence it was best to remove the back side or the interiors of the buildings and also the Fort which the user is not going see or interact with.

Triangulating is another important step of optimization (Silverman, 2013). This step simply consists in adding edges to faces that are made of more than three vertices, so that all the faces

are triangles. This is important since 3D renderer, such as WebGL, usually triangulates models for easier calculations when rendering. Doing this work upstream from the renderer may lead to better performances.

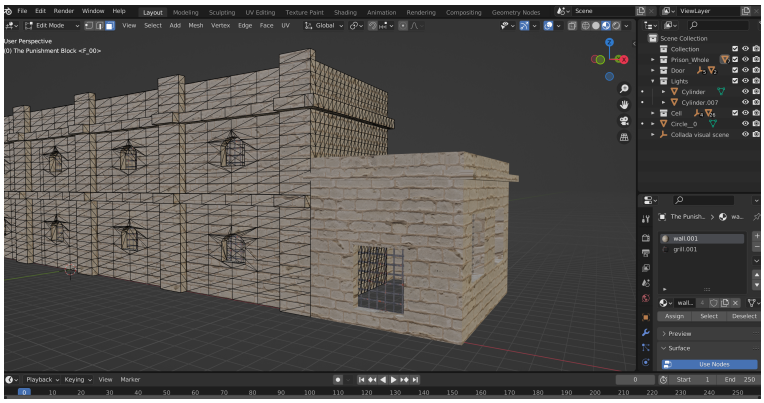

Figure 4.1: The Punishment Block with triangulated polygons

4.3.2 Texturing

Having low-poly models are not the only factor that will ensure low render times. After the modelling step comes the texturing of the 3D models which for this project has gone through a lot of trial and errors. Initially it was decided to attach high detail textures to 3D models to maintain a high visual detail while keeping polygon counts low. 16k .png textures were used which significantly caused a toll on memory and rendering time, hence subsequently it was decided to go with 4k textures for larger meshes and 2k textures for smaller meshes.

For a texture to map correctly with the geometry of an object, UV unwrapping has been performed prior to texture mapping. This technique generates UV maps by virtually unfolding the 3D model in order to create a 2D representation of the model. The 2D textures are then mapped over these 2D UV maps.

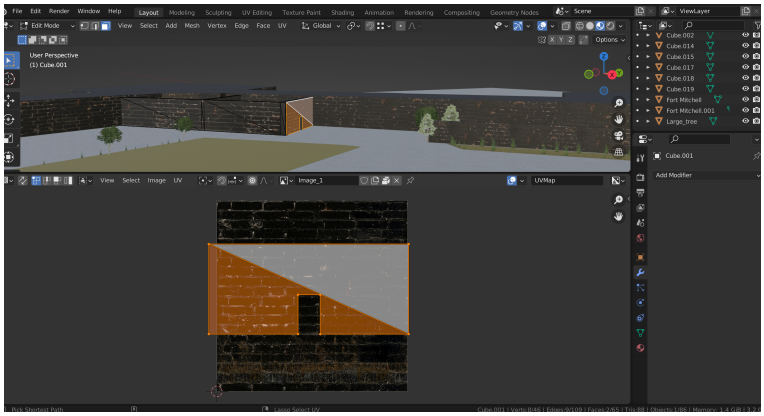

Figure 4.2: An example of UV mapping. The orange edges represent the seams used to unfold the mesh.

Different types of textures are usually used to achieve more realistic result. The most important texture types are Diffuse Maps, Normal Maps, Specular Maps and Alpha Maps (Silverman, 2013). For most of the models in the project a set of diffuse map, normal map, roughness map and displacement have been used. For certain objects specular maps and alpha maps have been used such as for the window grills to make it see through.

Diffuse maps represents the flat / base colours of the objects. On top of the diffuse map, it is a common practice to use a roughness map and a normal map to give the illusion of greater detail on a model, simulating some depth to the texture. Even though using all these maps produced great results in Blender, exporting to Babylon.js produced relatively poorer quality especially with the roughness and normal maps, hence another reason to go with low resolution textures.

4.4 Blender to Babylon.js – The Workflow

The final models were exported from blender in a .glb format which is a binary version of glTF and one of the suggested file format for Babylon.js. One reason to prefer glb format over glTF is because it incorporates the entire model, animation, and textures in one single file rather than multiple files. This meant that the materials for the mesh were then automatically generated in the engine and applied to the meshes on importing the .glb files. As a result the entire process became simplified as there was no longer a need to export and import the textures separately and therefore, materials didn't need to be applied to the meshes again.

4.4.1 3D rendering with Babylon.js

Once the modelling step was completed, it was time to create a Babylon scene which is responsible for loading and displaying all the 3D graphics of the project.

HTML structure

The first step was to create a standard .html file and embed the latest version of Babylon.js from Babylon CDN as well as other helpful dependencies such as one for the loaders.

After that a canvas element was created inside the body tag which babylon.js will use to render the content on to fill the entire available viewport space. That was it for the html file. A separate CSS file and a JavaScript file was created as it is desirable for an application to separate the presentation (CSS) from the content (HTML) and behaviour (JavaScript) in order to have better readability, maintainability, and error prevention.

Initializing the Babylon.js engine

The engine is the core of Babylon.js and a scene allows to create and manage entities that is drawn on the screen such as 3D models, lights, cameras and so on. An engine can be seen as a gateway to communicate with the video card (GPU) while a scene is high level interface that handles the multiple entities. In other words, a scene handles multiple entities and will call the engine to draw these entities on the screen.

Inside the scene.js file, firstly there is one helper variable which will store a reference to the canvas element from the web page (index.html) followed by the code to initialize the engine and other variables.

```
    const canvas = document.getElementById("canvas");
    let scene, engine, sceneToRender;
    const createDefaultEngine = function () {
        return new BABYLON.Engine(canvas, true, {
        preserveDrawingBuffer: true,
        stencil: true
    });
};
```

The BABYLON global object contains all the Babylon.js functions available in the engine.

Creating a scene

As mentioned above a scene is the place where all the 3D content is displayed with appropriate camera and lights. Once the engine was initiated, a createScene() function was defined inside which a new scene and a camera was instantiated.

```
    const createScene = async function () {
        const scene = new BABYLON.Scene(engine);
        const camera = new BABYLON.UniversalCamera("UniversalCamera", new
    BABYLON.Vector3(76.8, 2, 50), scene);
        return scene;
        }
```

Rendering the Scene

In order to make the scene actually visible on the screen, the final step is to render it using the engine.runRenderLoop call which is called each time the browser or device needs new frame.

```
    engine.runRenderLoop(function () {
    if (sceneToRender) {
        sceneToRender.render();
    }
});
```

The sceneToRender.render() function is called to draw every object of the screen during the frame.

Since the application encompasses the entire browser viewport, it is important to ensure that whenever a user resizes the browser window, the scene's dimensions update accordingly. To do so an event listener is added at the end for any browser resize that occurs.

```
    window.addEventListener("resize", function () {
    engine.resize();
```

Loading the Models

Once a basic babylon.js scene with a camera was setup, the next step was to load all the models. Initially it was decided to model the entire 3D scene and save it as a single .blend file

and transfer it to Babylon via a single .glb file. This caused an issue with the rendering time as the final .glb file was over 500MB and with that the target FPS was impossible to meet.

The second option was to import and position the assets within the Babylon.js scripting which again was not the best option for this project considering how big the scene was hence it was best to individually export each models from the .blend file with all transformations and scale applied so that there was no need to do any extra manipulations in the scripting. This was a less practical way for designing layouts and spaces than in a 3D software but is rewarded in loading times as models can be loaded in parallel.

```
    const ground = new BABYLON.SceneLoader.ImportMesh("","./assets/", "ground.
    glb", scene, function(allMeshes) {
    for (i in allMeshes) {
        allMeshes[i].checkCollisions = true;
        console.log("GROUND HAS BEEN CREATED");
    }
});
```

4.5 Creating the environment – Image based lighting and Skybox

Lighting is not just about making sure the user can see the scene. Light along with other design elements such as sound, actions etc. is also a mean to arouse emotional responses to visual components. This project is about creating a virtual environment of a tourist area which comes under a category of dark tourism whereby travellers seek something a little more macabre than the traditional trip to the seaside. With that in mind, it was important to make decisions regarding sky and lights for the environment which could instil a feeling of darkness and grim to the users.

Light is also expensive in terms of rendering time and in general, the rendering engine requires more time to render more lights, hence it was decided to keep the number of lights as minimal as possible in the scene and to use an Image based lighting (IBL) to lit up the entire scene. Skyboxes are commonly used in games, 3D scenes, and even real life movies to simulate a sky that covers the entire viewport. Using image based lighting, if the paint or texture on the skybox emits light in the same colour, it will light up the objects within it. Considering an stormy day, for instance, the skybox would give the scene grey and dull ambience, while a skybox with sunset paint would lit up the scene with a golden yellow ambience. Similar idea was implemented for this project.

With Babylon.js creating a scene and lighting it up with image based lighting is reasonably easy. For this project, a stormy sky HDRI image was used, which was converted to .env format using Babylon.js texture tools. Afterwards, a skybox was created with PBR enabled and a scale of 10000, and the .env file was used to texture it to create a sky for the environment and light up the scene.

```
    const texture = new BABYLON.CubeTexture("./assets/environment.env", scene);
    const skybox = scene.createDefaultSkybox(texture, true, 10000);
```

Figure 4.3: Skybox with IBL in the virtual environment

4.6 Creating immersive VR experience with WebXR

Enabling WebXR for the project involved adding a single line of code which is for the Babylon.js' default experience helper that determines whether a browser or device is WebXR compatible or not and it enable WebXR in VR immersive mode, including session init, input sources, the camera, teleportation and scene interactions. The WebXR support was enabled before the generated scene was returned from the main createScene() function. Using the relevant console messages, a base experience variable was constructed to see if the default experience had been initialized successfully.

```
1    const xrHelper = await scene.createDefaultXRExperienceAsync (
2    {
3        floorMeshes: [scene.getMeshByName("ground."), scene.getMeshByName("
     ground"), scene.getMeshByName("cell_ground"), scene.getMeshByName("stairs")
     ],
4    });
5    if (!xrHelper.baseExperience) {
6    console.log('WebXR support is unavailable');
7    } else {
8    console.log('WebXR support is available');
9    }
```

4.6.1 Pointer selection and teleportation

Both pointer selection (laser pointer and interaction with the scene's meshes) and teleportation features are immediately initialized in the babylon.js' default experience helper. In order to limit teleportation to specific meshes alone and not the entire scene, the specific floor meshes were included in the default experience helper option. For example, teleportation was enabled for the ground but excluded for the grass area.

A state change observable was added after initializing WebXR that is executed when the user enter XR. This was for setting the height of the user inside the environment and also to ensure the height remains the same when teleporting by positioning the XR camera to a y value.

4.6.2 Snap to Hotspots

There is a teleportation snap feature within Babylon.js which allows defining spots where the teleportation ray snaps with a specific threshold. As this comes with the WebXR features manager, which isn't required for this project, a custom piece of code was written which will take the user to a specific position in the scene, when appropriate.

Therefore an on pointer observable has been used which consists of a function that states, when the picked event is the panel mesh which is a kind of information board placed inside the punishment block and outside the C Block, the XR camera will position itself inside the confinement cell and modern cell respectively with appropriate rotation as well. Additionally, there is an exit board inside the two cells that the user can interact with and that leads to the outdoors.

This has to be implemented because the first prototype of this project consisted of no graphical interfaces inside the scene as it was decided to keep the scene as natural as possible without any extra artificial inputs. This also meant the users had to navigate around by themselves without any help which could have been difficult for novice users especially when trying to locate the two cells. The four panels above mentioned will be the only user interface that will be presented to the user and no other interfaces on the screen will be visible during the VR session. This decision was made to immerse the user in the Virtual environment without any unnatural interfaces that could cause distraction.

4.7 Babylon.js Actions and Triggers

In Babylon.js, there are two ways to create animations. As with Three.js, they can be generated by functions called within the rendering function or through built-in animation functions. By using the built-in animation engine, animations can be controlled by certain key values that determine how animations will be performed.

For this project the animations from the .glb file have been used as actions in the scene. Actions are a simple way to add interactions in the scene and it is launched when its triggered. For instance, the user clicks or touches a mesh an action is executed.

The door for entering the punishment block has a set of keyframes that animates a door opening and closing which has been exported along with .glb file and will only be started when there is a user interaction on it.

Babylon.js provides a number of actions and triggers which can be used as per the requirement. For the door, since it's an animation which has to be triggered, Babylon.js play animation action has been used with a on pick trigger so that when the user points the controller over the door mesh and clicks the animation begins. It has been chained in two parts so that with the first click the door will open and with the second click the door will close.

4.7.1 The Confinement Cell

The confinement cell in the punishment block has been built with a purpose of giving the user a feeling of storytelling, and for this a handwritten letter has been placed below the pillow which the user can interact with. The interaction here is about playing a 3D audio, which will begin when the user clicks on the letter. This interaction is also from Babylon.js actions i.e. Play sound action which is again chained in two parts so with the first click the audio will begin and with the second click the audio will stop. Since this audio has been set to spatial sound with a maximum distance, the audio won't be heard as the user walks away from that mesh.

4.7.2 The Modern Cell

The interaction inside the modern cell is similar to the one mentioned above but in a video form. The screen of the TV has been textured with a mp3 video file. In this case, the issue was that video files do not produce spatial sound, but rather ambient sound, which means that no matter where the user is, if the video is playing they will hear it as ambient sound. In order to fix this initially, it was decided to fake it by playing another audio file along with the video file, but in a spatialized format. This, however, caused audio and video to become unsynchronized.

The final option sought was to separate the audio from the video file, store it as a separate variable, and treat it as a Babylon sound with spatial sound set to true. Finally, both the video file and the audio were set to play on user interaction through a simple on pointer observable.

4.8 Chapter Summary

This chapter began with an overview of the various decision taken in terms of scale and units before the actual modelling and texturing was started. This was followed by a discussion on how the modelling and texturing of 3D models was done in blender along with the various optimization measures taken before exporting them in .glb format. This was the most time-consuming part in the project. Once the modelling and texturing stage was done, the next step was to import them and render them on the scene with Babylon.js. The chapter concludes with an overview of the scripting elements that were created in the implementation process.

Chapter 5

Testing and Evaluation

The testing and assessment of the created virtual environment will be covered in this chapter. In order to maintain a perfect performance rate at the end, an ongoing evaluation was done throughout the project's development stage before the post-project evaluation was conducted with user testing. This was necessary because the project's target is the web and any inconsistencies would have negatively impacted the frame rate and rendering speed.

5.1 System evaluation

During the implementation stage continuous assessment was carried out in order to keep a track of fps performance of the application. The Performance Profiler, which is introduced in Babylon.js 5.0, is an excellent visual performance debugging tool. This tool can assist in quickly identifying performance problems and glitches in the scene during the development phase.

This feature, which can be found on the inspector's statistics tab, enables users to visually view data that has been mapped out over time as opposed to just seeing the most recent value at any given time. The data is normalized on the displayed range, meaning the smallest value corresponds to the bottom y-position on the graph, while the highest value corresponds to the top y-position.

Once the scene was completely loaded, in the headless recording mode the performance was recorded by moving around the space and interacting with elements within the scene. This performance recording was then exported to CSV file, inside which it was possible to get the highest fps and the lowest fps attained during the experience. Also the GPU frame time was generated at different time stamps.

Even though fps and frame time are related, as frame time is calculated from fps, recording the frame time can be an even better indicator of performance because frame time follows a linear trajectory, whereas fps does not. To simplify, the total number of frames per second is divided by 1000ms to determine the frame time in ms. Therefore, the lower the frame rate, the higher the frame time.

5.1.1 System Used

For a general system evaluation the project was tested on three machines with different system specifications.

	System-1	**System-2**	**System-3**
Operating system	MacBook Air (M1, 2020)	Windows 11 Home	Windows 10 Pro
Processor	Apple M1 chip	Intel® Core® i7-9750H CPU @2.60GHz	Intel® Core® i7-7700K CPU @4.20GHz
Installed Memory (RAM)	16 GB	32GB	16GB
System Type		64-bit	64-bit
Graphics Card		Nvidia GeForce RTX 2080 with Max-Q Design	Intel® HD Graphics 636

Table 5.1: System specifications

5.1.2 Performance Results

The results from the Babylon.js performance recorder for both fps and GPU frame time on the three systems are displayed in the figures below.

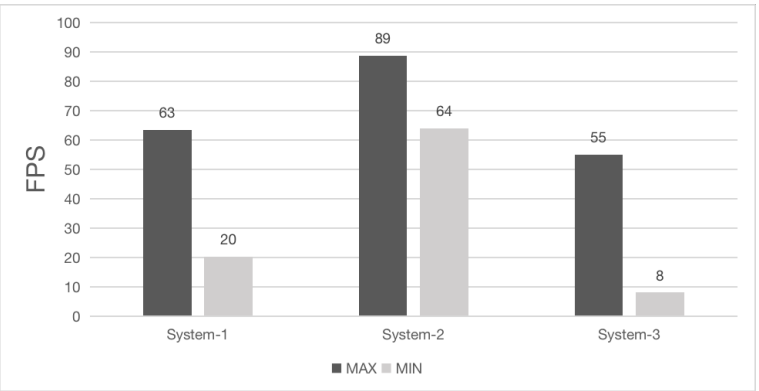

Figure 5.1: FPS values derived from the performance profiler for the three systems

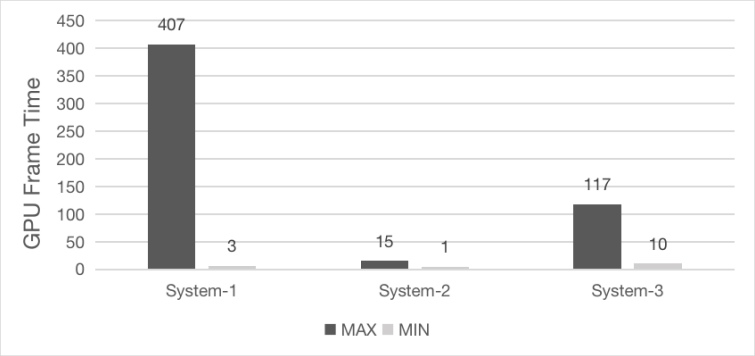

Figure 5.2: GPU frame time recorded at different timestamps on the three systems

5.1.3 Observation

As can be seen from the chart above, the frame rate was significantly higher in all areas while running the scene on system 2.

During testing on the MacBook Air, on which the project was entirely built, the frame rate dropped only when changing views suddenly, otherwise it remained nearly 60.

On system 3, however, a significant drop in frame rate was observed almost reaching 8, especially around active meshes like punishment block and C Cell.

The frame rate only reached a decent level when the camera was located close to static models, such as Block B, Hall, which are also the furthest meshes from active meshes.

In the beginning, it was believed that large texture size was the only reason that was causing the scene to render slowly and making lag more of an apparent issue. However, the performance data suggests that active meshes are actually the biggest factor affecting frame rate and render time. Here, active meshes refer to meshes with some interaction associated with them, and the two cells are the only locations where light has been positioned in the scene, which explains why fps dropped dramatically around these meshes.

5.2 Measuring User Experience in VR

According to the official ISO 9241-210:2010 standard the definition for the term User-Experience is “a person’s perceptions and responses that result from the use or anticipated use of a product, system or service” (for Standardization, 2010).

However, despite the existence of the ISO standard, the term User-Experience still remains a convoluted and complex term that is continuously being discussed in both the industry and academia. (Papadimitriou, 2019).

Measuring user experience can be challenging as it is subjective and completely depends on how the user feels ant a certain point of time and the context in which they are using the product. It can be even more difficult to find a suitable approach to measure user experience when it comes to Virtual Reality. The lack of standardized methods for researchers to access and evaluate virtual environments and other cutting-edge technologies is one of the key problems (Winn, 2002).

According to Bowman et al. (Bowman et al., 2002), due to the distinctive ways that users might engage with the content, conventionally accepted approaches for evaluating software programs may prove to be inappropriate and ineffective when it comes to Virtual Reality.

"For researchers to capture aspects of this complexity, there are multiple questionnaires that assess experiences of users, such as general reactions, user incentive, presence, entertainment and cybersickness"(Papadimitriou, 2019).

For virtual reality applications, it would be beneficial to study as many aspects of the experience as possible, in order to make adequate observations and measurements about the context. The purpose of this thesis, however, is not to provide a detailed description of every possible aspect of virtual reality applications mentioned above. For the current project, Witmer and Singer's standard presence questionnaire was utilized to assess the degree to which the user felt present in the developed virtual environment.

5.3 Presence Questionnaire

The presence questionnaire was used for the evaluation. Developed by Bob Witmer and Michael Singer in 1992, it aims to measure the amount of presence a person experiences in a virtual environment, as well as how specific elements impact experience intensity.

Based on the semantic differential principle, users had to answer the questionnaire with a seven-point scale. At the ends of each item are opposing descriptors, much like the semantic differential. This scale includes a midpoint anchor, unlike the sematic differential. This anchor system relies on the content of the question, and is a lot like the anchors found in rating scales. Respondents were asked to mark the appropriate box of the scale in accordance with the description of the question (Witmer and Singer, 1998).

An example item from the presence questionnaire is shown in Figure below.

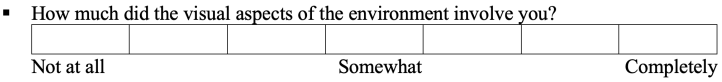

Figure 5.3:

In total, 14 questions were selected based on a standard Presence Questionnaire, which were divided into four categories, each addressing one of the contributing factors of presence, namely: Control factors, Sensory factors, Distraction factors and Realism factors. Presence in the questionnaire is considered to be the averaged sum score of these factors.

Control Factors	**Sensory Factors**	**Distraction Factors**	**Realism Factor**
Degree of control	Sensory modality	Isolation	Scene realism
Immediacy of control	Environmental richness	Selective attention	Information consistent with objective world
Anticipation of events	Multimodal presentation	Interface awareness	Meaningfulness of experience
Mode of control	Consistency of multimodal information		Separation anxiety/ disorientation
Physical environment modifiability	Degree of movement perception Active search		

Table 5.2: Factors Hypothesized to Contribute to a Sense of Presence by (Witmer & Singer, 1998).

5.4 Participants

The study had 10 people participating. The number was deemed to be adequate as it has been shown that a total of 5 to 8 participants can expose around 80% of usability problems through testing with users (Nielsen and Landauer, 1993). Because this project is expected to be continued and the foundation of the experience is only being built at this point, 10 participants were sufficient to carry out a quantitative analysis.

There were 3 male participants and 7 female participants with ages ranging from 22 to 34 years. Eight of the participants had technical backgrounds in computer science or related fields, one had a philosophy background, and one had nursing background. A number of 7 participants wore glasses and 6 had previous VR experience. Lastly all the ten participants reported having no cognitive or physical impairments that could impede them from taking part in this study. The study received ethics clearance according to the ethics and privacy regulations of our institution.

After welcoming the participants, they were handed over an information sheet providing details about the study and then asked to sign the consent form. For those who were completely new to VR, firstly they were introduced how to use to the controllers to navigate and interact within the designed virtual environment and then the HMD was adjusted to the participant's head. The inter-pupil distance was already set for best visual results.

5.5 VR Headset Used

Initially it was decided to go with a wireless headset for the project since it completely detaches one from the real world with no need to be tied to one place, worrying about stepping or tripping

on the cable. However as mentioned above an on-going evaluation was conducted at different stages of the implementation phase of this project and it was noticed that due to the large file size of the 3D models in the scene, wireless headset was not able to render it properly, rather the scene was completely breaking up even before it has fully loaded. Hence it was decided to go with the wired VR headsets since they rely on the power (processor or graphics card) of the workstation it is connected to. The head-mounted display that has been used for the testing purpose of the project is Oculus Rift S. Oculus Rift S is a fully immersive, wired, 6DOF virtual reality headset which was manufactured and released by the company Lenovo in 2019. The key attraction of the Rift S is how easy it is to set up. Thanks to its inside-out tracking, there's no need for external sensors unlike its predecessor Oculus Rift. The headset is connected via a cable to a spare DisplayPort and USB3 port on the workstation. Rift S features a 2K binocular resolution display, 80Hz refresh rate and a total weight of 500g.

5.6 Evaluating the questionnaire

Following the user testing, the participants filled out the presence questionnaire. The questionnaire was then analysed to produce a descriptive report. To produce the total score for the questionnaire, each category had to be calculated independently among all the 10 participants first. For example, when calculating the overall score of 'Sensory factors', you add the scores from questions 1, 2, 3 and 4. This produced the score for one participant in that category. For each of the other categories, the sums were computed similarly. Finally, the sum of the averages from all categories was used as the final score for presence.

5.6.1 Result

The average total score of Presence of the 10 participants was 86. Similarly, the average scores of the subset categories/ factors that contribute to presence and their average scores are presented in Table 3 below.

	Average
Total Presence Score	86
Control Factors	41
Sensory Factors	24
Realism Factors	12
Distraction Factors	9

Table 5.3: Result from the Presence Questionnaire

5.6.2 Observation

The questionnaire was used to examine the degree of presence that participants experienced while inside the virtual environment. The overall presence score that was produced was 86.

It was interesting to observe that the category of 'Control Factors' received the highest points out of the 4 categories despite half of the participants reporting the navigating mechanism to be difficult. Due to the nature of teleportation participants were the only ones who could control their movement, which meant for the new users of VR it was very difficult to find their way around the virtual environment. One of the participant who had some prior VR experience also had issues with the navigation, eventually leading to motion sickness due to which the testing had to be stopped in the midway. The biggest issue with the experience which almost all the participants commented on was collision. Outside the VR the scene had some collision detection attached to the scene camera, but inside VR enabling collision for the main WebXR camera did not affect anything. With some research into Babylon.js and WebXR it was understood that collision detection only works when there is a camera direction which is defined by the movement applied to the camera by input sources (mouse, keyboard, gamepad etc.). As this is not yet implemented in WebXR, the camera has no direction and therefore cannot collide with the world. Collison works when there is a smooth movement of camera and not when a new position is set, as in teleportation. Hence this issue could not be resolved for this project which led to users getting out of the room state if they did not control the movement. Getting out of the room state here means going outside the Mitchel Fort as in this project only the interior of the fort have been developed. Apart from the navigation mechanism, all other features related to control factors inside the virtual environment was found to be quite efficient which is evident by the average score.

Distraction factors received the lowest score in terms of presence. The testing took place in a computer lab that had some students using it, and it appeared that there was constant distraction going on during the session due to individuals entering and exiting the room.

While the project has a decent level of graphics quality, there have been instances of lag inside the scene. It caused participants to notice the scene loading incrementally during the session. The user testing was conducted on system 3, which had the lowest frame rate among the three systems and also which lagged the most, which explains the reason for slow rendering. Some of the participants reported that this was more apparent with constant head movements.

On another note, most of the participants reported that the experience felt realistic which helped them be immersed. Three of the participants also commented on how they felt scared inside the virtual environment due to ambient sound and overall visual components, which for this project is a positive point as the virtual environment developed is of a place with dark history and inflicting a feeling closely associated with that place to the users through a virtual space was indeed a success.

The questionnaire was able to provide an overall score for presence and indicated that control factors was considered to be the most influential aspect out of all the presence categories. Comments from the participants during the session contributed to the understanding of the aspects of virtual reality that actually immerses one completely.

Although using think aloud approach during the session helped in gathering qualitative

data, it should be noted that it can be harmful in the case of virtual reality as it can remind the participant that they are being tested and can affect the level of immersion inside a virtual environment.

5.7 Conclusion

This chapter began with a system evaluation that was done throughout the implementation stage in order to assess how well the application performed in a web browser. The user evaluation section started with a basic review of the user experience in virtual reality (VR) followed by the design of the study, which describes the selection of the participants, questionnaire, and VR headset used for the testing. The chapter concludes with an assessment of the questionnaire from the user testing, followed by a discussion of the findings and observations.

Chapter 6

Conclusion

The principle objective of this project was to create a 3D virtual environment of Spike Island Prison using standard web technologies. The field of virtual heritage is growing and becoming increasingly prominent in the public's eye as new technologies, equipment and software make it possible to document heritage sites with greater detail and efficiency and display this data using visually appealing methods.

From the case studies reviewed in chapter 2 it was found that many virtual environments of heritage sites exist, many of which were created with game engines, which led us to choose a WebGL game engine 'Babylon.js' for implementing the project. The reason for choosing a web based engine rather than a native engine lies in the fact that Web technologies are rapidly developing, and unlike native engines WebGL engine assures backward compatibility which for this project was a required factor as this project is intended to be carry on in the future. Although Babylon.js is currently not at the level of a native engine like Unity, it still has a lot of potential to be developed further and hopefully both will be in the same level someday.

The four-month duration of this project meant that time had to be utilized efficiently. With limited experience in 3D modelling and no previous experience with game engines, the execution of the project was very challenging.

It was incredibly difficult to find appropriate optimization techniques as mentioned in chapter 4. The final .glb file was above 500 MB which took a heavy toll on rendering time and fps, hence numerous decisions had to be made at different points in time during the implementation stage so that the environment could be displayed in real-time and also in different browsers.

Throughout the project development phase, I have gained an extensive understanding and knowledge of creating three-dimensional virtual environments. In spite of the fact that there's ample room for further development in this project, the final results have left me feeling satisfied and accomplished.

6.1 Recommendations for Future Work

Following are some thoughts on how the virtual environment can be improved and further developed in the hope that the project will continue in the future.

Although the project began with the goal of building the entire Spike Island, due to the limited time and experience with 3D modelling it could not be completed, and thus only the interior portion of the fortress was constructed. In addition to using higher quality models and textures, the virtual environment can be improved by including more areas of the Island and Prison. Further research should be conducted on lighting in the virtual environment.

At the moment, only wired headsets can be used to visualize the virtual environment. There should be better optimization techniques incorporated so that the project can run on wireless headsets, which are most commonly used today.

In order to avoid creating any kinds of unnatural interfaces during the VR session that could distract users, only a few GUIs were added to the scene, which during testing proved to be insufficient and resulted in users not being able to navigate properly.

Instead of using GUIs inside the scene a much better solution would be incorporating a fully animated avatar with some AI for the purpose of guiding users and narrating stories about the island. These elements would add intangible content to the virtual environment which would enhance the user's experience of virtually visiting the site.

Bibliography

3d assets from sketchfab. (n.d.). https://sketchfab.com/feed

Anderson, E. F., McLoughlin, L., Liarokapis, F., Peters, C., Petridis, P., & De Freitas, S. (2010). Developing serious games for cultural heritage: A state-of-the-art review. *Virtual reality*, *14*(4), 255–275.

Bowman, D. A., Gabbard, J. L., & Hix, D. (2002). A survey of usability evaluation in virtual environments: Classification and comparison of methods. *Presence: Teleoperators & Virtual Environments*, *11*(4), 404–424.

Champion, E. (2006). Playing with a career in ruins: Game design and virtual heritage. *Treballs D'arqueologia*, 45–61.

Champion, E. (2014). History and heritage in virtual worlds.

DeLeon, V., & Berry, R. (2000). Bringing vr to the desktop: Are you game? *Ieee Multimedia*, *7*(2), 68–72.

Depledge, M. H., Stone, R. J., & Bird, W. (2011). Can natural and virtual environments be used to promote improved human health and wellbeing? *Environmental science & technology*, *45*(11), 4660–4665.

Edwards, C. (2007). Another world. *Engineering Technology*, 28–32.

for Standardization, I. O. (2010). *Ergonomics of human-system interaction: Part 210: Human-centred design for interactive systems*. ISO.

Granström, H. (2013). Elements in games for virtual heritage applications.

Jacobson, J., & Holden, L. (2005). The virtual egyptian temple. *EdMedia+ Innovate Learning*, 4531–4536.

Järvilä, M. (2021). Recreating a unity game project in a 3d html5 webgl environment.: Research and comparison of 3d capable html5 webgl game engines.

Kaplan, A. M., & Haenlein, M. (2009). The fairyland of second life: Virtual social worlds and how to use them. *Business horizons*, *52*(6), 563–572.

Koutsoudis, A., Stavroglou, K., Pavlidis, G., & Chamzas, C. (2012). 3dsse–a 3d scene search engine: Exploring 3d scenes using keywords. *Journal of Cultural Heritage*, *13*(2), 187–194.

Lepouras, G., & Vassilakis, C. (2004). Virtual museums for all: Employing game technology for edutainment. *Virtual reality*, *8*(2), 96–106.

Matsuda, K., & Lea, R. (2013). *Webgl programming guide: Interactive 3d graphics programming with webgl*. Addison-Wesley.

McCarthy, C., & O'Donnabhain, B. (2016). *Too beautiful for thieves and pickpockets: A history of the victorian convict prison on spike island*. Cork County Library.

Nielsen, J., & Landauer, T. K. (1993). A mathematical model of the finding of usability problems. *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*, 206–213.

Papadimitriou, D. G. (2019). *User experience in virtual reality, conducting an evaluation on multiple characteristics of a virtual reality experience* (Master's thesis).

Prensky, M. (2001). Fun, play and games: What makes games engaging. *Digital game-based learning*, *5*(1), 5–31.

Prisoner daniel lane - convicted arson 1849 - letters from spike island series (2/3), cork ireland. (2020). https://youtu.be/KPU93V_Dkjc

Roussou, M. (2002). Virtual heritage: From the research lab to the broad public. *Bar International Series*, *1075*, 93–100.

Roussou, M., Oliver, M., & Slater, M. (2006). The virtual playground: An educational virtual reality environment for evaluating interactivity and conceptual learning. *Virtual reality*, *10*(3), 227–240.

Sanders, D. H. (2001). Persuade or perish: Moving virtual heritage beyond pretty pictures of the past. *Proceedings Seventh International Conference on Virtual Systems and Multimedia*, 236–245.

Sanders, D. H. (2014). Virtual heritage: Researching and visualizing the past in 3d. *Journal of Eastern Mediterranean Archaeology & Heritage Studies*, *2*(1), 30–47.

Silverman, D. (2013). 3d primer for game developers: An overview of 3d modeling in games. *Viitattu*, *11*, 2016.

Slater, M., Usoh, M., & Steed, A. (1994). Depth of presence in virtual environments. *Presence: Teleoperators & Virtual Environments*, *3*(2), 130–144.

The spike island prison riot 1985. (2020). https://youtu.be/tTQXDN0e0ko

Stone, R., & Ojika, T. (2000). Virtual heritage: What next? *IEEE multimedia*, *7*(2), 73–74.

Svånå, D. (2010). *Environment re-creation methods for virtual heritage using a game engine with discernment of visual learning cues* (Master's thesis). Institutt for datateknikk og informasjonsvitenskap.

Urbanus, J. (2020). The sorrows of spike island.

Warburton, S. (2009). Second life in higher education: Assessing the potential for and the barriers to deploying virtual worlds in learning and teaching. *British journal of educational technology*, *40*(3), 414–426.

Wells, S., Frischer, B., Ross, D., & Keller, C. (2009). Rome reborn in google earth. *Proceedings of the 37th CAA Conference*, 22–26.

Wessels, S. G. (2015). *Design and creation of a virtual world of petra, jordan* (Master's thesis). University of Cape Town.

Winn, W. (2002). Research into practice: Current trends in educational technology research: The study of learning environments. *Educational psychology review*, *14*(3), 331–351.

Witmer, B. G., & Singer, M. J. (1998). Measuring presence in virtual environments: A presence questionnaire. *Presence*, *7*(3), 225–240.

Appendix A

JavaScript Code Sample

```
//enabling xr
const xrHelper = await scene.createDefaultXRExperienceAsync (
    {
        floorMeshes: [scene.getMeshByName("ground."), scene.getMeshByName("ground"), scene.getMeshByName("cell_ground"), scene.getMeshByName("stairs")],
    });
if (!xrHelper.baseExperience) {
    console.log('WebXR support is unavailable');
} else {
    console.log('WebXR support is available');
}

//SETTING HEIGHT OF THE USER INSIDE VR
let personHeight = 2;
let xrCamera = xrHelper.baseExperience.camera;
xrHelper.baseExperience.onStateChangedObservable.add((state)=>{
    if(state === BABYLON.WebXRState.IN_XR){
        xrCamera.position.y = personHeight;
    }
});
xrCamera.onAfterCameraTeleport.add((targetPosition) => {
    xrCamera.position.y = personHeight;
});

xrHelper.baseExperience.onStateChangedObservable.add((state)=>{
    if(state === BABYLON.WebXRState.IN_XR){
        music1.play()
        music1.setVolume(2, 3);
    }

});
```

Appendix B

Performance Profiler Data Sample

Timestamp	FPS	Active meshes@Count	Total lights@Count	Render@Frame Steps Duration	GPU frame time@Frame Steps Duration
21.7	14.628438	81	2	2	64.56249237
91	14.76595974	81	2	2.3	64.56249237
153.8	14.66705799	81	2	1.9	64.56249237
217.5	14.68716335	81	2	1.3	64.56249237
278.7	14.85295582	81	2	1.6	64.56249237
353.8	14.79873753	81	2	1.8	61.8350029
412.8	14.89055443	81	2	1.3	61.8350029
477.8	14.94172764	81	2	1.4	61.8350029
552.9	14.9835186	81	2	3.6	61.8350029
614.3	15.00900555	81	2	1.6	61.8350029
687.9	14.99849987	81	2	4.8	65.22665405
753.1	14.97379589	81	2	1.1	65.22665405
821	14.91646767	81	2	1.3	65.22665405
980.9	14.46898842	81	2	2.9	65.22665405

Appendix C

Presence Questionnaire

PRESENCE QUESTIONNAIRE
(Witmer & Singer, Vs. 3.0, Nov. 1994)

WITH REGARD TO THE EXPERIENCED ENVIRONEMENT

1. How much did the visual aspects of the environment involve you?

Not at all Somewhat Completely

2. How much did the auditory aspects of the environment involve you?

Not at all Somewhat Completely

3. How well could you identify sounds?

Not at all Somewhat Completely

4. How well could you localize sounds?

Not at all Somewhat Completely

5. How aware were you of events occurring in the real world around you?

Not at all Somewhat Completely

6. How much did the visual display quality interfere or distract you from performing assigned tasks or required activities?

Not at all Interfered Somewhat Prevented Task performance

7. How responsive was the environment to actions that you initiated (or performed)?

Not Responsive — Moderately Responsive — Completely Responsive

8. How natural did your interactions with the environment seem?

Extremely Artificial — Borderline — Completely Natural

9. How natural was the mechanism which controlled movement through the environment?

Extremely Artificial — Borderline — Completely Natural

10. Were you able to anticipate what would happen next in response to the actions that you performed?

Not at all — Somewhat — Completely

11. How involved were you in the virtual environment experience?

Not Involved — Somewhat Involved — Completely Involved

12. How much delay did you experience between your actions and expected outcomes?

No delays — Moderate Delays — Long Delays

13. How quickly did you adjust to the virtual environment experience?

Not at all — Slowly — Less than

14. How proficient in moving and interacting with the virtual environment did you feel at the end of the experience?

Not Proficient — Reasonably Proficient — Very Proficient