FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor’s thesis

Title:	Post-Apocalyptic Model and Training for Survival in Virtual Reality
Student:	Duc Hoang Duong
Supervisor:	doc. Ing. Mgr. Petr Klán, CSc.
Study program:	Informatics
Branch / specialization:	Web and Software Engineering, specialization Computer Graphics
Department:	Department of Software Engineering
Validity:	until the end of summer semester 2024/2025

Instructions

1. Research the post-apocalyptic views throughout the history of humanity.
2. Familiarize with and investigate thoroughly at least two modern publications featuring the themes of post-apocalypse and survival.
3. Introduce briefly the use of Blender and the game engine Unity.
4. Design and create a post-apocalyptic 3D model inspired by past research (points 1 and 2). Include the avatar of the protagonist.
5. Import the designed 3D model into Unity.
6. Create at least three scenarios and program ways of the protagonist's survival in the post-apocalyptic 3D model. Be inspired by point 2. Use the C# programming language.
7. Introduce and evaluate pay-off matrices (known from game theory) for each scenario and establish the optimal strategies for the protagonist.
8. Realise the protagonist's movement and actions as an interactive game, where the protagonist trains optimal survival strategies.
9. Test the game and evaluate the feedback of users.
10. Publish the 3D game on an appropriate public platform (can be done under a pseudonym/different username).

Electronically approved by Ing. Radek Richtr, Ph.D. on 13 June 2023 in Prague.

Bachelor's thesis

POST-APOCALYPTIC MODEL AND TRAINING FOR SURVIVAL IN VIRTUAL REALITY

Duong Duc Hoang

Faculty of Information Technology
Katedra teoretické informatiky
Supervisor: doc. Ing. Mgr. Petr Klán, CSc.
January 11, 2024

Citation of this thesis: Duong Duc Hoang. *Post-Apocalyptic Model and Training for Survival in Virtual Reality*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2023.

Contents

List of Figures

List of Tables

List of code listings

I want to thank my supervisor, doc. Ing. Mgr. Petr Klán, CSc., who supported and supervised me throughout this bachelor thesis.
I also want to extend my gratitude to my family and friends for supporting me, encouraging me, and always being there for me. Finally, I'd like to thank Ninomae Ina'nis for helping me through tough times and inspiring me to try more art-related projects.

Declaration

In Prague on January 11, 2024

Abstract

This thesis deals with virtual models of surviving a post-apocalyptic world in predetermined scenarios. The technologies used are briefly explained, and the history of post-apocalyptic views throughout history and two publications of the same genre are analyzed. The output of the thesis is built as a video game for virtual reality created using the Unity Engine and C#, and is available to be downloaded and played from the author's GitLab page. The 3D assets are created in Blender, with the assets created by the author ranging from the small props to the player's avatar and even full environments. Scenarios are inspired by the aforementioned analysis, and pay-off matrices have been calculated describing the optimal way of solving them, which are fine-tuned using feedback from various play-testers. The game itself has the ability to evaluate the player's choices for survival training.

Keywords post-apocalyptic game, scenario-based training, virtual reality, Unity engine, C#, Blender

Abstrakt

Tato práce se zabývá virtuálními modely přežití v post-apokalyptickém světě v předem daných scénářích. Použité technologie jsou ve stručnosti vysvětleny společně s historií post-apokalytického myšlení napříč dějinami a dvěma publikacemi týkající se daného žánru. Výstupem práce je počítačová hra ve virtuální realitě vytvořena pomocí Unity Engine a C#, a je dostupná ke stažení a hraní na GitLab stránce autora. 3D modely jsou vytvořeny v programu Blender, kde autorem vytvořené modely zahrnují drobné věci, avatar hráče a také celá prostředí. Scénáře jsou inspirovány výše uvedenou analýzou a mají vypočtěné pay-off matice popisující jejich optimální řešení, která byla laděna podle zpětné vazby od uživatelů testujících hru. Hra samotná má schopnost vyhodnotit hráčova rozhodnutí a šanci na přežití v těchto scénářích.

Klíčová slova post-apokalyptická hra, trénování na scénářích, virtuální realita, Unity engine, C#, Blender

List of abbreviations

2D	Two Dimensional
3D	Three Dimensional
API	Application Programming Interface
BSDF	Bidirectional Scattering Distribution Function
CPU	Central Processing Unit
DOF	Degrees of Freedom
EEVEE	Extra Easy Virtual Environment Engine
GPU	Graphics Processing Unit
HDRP	High Definition Rendering Pipeline
HLSL	High Level Shading Language
LWRP	Lightweight Rendering Pipeline
PBR	Physically Based Rendering
RGBA	Red Green Blue Alpha
SDK	Software Development Kit
URP	Universal Rendering Pipeline
VR	Virtual Reality
XR	Extended Reality

Introduction

This chapter is reserved for a quick introduction to the thesis at hand, in which the individual parts of it and the objectives of this thesis are described. At the beginning, the problem and the genres are also briefly explained.

Post-apocalypse themes are a common topic in media, especially in literature and video games, most likely because they are very flexible, leave a lot of space for creativity, and allow exciting ways to build unique settings and worlds. The apocalypse that leads to the end of the world varies often in what has caused it and how it has started. One of the common narratives is wars between countries; at times, they can be elaborated with nuclear weapons being used to destroy the world. The media then often focuses on different aspects of survival.

Surviving in a harsh environment is also a very broad topic, especially mixed with the fact that not all apocalypses are equal, as mentioned above. In modern post-apocalypse media, more often, it is about communicating with other survivors or groups of survivors, trying to solve conflicts, and exploring the topics of greed and trust in a lawless world. Other media, primarily games, can focus on the scavenging and exploration of the world, with the player attempting to gather resources that would further help them survive the following days.

Video games often allow us to play as different people in different worlds, which is why many of them strive to be as immersive as possible. This aspect can be done either by stellar writing of characters and their dialogues, having the player's choices affect the world and the surroundings, or by simulating the aspects of the real world in the game as faithfully as possible, allowing the player to do whatever they want. The latter is an especially powerful concept in virtual reality, where the player's vision is replaced with the video game, and their hands directly control the in-game hands.

The final product of this thesis is a virtual reality game set in a post-apocalyptic world, with the only people in there being the players themselves. The target audience is people who wish to train for survival in selected scenarios or seek enjoyment in exploring worlds devoid of civilization.

The first part of this thesis is dedicated to briefly explain the terminologies and concepts used throughout the thesis.

How the post-apocalypse views have developed throughout humanity will be the topic of the second part of this thesis, alongside an analysis of different works and media the author has decided to use as inspirations and the technologies used to produce the final product. The inspirations will range from simple floor plan designs to core ideas of scenarios.

As the deliverable of this thesis is a video game, it requires visual materials and logic regarding how the player interacts with it. Therefore, the third part of this thesis is dedicated to how the assets are created, how the player controls themselves in the game, and how their interactions are implemented. It will also contain documentation on how sound effects are done in response to the player's action.

The fourth part of this thesis is solely dedicated to the three scenarios the author has come up with for this thesis, which are based on the works analyzed in the second part. Each scenario has a dedicated section outlining its goal, what inspirations were explicitly taken, how the author has envisioned the scenario to be solved, how it looks, and the unique assets used in it. As part of the assignment, pay-off matrices are provided for each scenario, outlining the player's strategies in opposition to the environment.

The last part of this thesis documents the experiences of various users selected to test the game and its scenarios and how their feedback has influenced the development.

Objectives

Amongst the objectives of this thesis are two analyses, one of which is to perform research on a few post-apocalyptic views throughout the history of humanity and how it has been developing with the state of the world at its time, and the second being an analysis of two modern publications.

For the assets, a 3D manipulation program called Blender will be used to create them, with the game being done in the Unity game engine using the C# programming language. The thesis's final product is a game played in virtual reality with worlds/scenes that the player can freely interact with and walk in.

Using the aforementioned program Blender, the models created will include the player's avatar, environments, and props in them. The creation of said assets will also be documented, with the elements obtained from third parties credited. The assets are then placed in a scene accommodating a scenario whose creation and details would also be described. For each scenario, pay-off matrices will be provided to establish the player's optimal strategy.

A few people with different levels of experience using virtual reality will be entrusted with testing the game and providing feedback to help further fine-tune the scenarios and possibly add additional interactions with the world.

At last, the game will be published with prior approval from the thesis supervisor on the faculty's computers and on the author's GitLab page, where it will be free to download and play.

Chapter 1

Terminology

This chapter briefly explains the terminologies used in this thesis, including the basic concepts of 3D graphics and rendering. The individual properties of materials in physically based rendering systems used in this thesis and the concept of virtual reality are also explained.

1.1 Basic concepts of 3D

The most fundamental concept in 3D graphics is an ordered set of three dimensions. This set is often called a “vector” and can carry different meanings depending on the context (e.g., position, scale, direction, …). The individual values in the set are often referred to by the letters x, y, and z, respectively. A point in a three-dimensional world would be specified using a single vector, determining its position or, in other words, its offset from the world’s zero point[1] [1]. Points that are connected to each other are called polygons, and three connected points would then form a triangle, which is the most primitive shape that is visible to the human in computer graphics, with most graphics hardware being optimized for handling triangles. While it is possible to form a polygon using more points, it can have different representations based on how the application draws the given polygon [2]. Figure 1.1 shows an example of this behavior. The way the vertices are connected with each other forms a topology, which may not be as important for static models but is crucial for models that are supposed to be bent or deformed (e.g., body parts).

Models are built from a mesh, materials, and textures [3]. Meshes are made of multiple polygons, most often triangles, connected to each other to convey a more complex shape. Materials determine the appearance of either parts or the entire model and can utilize textures as a data source. Textures are then applied to the mesh accordingly with the materials, conveying its visual appearance in terms of color or any additional detail that would not be computationally feasible using polygons [1]. Applying a texture, usually a 2D image, onto a 3D mesh is often done using UV maps[2], which specify the coordinates of the individual points of a polygon on a 2D plane. They are also commonly referred to as texture coordinates [1]. The process of creating these UV maps is called “UV unwrapping”. To simplify the thought process, one can think of it as deconstructing a papercraft model, going from the final model and unwrapping it onto a paper. For the 3D software to know how to unwrap the model, edges have to be marked as “seams”, which in papercraft terminology would be where it would need to be cut to flatten it with the least distortion or stretching [4, 5].

[1]Also known as the world origin.

[2]Similar to how 3D vectors use the letters x, y, z, UV maps use the letters u and v. In some applications, they are referred to as s and t coordinates.

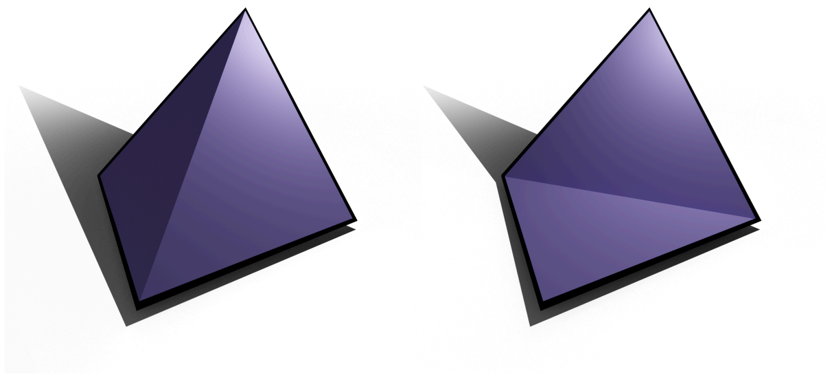


Figure 1.1 Same set of four points forming a polygon; however, each calculated in a different order. A light source is added to help demonstrate the shape difference.

A three-dimensional world is often called a Scene, which is described using objects. Each of these objects holds information about their position, rotation, and scale and can be either nothing[3], models, or lights. Objects can be parented to another object, which would make changing the parent's transformation affect its children, but not the other way around. These relationships are often described in scene graphs. [2]

```
SceneRoot
├── barrel_03
└── small_wooden_table_01
    ├── food_apple_01
    └── marble_bust_02
```

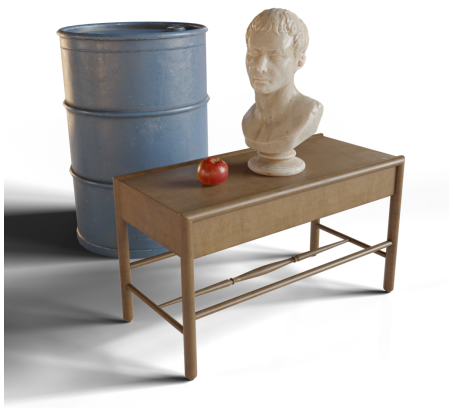


Figure 1.2 Example of a simple scene graph. Moving the table would affect the apple and the bust; however, moving the barrel would not change any other object in the scene.

For a model to be set in motion, one can update the transformation values in the scene graph over time. Doing so would only allow the model to be moved, rotated, or scaled with no deformation to its shape as if it were rigid. For cases where we wish to deform the model in a hierarchical way, the model needs to have bones and a numerical value called "weight" for its vertices. This method is often referred to as "skinning," with the process of adding weights called "weight painting" [6]. Bones can be considered standard empty objects that store the basic three transformations, and each vertex can have a weight value correlating to the said bone. The weight determines how much the vertex would change in relation to the change in the bone's transformation. [1]

[3]Often used to group objects to transform them together or to serve as bones in rigs.

1.2 3D Rendering

In computer graphics, scenes, models, and objects are typically stored as data, which can be difficult for humans to visualize. A visual representation of the scene is created by a process called rendering. In this thesis, two rendering techniques were used depending on the use case: rasterization and path tracing.

1.2.1 Rasterization

Rasterization is a very fast type of rendering, usually done on a GPU using graphics API such as OpenGL, DirectX, Vulkan, or Metal. It is often used in game engines or workflows where real-time previews are necessary. While it is fast, it may not be physically accurate, as in most cases, they are approximating the lights' behavior. On input, the rasterizer receives the geometry assembled from primitives, such as the triangles mentioned above, and returns a fragment in a pixel grid. [7]

A fragment is a set of values of the geometry at a given position on the raster, meaning that it has no visual appearance. For that, special programs called *pixel shaders*[4] are used to give the fragment a color, making it a pixel on the screen. [2]

When rendering opaque objects, they can be rendered in any order since each of the fragments usually holds a depth value, which is the distance of the geometry at that point to the camera. This method is called the "Z-Buffer Algorithm" since the depth values are often referred to as Z values and are stored in a buffer. If two fragments are drawn at the same pixel, the one closer to the camera wins and is displayed instead of the other one [8]. This behavior is referred to as "over-draw" [2] and can be modified, commonly to make an object visible even if it is occluded [9]. In the case of transparent objects, however, they must be rendered after opaque objects and then in order from the outermost transparent object to the closest one [1]. This approach is called the "Painter's algorithm", as it performs similarly to how an artist would paint with perspective in mind on a canvas.

Shading and lighting are usually also done using pixel shaders, and there are two approaches[5] to this called Forward and Deferred. Both of them are used in the industry; however, each of them has a set of advantages and disadvantages, both in terms of performance and abilities, and are usually chosen based on the use case or the scene [10].

Forward shading works by going through each of the fragments and lights in the scene to calculate the final appearance and can be done using a single pixel shader. This approach can be very computationally demanding in scenarios where many dynamic lights need to be calculated in a dense environment since computing time might have been spent on lighting operations on fragments later drawn over by a different fragment.

Deferred shading works by initially calculating various information about the scene into "*G-Buffers*"[6], which are then used in a second stage, which performs the lighting calculation. In this case, all of the lighting that is being calculated is visible and does not go unused. The downside, however, is that the G-Buffers can be very large in terms of memory and are unable to render transparent objects, among other limitations. In Unity, for example, transparent objects are rendered using the Forward method after the Deferred method has finished [2, 11].

[4] Also known as Fragment Shaders in some APIs
[5] Commonly referred to as rendering paths
[6] Various buffers containing information of the rasterized scene.

1.2.2 Path Tracing

Path tracing attempts to render the scene by faithfully simulating the light rays in the scene as they are being emitted, reflected, and scattered to find their path from the camera to the light source [12]. This method is very performance-demanding; however, the quality tradeoff for it is often worth it. It is often performed on the CPU; however, it has also recently been possible to use GPU for such tasks. Nowadays, it is even possible to render scenes in real-time with acceptable quality thanks to advancements in GPU technology.

This method is not used to display the final game in this thesis since it would be a tremendous task to have it work fast enough for virtual reality. It is, however, used to prepare textures for 3D models using a process called "baking," which is the process of calculating time-consuming tasks (e.g., lighting) and storing its results in a texture. Doing so allows the models to be viewed with better lighting in real time.

1.2.3 Physically Based Rendering

Physically based rendering, in a sense, are guidelines that help the artist describe the materials in the scene and how they should interact with light and its surrounding environment [13]. It is commonly used in computer graphics, both in video games and movie industries, to the point that it has become the de-facto standard in the industry [14], allowing these materials to be exported from one software and imported to another with little to no additional work. A material can be defined as a set of properties that dictate how the light interacts with the surface on which it is used. These properties can be encoded in either a numerical value, meaning that it applies to the entire material, or using a map, which would only affect specific areas of the material.

Two workflows are commonly used: the *specular workflow* and the *metallic workflow*. The former is more flexible in terms of artistic freedom, allowing the artist to change the color of the specular lighting, which may lead to light interactions that would not be possible in real life [15]. The latter dictates the color of the specular lighting using a metallic map, which, as the name suggests, determines which portion of the material is made out of metals, which is more useful, as it will follow more faithfully how light behaves in the real world for most common materials. For the entirety of the thesis, the metallic workflow is used, as it is the default in both Unity and Blender [16, 17].

For most materials in this thesis, only a subset of properties are used to describe them in a believable way:

Albedo: Describes the color of the material. In most cases, it should be a texture with no shadows and shading for the best results, as it is the renderer's job to provide them. Including shadows in the albedo can cause overlapping shadows or misdirected light, resulting in physically inaccurate renditions. [14]

Roughness: Determines how rough the surface is in the material, which can make a material have a glossy or matte appearance. In some software, a value called "Smoothness" or "Gloss" is used instead, which is just the inverted value of roughness. [18]

Metallic: Describes which parts of the material are made out of metal. Usually, the map or values should be set either to 1.0 for metallic or 0.0 for dielectric, with values hardly ever being anything in between. This is because a material described should have a clear-cut distinction whether it is metal or not. [14]

Normal Map: It is a three-dimensional map representing the offset direction of the surface normal on which it is being projected [18]. This map allows additional details to be added that would have been hard to portray or expensive in terms of performance in a mesh using polygons [19].

Emission: Determines which parts of the material emit light and with what color and intensity [18]. Since most materials do not emit light, most materials have this set to pure black, meaning it does not emit any light.

Many other maps can, for example, describe the material's transmission properties down to their IOR[7] values, or subsurface scattering maps that determine which colors of the incoming light are being scattered as it penetrates the surface; however, they are most commonly used in specific materials such as glasses and skin. [17]

These maps can be created procedurally using specialized software such as Adobe Substance Painter or Blender or can be approximated by utilizing the photogrammetry techniques on a material in real life. Procedurally generated materials are often specific to the given software or computationally expensive to calculate all the time and need to be baked into a texture for the maps to be used in other software. The process and workflows describing baking are further elaborated in Chapter 5.

1.3 Virtual Reality

Virtual Reality (VR) is the concept of creating a world in a virtual space, which is often viewed using specialized equipment called virtual reality glasses or headsets. The most basic ones contain one or multiple screens, which are used to display the virtual world to the user, and a gyroscopic sensor, tracking the user's head rotation. These headsets are called 3DOF[8] headsets since they allow the user to track their head's rotation only. The more advanced ones include more sensors (e.g., accelerometers or light sensors), which track the headset's position and are called 6DOF headsets. Most VR headsets nowadays include controllers, which can be either 3DOF or 6DOF, similar to the headsets themselves. They are used to interact with the virtual world in a greater capacity, usually to give the user virtual hands. [20]

The most common issue people have with VR headsets, especially newcomers to the technology, is the feeling of discomfort after using them, such as nausea and motion sickness [20]. Motion sickness generally happens when the movement perceived by the player's eyes differs from what is actually happening around them. This mismatch is often caused by insufficient framerate, meaning that the eye would have jittery visuals or, at times, by design (e.g., fast-paced action games).

An avatar is used to represent the user in the virtual space. In virtual reality, avatars are essential, as without them, users would have trouble orienting themselves in the virtual world. These avatars can be as simple as floating models of hands or controllers or as complex with full-fledged characters, which may be bipedal or even quadruped if one so desires, as there are no theoretical limits, only practical constraints to how one would be controlling the avatar. Thanks to the limitless potential, it is an excellent way for people to personalize themselves, be unique from others, and become who they wish to be.

[7]Index of Refraction
[8]Degrees of freedom

Chapter 2

History of post-apocalypse views

The first part of the research in this chapter is mostly based on David Paulík's master thesis [21] and has been the primary inspiration on how to approach this topic.

According to Dictionary.com [22], the word “apocalypse” has different meanings depending on the context. For one, it can mean the revelation of a prophecy, in which case it would mostly be linked with religious texts. The word originates from the Greek translation “apoálypsis”, meaning “uncovering”. In modern understanding, however, it is mostly attributed to the disastrous end of civilization as we know it. The dictionary has noted that the latter meaning was not recorded until the late 19th century.

Most of the apocalyptic texts from ancient times are primarily prophecies describing a revelation of sorts. Noah's Ark, described in the Old Testament, can be considered to be apocalyptic in the modern understanding of the word. In these texts, it is described that God has decided to essentially reset the world using floods, effectively killing anything on the planet. The reason why God has decided to do so is because they saw “how corrupt the earth had become, for all the people on earth had corrupted their ways.”[1] However, God has spared Noah and his family and has assigned him to build a ship to house them and a pair of each of the animal species.

Within the New Testament, the last book called the “*Book of Revelation*” which the author of the aforementioned master thesis refers to as a prophecy, preaches about “*the events that would transpire in order to cleanse the world from its sins, fight between good and evil and the eventual victory of God and his kingdom on earth*” [21]. The author of the thesis mentioned, alongside various sources on the internet, that the book has different ways of understanding the text, with the former even providing four different perspectives.

During the Middle Ages in Europe, the black plague became a pandemic and took many lives. According to Paulík's thesis [21], due to the limited knowledge of diseases at the time, the plague was attributed as a divine punishment for their sins.

In recent years, the author of this thesis remembers two significant years when the world was thought to be ending, the years 2000 and 2012.

The mass panic that occurred shortly before the year 2000 was caused by a software bug, often dubbed the millennium or Y2K bug, where the years were stored as a two-digit number, with the assumption of appending “19” in front of it, meaning that once the year had gone to 2000, the internal representation would be “00”, with no way of determining if it was 1900 or 2000 [23]. What people were afraid of was that various computer systems would malfunction and cause havoc across the globe. While such claims were not necessarily unfound, given that many systems' failures during this period were attributed to the bug, it was not at the apocalyptic scale [24]. According to an article from Forbes [25], various people attributed this bug to the end of the world, linking their beliefs to the aforementioned “Book of Revelation”.

[1] Genesis 6:12

The year 2012, or specifically the 21st December of 2012, was thought to be the end of the world, with its basis being solely on the Mayan Long Count calendar ending on that date. Apart from this fact, there are no Mayan inscriptions depicting that date as apocalyptic [26]. The popularity of that year's phenomenon can be attributed to the movie titled *2012*, which is a post-apocalyptic movie based on the aforementioned premise, with the expectation of natural disasters such as earthquakes, volcanic eruptions, and floods occurring. Interestingly enough, it appears the movie has been inspired by the story of Noah's Ark, as near the end of the movie, they have built similar arks to preserve humanity. [27]

One of the prevalent causes of apocalypse in the genre is based on war, more specifically nuclear war. The fear of nuclear weapons being used can be attributed to the two bombs dropped on Hiroshima and Nagasaki in Japan, with the devastating aftermath of destroyed cities, thousands of people dead, and the lasting illness that followed people who survived the impact. The life shortly after the bombing can be witnessed in the work of the Japanese writer *Keiji Nakazawa*, *Barefoot Gen*, which was a manga[2] and later adapted into a movie. It is an autobiographic work, as the author is describing their experience as a survivor of the atomic bomb dropped on Hiroshima [28].

The fear of a nuclear war was not limited to just Japan, however. In the West, after witnessing the effects of said weapons and the rising tension between the United States of America and the Soviet Union during the Cold War era, with both sides having nuclear weapons, it was relatively justified. Even at the time of writing, the fear of a nuclear war has waned but not disappeared, especially with the ongoing world politics and conflicts.

The Cold War has also served as inspiration for a lot of movies, games, and books. *Fallout*, the video game franchise inspired by the tension of the Cold War, is one such example; however, with the deviation of it being the USA against China [29]. Another work that has taken inspiration from the potential of nuclear war was the novel series "*Metro 2033*" by *Dmitry Glukhovsky*, who has also quoted Fallout to be one of the inspirations [30]. Both take a similar approach to world-building, focusing on the post-nuclear world with limited resources and the tension and political situation between various groups of survivors.

Another apocalyptic scenario that often occurs in media are diseases causing a pandemic, wiping out large portions of humanity. The fear of elements small enough to be invisible to the naked eye, slowly killing the infected people from the inside out and spreading itself to others unbeknownst to anyone, is entirely justifiable. Even if the disease does not affect the human body directly, it may affect food production and cause an apocalypse by famine. The origins of such fear can be traced back to the biblical books and stories describing various plagues[3] causing harm to people, or their crops and livestock, thus hurting their food source.

The fears of a disease killing the general population have been brought back into relevancy due to the global coronavirus pandemic in late 2019, often referred to as COVID-19. According to the article from Our World in Data [32], there are 6.99 million deaths attributed to the virus as of the time of writing. The focus of the pandemic, however, appeared to be riddled more with conspiracy theories and political beliefs rather than about the disease itself. A few notable examples are "*5G antennas are causing the virus*" [33] and "*virus created to force biochip injections into people through vaccines*" [34] among others. The common talks among people on the internet, apart from the conspiracy theories, are often about the poor handling of the situation by the government, with people in power not listening to scientists and claiming contradictory statements, and people flat out refusing to acknowledge and attempt to prevent the spread of the virus by any means. Some have also drawn a parallel with apocalyptic movies, previously describing them as unrealistic for the portrayal of ignorance of the disease, now acknowledging that such events are possible.

[2]Japanese comic books

[3]According to Dictionary.com, it may be "any widespread affliction, calamity, or evil, especially one regarded as a direct punishment by God". [31]

Apart from diseases causing mass extinction, a spin-off of this topic has recently seen a rise in popularity, that being the zombie apocalypse. A zombie is often described simply as "*an undead creature with a reanimated human body [...]*", with the origin of the word traced back to Hiatian's word "zonbi" [35]. While in their culture, it symbolized the dead being reanimated to be a slave to their master.

> "*The concept of the zombie represents the newest of the group [of living dead]*[4]*, initially popularized by sensationalist travel writer Willian Seabrook, [...]. Seabrook's influential 1929 book The Magic Island told of deceased Haitians, frequently victims of voodoo vengeance, taken from their graves and forced to toil as slaves for their rapacious re-animators.*" [36]

In the modern sense, it has mostly departed from this notion and mainly focuses on the behavioral change of people to attack other people. The cause of the behavioral change would often be attributed to some disease, and for it to spread from one person to another, it would usually be by biting them. As these beings are already considered dead, they are unable to be killed by regular means. Zombies often need to be shot through the head, similar to how vampires in folkloric stories need to be stabbed through their hearts. One of the first movies that portrayed the zombies in this modern understanding would be George A. Romero's movie called "Night of the Living Dead" [37], where the zombies are portrayed as slow-walking, cannibalistic, and mindless creatures, with the only method of killing them being either shooting them in the head or burning their body. In a book regarding the behind-the-scenes of the movie, the writers have been inspired by the apocalyptic novel "*I Am Legend*" by Richard Matheson [36, p. 22]. While it is not about zombies but about *vampires*, it does share similarities with the movie about the disease-based apocalypse, the spread of infection through bites, and the limited methods of disarming them being a specific strike-zone on the infected.

[4]This was added by the thesis author, as the quoted text refers to the group mentioned before the cited part.

Chapter 3

Analysis of existing media

This chapter will present a brief analysis of four modern publications that feature the theme of either post-apocalypse or survival. The subjects of this analysis are the Fallout franchise, Half-Life: Alyx, Introverts Till the End and Girls' Last Tour. The author has also taken inspiration from each during the creation of the scenarios in the thesis.

3.1 Fallout franchise

Fallout is the long-running series of games set in a nuclear post-apocalypse, initially created by Interplay Entertainment, with later series done by Bethesda Game Studios and additionally by Obsidian Entertainment, who helped to produce the game Fallout: New Vegas.

The games that the author had played were Fallout: New Vegas [38] and Fallout 4 [39]; however, they were inspired mainly by the latter since the former focused mostly on the political side of the post-nuclear world, while Fallout 4 had more emphasis on scavaging and collecting materials for crafting purposes.

Exploring the open world, commonly referred to as the Wasteland, can lead the player to various locations, which can be filled with enemies, destroyed buildings, caves, and many other points of interest. In these areas, there are usually a lot of items scattered around, which the player can pick up. Some may be on the ground lying around, on the table, or even stored in shelves, boxes, and cupboards. Across the environment, the player may also stumble upon many radio stations, which can be just transmitting music, instructions, or directions towards a certain place, or even distress signals from individuals or various groups.

The player themself is not limited by the number of items they wish to carry but by the weight limit. It is a soft limit, meaning that the game allows them to carry over the limit; however, they will be handicapped with slower movement and the inability to sprint. Each item also has a "Value" stat, which is the value of the item in caps, the game's universe's currency. Fallout 4 has a simplified crafting and scavaging mechanic, where instead of a crafting recipe requiring a specific item, it requires general materials (e.g., metal, wood, oil, screw, ...) that can be obtained by decomposing found items. An example of such an item can be seen in Figure 3.1.

In the Wasteland, during one of the quests, the player will be given a chance to visit a place called the "Old Corner Bookstore", which was used further as an early inspiration for the building's layout of the library in the scenario that will be described in Section 8.1. When the player enters the building, they are met with a front hall, rows of bookshelves, a reception desk, and stairs that lead to the second floor of the bookstore. Both of these examples can be seen in Figure 8.2.

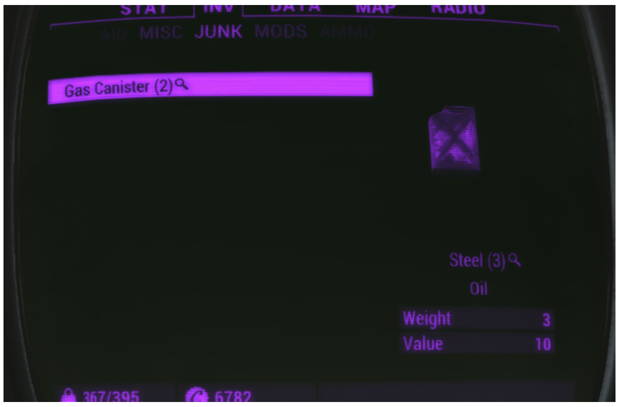


Figure 3.1 Example of an item called "Gas Canister", which can be decomposed into 3 Steel and one Oil, weighs 3 units, and could be sold under 10 caps.

3.2 Half-Life: Alyx

Half-Life: Alyx [40] is a prequel to the popular video game franchise Half-Life by Valve, released in 2020. The game is set in a dystopian world, where an alien race called the Combine has enslaved humanity. It is a virtual reality game that has won the 2020 Game Awards in the "Best VR/AR" category [41] and is shipped with Valve's VR headset (called Valve Index) [42].

The main inspiration the author had taken from this media was the control scheme of the player in the environment, specifically the continuous motion mode controls. The continuous mode essentially moves the player through the world without teleporting or sudden changes in velocity. The other modes left are "Blink" and "Shift", which teleport the player with different ways of transitions. The thesis will only work with continuous locomotion for full immersion with the cost of possibly causing motion sickness.

Continuous mode allows the player to move themself using the controller's joystick, which can move the player in the direction of either the hand or the head, depending on the player's preference. During the move, the collision model of the player moves with them, with gravity's force being applied to it. It will also collide with objects more crudely. When the player has stopped using the joystick, the collision is more lenient, as the player's collision shape is now only around the head and not the entire body. This allows the player to lean over tables, look out of the windows, peek into shelves, and many more actions. Figure 3.2 demonstrates this behavior. In cases where the head collides with anything, the player's vision is obscured with only a small window, into which they are required to move their head and body in order to resume the game. The game also permits the player to teleport if they hold the joystick back on their primary hand, aim at the desired location they wish to move to, and release the joystick.

Turning the view around to look behind or adjusting the player's orientation can be done by physically rotating themselves in real life or flicking the joystick horizontally. Similar to how the movement is handled, as mentioned above, the rotation can be instantaneous or gradual, depending on the player's preference. The latter is, however, more disorienting and prone to cause motion sickness.

As mentioned before, the force of gravity is only applied to the player when their collision shape is in the air or when they are moving around using the joystick. Because of that, it is possible to physically move your real body off a virtual ledge. To prevent this, the game has a certain leniency when the joystick is untouched. However, if the head moves too far from the collision shape of the body, the game forces the collision shape of the body to be updated to be below the head, after which, if the player's head were indeed located off a ledge, they would fall. A diagram showcasing this behavior is shown in Figure 3.3.

Within the game, there are various levels where one has to go up a ladder, and the game allows multiple ways to perform the ascension. The player can climb manually step by step by grabbing the ladder and pulling downwards, effectively lifting themselves off the ground. Once the player has been lifted a predefined amount, the game teleports them to the top platform. Another way to ascend is to aim the controller at the ladder and hold down the teleport button, and after some delay, releasing the button will teleport them to the top as well. The same methods are applied when the player needs to descend on a ladder. In most cases, the former is physically impossible, as the location of the ladder in the physical world would have been in the ground and unreachable by the player.

Another point of inspiration was how the game handled the player's hands, as they are part of the physics world, and items such as buttons and levers have to be physically pushed or pulled to be activated. The hands also act as the player's avatar, representing them in the virtual world.

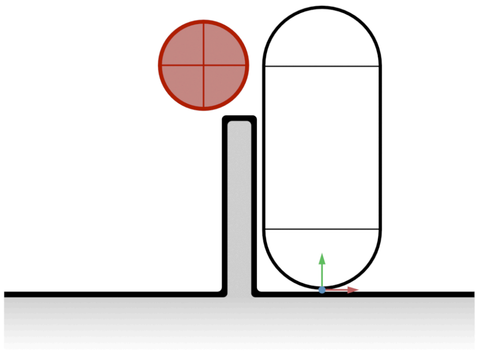


Figure 3.2 An example of the head having separate collision bodies from the player controller, allowing the player to lean over a wall. The capsule represents the player controller, while the tracked head is a red circle. The gizmo axis represents the origin of the tracked player space.

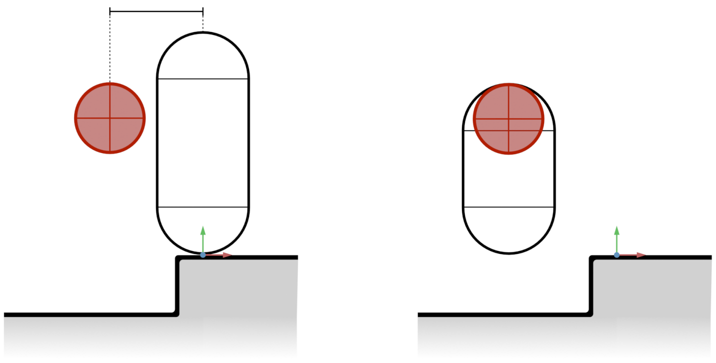


Figure 3.3 An example of the controller character updating to match the tracked head position. The capsule represents the player controller, while the tracked head is a red circle. The gizmo axis represents the origin of the tracked player space. On the left, the distance between the head and the controller is still within the threshold. When this threshold is crossed, the player controller is updated by moving its center and adjusting its height. After this update is done, the player will fall.

3.3 Introverts Till the End

Introverts Till the End [43] is a fanfiction[1] by an author who goes by their pen-name “tripsout2”. The author writes about characters from a group of online streamers who use animated characters to represent themselves[2] called Hololive, specifically the Hololive English Myth group and its five members, and included them in an alternative universe located in a zombie post-apocalypse of unknown cause.

The story is set in winter and is told from the perspective of the protagonist, Amelia Watson, with the premise that she is a lone wolf surviving the apocalypse with her dog and not knowing any of the other girls. The first few chapters are about how she met the other members who travel together. Each encounter with them is set in a different location, circumstance, and activity over which they bond. After that, she develops trust with the traveling group and provides food to them, even killing a deer for them.

Two chapters, specifically the second and fifth, have been used as inspirations for various aspects of this thesis’ scenario designs, which are further elaborated in Section 8.1 and Section 8.3.

In the second chapter, the narrator describes that winter is beginning, and Amelia requires warmth to survive. In the library, the protagonist encounters Ninomae Ina’nis, a member of the traveling group who arrived before her. They both considered using the books as fuel instead of gathering firewood, as it was the faster option. The paragraph in question can be found in Figure 3.4.

> “*She wasn’t about to light a campfire every night, but she still needed warmth. Warmth required wood. Unless she wanted to go out and hack at trees all day, her best bet was to find something small. She’s also lazy, so she makes a path for the library two blocks away with full intentions to burn literature.*”

Figure 3.4 Paragraph from the second chapter of Introverts Till the End, mentioning the character’s method of obtaining fuel for a campfire [43].

In the fifth chapter, Amelia ventures to a river near the town to gather water with her dog. While the story is set in the winter, the river’s stream is not frozen due to its strong flow. The method of gathering water she used was scooping it from the river using buckets she had brought with her. However, while she was gathering water, unbeknownst to her, her dog had run away, and after a quick investigation, she noticed that the dog’s paw prints in the snow ended at the river bank. Amelia then, without thinking, jumps into the river in an attempt to find her dog, who she presumed has been swept away in the strong stream. It is implied that the flow is strong enough to make a person drift away, as seen in Figure 3.5.

> “*Winter makes it difficult. The river doesn’t freeze because the waters move so quickly. [...] She can’t find where she left her bucket. She has no idea how far she drifted downstream.*”

Figure 3.5 Selected sections from the fifth chapter of Introverts Till the End, describing the river’s flow [43].

[1] Literary work that’s based on a franchise and has been made by its supporters
[2] Commonly referred to as “Virtual YouTubers”

3.4 Girls' Last Tour

Girls' Last Tour[3] [44] is a Japanese comic series written and illustrated by *Tsukumizu*. The story is set in a post-apocalyptic environment of unknown cause; however, there are strong inclinations of war being one of the factors leading to the end of civilization, with only a handful of people being alive and scattered far from each other. The series follows two female protagonists as they explore the wasteland, occasionally stumbling upon a few survivors.

While the author of this thesis has read both the comic series and watched the animated adaptation, all references going forward will be related to the latter. The notable sections for this thesis are located in episodes 3 and 5 of the animated series.

In episode 3, the two main characters are in a scenario where they need to cross a large hole between two sidewalks, with no way of walking around it or a bridge in sight. The protagonists also notice that there is a still smoking cigarette butt on the floor and footprints in the snow, suggesting that someone other than them is in the vicinity. As they look around for any threats, bracing and arming themselves for potential conflict, an explosion suddenly occurs on one of the ground floors of a tall building behind them, causing it to fall over, forming an impromptu bridge between two sides. After that, they meet a survivor coming out of the dust clouds caused by the collapse, who had gathered the explosives and caused it intentionally to form the bridge, as shown in the Figure 3.6.

In episode 5, rain suddenly occurs during their exploration in the wasteland, forcing them to take cover in a cave made of ruins of what seems like a building. The environment resembles a forest; however, all the trees are steel beams stuck in the ground. After they arrive at the cave and settle, one of the protagonists orders the other to use buckets to collect the water dripping from the holes of the imperfect roof. The cave and the surroundings can be seen in the Figure 3.7.

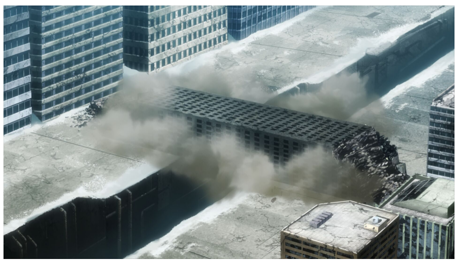


Figure 3.6 A blown up and collapsed building that bridges two sidewalks together.

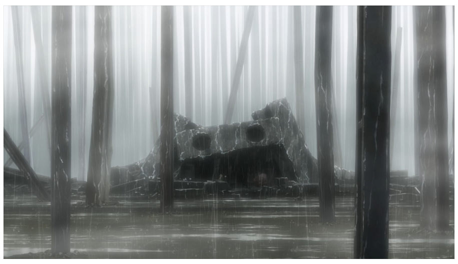


Figure 3.7 The cave the protagonists settled in during the rain, surrounded by steel beams protruding from the ground.

[3]Also known as "*Shōjo Shūmatsu Ryokō*"

Chapter 4

Technologies used

This chapter is dedicated to explaining the technologies used in this thesis, mainly the game engine Unity and its rendering pipelines and physics engine, the 3D modeling suite Blender, and the OpenXR API.

4.1 Unity

Unity, developed by Unity Technologies, is a multi-purpose 3D and 2D engine that has found itself in the gaming, movie, medical, and many other industries. The engine was chosen for its modularity, simplicity, and great support for XR-related applications. The editor is also readily available on Windows, Linux, and even macOS.

The version and edition used in the thesis is 2022.3.4f1 Personal, due to it being the LTS[1] version at the time of writing and includes packages that had proper feature maturity for this project. The Personal edition is also the only Free Tier available without going through any sort of additional verification.

In Unity, Scenes are used to group GameObjects, which are objects that are defined by their transformations (position, rotation, scale)[2]. Their purpose and behavior are then further defined by Components, which can be either written by the user using the scripting language C#, or one of the Unity's built-in components [45, 46]. These components can serve to play sounds, move objects in accordance with the user's input, render a model, and for many other purposes.

Unity provides the user the ability to choose different rendering pipelines, which dictate how objects are going to be drawn on the screen. By default, it provides these three rendering pipelines [47], all of which differ from each other to some degree:

Built-in Rendering Pipeline: It is the default pipeline Unity would use on a completely blank project. While it is the most mature out of the three, it lacks proper visual shader editor support [48] and proper decal support. It is possible to supplement them using third-party packages.

Universal Rendering Pipeline (URP): It was previously known as the Lightweight Render Pipeline[3]. It is very similar to the built-in rendering pipeline, but it is still in development, and thus, a few features that would be there are not available here. However, it supports the visual shader editor, decals, and a few other features, which will be discussed further in the subsection 4.1.1.

[1]Long-Term Support
[2]Same concept as scenes in section 1.1
[3]LWRP

High-Definition Rendering Pipeline (HDRP): It is the most performance-demanding rendering pipeline with great visual fidelity and the most advanced of the three, being tailored for photo-realistic purposes (e.g., movies, architectonic previews, games). It is not suitable for this thesis, given that the performance cost of rendering in VR would be too high for most devices.

In this thesis, the author has chosen the Universal Rendering Pipeline, especially since the aforementioned version has finally received long-awaited features that outweigh the reason to stay on the built-in pipeline. As mentioned before, this is going to be further elaborated in the subsection 4.1.1 below.

4.1.1 Universal Rendering Pipeline

Universal Rendering Pipeline (URP, previously known as LWRP) is directed to be the successor to the built-in render pipeline [49]. The pipeline was chosen based on its features and performance, which made it suitable for this thesis.

As mentioned above, while comparing it with the current built-in render pipeline, there are some features that URP has that the Built-in does not and vice-versa. While the differences in features used to be much greater, at the time of writing, the only feature that was missing and would have been helpful for this project is the Auto-Exposure post-processing effect[4] [50]. This effect simulates how the human eye adapts to different light sources of varying brightness by adjusting the eye's iris and changing the amount of light entering the eye. It mimics this behavior by adjusting the exposure of the camera in a similar fashion, and it allows the developer to add lights into their scenes without worrying about dark areas being pitch black or bright areas being pure white [51].

The main advantages that decided the use of this pipeline were mainly the *Decal system*, *Shader Graphs*, and the *Forward+ rendering path.*, all of which are elaborated below.

Decal system

Decals are essentially textures being overlaid over a model. There are different ways of achieving it. For example, a flat surface applied onto a wall is one method; however, it is limited to only flat surfaces. The system that both URP and HDRP have allows the user to apply decals to virtually any surface of any shape.

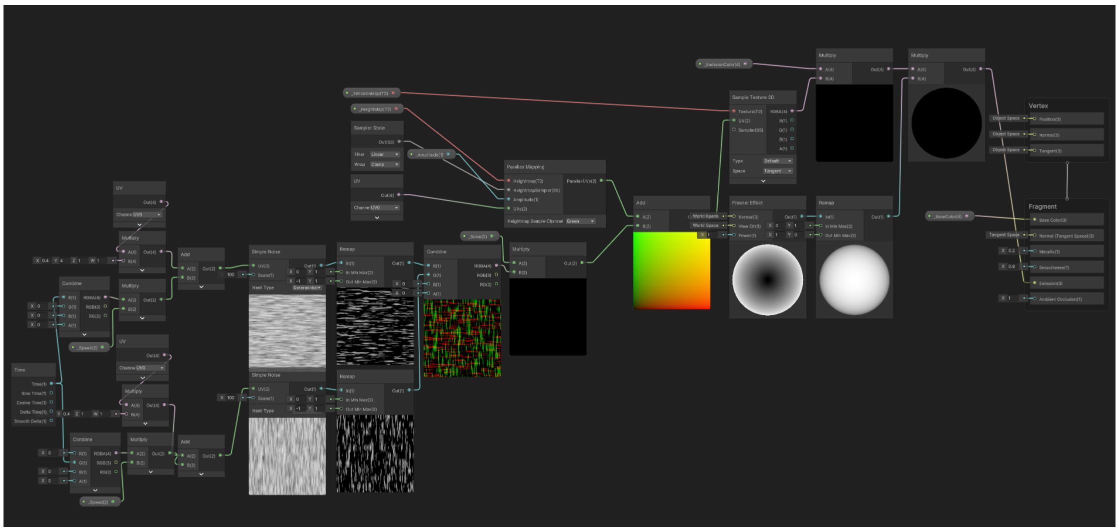
While it is possible to create decals on complex shapes in the built-in render pipeline as well, it does not have the same configurability, flexibility, and user-friendly experience as it does on URP and HDRP. For example, in URP and HDRP, limiting which objects the decal would be applied to is done by specifying the rendering layer that the object belongs to [52]. However, in the built-in rendering pipeline, this would have to be done by changing the Layer of the entire object, which may affect other parts of the engine, most often the physics engine, which also utilizes said layers [53].

Shader Graph

The Shader Graph is available in the current version for all rendering pipelines, including the built-in one [54]; however, it is included only for compatibility reasons [48]. It allows the developer to build shaders using various nodes as an alternative to writing shaders in shading languages such as HLSL[5], the language Unity uses.

Apart from simple color modulation and arithmetic color effects, it also supplies the user with various types of noise generators and even parallax mapping effects without any coding work from the developer's side. [55]

[4] An effect that is performed after the image is rendered, but before it is shown to the user.
[5] High Level Shading Language



■ **Figure 4.1** Example of a shader graph for an old-school monitor screen

Lighting

Lighting in Unity utilizes the light components, which determine what type of light it is, if it should cast shadows, what color, and how bright the emitted light should be, etc. Furthermore, lights can be either dynamic or static, which have to be set in the editor.

Dynamic lights are, by nature, light objects that are going to change their transformation or their light settings. It is calculated while the game is running and will require additional computing power. Because it is done in real-time, it lacks the ability to simulate secondary bounces[6]. [56]

Static lights are lights that are immutable in both their transformation and settings. These lights have to go through a process called lightmap baking in Unity, which is precalculating bounces of light rays throughout the scene against other static objects. The advantage of this process is significantly lower performance requirements and more physically accurate results. However, it lacks the ability to be changed in real time, and the baking process can take a long time, depending on the scene's complexity. [57]

The lightmap baking, however, only gives the appearance to the static objects, while dynamic objects would not benefit from this baking. For this purpose, light probes are used, which are points in space that store the lighting information as if a sphere was being lit in its place [58]. Usually, a scene would contain light probes at positions where dynamic objects could get and places where the lighting could change drastically. The placement of such probes is crucial, as incorrect placement can easily result in lights leaking through walls since an object would gather probes from both sides of the wall, which can be seen in an example scene in Figure 4.2. The object would then choose the nearest light probes based on their origin and calculate the average of the lighting information from said probes.

Additionally, Reflection Probes are used to bake in the reflection of objects by storing a full cubical view of the probe in its place [59]. This stage of baking has to be done after the other baking process. Unity also supports Box Reflection Probes, which attempt to project the baked reflection probe in a cube instead of a sphere. It is especially useful in areas of a box shape, as the reflections will be more visually pleasing and accurate regarding its walls. [60]

[6]It is possible to use Enlighten Real Time Global Illumination, which can simulate this effect, however, at the cost of performance and still requires additional data to be baked.

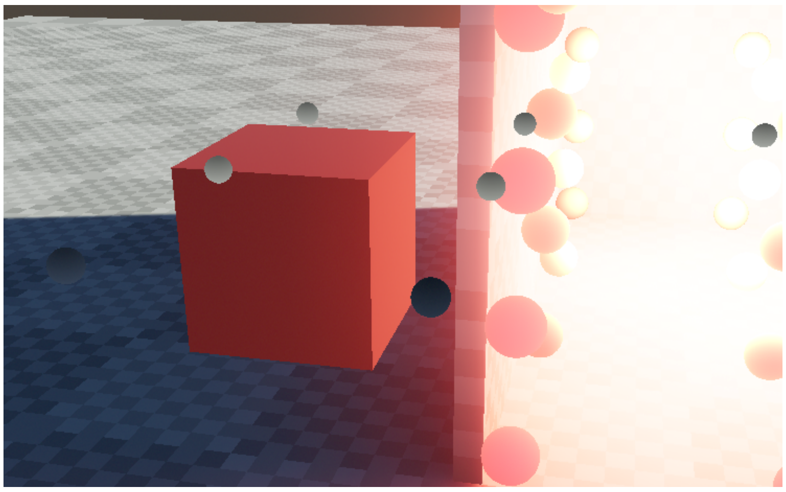


Figure 4.2 An example of a light leak due to incorrect placement of light probes. The individual spheres represent the light probes, which are static, along with the environment, including the room itself. The cube is a dynamic object and colored white; however, it samples more probes that are inside the lit room instead of the outside shaded area, causing it to appear glowing.

Forward+

Forward+ is a rendering path extending the Forward rendering path. It features the ability to have multiple lights in the scene without significantly reducing the performance. This is achieved by effectively culling the lights that do not contribute to the shaded pixel, thus reducing the performance cost of rendering lights that are not visible to the camera. It, however, has the additional hardware requirement of compute shaders, which should have been available to all devices that are supposed to run this thesis at a reasonable performance. The culling of lights is done in tiles instead of per-pixel [61].

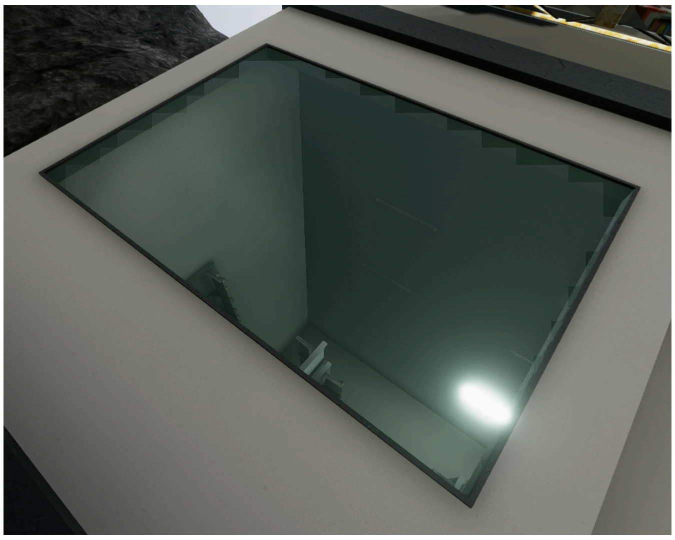
The Universal Rendering Pipeline, since version 14 shipped with Unity 2022 versions, has implemented the Forward+ rendering path. Apart from the flexibility of having many lights in the scene, objects can also blend between two or more reflection probes [62]. This rendering path is not fully supported in VR and results in artifacts, especially near light falloffs [63]. An example of such artifact can be seen in Figure 4.3. Although it may be distracting, this thesis does not use a lot of dynamic light sources and relies mostly on baked lighting. Forward+ in this thesis is mostly for its ability to use more than one reflection probe per object.

4.1.2 Physics System

Unity internally uses the NVIDIA PhysX engine for physics interactions in 3D and 2D [64]. This system is used not only for simulating the physics of various rigid bodies in the scene but also for the player's movement and interaction with the physical world. It could be labeled the "core of this project" for this thesis.

To ensure an object is considered in the physics engine, a kinematic RigidBody component or a collider of any type has to be added to it. However, doing so will make the system consider it a static object. To make it properly dynamic[7], a non-kinematic RigidBody component must be added. [65]

[7]It is possible to update the object's transformation; however, that may result in unstable physics simulation.



■ **Figure 4.3** An example of such artifacts of Forward+ in VR, where jagged edges around the window are visible.

Every object in the scene can have a Collider, which tells the physics engine the shape of the given object. Usually, it is a simplified model of the displayed one for performance reasons. By default, it is possible to choose one of these colliders [66]:

BoxCollider A box-shaped collider is defined by a center position offset and dimensions of said box.

SphereCollider A spherical collider is defined by a center position offset and radius of the sphere.

CapsuleCollider Capsule[8] collider, defined by a center position offset, radius, and height of the capsule.

TerrainCollider Special collider used only on Terrain objects.

WheelCollider Special collider for vehicles and their wheels.

MeshCollider Allows the collider to take the shape of any mesh. For non-kinematic rigid bodies, it is required to be a convex shape with a maximum of 256 polygons. Static mesh colliders or kinematic rigid bodies do not have this limitation.

There are cases where the built-in colliders are not good enough to describe the shape of our rigid body mesh (e.g., trash can), for which Unity allows the creation of “compound colliders” made of multiple colliders belonging to the same hierarchy [67]. An example of such a structure can be seen in Figure 4.4.

Interaction with the rigid bodies is done using forces, which can be applied from code, with an example snippet shown in Code listing 4.1. Another way to do so would be using joints, which are components that can be used to define the relationship between different rigid bodies. The number of joints one object can have is not limited. It is even possible to use different types of joints on the same object simultaneously. There are various types of joints that Unity provides [68]:

[8]Could be thought of as two spheres with a cylinder connecting them

```
simpleChair ....RigidBody, MeshRenderer
├─leg.0 ......................BoxCollider
├─leg.1 ......................BoxCollider
├─leg.2 ......................BoxCollider
├─leg.3 ......................BoxCollider
├─seat .......................BoxCollider
└─backrest ...................BoxCollider
```

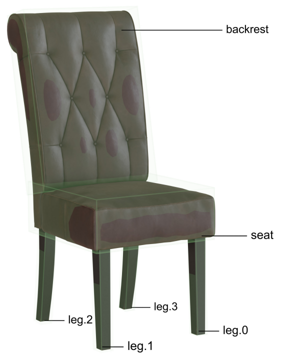


Figure 4.4 Example of a compound collider made out of box colliders, with `simpleChair` being the visible mesh and the colliders being appended to objects that are parented to the said chair

Code listing 4.1 Component that applies a constant force to a given object as long it is active.

```
using UnityEngine;

private new Rigidbody rigidbody;
void FixedUpdate(){
  rigidbody.AddForce(new Vector3(0.0f, 1.0f, 0.0f) * Time.fixedDeltaTime);
}
```

FixedJoint Allows one joint to stick to another, acting as if they belong to each other.

HingeJoint Allows one object to be connected to another using a hinge.

SpringJoint Allows one object to be connected to another using a spring.

CharacterJoint Allows to simulate ball and socket joints, commonly used for ragdoll characters on their knees and shoulders.

ConfigurableJoint Allows the developer to adjust the joint in any degree of freedom.

For the player to traverse the physical world, there is a special component for this specific case called the `CharacterController`, which implements the "collide and slide" technique often used in games. This component also applies gravity and handles collision detection [69, 70].

The "collide and slide" algorithm describes the behavior of a situation where the player wants to move in a way that would result in a collision with an obstacle, in which the sliding comes into play, and the player would slide to the side instead of stopping them. A simple diagram of how this algorithm works is demonstrated in the Figure 4.5.

4.2 OpenXR

OpenXR is an open standard for creating games and applications for both virtual and augmented reality. OpenXR is developed by the Khronos Group [71], which has a rich history of standards that are still used to this day (e.g., OpenGL and Vulkan rendering APIs) [72]. The specifications of the standard not only include technical information about how the individual components need to interact with each other, but it also contains the standard of what controls the controllers should have and even the facial structure for the potential facial and eye tracking features, among other topics.

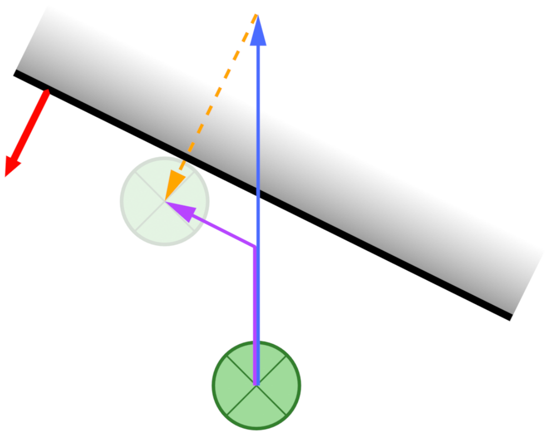


Figure 4.5 An example of the collide and slide algorithm, where the green circle represents the player and the faded circle the resulting position. The arrows represent the various vectors related, with blue being the vector the player wishes to move with, red being the normal vector of the collision the player would encounter, and purple being the resulting move the player would perform.

While it is possible to use SteamVR, a higher level of abstraction, since most headsets nowadays have a way to bridge their API and this one together, it is an additional dependency that is not required for this use-case. Additionally, OpenXR was chosen over SteamVR since it allows the game to be played on many VR/XR headsets and even on different platforms, not limited to PC, with little to no additional work by the developer or the player.

Most controllers provided with an OpenXR-compliant VR headset usually consist of three touch zones: the top, the trigger, and the grip of the controller [73]. The top of the controller usually consists of buttons and a joystick, so it is possible to detect when the thumb is located here either by detecting when the components are pressed or possibly when touched[9]. The trigger is where the index finger is located, and it is possible to detect how deep the trigger is being pressed. The last zone is the grip, which is located below the trigger, from which it is possible to determine how firm the grip is. [74]

4.3 Blender

Blender is a free and open-source 3D manipulation tool licensed under the GPL license, officially available for Windows, Mac OS, and Linux [75]. It allows the user to create 3D models, texture them, and render them into images or animation. The user can also create physics simulations of either rigid or soft bodies or even liquid physics simulations [76]. The version the author used was 3.6 since it was the current LTS version at the time of writing. For this thesis, it was used to create models and textures as assets for the Unity game and to optimize models that were not in an ideal state.

The classic installation of Blender comes with two rendering engines: EEVEE[10] and Cycles, which are rasterization and path-tracing engines, respectively [77, 78]. The former requires a GPU and is ideal for use cases, where one wishes to quickly preview how the model would look or how it would look in a game, given that it renders in a similar way as most game engines. The latter can be run without a GPU; however, having a compatible GPU can significantly reduce its processing time [79]. It is mainly used for scenarios where photo-realism is desired as it simulates individual light rays and their bounces in the scene and to bake textures for models [80]. Both

[9]According to the OpenXR specification, controllers are not obligated to report touch.
[10]Extra Easy Virtual Environment Engine

of them define the materials using Shader Nodes and allow a wide range of customization to be done to the final look. These nodes are then interconnected into a tree structure, which is called a Shader Node Tree [81].

For the PBR workflow found in many industries, the Principled BSDF[11] is often used [17]. As mentioned before, Blender, and subsequently this node, follow the Metallic workflow. If, however, the artist demands, a Specular BSDF node is provided for the Specular workflow [82]. If they wish to deviate from the standard and create their materials, it is possible to do so by combining different BSDF nodes, such as the Diffuse BSDF, Glossy BSDF, and others. Within these Shader Nodes, the artist can even create their textures using mathematical operations and different types of noise generators (e.g., White Noise or Voronoi) to provide them with some entropy. These textures are commonly referred to as procedural textures and need to be baked in order to bring them into different software [80].

Modeling can be done by the traditional box modeling method or by sculpting, both of which are considered “destructive workflows,” meaning that it is not trivial to return to the previous iterations of the model after an edit has been made. In Blender, *modifiers* can be added to models, which, as their name suggests, modify the model in a non-destructive way. It allows the artist to change its parameters later on or even remove it [83]. Out of the box, the artist would be supplied with modifiers such as Bevel, Subsurface Subdivision, and many others. If they want to create their own, then Geometry Nodes, modifiers the user can customize, can be utilized [84]. These nodes, similar to Shader Nodes, modify the geometry with whatever parameters and methods the artist desires, and their usage can be from quickly instancing multiple objects in a set pattern to creating full-fledged buildings. While this is great for editing as it gives the artist freedom and flexibility, it cannot be used in other software directly and needs to be converted into a classic mesh by applying the modifiers, akin to the Shader Nodes.

A quick side note that most production pipelines in the video game industry often use Autodesk FBX, which is the proprietary format for storing 3D models and animations, and due to Blender’s license, they are unable to use the official FBX SDK[12] in Blender. The format has been reverse-engineered to work around this issue, and an import/export addon was made for it. [85] While its features are stable and sufficient enough for most cases, there are caveats in certain areas that require artists to work around said issues. One of those issues, specifically regarding the export, is further mentioned in Chapter 5.

[11]Bidirectional Scattering Distribution Function. A mathematical formula describing light’s behavior.
[12]Software Development Kit – It allows the software to be built upon or to be integrated as part of other.

Chapter 5

Creation of Assets

In this chapter, the process that goes into creating assets in Blender is going to be described, alongside their subsequent export and import to Unity. Most of the work remains consistent for all the models, so it will be only described once on a selected asset, and the ones where the process differs will be further elaborated on. Some assets have been obtained from third parties, so their sources will be documented here.

The assets inside this thesis are both hand-made by the author or borrowed from PolyHaven [86], which are licensed under the CC0 license. The models that the author made follow the same workflow, with minor adjustments to accommodate the specific needs of some models. The general workflow starts from modeling, texturing, and baking, all done in Blender and then prepared to be imported to Unity. Models borrowed from third parties occasionally required additional attention for them to be brought over to Unity, described in Section 5.2.

The modeling process is the process of creating the shape of the model and usually starts with obtaining references. Some models (e.g., water bottles) used references from sites such as Dimensions.com, which contains the dimensions and orthographic views of the subject. A handful of models were designed by the author just by their visual look (e.g., the cabin). After that, the process continues by creating the model with the box modeling technique, which involves building the complex model by composing and modifying primitives (such as cubes and planes). This method is universal for all models; however, for models that have less of a rigid shape[1] an additional method called sculpting can be employed, which allows the artist to add more details to the mesh itself as if they were drawing on it and not by manually moving each of the faces, edges, or vertices around. However, this technique often results in meshes with very high polygon counts, which are unsuitable for video games and, hence, are later reduced during the baking process.

Texturing gives the model its visual look and also describes its physical material. Within the thesis, most of the textures are sourced from CC0-licensed asset libraries such as PolyHaven [86], AmbientCG [87], and Textures.com [88]. In Blender, one can make procedurally generated textures using Shader Nodes, which have been used in this thesis for various purposes, such as mixing two materials based on a mask or a given criteria, or adding scratch marks and wear and tear look to the models. These procedural textures are not easily transferable to different programs, meaning they must be baked into a regular texture that can be used anywhere.

Models and textures are baked to precompute details into more optimized meshes and textures. This is done to increase the rendering performance at the possible cost of reducing the visual fidelity. The loss of quality can be useful for objects that are barely visible to the player, such as faraway objects. Baking of textures usually means creating different texture maps that

[1] An example of a non-rigid shape could be a plastic bag.

describe the model's color, metalness, roughness, and normals. While baking the normals, one can also bake a model using a simplified model with a lower polygon count than the original but roughly looking resembling the original shape. By doing so, the baked normals will make the simplified model shade similarly to the original model with a smaller performance impact.

Exporting is the process of converting the model from Blender to be usable in Unity. This process consists of saving and converting the model to the FBX format[2] in Blender, packing the textures, and then importing them into Unity. Additional precautions must be taken while exporting a model from Blender; otherwise, once the model is imported into Unity, it can have unexpected rotation and scaling issues. The rotation can be partially remedied using Unity's option to "Bake Axis Conversion"; however, that will not solve the object's scale always being multiplied by 100 to be correct. During the exporting process, an experimental option of "Apply Transform" is used; however, due to the limitation of the Blender's built-in FBX exporter, all models can be nested utmost to a depth of two[3].

Texture packing or channel packing is the process of storing multiple textures in an image's individual RGBA[4] channels. The textures that are being packed have to all share the exact image dimensions and describe a single numerical value per pixel. In Unity's URP and HDRP default shaders, textures usually expect what the author has dubbed the "MADS" arrangement [16], which correlates the channels to the maps as described in Figure 5.1, with a practical example shown in Figure 5.1. Packing is done to reduce the memory bandwidth.

Some textures for objects such as magazines and scattered papers were created by capturing local newspapers and newsletters using a camera. The organizations featured in the papers are not affiliated with this thesis and are solely used for convenience.

Table 5.1 Texture packing arrangement used in this thesis. The default column describes what the channel can be replaced with in case the assigned map is not available.

Channel	Assigned map	Default
Red	Metalness	Black
Blue	Ambient Occlusion	White
Green	Detail Map	Black
Alpha	Smoothness[5]	White

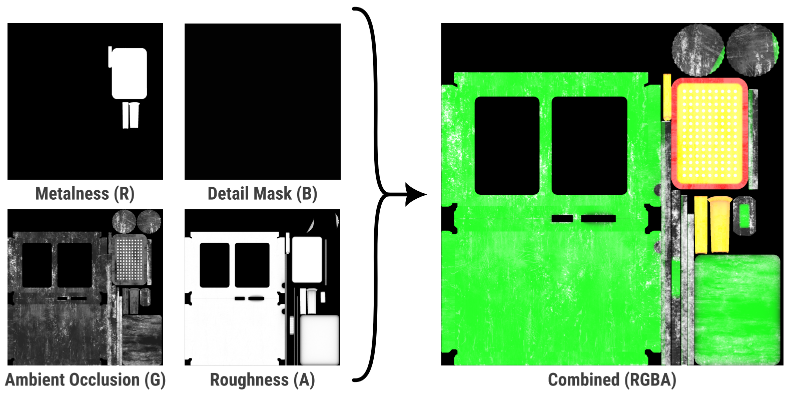


Figure 5.1 Different texture maps being combined into a single texture.

[2]Proprietary format by Autodesk for storing 3D models and animations.

[3]This limitation was already reported to the Blender's issue tracking site, but was closed due to it being an experimental feature [89].

[4]An acronym for Red, Green, Blue, Alpha

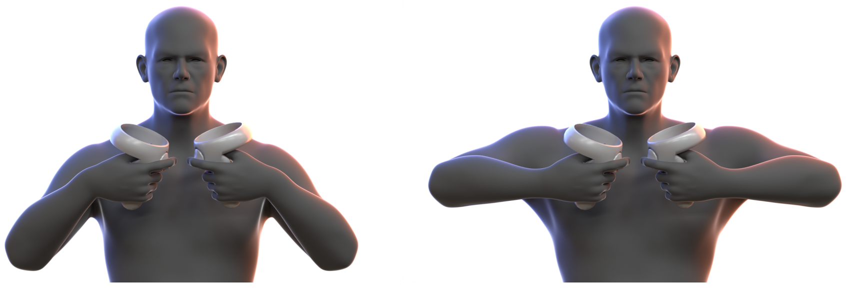


Figure 5.2 Mismatch between virtual and real arms, with the only tracking data used being the headset's and controller's position and rotation. The controller's position and rotation remain unchanged between the poses in the pictures. The male model obtained from Blender Studio Sculpt Base library [90], and the controllers from Meta Quest Developer Center [91].

5.1 Hands

The model of the hands was inspired by the gardening gloves the author had in their family's garden. According to the author, it is quite suitable for a post-apocalypse situation, given that these gloves can be obtained quite easily from gardens or sheds and are durable enough to protect the hand from sharp objects that they may wish to pick up.

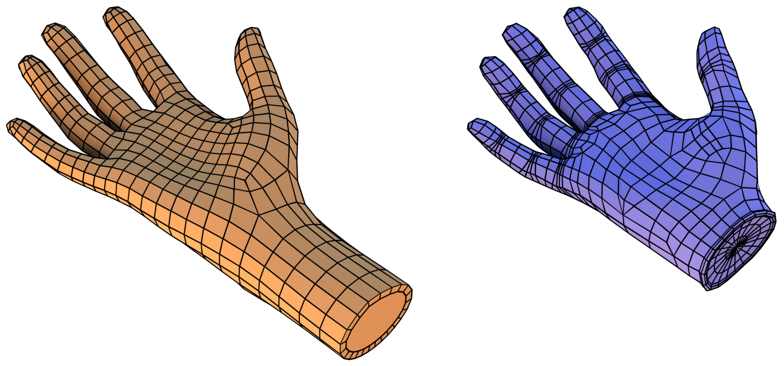
These hands also serve as the player's avatar, similarly to how Half-Life: Alyx portrays the player as described in Section 3.2. This has been done to avoid disorienting the player with limbs that most devices are not capable of tracking, such as arms and feet. While it is possible to extrapolate the tracking data available from hands and apply it to arms, it will not cover all possible positions in which the arms and hands can be. It will often cause a mismatch between virtual and real hands, as demonstrated in Figure 5.2.

The creation process of the hands differs from the other models by the fact it is one of the only organic models explicitly created for this thesis. Instead of box modeling the hand, an existing model from the Blender Sculpt Base library [90] is used. These models, as the name suggests, are designed for sculpting purposes, and their topology is not optimal for this use case since it is expected for the hand's fingers to bend. To remedy this, a process called "retopology" is used.

Retopology is the process of modifying the topology of an existing model while retaining its general shape. This can be done for reasons such as easier manipulation of the model itself [92], adjusting the polygon count to be lower or higher, or, in this case, to ensure the model bends correctly. To do this, the Shrinkwrap modifier in Blender is used. It allows for one model to be cast onto a different one, which is used here to create a new model with proper topology, and by casting it onto the original model, it copies its shape. The result of the retopology can be seen in Figure 5.3

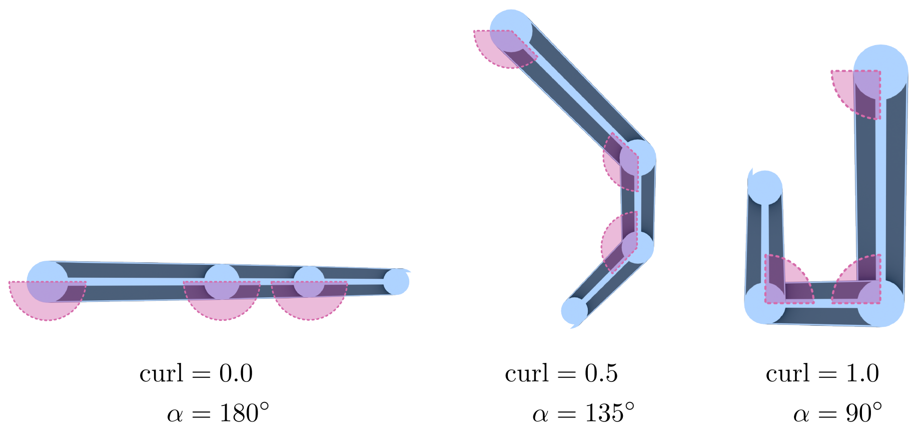
As the hand is required to be animated by the controller's input, it needs to be skinned and weight-painted. To create the bone structure, a Rigify[6] human rig was used. Bones for its limbs, such as legs and arms, were not needed and were removed, so only the hand and finger bones were kept. These provided rigs need to have their bones' position and rotation adjusted to match the model at hand, where the model's topology has aided as it was possible to easily determine the center section of a finger. The automatic weight method from Blender has been used to add weights to the vertices of the model. However, as it did not provide satisfactory results when the thumb was bending, making the hand's thenar bend inwards, a manual adjustment of these areas was necessary.

[6]An addon shipped with Blender that allows simple creation of many different rigs.



■ **Figure 5.3** The original hand model from Blender Sculpt Bases on the left and the retopologized version used in the thesis on the right. The most notable change is the increased density of polygons around the joints on the retopologized version.

Importing the model into Unity follows the same procedures outlined above. An additional script was required to drive the rotation of the finger joints in harmony with the player's input. The script operates in two modes: either it can curl all of the fingers equally or drive each finger using a single value provided by the controller. The value itself is limited from 0.0 to 1.0 inclusive, and they are multiplied against the amount of rotation each of the finger joints receives. Even though these rotations should be up to 90 degrees on the given axis, they have to be "calibrated" by trying them out in-game with different parameters and manually determining what is correct. The way the finger curls in relation to the driving value is simplified in 2D in Figure 5.4.



■ **Figure 5.4** Simplified model of a finger's skeleton and how it curls given a factor. The angles of the joints (indicated by the pink overlay) equate to $\alpha = 180° - \text{curl} \cdot 90°$, where curl indicates the factor of how curled the finger should be.

As it has been established in Section 4.2, controllers following the OpenXR standard have three interaction zones. The thumb is curled when the buttons or touch is detected on top of the controller. The index finger is curled in accordance with how deep the trigger is held down. The grip curls the middle finger, ring finger, and little finger in a similar fashion as the trigger does.

5.2 Third party models

Many of the assets in this thesis are borrowed from the public CC0 licensed asset library called PolyHaven [86], which includes textures, skies, and models. These models are often designed to be used in animations and still renders, not necessarily for real-time purposes such as games, meaning that the models can have a considerably high polygon count. Two methods were utilized in this thesis for the polygon count reduction: manual labor and using a Decimate modifier.

To perform the manual labor, the model needs to have a suitable topology first, from which it is possible to remove edge loops around places where the player would not see and notice the more crudely defined curves. An important note is that removing loops is only possible if they are not acting as a seam to the UV map. If it is, the UV map must be edited or the entire texture rerendered. This method, while providing theoretically the best results in terms of similarity to the original model and flexibility, allowing the artist to redistribute the quality density based on its importance, is very time-consuming. Therefore, the second method was used more often.

Using the Decimate modifier in Blender allows the model to reduce the poly count quickly, however, with little to no attention to its topological structure. The modifier itself has three modes it can work in:

Collapse: “Merges vertices together progressively, taking the shape of the mesh into account.” [93] Using this method often results in stretched and invalid UV maps. It decimates to approximately $triangleCount \cdot ratio$, where $ratio$ is specified by the user.

Un-Subdivide: Reverses the Subsurface Subdivision modifier, basically attempts to find a grid of quads[7] and merge it into one. It requires the model to be subdivided and unmodified after that, which was not applicable in most cases in this thesis, given that their models usually come unsubdivided with the presumption that the software would do it for the user.

Planar: Attempts to merge faces under a certain angle threshold, making them flat. This method also allows the user to limit which faces are to be considered based on their UV seams, making it possible to keep the map intact.

Given the advantages of the Planar method, allowing the existing UV map to be kept, it was used extensively in this thesis. There are, however, some models that the player would be able to inspect up close, so the details were kept intact using an LOD[8] approach. This entails keeping multiple versions of the model in varying poly-count and quality, rendering only one of them depending on how much of it is visible on screen, where distance is often the most significant contributing factor. Doing so allows the most detailed models to be seen up close while displaying a lower quality version from afar that is visually indistinguishable due to the size it is rendered on screen.

[7]Polygons made out of four vertices
[8]Level of Detail

Chapter 6

Player controls

This chapter explains the thought process behind the player's appearance, the control scheme, how it is implemented in the game, and the gameplay mechanics of storing items in a backpack.

The movement of the player is mostly inspired by the game Half-Life: Alyx and its continuous hand control mode described in Section 3.2. The differences between their control scheme and this thesis' were the removal of the ability to climb over obstacles and the ability to turn using the joystick. The reason why the turning system was removed is to enhance the immersion, forcing the player to rotate physically in the real world.

For the player, there is a new space the author has dubbed the Player Space, which contains the tracked objects representing the headset and controllers. This entire space is moved using the joystick and `CharacterController`. This, however, means that the camera/headset will not match with the location of the character controller at all times. Because of this, the `UpdateCharacterCenter` function in the script responsible for the player's movement correctly updates the character controller's position and height to the headset's current position. Furthermore, when moving, the game checks the slope on which the player is located and adjusts the speed of the player, with steeper slopes causing the player to be slower to simulate fatigue on hills. If, however, the slope is higher than a predefined value, the player will not be able to move upwards, only to the side or downwards. The logic of the player controls is contained within the `PlayerControl.cs` file.

In contrast to the game that inspired the author, where the player's vision is obscured if they physically move their head into a colliding object, leaving a small hole that the player needs to move through to resume the game (as described in Section 3.2), this thesis' game moves the entire player space back to resolve the collision.

The hands are also inspired by the aforementioned game. However, in the Half-Life: Alyx when the player picks up an item, the fingers are wrapping around the model, but in this thesis, the hand forms a fist, while the object is only simply parented to the hand, which results in held objects seemingly floating in front of the hand.

The controllers, which represent the hands, are each tracked individually to an object which the author refers to as "the ground truth transform" and does not interact in any way with the simulated physics world and is not visible to the player[1]. The model of the hand, which is physically simulated with proper collision setup, is constantly following the ground truth transform using a `ConfigurableJoint`. The usage of joints allows the developer to set the maximum amount of force it can apply to the model in order to move it to the correct position, which makes the hand occasionally lag behind the ground truth position during quick hand motions. Doing so will, however, provide more predictable and physically correct results as other forces

[1]Exception being adding visualization to this object for debugging and development purposes.

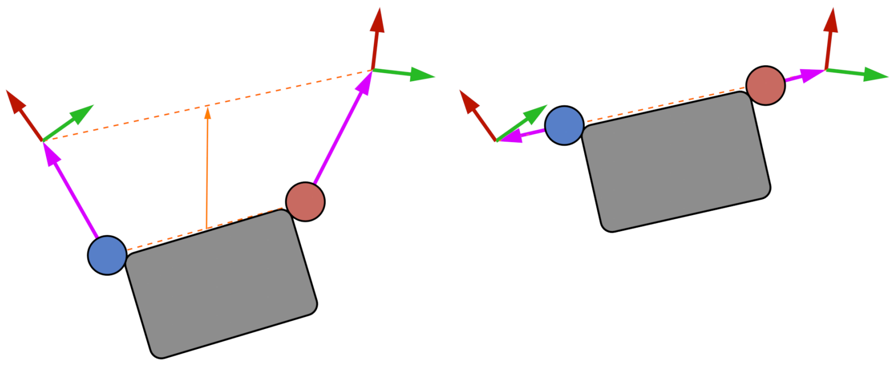


Figure 6.1 A diagram of how the hands are implemented in the physics simulation within this thesis. The red-green axis gizmo represents the ground-truth position of the controllers. The visual hands are shown by the blue and red circles, connected using fixed joints to the held object, represented by the gray box. The visual hands have forces applied using configurable joints, represented by the pink vectors, to make it move towards the ground-truth position. On the left, the hands have been moved from the position during the initial grab; on the right, the object has moved up to respond to this change.

would push the hand back during a collision. This further comes into play when the player wishes to grab objects around the world, as they would be unable to lift heavy objects or objects that are bound to an immovable prop.

Grabbing is done using FixedJoints, which determine the location where the player has grabbed the object and acts as the point from which it applies forces to keep up with the grabbed hand. This means the hand model is connected to the ground truth position using a `ConfigurableJoint` to match the expected hand position and is also connected to the grabbed object using a `FixedJoint`. Joining these objects in this way allows the object to be held by multiple hands, thus permitting the player to lift heavier objects. Diagrams of physical interactions with hands and objects can be seen in Figure 6.1. If the `FixedJoint` requires too much force to move the grabbed object, the joint is destroyed, and the hand no longer holds onto the object, causing it to drop.

Objects the player can grab contain a `Grabbable` component or its derived class, which handles the interaction between the hand and the object. By deriving a class based on it, it is possible to pass additional controller inputs to it, which can be used to control held objects that would have been inconvenient physically as the controls would be too small and require high precision from the user (e.g., turning a knob on a radio to change frequency). When a grab action is requested, the game picks up the nearest object to the hand. As this may be unclear to the player, the game highlights the object they are about to interact with using a purple highlight, as shown in Figure 6.2.

In addition to grabbing physics objects, the other way the player can interact with the world is by using what the author has labeled as "gestures," which are areas that perform an action if a hand is within their boundaries and has performed a grab. If they reach the bounds of any gesture, a short haptics feedback is sent to the controller to signal the player they may grab to perform the action. This is used, for example, for the flashlight on the player's head, where the gesture zone is located right above their head. Reaching into it and invoking it will cause the flashlight to turn on or off.

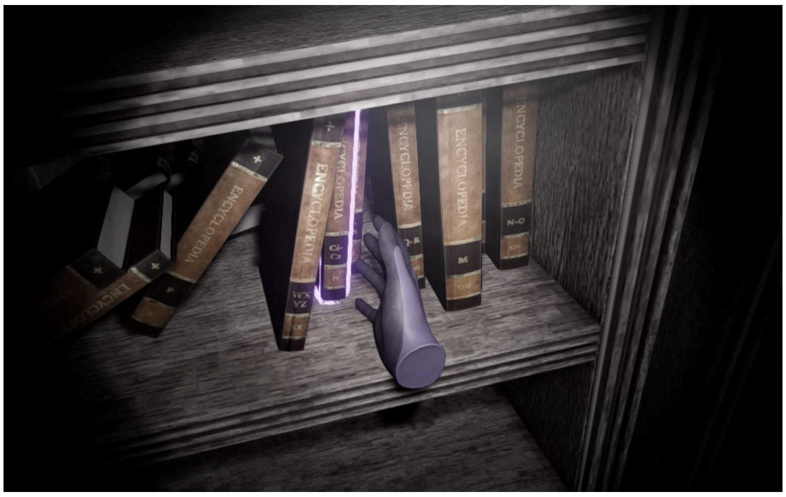


Figure 6.2 A book that the player would grab, being highlighted using a purple outline.

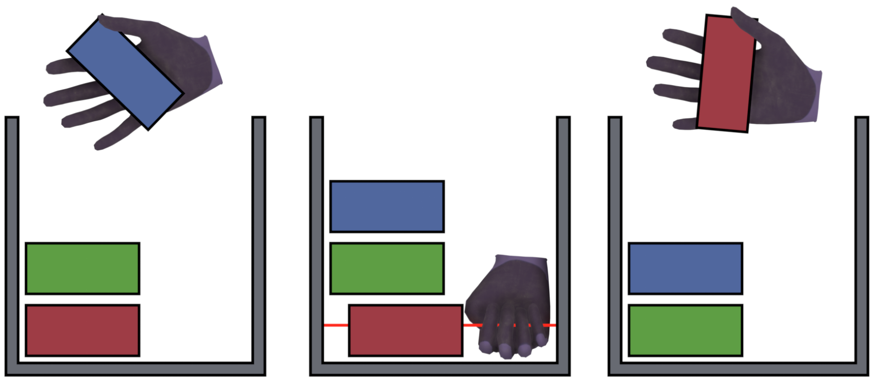
6.1 Backpack

One of the key points of the game is to collect items and to be evaluated based on which items the player has grabbed and stashed away in their backpack[2]. Because dragging it across the world would be tedious and not enjoyable, the player can grab the backpack's handle, which will cause the backpack to be technically hidden. This is to simulate it being slung over the player's back. To make it appear again, the player would have to reach over their shoulder, where the gesture zone for making the backpack is, and grab it, which would make the backpack appear near them. The player needs to ensure they have slung it back on their back; otherwise, attempting to grab it over the shoulder would yield nothing, as it was left somewhere in the world.

The backpack essentially works as a LIFO[3] structure. To add items to it, the player needs to toss it into its open mouth. To retrieve an item, the player needs to reach into the backpack, and depending on how early on they have stashed it, the deeper they need to put their hands in and grab it. To demonstrate, an example diagram can be seen in Figure 6.3. This also means that items that are too big to fit in will not be able to be stashed, as the player would not be able to put them physically into the backpack. Additionally, objects can be marked as unable to be stored regardless of their size. The backpack itself also has a limit on how much it can carry, with each object having its own weight property. If an object is dropped into it and accepting it would exceed the backpack's limit, it is not stashed and is forcefully pushed out of it. It is important to note that while it is called weight, it does not refer to the object's weight but to the volume it would take in the backpack. The limit of the backpack is set to 40, with the units being in liters. All the items have a property describing the approximate volume they would take in the same units.

The backpack's contents are then checked by the component managing the given scenario, as each scenario can have different goals and require different evaluation methods, described in Chapter 8.

[2]In the project files it has been dubbed the `kaban`, which is a romanized Japanese word for backpack
[3]Last in, First out



■ **Figure 6.3** Diagram of the backpack. In the first image, there are two items in it, with a blue item the player wishes to stash. In the second image, the player has already deposited the blue item which is on top of the other items in the backpack, and wants to take out the red item at the bottom, so the player has to reach deep into the backpack to select it, with the red line visualizing how deep the hand is considered to be. After grabbing it, the remaining items are shifted down, and the player holds the red item.

Chapter 7 Sound effects

This chapter describes the sound effects, their source, and how they are invoked within the game.

One of the most important parts of an immersive game is sound effects that are played based on the actions of the player and the environment. It can tell the player a lot of information about the surroundings without being directly told, for example, something occurring behind closed doors, what they are standing on, and much more.

The sound effects used in the thesis are borrowed from the game Half-Life 2 [94] due to its wide library of effects used and easily recognizable for those who have played any of the games in the series. The author themself recorded a couple of sounds that were used in the game, one being an ambient sound track obtained by going out and recording audio in times and places with no people or vehicles nearby, and the other a radio transmission used in the scenario described in Section 8.3.

Unity allows the developer to tweak how the audio is being perceived using various effects. The most commonly used effect in this thesis is the inclusion of reverb to simulate different environments, for example, the echo in the forest or the reverb in tightly closed spaces. This is done to simulate how the audio would have bounced physically in the world. This is noticeable when the player is walking through the city streets and then entering a building, where the reverb and echo sound different. It is possible to have a more physically accurate physical simulation of the audio using third-party solutions such as “Steam Audio”; however, due to its overwhelming complexity, it was decided that it was not suitable for the thesis at the moment.

In the thesis, sound effects are grouped into “Sound pools”. This is done to provide the sound clips to components in a consistent way and to easily pick the clips randomly. These pools are further utilized in audio emitting systems such as footsteps, impact sounds, and certain interaction sounds.

Most, if not all, components and objects that emit sound utilize Unity’s Audio Mixer function. These mixers function similarly to their real-world counterparts, allowing the volume adjustment to a group of audio sources or adding effects to them. This thesis uses it to tweak the volume globally to avoid going to extremes: being too loud or too quiet. To ease access to these mixers, it is possible to access them using the `AudioManager` class, which, alongside functions to play a sound effect anywhere in the world, gives access to the master mixer[1], ambiance mixer, footstep mixer, and impact mixer.

[1]The master mixer controls all of the other mixers.

7.1 Footsteps

Footsteps are emitted from the player's feet and differ based on the surface on which the player is walking. The game keeps track of the distance the player has walked, and once it has reached a certain distance corresponding to the distance of a human step, a random footstep audio clip is played beneath them, and the distance counter is reset to zero. If the player is walking up or down a slope, the counter's increment rate is higher to make the footsteps occur more often. The footstep audio clips are also grouped into sound pools based on the material. If the ground has no audio material defined, the concrete material is used as a fallback to prevent the illusion of the player being silent while walking.

The limitation of this is that the level designer, in this case, the author of the thesis, has to specify the material of the ground or any surface the player could step on manually. This means that if a ground is made out of multiple materials, only one audio pool of footstep sounds can be used unless it is separated into multiple colliders. There was an option to do this by checking the triangle of the mesh it has collided on; however, this is more computationally expensive and would require the mesh to be set as readable during its import, which also increases the memory usage [95].

7.2 Interaction sounds

Interaction sounds include both sounds emitted by the player interacting with items, such as buttons, and sounds emitted by physically simulated objects in collisions. The system in place has a similar problem as the footstep system, as for every object desired to have sounds being played during its impact, a component has to be added to them since Unity does not have a way to detect collisions globally and only on a per object basis.

The audio clips are now gathered in two sets of audio pools, one for "soft" impact and the other for "hard" impact. Depending on the velocity of the impact, a different sound is played, with soft audio clips being played at lower velocities. Additionally, the volume of the audio clip is also further scaled by the velocity to convey the strength it delivered upon impact.

There are some objects that perform sound effects based on the state they are in or the player's actions. In the simplest cases, since they already need to perform an action upon their interaction, a sound effect is added to be played after the said action is performed. This applies to the buttons when pressed, to doors when opened or closed, or to certain windows that are allowed to be broken.

Some objects that do not necessarily perform anything utilize various monitoring scripts, such as `MonitorDistance`, which is a script to detect when an object has moved away too far from or into its initial position and to invoke events and actions based on it. This was used in the case of the cash register's sliding drawer when a sound effect would be played if the drawer had fully retracted back.

However, there are objects that play sound effects not necessarily due to the player's interaction but due to physical properties being applied to it. The difference between these and former sound effects is that it does not utilize the sound pools but play a single looping audio track. To achieve this, a script called `SoundOnVelocity` is used, which monitors either positional or angular velocity. If the magnitude of the monitored velocity is higher than a certain threshold, then the audio track is played with its volume adjusted to the magnitude of the velocity. The use of threshold limiting prevents the audio from being played when the object is barely moved, and the volume modulation conveys the strength of the force that was applied. The adjustment is also smoothed out over time to prevent stuttery-sounding audio. In this game, this was used to add sound effects to both the regular doors and shutter doors, with the former utilizing the angular velocity of the rotating door and the latter the velocity of the sliding vertical door.

7.3 Ambient sounds

By design, the game is devoid of any background music and uses ambient audio tracks, some of which were recorded by the author. These tracks mostly consist of sounds of chirping birds, wind whistling, tree rustling, and miscellaneous distant sounds.

However, if the game were to treat these tracks as background music, then it would play regardless of the player's location. This was especially evident in areas where the player is able to get into a building and would still hear the wind and trees inside. To remedy this, a script called `AmbienceManager` is used, which is also responsible for playing the ambient tracks. In addition to that, it allows the level designer to define volume boxes in the world to determine where the volume should be reduced. The volume is then smoothed over time to avoid stutter and sudden changes in audio. These box volumes also allow changing the ambiance tracks, which are used, for example, to play wind rustling through trees in the forest and not in the city.

To simplify the defining process of these boxes, the component itself adds controls into the editor that allow intuitive resizing and moving of existing boxes. An example of different boxes with their controls can be seen in the example scenario in Figure 7.1.

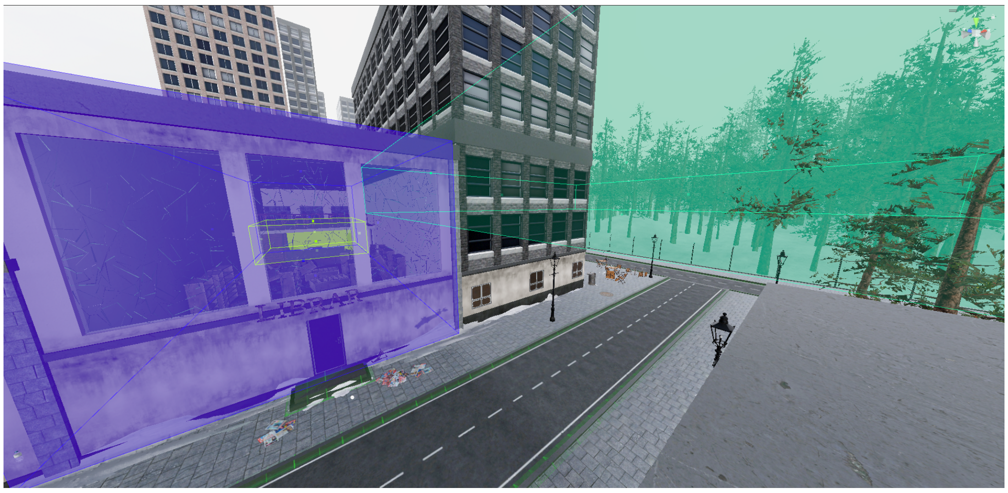


Figure 7.1 Example of ambience zones in the *library* scenario. The purple zone covers the entire library, where the outside ambiance is quieter, and the green zone inside the library makes it even more quieter. The teal zone on the right covers the forest and switches the ambient track into a leaves rustling one.

Chapter 8
Scenarios

All of the scenarios listed below and included in the game have the common idea of exploring the environment and gathering resources; however, each of the scenarios attempts to make the player gather different resources for different purposes and evaluate the score differently.

A similar method of calculating the player's score is used throughout the scenarios, each with minor tweaks. If the player has died during the scenario, the score is always zero. For simplicity, the evaluation is split into four categories, which are represented by a pay-off matrix from the player's point of view. These matrices determine the loss and gain of the given category depending on the player's decisions. They are primarily based on the items the player has stashed away in their backpack, which they have to exit the scenario with. If they had forgotten it in the world, the score calculation for the items is skipped. A rudimentary description of these categories would be as follows:

Cold : The number of seconds the given item would burn as fuel for fire.

Food : Nutritional values of the given item. To simplify the model, energy in kcal was used.

Water : Mililitres of water the item contains. This also includes the water a food item would have in their nutritional values.

Resources : Arbitrary value that determines the future usefulness of the item, where negative values define items that are either useless or would become useless in the future. Positive values would be reserved for items that have the potential to be reused or possibly recycled. A value of 100 is reserved for tools and is the highest obtainable value in this category.

The cold category would be estimated by the material and extrapolated from there. The food and water would be referenced by the real-life counterparts of the object or food and their nutritional values or the amount of liquid it could carry. The thought behind the resources category and the values is to deter the player from gathering items for just short-term usefulness.

For each state the player could be in, a pay-off matrix is provided, determining the *average gain or loss* in terms of items they could obtain by deciding one way or another given the state. In special cases, for example, to reward the player, they can have the values of individual categories artificially modified. These pay-off matrices would all be summed into a single matrix, from which the final score would be calculated. Individual values in the matrices below are calculated based on the author's playtesting sessions and multiplied by $\frac{4}{3}$ to offset the potential bias of the author knowing the solution already.

8.1 The Library

The scenario in the library focuses on obtaining resources in order to survive in a colder environment, which means finding items that can be used as fuel. This scenario also served as a testing ground to test out various mechanics in the game and features that Unity offers and to tweak them for better performance. Screenshots of this scenario can be seen in Appendix A.

It also differs from other scenarios because it does not allow the player to play indefinitely, as they have a time limit dictated by their body temperature. Based on how long they stay in certain areas will decrease the temperature at different speeds, and for the player to gauge this statistic, they have a watch in the game that shows them their current body temperature, as seen in Figure 8.1. Once the body temperature drops below 35°C for an extended period of time, the game will end under the premise that the player has fainted due to hypothermia and has succumbed to the elements. Body temperature is displayed on the watch because having a timer in seconds would be confusing, as the time decrement is not linear and depends on the player's location.

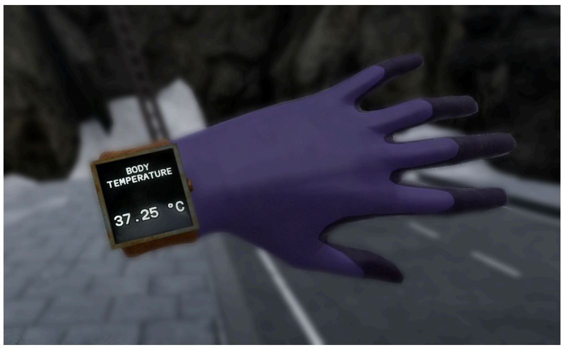


Figure 8.1 The player's watch on their left wrist displays their current body temperature.

8.1.1 Inspirations

The idea behind the scenario was inspired by the actions of the characters in the Introverts Til the End story, specifically by the paragraph noted in Figure 3.4. Furthermore, the idea of the library having multiple floors was inspired by the story as well, as further down in the chapter, there is a short snippet about the library's layout.

"*The library has two floors. Both floors have books.*" [43]

The finer details about the layout were inspired by a certain location in the video game Fallout 4 called the "Old Corner Bookstore", which coincidentally also has two floors. The small front hall area right after entering and the pillars supporting the building and floor were directly inspired by this. The source material can be seen in the Figure 8.2, and the visuals of the library in this thesis are further below in the screenshots.

Library aside, the player has the ability to explore the small section of the town that's available, with all of the roads and paths being blocked off either by a barricade or rocks. There are other buildings; however, they are inaccessible.

8.1.2 Gameplay

The area where the player is located is surrounded by rocky cliffs and walls, which the protagonist has presumably climbed down using a ladder located behind them. Right after the player arrives

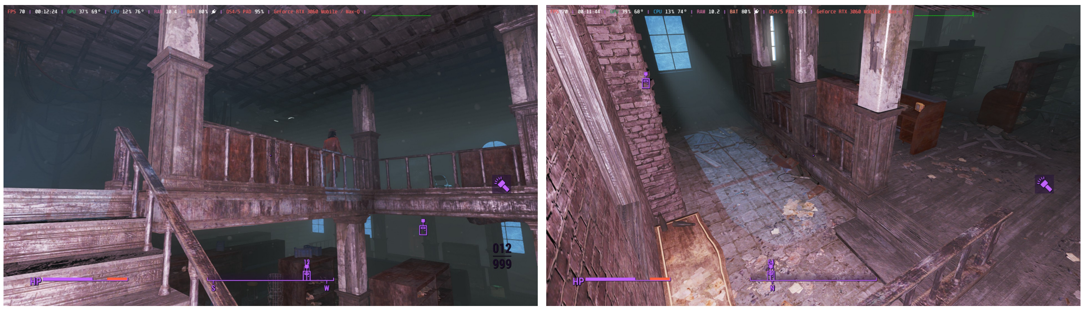


Figure 8.2 The interior of the Old Corner Bookstore, with the front area displayed on the left picture and the second floor on the right picture.

at the scenario, they will be presented with the library in front of them, which is the main building the player is led to. After opening the door to the library and entering it, the body temperature reduction is slowed down. Inside, the player will see fallen and toppled bookshelves with many books scattered on the floor. These books can be picked up and stashed away under the presumption that they will be used as fuel. Some bookshelves also block the path to the back of the floor, and the player has to find their way, which is designed to be narrow and to force the player to physically crouch in real life to fit in there.

Behind the pile of bookshelves, the player will be met with the stairs to the second floor, a reception desk, and a door to an employee-only area. The employee-only area is locked and cannot be accessed unless the player finds the key card, which is found in one of the drawers at the reception desk.

The employee-only area is mainly furnished with shelves littered with cardboard boxes and a table surrounded by chairs. On the table itself, various food supplies are scattered around, and a map overview of the scene is presented, which incites the player that they can explore further for additional supplies. The food items the player decides to stash are subject to the evaluation process. A single lantern lights the table hung from the ceiling, which significantly slows down the rate at which the body temperature decreases.

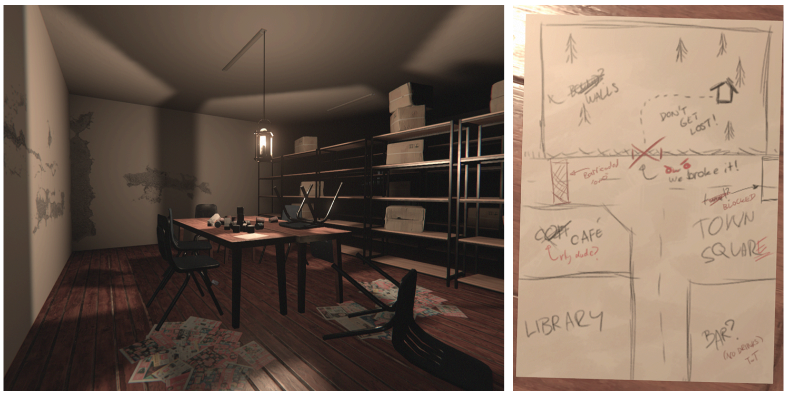


Figure 8.3 The employee-only room, with the table in the middle shown on the left image. On the right, the hand-drawn map of the area can be seen.

On the second floor, there are tables and a wall of lockers in the back. There is nothing in them or on the floor in general, and it serves just as a distraction to hinder the player.

If the player is done with the library or wishes to explore the environment further, then on the right of the library, they will be met with a view of a small town square and a forest, precisely as the map in the employee room has described. Within the town square, apart from the visual decorations, they can see a few outdoor coffee chairs and tables with plastic water bottles scattered around, which the player can stash away.

The town square also borders a snowy forest area, which is fenced off by a wire fence. Upon further exploration, however, the player may notice that one of the fences has been toppled down, allowing them to enter the forest. However, the player is not allowed to explore to their heart's contents, as the body temperature will start dropping significantly, forcing them to be quick and decisive about the route they will take. Additionally, the steep terrain reduces the player's speed, hindering them even further. In the forest, there will be various tree branches of different sizes scattered around that they may pick up under the presumption of using them as fuel.

In the forest, they may find a wooden cabin. The body temperature reduction will be slowed down slightly, but not by any significant margin. Inside there, they will find a hatchet, a lantern, and many branches in one place that they may take with them if their backpack has the capacity left to do so.

At this point, the player has explored all the areas. If they wish to quit and evaluate their score while being alive in this scenario, they will have to backtrack to where they have begun, as the ladder acts as the only way out of the scenario.

8.1.3 Evaluation

From the items the player could stash into their backpack, the pay-off matrix with the rows being the item itself and the columns the gain or loss in that given category can be seen in the Table 8.1.

Table 8.1 Pay-off matrix of items the player may collect in the library scenario.

	Cold	Food	Water	Resources
Book	1200	0	0	1
Card	0	0	0	-1
Fish Cans	0	200–350	25	1
Ramen	0	280	0	1
Map	4	0	0	0
Water Bottle	0	0	250	10
Branches	2700–3600	0	0	0
Lantern	10800	0	0	100
Hatchet	0	0	0	100

The burn time of a book is estimated to be 1200 seconds by extrapolating the assumption that a paper would burn for 8 seconds over the span of 150 pages. The values of fish cans and branches can vary since, in the game, the models themself have multiple variations of different sizes.

Books have the resource value set to +1, as they could contain valuable information inside, while the card has −1, as it is only usable in the library. Branches and the map yield 0 in the same category, given that they are either burned in their entirety or would not hinder the survivor in a significant way if they were to stash it. The food items, namely the fish cans and ramen, have it set to 1, as the left-over containers can be recycled in an useful manner.

As the premise of the scenario is being cold and the goal is to obtain materials to fuel a fire to make themselves warm, the pay-off matrix starts with the decision to either go to the library or explore the city and inevitably go to the forest. The Table 8.2 shows the gains of the player based on this initial decision, where the player stashes whatever they find in the location or on the way to it until their backpack is full. When the player enters the library, the only things they can take are the books; however, if they go to the forest, they can find branches and water bottles along the way. This pay-off matrix is similar to the one in the 8.1 table; however, it has the addition of two columns for the rough time estimations of the player completing the action, and for this scenario, the body temperature loss[1]. Completing the action here means arriving at the destination and stashing whatever they can.

Table 8.2 The initial decisions the player can take after starting the library scenario.

	Cold	Food	Water	Resources
Go to the library	14400	0	0	12
Go to the forest	37800	0	250	100
Go to the forest, find the cabin	42300	0	1250	250

	Time [s]	Temperature
Go to the library	40.56	0.09534
Go to the forest	111.91	0.80975
Go to the forest, find the cabin	266.040	1.85149

From the Table 8.2, it can be observed that while going to the forest, it would gain better items that'd burn longer, alongside the water gain, but it takes more time and body temperature to do so. Because of this, the option to go to the library *first* seems to be a more reasonable decision, even if the option to find the cabin yields better values since the player has no idea that it exists and will mostly wander around the cold forest until they stumble upon it.

In the library, if the player notices and manages to open the employee-only area, they are greeted with various food options scattered on the table, as well as a map of the surrounding area. From here, the player has either the decision to exit the scenario with items they have or continue their trek through the forest, as described in Table 8.3. This time, it differs from the decision to go to the forest first since the player has a map and a general idea of where to find the wooden cabin. It is assumed that they would only trade space in their backpack with a few water bottles during the walk to the forest.

However, since there is a possibility of them getting lost in the forest and not locating the wooden cabin, they can leave tightly before their body hits the point of hypothermia. If they do find the cabin, they are rewarded with two tools, one being a lantern and the other a hatchet.

Table 8.3 Decisions the player can take after entering the library.

	Cold	Food	Water	Resources
Enter the employee area	12000	4420	300	26
Go to the forest	29250	3045	675	38
Go to the forest, find the cabin	36004	3045	425	221

	Time [s]	Temperature
Enter the employee area	101.64	0.26289
Go to the forest	192.91	1.06199
Go to the forest, find the cabin	302.24	2.00307

The Table 8.4 represents the total gain of all the possible decisions the player can take, meaning it takes into account all of the previous pay-off matrices, as well as the time and body temperature loss of going to the exit area to finish the scenario. The list of names in the first column represents the locations the player would go to in a sequential order.

From the table, it appears that the optimal way in this scenario is to go to the library or the forest, take the books, branches, or bottles on the way there, and then leave immediately. While the other strategies have better gain in other categories, the focus of this scenario is about

[1]It is challenging to make them fit on the page, so they have been visually split into two tables.

Table 8.4 All the decisions that are accounted for in the library scenario, which start from the moment the player can move until they reach the exit of the scenario.

	Cold	**Food**	**Water**	**Resources**
Library, Finish	14400	0	0	12
Library, Employee, Finish	12000	4420	300	26
Library, Employee, Forest, Finish	29250	3045	675	38
Library, Employee, Forest, Cabin, Finish	36004	3045	425	221
Forest, Finish	37800	0	250	100
Forest, Cabin, Finish	42300	0	1250	250

	Time [s]	**Temperature**
Library, Finish	60.47	0.16756
Library, Employee, Finish	154.81	0.37268
Library, Employee, Forest, Finish	251.59	1.57441
Library, Employee, Forest, Cabin, Finish	444.91	2.95904
Forest, Finish	241.19	1.39063
Forest, Cabin, Finish	336.04	2.94391

gathering fuel for fire. The final score the player is given is set by the Equation 8.1, with cold being the highest contributing factor. The other factors have less influence, mainly food and water, since they were not the main focus of the scenario. The division of values time and temperature are constants the author has tried during the playtesting sessions. Final scores for each decision can be seen in Table 8.5.

$$\text{score} = \frac{3.0 \cdot \text{cold} + 1.5 \cdot \text{food} + 1.2 \cdot \text{water} + 2.0 \cdot \text{resources}}{(1.0 + \frac{\text{time}}{180}) \cdot (1.0 + \frac{\text{temperature}}{2.4})} \tag{8.1}$$

Table 8.5 Scores of each of the decisions in the store scenario

	Score
Library, Finish	30243
Library, Employee, Finish	20030
Library, Employee, Forest, Finish	23473
Library, Employee, Forest, Cabin, Finish	14645
Forest, Finish	30819
Forest, Cabin, Finish	20193

8.1.4 Assets

Since this scenario was the first that was created amongst the other three, and because it was used as the testing ground for various mechanics, it had a lot of assets created that were later used for other scenarios. Because of that, these models will be only stated here and not in other sections to avoid constant repetition. The list of 3D models that are used in this scenario is shown in Table 8.6.

Table 8.6 3D models that were used in the Library scenario.

Asset Name	Source	Description
Library Sign	Author	It is a sign of letters spelling out ”LIBRARY”, with certain letters being grouped to allow them to be placed with random offsets, giving them a destroyed look.
Door	Author	Simple and generic metal door that is used throughout the thesis.
Windows	Author	Generic looking window that can be placed on various walls to give the impression of an actual hollow building. The window, however is not transparent, and there are wooden planks behind them with a parallax effect to simulate the windows being boarded up.
Book	Author	Books that were used to fill the library and its bookshelves.
Lockers	Author	A column of 3 lockers, used on the second floor of the library. Each of the lockers has doors that are connected with a hinge to the main body.
Card and Card Scanner	Author	Used to open up the door to the employee-only room.
Reception desk	Author	Reception desk that is stationed in front of the employee-only room. It is a corner desk tailor-made for this scenario.
Water Bottles	Author	Made as a generic prop to be scattered across the world. Modeled using references from Dimensions.com [96].
Snow Piles	Author	There are various versions of this model, depending on the surface it'd be placed against. They were designed to be intersecting with other models, making them look as if they were buried in the snow.
Cabin	Author	The wooden cabin found in the forest. The design was purely sketched out by the author and contains a wooden door that is hinged to the body.
Cup Ramen	Author	Cup ramen that are found in the employee-only room.
Broken Glass Inlay	Author	The effect of shattered glass that is placed inside a transparent object to give them the look of brokenness. This effect the author has noticed in a level of the Portal 2 game.
Wired fence	Author	A simple and common fencing used to barrier the player, forcing them to find different routes and such.
Light Fixture	Author	There was a need to add light into buildings in a convincing way in relation to the time the game takes place in. This model comes with two variations, depending on their state of power.
Map	Author	A simple model size of a paper, that allows anything to be printed on it.
Wooden Barrel	Author	A barrel can be found in the wooden cabin, into which the hatchet is hacked into.
Bookshelves	PolyHaven	Bookshelves that are scattered on the first floor of the library with all sorts of books over them
Encyclopedia Books	PolyHaven	Another type of books to diversify the selection of books.

Asset Name	Source	Description
Wooden Table	PolyHaven	These tables are used in the library and other scenarios. The original table from the source was quite short, so the author has created a prolonged version that is about twice the length.
School Chair	PolyHaven	Chairs that are used throughout the thesis.
Wooden Table 2	PolyHaven	This wooden table is located behind the reception desk. It was used because of its design, which has both a door and various drawers the player can interact with.
Salt and Fish cans	PolyHaven	This collection of models consists of a box of salt and two different sizes of fish cans.
Metal Shelves	PolyHaven	Metal shelves are used throughout the thesis to fill areas that are supposed to be storage areas.
Cardboard box	PolyHaven	Used to fill the metal shelves to give the impression of stores having some sort of stock.
Café table and chair	PolyHaven	For decorating the outside area to give the impression of the world where people had left it in a hurry.
Concrete Road Barrier	PolyHaven	A simple and meaningful way to block the roads in the game to restrict player's movement.
Trash Can	PolyHaven	A fully functional trash can into which the player can put things inside. It has an accompanying lid to cover it.
Street Seats and Lamps	PolyHaven	To furnish the city square.
Cliff	PolyHaven	Used as a natural wall to restrict players' movable area.
Branches	PolyHaven	These branches are scattered around the forest area on the ground to give the user a different option other than burning books.
Hatchet	PolyHaven	Found inside the cabin in the forest to reward the player for exploration.

8.2 The Store

This scenario is set near a store and is smaller than the library scenario. Here, the goal is to obtain the items that are located within the store, especially food and water. The player is encouraged to explore perhaps unconventional methods and be wary of the surrounding environment. Screenshots of the scenario can be seen in Appendix B.

8.2.1 Inspirations

Most of the inspirations for this scenario stem from the video game franchise Fallout, in which the player can get various items, including foods and drinks, from old and abandoned stores. These stores can, however, be inaccessible by regular means, and if the player wishes to continue, they would need to find an alternative way. In the Fallout franchise, it usually involves finding a hidden entrance, breaking in, or lockpicking the door.

Within the aforementioned game, there are hints and caution messages being dropped by the other people in the world in the form of notes scattered across the wasteland, which can range from telling the player where something is located to warning them about a trapped entrance.

8.2.2 Gameplay

The scenario begins with the player appearing in front of an abandoned store, which is in relatively good condition due to it being well-secured from intruders. Next to the building is the docking area for the store, which is fenced off. Behind the player is also a ladder, which, similar to the Library scenario, acts as a way to end the scenario.

Attempting to open the door to the store once the player reaches the store would be futile as they are closed and locked. Next to the store's entrance is a trash can with a baseball bat inside of it. They may use it to attempt to use nearby items to break the storefronts' windows; however, they will only succeed with a cracked window, as it is a laminated glass window. At the wired fence near the docking area, the player can notice that there is a hole cut off at the bottom, which allows them to crawl into the area. Note that the player needs to physically crouch to lower themselves to fit in the hole.

From the docking area, the player may enter the store using the shutter door, which has to be physically lifted up. Additionally, it should be either propped up by something or the player should be constantly holding it because if the door were to hit their head, the game would end with them having a concussion. If the player succeeds in getting in, they will be located in the employee lounge of the shop.

The employee lounge is furnished with a small kitchen-esque area, with a large table surrounded by chairs and a noticeboard for the employees. In the corner, they would find collapsed metal shelves filled with boxes. If they look carefully enough, they can see that through the metal shelves, a path leads to someone's previous hiding spot, and the player could collect water bottles they left behind. In the room, there are two doors, and the player can now enter the store through one of these doors and stash whatever they wish from the store shelves and reception desk. Items located here are random assortments of foods, drinks, and magazines. The door to the outside is still locked and not usable. Returning to the lounge, opening the second door will lead to the store's brick cellar, which contains two huge walls of wine barrels. There, they will find a person who had met an unfortunate fate, who was a survivor attempting to flee for safety into the cellar but was crushed by one of the barrels. Only the person's legs and feet are shown sticking from underneath a barrel to prevent the scene from being too graphic. Next to the person, however, they will notice a sleeping bag they may take. If the player wishes to leave the cellar, they would have to prop the entrance with some items, as the door handle only functions from the outside and not the inside, as mentioned on the employee's noticeboard.

At this point, the player has reached all the possibilities and events that this scenario has to offer and may end it by returning to the ladder at the beginning.

8.2.3 Evaluation

Similarly to the evaluation of the scenario described in Section 8.1.3, it is mainly based on the pay-off matrix of the items that are found in the scenario. The items and their values can be seen in Table 8.7. However, given that this scenario, apart from the item collection portion, also focuses on the player finding unconventional ways into the store, the pay-off matrices of the strategies will have the resources' value artificially added to reward the player. As for the baseball bat, it is made of wood and can be used as fuel, with an estimated time to burn to be effectively around 5 minutes.

When the player starts, they are immediately met with the decision of either attempting to break the window using the bat or to go crawl under the wire fence right away.

Once they continue to the docking area, they can enter the store by lifting the metal door. Here, they will be additionally rewarded for using an object to prevent the shutter door from closing all the way, allowing the player to safely crawl under it. While in the store, they can gather whatever they see fit into their backpack, with the values for each category being calculated using the pay-off matrix in Table 8.7. At this point, they may continue, or they can explore the

Table 8.7 Pay-off matrix of items the player may collect in the store scenario.

	Cold	Food	Water	Resources
Fish Cans	0	200–350	25	1
Ramen	0	280	0	1
Water Bottle	0	0	250	10
Magazines	120	0	0	0
Baseball Bat	600	0	0	100
Carrot Cake	0	3200	50	0
Sleeping bag	1800	0	0	100

basement, in which they would find a sleeping bag that they could take with them. However, if they do not heed the warning on the noticeboard, the scenario will end there, and the final score will yield zero. The pay-off matrix for these actions can be seen in Table 8.8. Note that there are two windows; breaking each will give the player the values listed. If the player breaks only one of them, the value gain of the action would be halved.

Table 8.8 The initial decisions the player can take after starting the store scenario.

	Cold	Food	Water	Resources	Time [s]
Break both windows	0.0	0.0	0.0	10.0	28.15
Crawl under the fence	0.0	0.0	0.0	30.0	18.48
Prop the shutter door	0.0	0.0	0.0	20.0	36.21
Prop the basement door	0.0	0.0	0.0	100.0	20.23

As this is essentially the end of the scenario, the player can exit the scenario. In the Table 8.9 with the individual strategies, it is assumed that the player had reached an area and exited the scenario only with the baseball bat that they used to break the windows. The pay-off matrix for these strategies is not provided here, as they are summing the values from the Table 8.8 based on the events that are relevant to it. The score in this scenario is calculated using the Equation 8.2, which is designed to be biased towards resource-based items and events. Time has a very small role in the equation, as the author wishes to encourage exploration over a hurried exit. The Table 8.10 represents the extension of the aforementioned strategies, where the player had managed to reach the docking area and assumes they would spend the time collecting items and leave right after. It is assumed they would collect the same type and amount of items throughout all of the strategies in said table, except for the visit to the basement, where the player would trade some space in their backpack for the sleeping bag.

$$\text{score} = \frac{2.0 \cdot \text{cold} + 3.0 \cdot \text{food} + 5.0 \cdot \text{water} + 100.0 \cdot \text{resources}}{(1.0 + \frac{\text{time}}{6000})} \tag{8.2}$$

Table 8.9 Scores of each of the decisions in the store scenario, not accounting for any item searching or gathering.

	Score
Break windows, leave	1988
Break windows, docking area, leave	4946
Break windows, docking area, prop the shutter door, leave	6868
Break windows, docking area, prop both of the doors, leave	16646

Table 8.10 Pay-off matrix and the score of each of the decisions in the store scenario, assuming the player had gone inside the store and searched it for items.

	Cold	Food	Water	Resources
Gather items	1800	14775	3475	342
Prop the shutter door, gather items	1800	14775	3475	362
Prop both doors, gather items, visit basement	3840	4860	650	406

	Time [s]	Score
Gather items	185.44	96517
Prop the shutter door, gather items	235.45	97667
Prop both doors, gather items, visit basement	247.50	63491

From the score, it is visible that the scenario rewards the player for propping the shutter door; however, due to the fact the player was taking the sleeping bag, meaning they had to swap out items from their backpack, they had lost a significant amount of items to make the sleeping bag fit in their backpack.

8.2.4 Assets

As mentioned in the previous scenario, only assets that were not mentioned before will be listed here.

Table 8.11 3D models that were used in the Store scenario.

Asset Name	Source	Description
Billboard	Author	This billboard is located right on top of the store for a potential advertisement
Broken Wired Fence	Author	A slightly changed version of the wired fence with a hole cut out of the bottom right. The cutout part is kept as a separate model the player can interact with.
Metal Shelves	Author	These are located inside the store in the public area.
Posters	Author	Posters are used to decorate the barren walls. Currently, it displays the author's old poster for a past event.
Noticeboard	Author	Found in the employee area, littered with various papers containing additional information about the store's management and the state of the door to the basement. It also contains some filler documents.
Shutter Doors	Author	These separate the docking area from the employee room. The player can lift the door to open it.
Wooden Barrel	Author	These barrels are found in the basement and are stacked up on each other.
Dead body	Author	A dead body that is crushed underneath a barrel that had fallen. For simplicity, only the legs and feet are modeled.
Sleeping Bag	Author	This is found next to the dead body and serves as a reward to the player for exploring the basement.
Store Counter	Author	Place where the cashiers were operating. It is also designed to hold magazines in the front.

Asset Name	Source	Description
Blue Barrels	PolyHaven	Located beneath the docking area, it can be used to prop up the shutter door to prevent it from falling on top of the player.
Cash Register	PolyHaven	Two of them are on the store counter. Their shelf can be rolled out; however, they do not provide any benefit to the player.
Baseball Bat	PolyHaven	A bat located in the trash bin near the store to entice the player to break the glass window.
Stovetop	PolyHaven	Found in the employee-only room. It is operable by the player.
Carrot Cake	PolyHaven	Found inside the oven underneath the stove top.

8.3 The Cave

In this scenario, the player starts out in a cave with a water container next to them and a river behind the cave. The goal is to collect water and possibly navigate through the foggy forest. A map is also provided to the player, roughly describing their surroundings, and the player is further encouraged to explore. Screenshots of the scenario can be seen in Appendix C.

8.3.1 Inspirations

The main inspiration for this scenario comes from the animated show *Girls' Last Tour*, which was already described in the Section 3.4, with the design of the cave and environment matching the Figure 3.7, as it can be seen in the Figure 8.4. The idea of the scenario being about collecting water had also come from this animated series; however, without having the player capture rainwater by letting the water flow into the bucket, as this was replaced with obtaining water from a river. Furthermore, the concept of forming a bridge with explosives was also taken from here. Giving the player the option to have explosives would not be wise and most likely unrealistic, so instead, the explosives are present in the scenario as undetonated mines located in the forest, with warning signs indicating a minefield.

The idea of the river was inspired by a chapter in the story *Introverts Till the End*, as described in the Section 3.3, where it was described where the protagonist had jumped into a river and described them being dragged away, as can be deduced from Figure 3.5. It also serves as a perfect obstacle for players to force them to find a way across it.

The radio has been added to the scenario from the author's fascination with radio transmissions. As the author had a radio transmitter, they found it interesting that, using a receiver, they were able to see the audio degradation depending on the distance between the devices. In the *Fallout* franchise, as mentioned before in Section 3.1, there are various radio signals the player can tune in, which amongst all may contain instructions or directions, and such it serves as an inspiration for the broadcast content in this scenario.

8.3.2 Gameplay

The player begins in a cave located within a forest-like environment. Right from the start, they'll have a water container, a bucket, a map of their surroundings that they have seemingly sketched out, and a laid out sleeping bag, which, if the player interacts with, counts as exiting the scenario. The map can be seen in Figure 8.4.

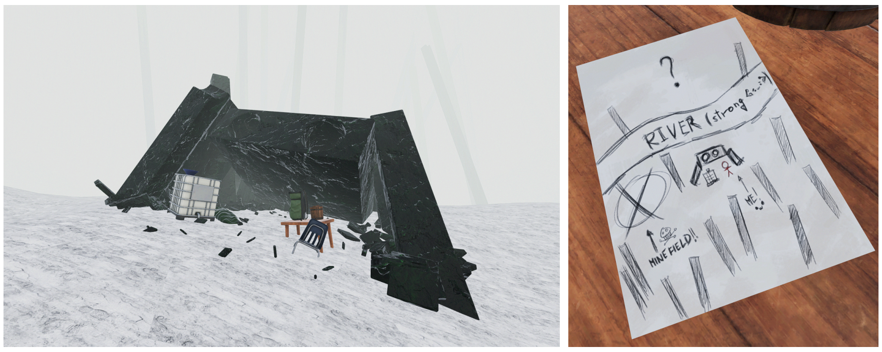


Figure 8.4 The cave the player starts the scenario in, with the only objects nearby being the backpack, a water container, a wooden bucket, and a map of the surroundings. The map is displayed on the right side.

They are directed to the nearby river and to use the bucket to take the water back and pour it into their water container. If the player steps too far into the river, they will be swept away due to the stream's strength, and the scenario ends under the premise of being lost, suffering from hypothermia, or drowning. Note that the water cannot be stashed in a backpack and needs to be carried upright; otherwise, the contained water would spill out.

Across the river, they may see a wooden cabin with a water pump; however, they are unable to reach it due to the problem mentioned above. On the map, the player could see that on their side of the river bank is a sign that warns them about a minefield in the vicinity surrounded by a bunch of pillars. Naturally, if the player disregards this message and walks into it, they will die, and the scenario ends. However, if they manage to trigger the mines without being in their proximity, the explosion will cause the nearby pillar to topple down over the river, effectively forming a bridge the player can walk over. The water pump allows the player to obtain water in a safer and faster way than using the river. In the cabin, they will find an assortment of tools, another laid-out sleeping bag, and a radio receiver, which can be turned on. This sleeping bag functions the same as the one in the cave; however, it would count as them moving from the cave to the cabin to live.

The radio needs to be tuned to a valid frequency; otherwise, it just emits static noise. The player is encouraged to walk and explore. If the player finds the correct frequency, they may wish to find where it is being emitted from, as the noise levels would increase with the distance from the emitter. As noted on the map in Figure 8.4, the area on the top-right is unexplored. However, using the radio, they can be led to a human-made structure in that unexplored area, containing a helipad and a little control booth. In this control booth, there is a button that calls for rescue. Doing so will end the scenario under the premise of them being rescued in a helicopter.

8.3.3 Evaluation

This scenario differs from the others by having three exit points the player could choose, each implying a different story to the ending. The player starts with an assortment of items in the backpack; however, there are still other items in the world that they may pick up, which are listed in the pay-off matrix in Table 8.12 The score is calculated using the Equation 8.3, which favors the water collection and resources, which in this case are events triggered throughout the scenario.

$$\text{score} = \frac{1.0 \cdot \text{cold} + 1.0 \cdot \text{food} + 10.0 \cdot \text{water} + 200.0 \cdot \text{resources}^2}{(1.0 + \frac{\text{time}}{3000})} \tag{8.3}$$

■ **Table 8.12** Pay-off matrix of items the player may collect in the cave scenario.

	Cold	Food	Water	Resources
Book	1200	0	0	1
Magazines	120	0	0	0
Baseball Bat	600	0	0	100
Hatchet	0	0	0	100
Fish Cans	0	200–350	25	1
Water Bottle	0	0	250	10

The pay-off matrix of the possible strategies is listed in the Table 8.13. The water container simulates a volume of 50 liters, which can be filled using the water bucket. To trigger the minefield, they have to use an item from the backpack, and for the pay-off matrix, it is assumed they have thrown a fish can out. When exiting the scenario in the cabin using the sleeping bag in there, the water player would gain would come from the potentially endless water pump. On the way to the helipad, the player may find a baseball bat sticking out of a pile of snow, which is assumed they would take with them. In the control booth at the top of the helipad, the player can scavenge further and obtain additional items, which, due to the spare space in their backpack, are assumed to take all of them before calling for help.

■ **Table 8.13** Pay-off matrix of the decisions the player can make in the cave scenario

	Cold	Food	Water	Resources	Time [s]
Go to sleep (cave)	0	0	0	0	14.97
Gather water (full)	0	0	50000	0	651.19
Trigger minefield	0	−275	−25	−1	47.31
Go to sleep (cabin)	100	0	100000	0	72.44
Call for help	600	0	0	100	104.15

All of the thoughts of strategies are then represented in Table 8.14, alongside their summed-up pay-off matrix and calculated scores. Only the strategies from the initial table that lead to an ending are elaborated there.

■ **Table 8.14** Pay-off matrix of all the decisions leading to ending the scenario successfully, alongside the calculated score.

	Cold	Food	Water	Resources	Time [s]	Score
Go to sleep (cave)	0	2050	950	38	14.97	298859
Gather water (full)	0	2050	50950	38	666.16	654922
Go to sleep (cabin)	100	1850	100925	37	119.75	1235676
Call for help	600	1850	925	137	223.90	3503983

Given that the scenario was designed from the get-go to be more linear, it is no surprise that the scores are ordered from lowest to highest based on the resourcefulness and awareness of the player, with the highest being the player being able to call for help.

Chapter 9

User tests

The game was tested by the author and three other people who have varying levels of knowledge and experience in VR gaming and gaming in general. These tests were conducted on the developer's machine and equipment with the specifications listed in Table 9.1. A third-party program called ALVR [97] was used to stream the VR content from the computer to the headset wirelessly. Because of this, however, all test subjects were affected by a latency issue. The results of their playtesting sessions were used to adjust various gameplay elements, visuals, and pay-off matrices.

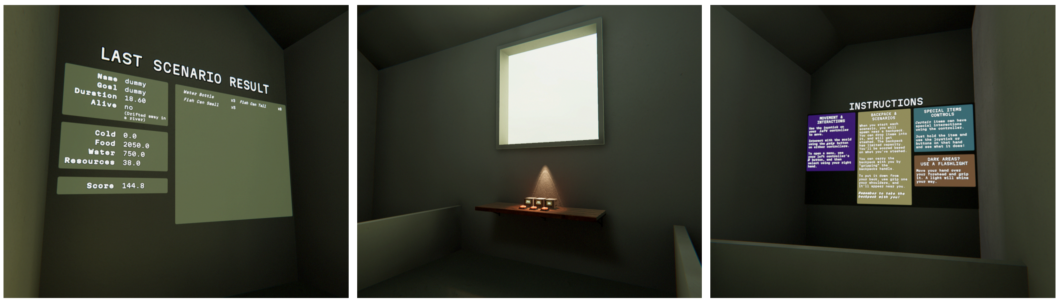
Table 9.1 Computer and headset specifications used during the playtesting

CPU	AMD Ryzen 7 6800HS
GPU	NVIDIA GeForce RTX 3060 Mobile
RAM	16 GB
Headset	Meta Quest 2

Before the testing had begun, test subjects were informed about the use of the controllers, safety precautions about the space around them, and an option to pause or quit playtesting if they had become motion sick or unwell. They were not told anything about the scenarios or game controls. Furthermore, they were asked for consent about having their gameplay footage recorded alongside their voice, as they were encouraged to vocalize their thought process out loud. After this, they were equipped with the headset, and started testing the gameplay.

The player starts in a square room, where they can find three buttons with labels indicating the scenario it would start upon pressing it and the goals of said scenario. On the right side of the room, they are informed about how to control their character and how to interact with the world. The room itself can be seen in Figure 9.1. After the end of each scenario, the player would appear back in this room, however, with the new addition of a wall of text on the left side of the room, showing the result of the scenario, details about the actions recorded, and the calculated score of the evaluation if they had succeeded. If they had not, they would be met with the reason why they had failed. The thought and methods applied during the calculation of this score are already described in the evaluation sections of individual scenarios in Chapter 8.

Before the test begins, the author ensures that the headsets are correctly calibrated to the environment, meaning that the floor the test subject would be standing on would match the virtual one in the game. Each of the selected test subjects will be testing all of the scenarios in the order listed in this thesis. During the test, the author is in the same room with the test subject and silently monitors the state of the game while keeping them safe by checking the physical boundaries to ensure they would not get hurt while in VR. Only questions unrelated to the game



Name dummy
Goal dummy
Duration 18.60
Alive no (Drifted away in a river)

Cold 0.0
Food 2050.0
Water 750.0
Resources 38.0

Score 144.8

Water Bottle x3 Fish Can Tall x3
Fish Can Small x5

■ **Figure 9.1** In the center is the main hub, where the player can select which scenario they want to try. Instructions on how to control the game are on the right, and the result of the last scenario is on the left if they have just returned after finishing one.

or the scenario would be answered, such as the technical aspects or potential issues that may arise. After experiencing each scenario at least once, individuals can redo previous scenarios with the new knowledge gained from previous attempts. The results from these test sessions consist of the recordings of what they have perceived in the headset with their thoughts voiced out. These were viewed by the author, who then decided whether adjustments are necessary and possible.

9.1 First test subject

The first test subject had a lot of experience with various gaming and game mechanics but had little to none with virtual reality. As they were the first play tester, they discovered a lot of oversights and issues in various aspects of the game that were hard to spot by the author, given their pre-existing experience with the game.

Right in the main hub, they noted that the instructions written out were sometimes confusing and were initially lacking essential information. They had also noted that reading it was uncomfortable, as the main hub was a small room, and the text was spread out over the entire wall, making it very big and close to the player. This issue has been remedied by pushing both the instructions and the result screen back; however, an obstacle was added to keep the player within the bounds of the original room.

In terms of the controls, they had difficulty understanding how to put the backpack down from their back. As noted in Section 6.1, a gesture of grabbing the air above the shoulder would make the backpack appear. During the play session, the gesture was for the player to grab the air above their shoulder using the opposite hand, such as using their right hand to grab the left shoulder and vice versa. This turned out to be very confusing for them, and as such, it has been changed so the gesture can be invoked by either hand over either shoulder.

Their experience with the item grabbing was seemingly frustrating. The observer noticed that nearby objects were flung due to the rapid transition of the hands from a fist to a fully extended position. To address this issue, the author has limited the extension of the hand to a partial curl, assuming that the player has a firm grip on the controller.

During the first scenario (the library), the subject had missed the backpack entirely and was exploring the world. They were left wondering how they were supposed to stash the items. The author acknowledges that the backpack's initial position was quite far and in a location that would cause it to blend with the background visually, so it was moved closer to the player's initial position. While exploring, they also often forgot if they had the backpack with them on their back or not since there was no clear indication. In the library, while they had discovered the employee-only door and even the employee card for it, they failed to connect those two entities and discarded the card.

In the second scenario, the subject has spent a long time attempting to break the window to the store, not once noticing the hole in the fence that'd allow them to progress. A visual indicator in terms of a decal on the floor at the broken fence was added to make it more visible. They have also gotten disoriented in the dark room, even with the flashlight. In an attempt to lessen the issue, a different light mapping system was used that simulated secondary light bounces from real-time sources, making the flashlight illuminate even the surrounding walls and making the scene, in general, brighter and more visible. They had also noticed the noticeboard but did not notice the warning and locked themself in the basement.

How to exit the scenarios had to be told to the subject, as it was not clear, given that it appeared in all of the scenarios as a prop near them that seemingly served no purpose other than being a background element. To help the future players, a small text indicating that the prop they are about to grab would exit them out of the scenario was added, which only appears if they are in its close vicinity.

The score calculation had to be adjusted in regards to the time being too big of a factor, heavily punishing the player in not time-limited scenarios for taking their time exploring. For these reasons, the time in these scenarios is further divided by a constant, so the players are still rewarded for completing the scenario quickly but not punished too harshly when they take their time.

9.2 Second test subject

The second test subject had no experience in virtual reality and had, in comparison to the first test subject, less gaming experience. Thanks to that, the feedback provided valuable insight into how the controls are to a complete newcomer to virtual reality. This also meant that, as a player, they had read the instructions more carefully; however, that did not stop them from asking to be reminded about the backpack controls later. They remarked that the wall of text, while understandable, could be better if it had a dedicated tutorial level where the players could try the mechanics out. However, due to the project being in very late stages of development, it could not be implemented.

During the library scenario, even with the adjustments done, they walked by the backpack and did not take it with them. At the start, they wandered blindly, testing out the grabbing mechanics with the books in the library. After a while, they noticed the employee-only room and recognized that it was currently inaccessible but could be unlocked using the card reader next to the door. Initially, they found the newspaper decals outside, which were not interactable, so when they found the key card in the drawer, they assumed that it was also just for decoration. After being informed that it could be grabbed, they managed to enter the room. After reaching the table inside it, they noticed the map, figured out the layout of the scenario, and started heading towards the forest. Once they had approached it, they noticed that their body temperature was very low and decided against going further, so they turned around and exited the scenario. They, however, did not make it and succumbed to hypothermia.

The second scenario was the store. Right from the beginning, they had attempted to break in using the baseball bat found near the windows. Later, they noticed the hole in the fence relatively quickly after the marks were added to the floor. In the dark room, they seemed less disoriented than the last test subject, as the room was now better illuminated. By pure coincidence, they had stumbled upon the hidden stash of water bottles near the collapsed metal shelves and had stashed some away. On their way back, they had found the noticeboard and the warning on it, and because of that, they had avoided going to the basement and attempted to exit the scenario. However, the testing session was cut short since they were getting very motion-sick.

They noted that during the second attempt at a scenario, given that they knew what the game was about and how to control it, it was much easier to focus on the scenarios themselves. For example, in the beginning, they had issues distinguishing between the trigger and the grip button. However, during the second attempt, they had already grown accustomed to the control scheme and were grabbing items without any significant issues.

9.3 Third test subject

The third and last test subject had the most experience in virtual reality out of all the subjects, and because of that, they had the least amount of trouble traversing the world and using the menu. There was, however, a brief moment where the backpack instructions regarding putting it on and off their back had to be explained again at the beginning. They noted that even with the instructions on how to put the backpack on the player's back, the gesture required was not very intuitive.

The first scenario they tried was the store instead of the library since they had accidentally pressed the wrong button. In there, they had trouble seeing the hole in the fence even with the marks on the ground and had only noticed once they were hinted that the fence played an integral part in the solution. Out of all of the test subjects, they understood that the roller door could be propped up by the barrels found underneath the docking area and were utilizing both hands to do so. While they did not find the hidden stash of bottles, unlike the second test subject, they went to the store and gathered some food resources. Curiously enough, they had gathered many boxes of salt. After that, they went back to the employee lounge and failed to acknowledge the noticeboard and its warning, heading straight to the basement, where they had found the body and the sleeping bag. While carrying the sleeping bag back up, they noticed that the door handle was missing, and the scenario ended there.

The next scenario they attempted was the library. This testing was relatively short. They had entered the library with the backpack with them and started stashing the books, as they had immediately noted that books could be used as fire fuel. While doing so, they used a technique: instead of grabbing the books individually, they placed the backpack near the shelves and just pushed the books down into the backpack. They were also the first test subject who had managed to fill the backpack to its fullest. However, they were confused about why the books were launched from the backpack, which occurs when the new item would overstep the backpack's capacity. After that, they left the library and headed towards the forest, where they had gathered some water bottles until their backpack was full. During the collection of the bottles, they inquired if the water bottles were filled with water or not. However, they concluded that because they were in the trash can, they were empty. Because of the full backpack, they had decided to exit the scenario.

They had briefly also attempted the cave scenario. In the first moments, they had noticed the bucket and the map on the table and grew vary of the minefield nearby. They were, however, disoriented and did not know which way the minefield was. When interacting with the bucket, they accidentally dumped all the water out and figured out that they could use the river to fill it back up. However, when trying to do so, they dropped the bucket into the river and attempted to retrieve it, which caused them to end the scenario as they drifted away, ending the scenario.

9.4 Summary of testing

From all of the three test subjects, there were a few remarks all of them had regarding the controls. One of them is that the instructions were not the clearest during the first playthrough, but after learning and trying the game mechanics out firsthand, they understood the point of the game. The third subject also noted that the latency introduced due to how the game was running on the computer and wirelessly transmitted to the headset greatly contributed to the motion sickness, affecting all test subjects. This could not be solved at the time of testing due to a lack of equipment on the author's side.

The backpack, the key part of the game, was commonly ignored on the first playthrough; however, they have always remembered to take it with them on subsequent attempts. At times, however, they seemed to forget if they were carrying the backpack on their backs or not.

For the library scenario, only one of them found their way into the employee-only room and ventured to the forest; however, none of them had found nor attempted to find the wooden cabin in the forest. The subject who got closest to the forest decided against it because it was dangerous for them. All of them had also gone up to the second floor in an attempt to search the lockers, which caused them to lose a lot of time. On that floor, all of them had also asked a sincere question: whether this game had horror elements.

In the store scenario, they all attempted to break the glass once they had found the baseball bat. However, none of them immediately realized it was impossible to break it completely. They also attempted to pull open the door to no avail. All of them managed to get into the store with or without help, with one of them even propping the shutter door to prevent it from falling onto the player. Only two subjects had noticed the noticeboard; however, only one read and heeded the warning. Within the store, all of them thought the counter gate was static and crawled underneath it.

The cave scenario was the least tested, and no one had completed it, as it was the last scenario, with most of them having become motion sick and two of them dying in a river. They had also noted that the sleeping bag was unclear and that it was the exit to the scenario.

The scores of the individual testing sessions of the subjects are summarized in Table 9.2, alongside the time they had finished and a short commentary, if applicable.

Table 9.2 Summary of testing sessions with the subjects and their obtained scores, sorted by the scenario.

Test subject	Scenario	Time [s]	Score	Comment
First	Library	920.61	1932	
Second	Library	1015.59	0	Hypothermia
Third	Library	379.48	10168	
First	Store	1848.17	0	Locked themself in the basement
Second	Store	1232.38	22225	
Third	Store	1024.04	0	Locked themself in the basement
First	Cave	146.26	0	Drifted away in a river
Second	Cave	–	–	Too motion sick to attempt
Third	Cave	128.19	0	Drifted away in a river

During test sessions, if players left the backpack in the world and exited the scenario, they would not receive a score of 0 for the items, as this was added after the author observed that players commonly forgot to retrieve their backpacks. If it had been added before, all scores would have been zeroed.

Chapter 10
Conclusion

In this thesis' documentation, research of post-apocalyptic views in media throughout history was performed and documented. Alongside that, four popular publications regarding the topic of survival or apocalypse were analyzed, with three of them serving as an inspiration to the created scenarios and one as the basis for how the gameplay should feel.

The main programs used within this thesis used for this thesis: the game engine Unity and the 3D manipulation program Blender, were also explained, with the addition of the basics of 3D modeling, texturing, and rendering elaborated upon, which were further used to provide the necessary context for upcoming sections, for example, to compare the benefits and approaches the author has taken.

The thought process behind the avatar of the player, its design, creation process, and how it was brought over from Blender to Unity was fully documented.

Within the thesis, three scenarios were created, which were inspired by the aforementioned analysis of the publications. They were created in conjunction with both Blender and Unity, the former to create the necessary assets and the latter to place them all into a scene. These scenarios and the events in them were scripted using several inter-connected components, all of which utilize the C# programming language. Pay-off matrices were provided for each of the scenarios to serve as a basis for the players' optimal strategy.

The movement and interactions of the player with the world the scenario is set in are implemented using the physics engine Unity provides out of the box. Within it, they can interact with the various dynamic objects in the scenario, such as buttons, doors, and shelves.

During the testing phase of the game, three people were monitored, and their feedback was documented and evaluated to fine-tune the individual scenarios and gameplay mechanics.

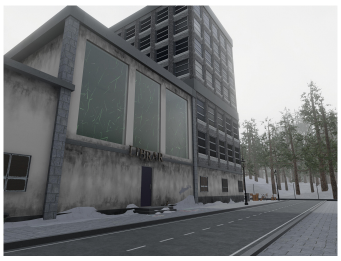
It is possible to obtain the compiled game from the author's GitLab page alongside this document, which can also serve as a reference to the solution of the scenarios.

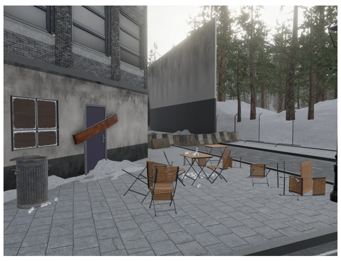
Additional work has been done beyond the initial assignment to accommodate the physics simulations with accompanying sound effects. These effects increased the player's immersion, alongside the effects of reverb and echo in select places within the scenarios. Some effects were added implicitly by detecting the collision between objects, and some were added by checking the various states of the object to determine whether or not an effect should be played.

During the creation, a few tools and addons were created to aid and accelerate the development. In Unity, a tool was created to allow in-editor physics simulations to place objects naturally and alleviate the need for physics simulations during runtime. Additional scripts to visualize the bounding boxes of reflection probes and light probes, with the latter even allowing the creation of a uniformly distributed grid matrix of them, were also added. The ability to create randomized instances of objects either in a sequence or in a box volume was added, together with the option to randomize their transformation values.

In the future, the author would like to touch upon a few aspects of the game, such as finer handling of fingers, where the held objects would have the player's fingers wrapped around them correctly, as opposed to the current solution, where the object is floating in front of a clenched fist. The addition of non-playable characters would be a great addition as well, requiring significant work on dialogue, behavior designing, and asset modeling.

Appendix A
Screenshots of Library scenario


(a) Outside view of the area where the player starts


(b) Abandoned café area in the city

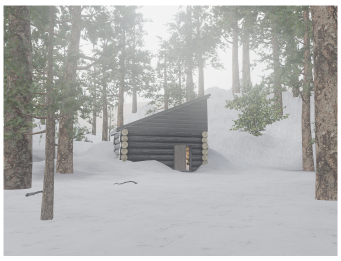
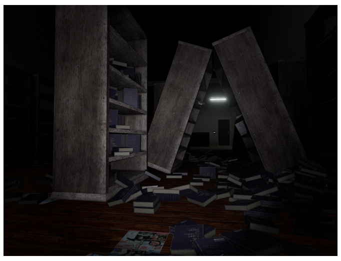
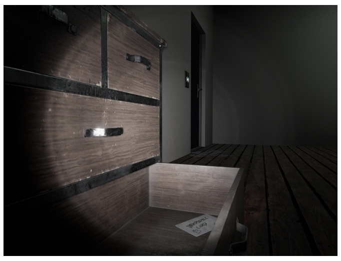

(c) Wooden cabin hidden in the forest area

Figure A.1 Exterior view of the library scenario.


(a) Bookshelves and books scattered around in the library


(b) The locked employee-only door and the key to it in a drawer

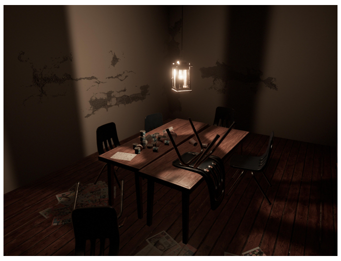
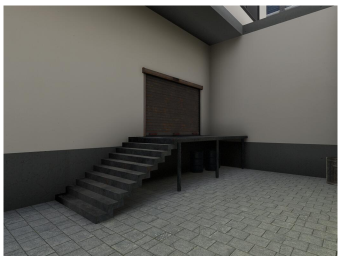

(c) Food objects, chairs, and the map on the table in the employee-only room

Figure A.2 Interior view of the library scenario and its key points

Appendix B

Screenshots of Store scenario

(a) Outside view of the store building, with the docking area on the right


(b) Closer look at the docking area and its shutter door

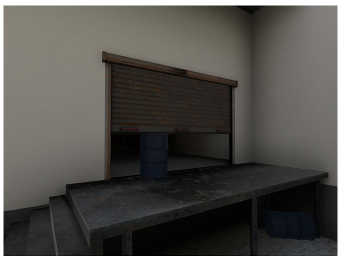
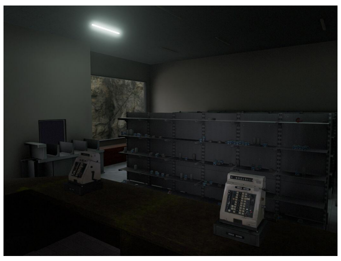
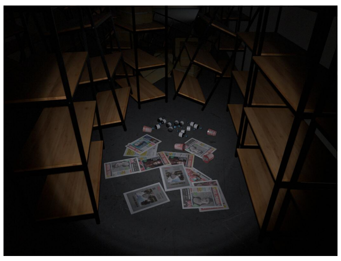

(c) Proposed solution to the shutter door using a barrel

Figure B.1 Exterior view of the store scenario.


(a) The inside of the store building


(b) Hidden stash of bottles in the employee area

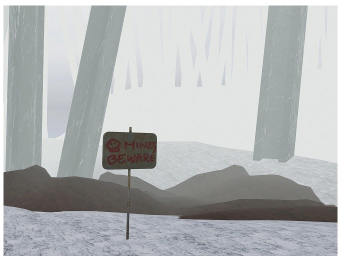
(c) Survivor who had met an unfortunate fate in the basement

Figure B.2 Interior view of the store scenario and its key points.

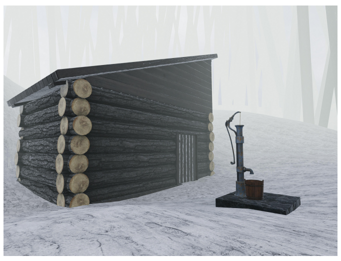
Appendix C

Screenshots of Cave scenario

(a) The cave the player starts in, with a sleeping bag, water tank, water bucket, and a map

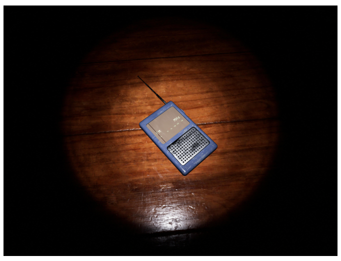


(b) Minefield found near the cave

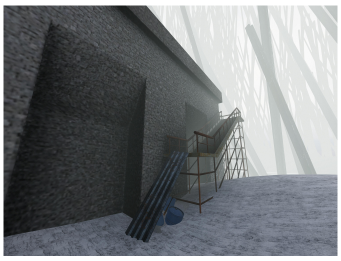


(c) Wooden cabin found on the other side of the river

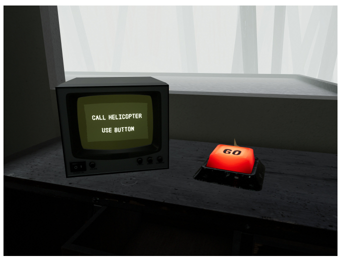
Figure C.1 Outside area of the cave scenario, including the wooden cabin.



(a) The radio obtainable in the wooden cabin



(b) Source of the radio transmission



(c) A method to exit the scenario by calling for help

Figure C.2 Required items and locations in order to exit the scenario using the button to call for help.

Bibliography

1. EBERLY, David H. *3D Game Engine Design: A Practical Approach to Real-Time Computer Graphics*. Second Edition. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2006. The Morgan Kaufmann Series in Interactive 3D Technology. ISBN 978-0-12-229063-3.
2. AKENINE-MOLLER, Tomas; HAINES, Eric; HOFFMAN, Naty. *Real-Time Rendering*. Third Edition. USA: A. K. Peters, Ltd., 2008. ISBN 978-1-56881-424-7.
3. UNITY TECHNOLOGIES. Models. In: *Unity Manual* [online]. Unity Technologies, 2024 [visited on 2024-01-06]. Available from: `https://docs.unity3d.com/Manual/models.html`.
4. BLENDER. Unwrapping: Introduction. In: *Blender 3.6 Manual* [online]. Blender, 2023 [visited on 2023-12-31]. Available from: `https://docs.blender.org/manual/en/3.6/modeling/meshes/uv/unwrapping/introduction.html`.
5. BLENDER. Unwrapping – Seams: Introduction. In: *Blender 3.6 Manual* [online]. Blender, 2023 [visited on 2023-12-31]. Available from: `https://docs.blender.org/manual/en/3.6/modeling/meshes/uv/unwrapping/seams.html`.
6. BLENDER. Weight Paint: Introduction. In: *Blender 3.6 Manual* [online]. Blender, 2023 [visited on 2024-01-07]. Available from: `https://docs.blender.org/manual/en/3.6/sculpt_paint/weight_paint/introduction.html`.
7. FERNANDO, Randima; KILGARD, Mark J. *The Cg Tutorial* [online]. Boston: Addison-Wesley, 2003 [visited on 2023-12-30]. ISBN 978-0321194961. Available from: `https://developer.download.nvidia.com/CgTutorial/cg_tutorial_frontmatter.html`.
8. BILEY, Mike; CUNNINGHAM, Steve. *Graphics Shaders: Theory and practice – Second Edition*. Boca Raton, FL: CRC Press, 2012. ISBN 978-1-56881-434-6.
9. UNITY TECHNOLOGIES. ShaderLab command: ZTest. In: *Unity Manual* [online]. Unity Technologies, 2024 [visited on 2024-01-06]. Available from: `https://docs.unity3d.com/Manual/SL-ZTest.html`.
10. UNITY TECHNOLOGIES. Rendeing paths in the Built-in Render Pipeline. In: *Unity Manual* [online]. Unity Technologies, 2024 [visited on 2024-01-06]. Available from: `https://docs.unity3d.com/Manual/RenderingPaths.html`.
11. UNITY TECHNOLOGIES. Deferred Shading rendering path. In: *Unity Manual* [online]. Unity Technologies, 2024 [visited on 2023-12-31]. Available from: `https://docs.unity3d.com/Manual/RenderTech-DeferredShading.html`.
12. PHABARR, Matt; HUMPHREYS, Greg. *Physiically Based Rendering*. From theory to implementation. San Francisco: Morgan Kaufmann Publishers, 2004. ISBN 0-12-553180-X.

13. ADOBE. Everything you need to know about physically based rendering. In: *3D & AR – Use Cases* [online]. Adobe, [n.d.] [visited on 2024-01-10]. Available from: `https://www.adobe.com/products/substance3d/discover/pbr.html`.
14. WILSON, Joe "EarthQuake". Physically-Based Rendering, And You Can Too! In: *Marmoset* [online]. Marmoset LLC, 2020 [visited on 2024-01-10]. Available from: `https://marmoset.co/posts/physically-based-rendering-and-you-can-too/`.
15. ADOBE. The PBR Guide – Part 2. In: *Learn Substance 3D* [online]. Adobe, [n.d.] [visited on 2024-01-10]. Available from: `https://creativecloud.adobe.com/learn/substance-3d-designer/web/the-pbr-guide-part-2`.
16. UNITY TECHNOLOGIES. Lit Shader. In: *Universal RP Manul* [online]. Unity Technologies, 2021 [visited on 2023-11-29]. Available from: `https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@14.0/manual/lit-shader.html`.
17. BLENDER. Shader Nodes: Principled BSDF. In: *Blender 3.6 Manual* [online]. Blender, 2023 [visited on 2024-01-03]. Available from: `https://docs.blender.org/manual/en/3.6/render/shader_nodes/shader/principled.html`.
18. POLIIGON. Texture Maps Explained. In: *Poliigon Help Center* [online]. Poliigon, 2023 [visited on 2024-01-10]. Available from: `https://help.poliigon.com/en/articles/1712652-texture-maps-explained`.
19. UNITY TECHNOLOGIES. Normal map (Bump mapping). In: *Unity Manual* [online]. Unity Technologies, 2024 [visited on 2024-01-10]. Available from: `https://docs.unity3d.com/Manual/StandardShaderMaterialParameterNormalMap.html`.
20. FUCHS, Philippe. *Virtual Reality Headsets: A theoretical and Pragmatic Approach*. EH Leiden: CRC Press/Balkema, 2017. ISBN 978-1-138-63235-6.
21. PAULÍK, David. *Apokalypsa – projevy a chápání v minulosti i současnosti* [online]. Brno, 2020 [visited on 2023-11-18]. Available from: `https://is.muni.cz/th/du7tw/`. Master's thesis. Masaryk University, Faculty of Education, Brno.
22. APOCALYPSE Definiton & Usage Examples. In: *Dictionary.com* [online]. Rock Holdings Inc., [n.d.] [visited on 2023-11-18]. Available from: `https://www.dictionary.com/browse/apocalypse`.
23. ERICKSON, Brandon. *Y2K: The Millennium Bug Effect* [online]. Helena, Montana, 1999 [visited on 2024-01-06]. Available from: `https://hdl.handle.net/20.500.12647/94`. Bachelor's thesis. Carrol College, Department of Business, Accounting, and Economics.
24. BBC NEWS. Y2K bug fails to bite. In: *BBC NEWS* [online]. BBC, 2000 [visited on 2024-12-07]. Available from: `http://news.bbc.co.uk/2/hi/science/nature/585013.stm`.
25. ALLEN, Frederick E. Apocalypse Then: When Y2K Didn't Lead To The End Of Civilization. In: *Forbes.com* [online]. Forbes Media LLC, 2019 [visited on 2023-11-18]. Available from: `https://www.forbes.com/sites/frederickallen/2020/12/29/apocalypse-then-when-y2k-didnt-lead-to-the-end-of-civilization/`.
26. The Meaning of 2012. In: *Living Maya Time* [online]. Smithsonian Institution, [n.d.] [visited on 2023-11-18]. Available from: `https://maya.nmai.si.edu/2012-resetting-count/meaning-of-2012`.
27. *2012* [movie]. Director Roland EMMERICH. Columbia Pictures and Centropolis Entertainment, 2009.
28. NAKAZAWA, Keiji; MOTOFUMI, Asai; MINEAR, H. Richard. Barefoot Gen, The Atomic Bomb and I: The Hiroshima Legacy. In: *Asia-Pacific Journal: Japan Focus* [online]. 2008 [visited on 2023-11-28]. Available from: `https://apjjf.org/-Nakazawa-Keiji/2638/article.html`.

29. CAIN, Timothy. The Biggest Influences On Fallout. In: *YouTube* [online]. 2023 [visited on 2023-11-28]. Available from: `https://www.youtube.com/watch?v=Q8XTWJRBFeM?t=6m50s`.

30. GLUKHOVSKY, Dmitry. I'm Dmitry Glukhovsky, the author of Metro 2033, base of the Metro video games. My new novel Metro 2035 has just come out. AMA! In: *Reddit.com* [online]. 2016 [visited on 2023-11-28]. Available from: `https://www.reddit.com/r/books/comments/5hyvwa/im_dmitry_glukhovsky_the_author_of_metro_2033/db42l1b/`.

31. PLAGUE Definiton & Usage Examples. In: *Dictionary.com* [online]. Rock Holdings Inc., [n.d.] [visited on 2023-11-18]. Available from: `https://www.dictionary.com/browse/plague`.

32. MATHIEU, Edouard; RITCHIE, Hannah; RODÉS-GUIRAO, Lucas; APPEL, Cameron; GIATTINO, Charlie; HASELL, Joe; MACDONALD, Bobbie; DATTANI, Saloni; BELTEKIAN, Diana; ORTIZ-OSPINA, Esteban; ROSER, Max. Coronavirus Pandemic (COVID-19). In: *Our World in Data* [online]. 2020 [visited on 2023-12-09]. Available from: `https://ourworldindata.org/coronavirus`.

33. SATARIANO, Adam; ALBA, Davey. Burning Cell Towers, Out of Baseless Fear They Spread the Virus. In: *The New York Times* [online]. The New York Times Company, 2020 [visited on 2023-12-09]. Available from: `https://www.nytimes.com/2020/04/10/technology/coronavirus-5g-uk.html`.

34. GOODMAN, Jack; CARMICHAEL, Flora. Coronavirus: Bill Gates 'microchip' conspiracy theory and other vaccine claims fact-checked. In: *BBC.com* [online]. BBC, 2023 [visited on 2023-12-09]. Available from: `https://www.bbc.com/news/52847648`.

35. ZOMBIE Definiton & Usage Examples. In: *Dictionary.com* [online]. Rock Holdings Inc., [n.d.] [visited on 2023-11-18]. Available from: `https://www.dictionary.com/browse/zombie`.

36. KANE, Joe. *Night of the Living Dead: Behind the Scenes of the Most Terrifying Zombie Movie Ever*. Kensington Publishing Corp., 2010. ISBN 0-8065-3331-5. Available also from: `https://archive.org/details/nightoflivingdea00joek`.

37. *Night of the Living Dead* [movie]. Director George A. ROMERO. Image Ten, 1968.

38. OBSIDIAN ENTERTAINMENT. *Fallout: New Vegas* [game]. 2010. [visited on 2010-10-22]. Available from: `https://store.steampowered.com/app/22490`.

39. BETHESDA ENTERTAINMENT. *Fallout 4* [game]. 2015. [visited on 2023-12-11]. Available from: `https://store.steampowered.com/app/377160`.

40. VALVE CORPORATION. *Half-Life: Alyx* [game]. 2020. [visited on 2023-12-11]. Available from: `https://store.steampowered.com/app/546560/HalfLife_Alyx/`.

41. 2020 Rewind. In: *The Game Awards* [online]. 2020 [visited on 2023-10-17]. Available from: `https://thegameawards.com/rewind/year-2020`.

42. VALVE CORPORATION. I already bought Alyx! In: *Half-Life: Alyx FAQ* [online]. [N.d.] [visited on 2024-01-07]. Available from: `https://help.steampowered.com/en/faqs/view/099C-EB8F-C7C2-9543`.

43. TRIPSOUT2. Introverts Till the End. In: *Archive of Our Own* [online]. 2022 [visited on 2023-10-17]. Available from: `https://archiveofourown.org/works/33756916`.

44. TSUKUMIZU. *Shōjo Shūmatsu Ryokō* [TV series]. Director Takahuru OZAKI. White Fox and Sentai Filmworks, 2017.

45. UNITY TECHNOLOGIES. GameObject. In: *Unity Manual* [online]. Unity Technologies, 2024 [visited on 2024-01-10]. Available from: `https://docs.unity3d.com/Manual/class-GameObject.html`.

46. UNITY TECHNOLOGIES. Create Gameplay – GameObjects: Use components. In: *Unity Manual* [online]. Unity Technologies, 2024 [visited on 2024-01-11]. Available from: `https://docs.unity3d.com/Manual/UsingComponents.html`.
47. UNITY TECHNOLOGIES. Introduction to render pipelines. In: *Unity Manual* [online]. Unity Technologies, 2024 [visited on 2024-01-10]. Available from: `https://docs.unity3d.com/Manual/render-pipelines-overview.html`.
48. UNITY TECHNOLOGIES. Getting started with Shader Graph. In: *Shader Graph Manual* [online]. Unity Technologies, 2023 [visited on 2023-10-23]. Available from: `https://docs.unity3d.com/Packages/com.unity.shadergraph@17.0/manual/Getting-Started.html`.
49. MOHEBALI, Ali; DEMEULEMEESTER, Aljosha; MULLER, Mathieu; KENT, Steven. Games Focus: Rendering that scales with your needs. In: *Unity Blog* [online]. Unity Technologies, 2022 [visited on 2024-01-07]. Available from: `https://blog.unity.com/engine-platform/games-focus-rendering-that-scales-with-your-needs`.
50. UNITY TECHNOLOGIES. Render pipeline feature comparison. In: *Unity Manual* [online]. Unity Technologies, 2024 [visited on 2024-01-11]. Available from: `https://docs.unity3d.com/Manual/render-pipelines-feature-comparison.html`.
51. UNITY TECHNOLOGIES. Auto Exposure. In: *Post Processing Manual* [online]. Unity Technologies, 2023 [visited on 2024-01-10]. Available from: `https://docs.unity3d.com/Packages/com.unity.postprocessing@3.4/manual/Auto-Exposure.html`.
52. UNITY TECHNOLOGIES. Decal Renderer Feature. In: *Universal RP 14.0.10 Manual* [online]. Unity Technologies, 2021 [visited on 2024-01-03]. Available from: `https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@14.0/manual/renderer-feature-decal.html`.
53. UNITY TECHNOLOGIES. Projector. In: *Unity Manual* [online]. Unity Technologies, 2024 [visited on 2024-01-10]. Available from: `https://docs.unity3d.com/Manual/class-Projector.html`.
54. UNITY TECHNOLOGIES. Changelog 12.0.0. In: *Shader Graph Manual* [online]. Unity Technologies, 2021 [visited on 2023-10-23]. Available from: `https://docs.unity3d.com/Packages/com.unity.shadergraph@12.0/changelog/CHANGELOG.html`.
55. UNITY TECHNOLOGIES. Node Library: Procedural Nodes. In: *Shader Graph Manual* [online]. Unity Technologies, 2024 [visited on 2024-01-10]. Available from: `https://docs.unity3d.com/Packages/com.unity.shadergraph@14.0/manual/Procedural-Nodes.html`.
56. UNITY TECHNOLOGIES. Light Mode: Realtime. In: *Unity Manual* [online]. Unity Technologies, 2024 [visited on 2024-01-10]. Available from: `https://docs.unity3d.com/Manual/LightMode-Realtime.html`.
57. UNITY TECHNOLOGIES. Light Mode: Baked. In: *Unity Manual* [online]. Unity Technologies, 2024 [visited on 2024-01-10]. Available from: `https://docs.unity3d.com/Manual/LightMode-Baked.html`.
58. UNITY TECHNOLOGIES. Lighting: Light Pobes. In: *Unity Manual* [online]. Unity Technologies, 2024 [visited on 2024-01-10]. Available from: `https://docs.unity3d.com/Manual/LightProbes.html`.
59. UNITY TECHNOLOGIES. Reflection Probes. In: *Unity Manual* [online]. Unity Technologies, 2024 [visited on 2024-01-10]. Available from: `https://docs.unity3d.com/Manual/ReflectionProbes.html`.
60. UNITY TECHNOLOGIES. Reflection Probes: Advanced Reflection Probe Features. In: *Unity Manual* [online]. Unity Technologies, 2024 [visited on 2024-01-10]. Available from: `https://docs.unity3d.com/Manual/AdvancedRefProbe.html`.

61. HARADA, Takahiro; MCKEE, Jay; YANG, Jason C. *Forward+: Bringing Deferred Lighting to the Next Level.* In: ANDUJAR, Carlos; PUPPO, Enrico (eds.). *Eurographics 2012 - Short Papers.* The Eurographics Association, 2012. ISSN 1017-4656. Available from DOI: `10.2312/conf/EG2012/short/005-008`.
62. UNITY TECHNOLOGIES. What's new in URP 14 (Unity 2022.2). In: *Universal RP 14.0.9 Manual* [online]. Unity Technologies, 2021 [visited on 2024-01-03]. Available from: `https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@14.0/manual/whats-new/urp-whats-new.html`.
63. PETERBAY. URP 14.0.4. VR and Forward+ results in broken lighting. In: *Unity Forums* [online]. Unity Technologies, 2022 [visited on 2024-01-04]. Available from: `https://forum.unity.com/threads/urp-14-0-4-vr-and-forward-results-in-broken-lighting.1367442/#post-8622333`.
64. UNITY TECHNOLOGIES. Physics. In: *Unity Manual* [online]. Unity Technologies, 2022 [visited on 2023-10-20]. Available from: `https://docs.unity3d.com/Manual/PhysicsSection.html`.
65. UNITY TECHNOLOGIES. Introduction to rigid body physics. In: *Unity Manual* [online]. Unity Technologies, 2022 [visited on 2024-01-03]. Available from: `https://docs.unity3d.com/Manual/RigidbodiesOverview.html`.
66. UNITY TECHNOLOGIES. Collider shapes. In: *Unity Manual* [online]. Unity Technologies, 2022 [visited on 2024-01-03]. Available from: `https://docs.unity3d.com/Manual/collider-shapes.html`.
67. UNITY TECHNOLOGIES. Introduction to collision. In: *Unity Manual* [online]. Unity Technologies, 2017 [visited on 2024-01-04]. Available from: `https://docs.unity3d.com/Manual/CollidersOverview.html`.
68. UNITY TECHNOLOGIES. Joints. In: *Unity Manual* [online]. Unity Technologies, 2022 [visited on 2024-01-03]. Available from: `https://docs.unity3d.com/Manual/joints-section.html`.
69. CORPORATION, NVIDIA. Character Controllers. In: *NVIDIA PhysX SDK 3.4.0 Documentation* [online]. NVIDIA Corporation, 2017 [visited on 2024-01-03]. Available from: `https://docs.nvidia.com/gameworks/content/gameworkslibrary/physx/guide/Manual/CharacterControllers.html`.
70. UNITY TECHNOLOGIES. CharacterController. In: *Unity Manual* [online]. Unity Technologies, 2022 [visited on 2024-01-03]. Available from: `https://docs.unity3d.com/Manual/CharacterController.html`.
71. OpenXR Overview. In: *The Khronos Group Inc.* [online]. The Khronos Group Inc., 2023 [visited on 2024-01-03]. Available from: `https://www.khronos.org/openxr/`.
72. Developer Resource Hub. In: *The Khronos Group Inc.* [online]. The Khronos Group Inc., 2023 [visited on 2024-01-03]. Available from: `https://www.khronos.org/developers`.
73. THE KHRONOS® OPENXR WORKING GROUP. Interaction Profile Paths. In: *The OpenXR™ Specification* [online]. The Khronos Group Inc., 2023 [visited on 2024-01-03]. Available from: `https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html#semantic-path-interaction-profiles`.
74. THE KHRONOS® OPENXR WORKING GROUP. Input subpaths. In: *The OpenXR™ Specification* [online]. The Khronos Group Inc., 2023 [visited on 2024-01-03]. Available from: `https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html#semantic-path-standard-identifiers`.
75. BLENDER. Features. In: *Blender* [online]. Blender, 2023 [visited on 2024-01-03]. Available from: `https://www.blender.org/about/`.

76. BLENDER. About. In: *Blender* [online]. Blender, 2023 [visited on 2024-01-03]. Available from: `https://www.blender.org/features/`.
77. BLENDER. EEVEE: Introduction. In: *Blender 3.6 Manual* [online]. Blender, 2023 [visited on 2024-01-03]. Available from: `https://docs.blender.org/manual/en/3.6/render/eevee/introduction.html`.
78. BLENDER. Rendering: Introduction. In: *Blender 3.6 Manual* [online]. Blender, 2023 [visited on 2024-01-03]. Available from: `https://docs.blender.org/manual/en/3.6/render/introduction.html`.
79. BLENDER. Cycles: GPU Rendering. In: *Blender 3.6 Manual* [online]. Blender, 2023 [visited on 2024-01-03]. Available from: `https://docs.blender.org/manual/en/3.6/render/cycles/gpu_rendering.html`.
80. BLENDER. Cycles: Baking. In: *Blender 3.6 Manual* [online]. Blender, 2023 [visited on 2024-01-03]. Available from: `https://docs.blender.org/manual/en/3.6/render/cycles/baking.html`.
81. BLENDER. Shader Editor. In: *Blender 3.6 Manual* [online]. Blender, 2023 [visited on 2024-01-03]. Available from: `https://docs.blender.org/manual/en/3.6/editors/shader_editor.html`.
82. BLENDER. Shader Nodes: Specular BSDF. In: *Blender 3.6 Manual* [online]. Blender, 2023 [visited on 2024-01-03]. Available from: `https://docs.blender.org/manual/en/3.6/render/shader_nodes/shader/specular.html`.
83. BLENDER. Modeling: Modifiers – Introduction. In: *Blender 3.6 Manual* [online]. Blender, 2023 [visited on 2024-01-03]. Available from: `https://docs.blender.org/manual/en/3.6/modeling/modifiers/introduction.html`.
84. BLENDER. Modeling: Geometry Nodes – Introduction. In: *Blender 3.6 Manual* [online]. Blender, 2023 [visited on 2024-01-03]. Available from: `https://docs.blender.org/manual/en/3.6/modeling/geometry_nodes/introduction.html`.
85. GESSLER, Alexander; BARTON, Campbell. FBX binary file format specficiation. In: *Blender Blog* [online]. Blender Foundation, 2013 [visited on 2024-01-03]. Available from: `https://code.blender.org/2013/08/fbx-binary-file-format-specification/`.
86. ZAAL, Greg; TUYTEL, Rob, et al. *Poly Haven: The Public 3D Asset Library* [online]. [visited on 2023-11-29]. Available from: `https://polyhaven.com/`.
87. DEMES, Lennart. *ambientCG: CC0 Textures, HDRIs and Models* [online]. [visited on 2023-12-13]. Available from: `https://ambientcg.com/`.
88. *TextureCan: Free PBR Textures and CC0 3D Models* [online]. [visited on 2023-12-13]. Available from: `https://www.texturecan.com/`.
89. CAVALCANTE, Germano. fbx export || !experimental! Apply Transform Bug: Comment from Germano Cavalcante. In: *projects.blender.org* [online]. 2022 [visited on 2023-11-29]. Available from: `https://projects.blender.org/blender/blender/issues/102575#issuecomment-91227`.
90. BLENDER STUDIO et al. Human Base Meshes. In: *Blender Demo Files* [online]. Blender Studio, 2022 [visited on 2023-12-31]. Available from: `https://www.blender.org/download/demo-files/`.
91. META. Meta Quest 2 Touch controller models. In: *Oculus Controller Art* [online]. Meta Quest Developer Center, 2023 [visited on 2024-01-10]. Available from: `https://developer.oculus.com/downloads/package/oculus-controller-art/`.
92. BLENDER. Retopology. In: *Blender 3.6 Manual* [online]. Blender, 2023 [visited on 2023-12-31]. Available from: `https://docs.blender.org/manual/en/3.6/modeling/meshes/retopology.html`.

93. BLENDER. Modifiers: Decimate Modifier. In: *Blender 3.6 Manual* [online]. Blender, 2024 [visited on 2024-01-04]. Available from: `https://docs.blender.org/manual/en/3.6/modeling/modifiers/generate/decimate.html`.

94. VALVE CORPORATION. *Half-Life 2* [game]. 2004. [visited on 2023-12-11]. Available from: `https://store.steampowered.com/app/220`.

95. UNITY TECHNOLOGIES. Mesh.isReadable. In: *Unity Manual* [online]. Unity Technologies, 2024 [visited on 2024-01-04]. Available from: `https://docs.unity3d.com/ScriptReference/Mesh-isReadable.html`.

96. Water Bottle - Single-Use Dimensions & Drawings. In: *Dimensions.com* [online]. Fantastic Offense, 2023 [visited on 2023-12-11]. Available from: `https://www.dimensions.com/element/water-bottle-single-use`.

97. ZARIK5; POLYGRAPHENE; SCTANF; JACKD83; SLABÝ, Ondřej; ROSCA, David; YURY; VIXEA; SANTIAGO, Trae; CKIE; XYTOVL; NUNEMAKER, Josiah; MARIAŃSKI, Łukasz; NEXITE; CHRIS; KOREJAN; GALISTER; OTAKE, Shota; R, James; MOŠENKOVS, Dāvis; ARMARR; RAGHU, Harsha; MEISTER1593; HECL, Jonathan; URIEL; SHEIN, Ian; DAGG; IRASTRIS; LE, Charlie; KIROTTU. alvr-org/ALVR. In: *GitHub* [online]. 2024 [visited on 2024-01-04]. Available from: `https://github.com/alvr-org/ALVR`.

Contents of the medium

```
readme.md ........................ description of contents of the attachment
src
  unity .......................................... Unity's project folder
  thesis ............................... LaTeXsource code for this thesis
    text .............................. text files for individual chapters
    assets ......................... image files for figures in the thesis
      screenshots ........................ screenshots of the scenarios
thesis.pdf ................... compiled documentation of this thesis in PDF
mkshrk
  mkshrk.exe ................................... compiled game executable
```