TU WIEN Informatics

Katastrophensimulation für mobile Virtual Reality Geräte

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Media and Human-Centered Computing

eingereicht von

Thomas Hannes Wechdorn, Bsc
Matrikelnummer 01427618

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Mag.rer.nat. Dr.techn. Hannes Kaufmann
Mitwirkung: Dr. Christian Schönauer

Wien, 14. Oktober 2024

Thomas Hannes Wechdorn

Univ.Prof. Mag.rer.nat.
Dr.techn. Hannes Kaufmann

TU WIEN Informatics

Wide Area Disaster Simulation for Mobile VR Devices

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Media and Human-Centered Computing

by

Thomas Hannes Wechdorn, Bsc
Registration Number 01427618

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Mag.rer.nat. Dr.techn. Hannes Kaufmann
Assistance: Dr. Christian Schönauer

Vienna, 14th October, 2024

Thomas Hannes Wechdorn

Univ.Prof. Mag.rer.nat.
Dr.techn. Hannes Kaufmann

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Thomas Hannes Wechdorn, Bsc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 14. Oktober 2024

Thomas Hannes Wechdorn

Danksagung

Es gibt zu viele Menschen, die mir während dieser Reise zur Seite standen, um sie alle namentlich zu erwähnen. Daher bitte ich um Verständnis, wenn ihr euch in den folgenden Zeilen nicht wiederfindet – ihr seid nicht vergessen. Mein Dank gilt meinen Großeltern, Hannes und Antoinette Wechdorn, die mich ermutigt haben, mit dem Studium zu beginnen, zu einer Zeit, in der es mir schwerfiel, an mich selbst zu glauben. Ich möchte mich auch bei meiner Lebensgefährtin Sabrina Horvath bedanken, die meine eingeschränkte Zeit (meistens) akzeptiert und mich all die Jahre unterstützt hat, auch an den Tagen, an denen ich vielleicht etwas weniger einfach war, als an anderen. Sowie bei meinen Eltern Christine Hartleb, Andreas Hartleb und Christian Wechdorn, die nie auch nur einen Moment an mir gezweifelt haben. Ein weiterer Dank geht an meine Kolleg*innen in der Arbeit, die – manchmal zu meinem Leidwesen – immer an meinen Studienfortschritten interessiert waren. Ein besonderes Dankeschön geht dabei an unsere Well-Being Officer Anita Angerer, die mich stets mit Tipps, Snacks und offenen Ohren unterstützt hat. Ein großes Dankeschön geht auch an Wassily Bartuska und Michael Pucher, mit denen ich nicht nur als Kommilitone verbunden bin, sondern auch als Freund. Dank euch kann ich auf die schwierigen Zeiten unseres Studiums mit einem Lächeln zurückblicken. Ein besonderer Dank gilt auch Andreas Peer für die Hilfe bei der Organisation der User-Study und die Übernahme der Operator Rolle. Abschließend möchte ich mich bei Christian Schönauer für all die Ratschläge und die Unterstützung sowie die schnellen Antworten auf meine – ich bin sicher, zahlreichen – Fragen bedanken. Ich bin jedem Einzelnen von euch dankbar für alles, was ihr für mich getan habt und immer noch tut.

Acknowledgements

There are too many people who stood by my side during this journey to be explicitly named, so forgive me if you are not named in the following sentences, but be sure you are not forgotten. I want to thank my grandparents Hannes and Antoinnete Wechdorn, who encouraged me to go and start studying during a time, where I had a hard time believing in myself. I also want to thank my partner Sabrina Horvath, for (mostly) accepting my limited time and being supportive all these years, even on days where I may have been a bit less easy to handle. As well as my parents Christine Hartleb, Andreas Hartleb and Christian Wechdorn for never doubting me even once. I also want to thank my colleagues from work, who were, sometimes to my annoyance, always interested in the progress of my studies. A special thank you goes to our well-being officer Anita Angerer, who always supported me with tips, snacks and open ears. I also want to say thank you to Wassily Bartuska and Michael Pucher, who I not only connected with as fellow students but as friends. It's because of you guys, that I can look back on the harder times of our studies with a smile. Also, a special thanks to Andreas Peer for your help in organizing the user study and playing the role of the operator. Last but not least, I want to thank Dr. Christian Schönauer for all your advice and guidance and your very quick replies to my questions, of which I'm sure there were many. I am thankful for every single one of you, what you did and still do for me.

Kurzfassung

Die Simulation von Katastrophen in realen Umgebungen kann kostspielig und gefährlich sein. Dennoch ist es wichtig, dass Einsatzkräfte entsprechend auf katastrophale Ereignisse vorbereitet und geschult sind. Virtual Reality Training ist ein möglicher Ansatz, um Einsatzkräfte bei der Vorbereitung zu unterstützen. Dank mobilen stand-alone VR Headsets sind VR Headsets heute erschwinglicher denn je. Die Mobilität und Erschwinglichkeit gehen jedoch mit einer geringeren Rechenleistung einher. Das Ziel dieser Arbeit war es, ein Katastrophen-Trainingsszenario, speziell einen Waldbrand mit sich ausbreitendem Feuer in angemessener visueller Qualität für die Ausbildung von Einsatzkräften zu implementieren und in die bestehende Unity 3D Trainingsanwendung VROnSite zu integrieren, sodass das Szenario auf einem mobilen stand-alone VR-Headset mit akzeptabler Performance ausgeführt werden kann.

Ein Workflow zur Erstellung weitläufiger Gelände wurde im Rahmen dieser Arbeit definiert und am Beispiel eines 3 km² großen Bereichs von Stammersdorf, Wien umgesetzt. Zusätzlich wurde eine interaktive Visualisierung von Rauch und Feuer implementiert, bei der sich das Feuer ausbreitet und sowohl Rauch als auch Feuer aus der Ferne wahrgenommen werden können. Diese beiden Funktionalitäten wurden in die bestehende VR-Trainingsanwendung VROnSite integriert. Ein Waldbrand Trainingsszenario wurde vorbereitet, um die implementierten Funktionalitäten mit einer Experten-Benutzerstudie zu evaluieren. Acht Teilnehmer aus verschiedenen freiwilligen Feuerwehren aus Niederösterreich nahmen daran teil und bewerteten die Akzeptanz und Benutzerfreundlichkeit des Trainings. Das Training wurde insgesamt gut angenommen und erzielte auf der System Usability Scale eine Bewertung von 88,125 Punkten. Alle Teilnehmer gaben an, dass das VR-Waldbrand-Szenario ihnen helfen würde, sich auf Einsätze bei Waldbränden vorzubereiten.

Im Rahmen dieser Arbeit wurden außerdem verschiedene Optimierungseinstellungen hinsichtlich ihrer Performance verglichen. Dabei zeigte sich, dass Leistungsoptimierung unumgänglich ist, um eine flüssige Darstellung mit akzeptablen 72 Bildern pro Sekunde für VR-Anwendungen zu erreichen. Zudem wurde deutlich, dass Optimierungstechniken gezielt und mit Bedacht eingesetzt werden sollten, da andernfalls die Bildqualität negativ beeinträchtigt werden kann, ohne dass dabei eine echte Verbesserung in der Performance erzielt wird.

Abstract

Simulation of disasters on real training grounds can be costly and dangerous, nevertheless it is important that first responder are prepared and trained for disastrous events accordingly. Virtual Reality training is one possible approach to help with preparedness. Thanks to mobile stand-alone Virtual Reality Headsets, VR devices are more affordable than ever, but mobility and affordability come at the price of lower computational power. The aim of this thesis was to implement disaster training scenario, specifically a forest fire, with extending fires and adequate visual quality for first responder training integrated in the existing Unity 3D training application VROnSite while being runnable on a mobile stand-alone VR device with acceptable performance. Specifically, a workflow to create wide roaming terrains was defined and implemented with the example of a 3 km^2 area of Stammersdorf, Vienna as part of this thesis. Additionally, an interactive smoke and fire visualization, with spreading fire, where both smoke and fire can be perceived from a distance were implemented. Both these features were added to the existing VR training application VROnSite. A training scenario was prepared to evaluate the implemented functionalities with an expert user study with eight participants from different volunteer fire departments from Lower Austria regarding acceptance and usability. The training was overall well received, with a SUS score of 88.125 and all eight participants thinking that the VR forest fire scenario would help them prepare for operations involving forest fires. As part of this thesis, I also compared some performance optimization settings showing that performance optimization is critical for rendering in acceptable 72 frames per second for VR applications, while also showing that optimization techniques should be used cautiously and targeted, otherwise image quality can be affected negatively with no real benefit in performance.

Contents

CHAPTER 1

Introduction

1.1 Problem Statement

Rescue workers face challenging situations on a daily basis. They are trained to handle dangerous situations with appropriate measures and care to minimize damage, rescue people, and avoid worst-case scenarios. Disasters like forest fires and floods which are appearing more frequently because of climate change are presenting rescue workers with new challenges. Simulation of such catastrophic events on real training grounds are rather expensive and mistakes of the trainees can lead to injuries. Thus, there is a strong need for new and innovative training methods. Virtual Reality (VR) is one possibility to create immersive and realistic training scenarios. VR hardware became cheaper and easier to use in recent years, not least because of wireless stand-alone VR Headsets. Since stand-alone VR Headsets are independent of computers, they need their own processing units (CPU and GPU), memory and battery, all while staying comfortable. While stand-alone VR Headsets are more affordable and less restricting because of a lack of cables, they also introduce some drawbacks when creating content for such devices, especially performance. For VR applications, performance is a key factor on the one hand, as low frame rates can introduce motion sickness or other forms of discomfort and therefore making an application completely unusable. On the other hand, training should prepare trainees for real situations as well as possible, thus also requiring a certain level of realism. For this reason, the thesis aims to find a process for creating wide roaming terrains and allowing for disaster simulation in the Unity 3D Engine, striking a balance between being performant and visually adequate while still running at an acceptable frame rate on a mobile VR Headset.

1.2 Aim of the work

The aim of this thesis is to develop a wide roaming 3D terrain and a simulation of forest fires, while being performance-friendly enough to run on mobile VR devices. The 3D terrain will be based on a designated 2.99 km² area of the real-world location Stammersdorf, Vienna. To ensure that the surface model of the area is as close as possible to its real world counterpart, open geographic information system data will be used as base. The prototype will be implemented in an existing Unity 3D Virtual Reality training application VROnSite [35], with a mobile stand-alone VR Headset, such as the Meta Quest 3, as target platform. The prototype will then be validated through a qualitative user study with experienced firefighters and a targeted group size of eight, leading to insights in its usefulness, usability, and possible improvements for further development and research.

Specifically, the following research questions will be answered:
Q1: How well do expert users assess and accept first responder training of wide area disaster events on a mobile stand-alone VR system ?
Q2: How well can optimization-methods reduce the computation power needed on a mobile stand-alone VR device for the simulation of wide area disaster events while maintaining the visual realism necessary?

1.3 Methodological approach

The methodological approach taken for this thesis was started with a literature review, followed by the implementation of a prototype in the VROnSite Unity application and evaluated by both a user study with expert users and performance measurements.

- **Literature Review**: A classic literature review focusing on the state-of-the-art research on first responder VR fire trainings and applications, Fire simulation, and VR performance and optimization techniques in Chapter 2.
- **Implementation of Prototype**: Implementation of a VR wide area fire simulation with spreading fire and smoke, visible from a distance on a Virtual Environment of a 2.99 km² area of the real world location Stammersdorf, Vienna in Unity for the existing VROnSite first responder training application with multi user support in Chapter 4.
- **Expert User Evaluation**: A user study with eight expert users focusing on the evaluation and acceptance of the implemented prototype with a prepared forest fire scenario in Chapter 5.1.
- **Performance Evaluation**: A performance evaluation with six different testing scenarios to measure the performance of the current prototype implementation and compare the performance different settings in Chapter 5.2.

CHAPTER 2

State of the Art

The state-of-the-art chapter tries to give an overview of current research and applications on VR fire trainings, fire models and optimization methods used for real time applications. First, the question when VR Training is appropriate in general and if it is appropriate for the Virtual Reality forest fire training which should be implemented as part of this thesis is answered. Next, a short description of the current status of VROnSite, the application which serves as a base for the implementation part of this thesis is given, which will be followed by an overview of research on VR fire trainings and some research on high-level performance optimization methods for Unity and VR. Lastly, this chapter will be concluded by taking a look at some related applications, which didn't fit in other sections but are still relevant or of interest for this thesis.

2.1 When is Virtual Reality Training appropriate?

Before going into depth on state-of-the-art and related work, one central question should also be addressed. When is Virtual Reality Training an appropriate choice as a training method instead or even preferred compared to other methods of training? Jeremy Bailenson, founding director of the Virtual Human Interaction Lab of the Stanford University, proposed the so-called DICE criteria as guidelines to decide if a use case may be suitable for VR training or not [19]. To apply the DICE framework, we take a look at the use case if it would be realized in the real world and check each one of the criteria for this use case. If at least one or more of the criteria apply, VR training can be considered as a training method. The DICE criteria are an acronym and consist of:

Dangerous - When training in real life is dangerous, it could lead to injuries or mistakes could cost lives, e.g. a training with real fire may lead to severe burns or even end deadly.
Impossible - When experiences can't be reproduced in real life like changing the color of your skin.

Counterproductive - When the goal of the training is counterproductive to the means of the training itself, e.g. burning down a forest is counterproductive to the goal of extinguishing fire and protecting a forest with minimal damage to the environment.
Expensive (or Rare) - When the execution of the training would be so expensive or an occurrence of something is so rare that it can't be easily replicated e.g. the cost of damage done on a forest and other environmental damage plus the cost of the materials used and the repairing of said damage through reforesting, which of course not only "costs" money but also time.

Being evident in the examples above training for forest fires in the real world is at least dangerous, counterproductive and expensive and while it's not impossible it arguably is at least impractical, thus a Virtual Reality Training application for forest fires can be considered a valid VR training application according to the DICE framework.

2.2 VROnSite

VROnSite [35] is an immersive multi-user first responder VR training simulation with the goal to provide training for squad leaders. The basic setup consists of one person being the operator on a personal computer and n persons (6 was the highest person count tested yet, but more could be feasible) being the trainees wearing a mobile VR Headset. The operator has the control over the Virtual Environment (VE), called scenarios in VRonSite, and can trigger certain events like the appearance (or disappearance) of virtual persons or objects like cars or fires. The trainees have a first person view from the position of their avatar and depending on the setup can either move via standard controller of the used mobile VR Headset or with the Cyberith Virtualizer[1], a treadmill like device allowing to move in the virtual world by simply walking on the spot as seen in figure 2.1.

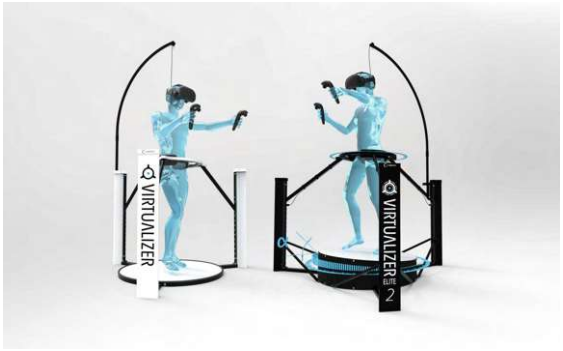


Figure 2.1: Cyberith Virtualizer

VrOnSite also records training sessions and allows reviewing and replay the recordings in its own designated replay mode. Since the fire spreading, the smoke and the VE of

[1] https://www.cyberith.com/

Stammersdorf will be implemented directly in VROnSite, a more in depth look will be provided in the Design and Implementation chapters of this thesis.

2.3 Virtual Reality Fire Training

Research on VR fire training is not a new topic. In 1997 Tate et al. developed a prototype for shipboard firefighting with a head mounted display and joystick and found that Virtual Environments could indeed prove effective as a training method [43]. Compared to then technology has evolved rapidly with the appearance of consumer level Virtual Reality Headsets led by the Oculus Rift in the year of 2013 (later acquired by Meta), followed by Headsets of other companies (such as HTC or Valve) and lastly mobile stand-alone Headsets, which are not dependent on a connection to a PC. It may not come as a surprise, that these VR Headsets (stand-alone or not) were and still are used in recent VR research.

Recent and current research on VR fire training can be roughly grouped in two categories - training for civilians and training for professionals. Training for civilians focuses on teaching how to behave in case of fire emergencies [27, 37] or training on how to effectively use firefighting tools like a fire extinguisher [36]. Training for professionals focuses on first responders and firefighters and how to effectively extinguish fires [28] or how to coordinate operations [31] or gaining insight and understanding on the spreading of fire in different environments [29].

Most of existing literature is focused on training for civilians, but since VROnSite [35] is aimed at training professionals, specifically squad leaders, the focus in this section will also be on recent VR training research for professionals and applications for professionals.

Belleman et al. developed a firefighter VR training prototype with a focus on ship fires in cooperation with the Royal Military Academy and the Belgian Navy to help face the challenge of possible knowledge loss because of upcoming retirement waves combined with extensive hiring of new recruits [21]. The prototype was developed in the Unreal Engine and provides multi-user functionality and a separate trainer view, from which for example fire and smoke can be controlled. A Vive Pro was used as VR Headset connected to a backpack PC, for allowing relatively unhindered movement. While the paper does not explicitly mention it, it can be assumed that the Vive Pro tracking via base stations was used for movement, as a limit of 6 by 6 meter is mentioned, which is the same area two Vive Base stations approximately track. The application was taken up by the Cyber Innovation Hub of the German Bundeswehr [38], but seemingly no further research based on the paper of Belleman et al. has been published yet.

Other researches like Tao et al.[42] or Braun et al. [22] also researched VR ship firefighting prototypes and Vukelic et al. [45] published a literature review in 2023 as part of the INNO2MARE project giving an overview of VR fire training research for ships (also known as maritime firefighting) and summarizing common gaps . Even though ship fires are usually enclosed fires compared to open "wide area" fires, Some of the challenges

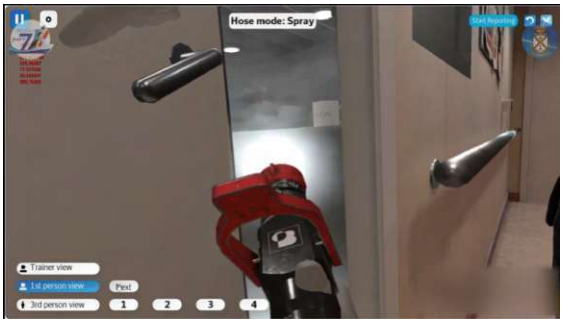


Figure 2.2: Trainee View of VR marine firefighter training developed by Bellemans et.al

between Maritime firefighting and "wide area" firefighting overlap, like the visualization and spreading of fire and smoke.

Most of the literature relies on VR Headsets in their prototypes and studies, but there also is research using other VR technologies. Lee et al. researched VR firefighter training for Commanders during the COVID pandemic to avoid possible infections during group trainings [31]. The training setup did not use an VR Headset but instead relied on VR Projection by using three projectors and three screens with a size of 4.8 meter by 2.6 meter as seen in figure 2.3.

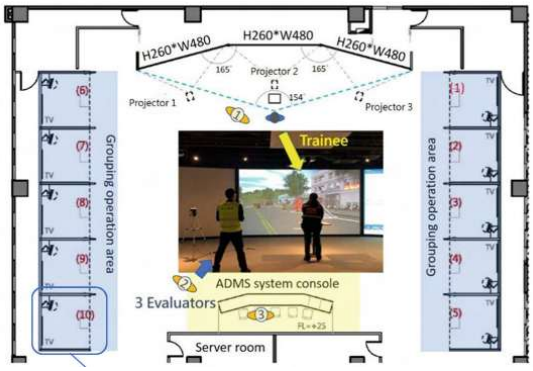


Figure 2.3: Test setup used by Lee et al.[31].

The trainee can move through the VE via a joystick and has to give commands through a wireless radio. The trainer receives the commands and acts according to these received commands with virtual firefighters inside the simulation. For the simulation, the commercial software Advanced Disaster Management Solution (ADMS) was used, where two scenarios based on real fires of tin houses and five-story buildings were created. For the evaluation, the trainees were evaluated by three domain experts before receiving a fire command course and directly after according to predefined criteria in a VR Scenario, and generally performed better after. Lee et al. argue that since every trainee has received traditional training the pre fire commander training evaluation results can be assumed as

the traditional training results, as every trainee is regularly receiving traditional training, while the post fire commander training evaluation results are the VR training results [31].

Grabowski compared the effectiveness of an VR projection training based on the Cave Automatic Virtual Environment (CAVE) and VR Headset based trainings. The CAVE setup consisted of three projected images on the walls and one on the ceiling and HTC Vive Base Stations with a controller attached to the end of a water hose to being able to measure the rotation and position and use it in the training application [28]. Additionally, the trainees can also change the amount of water shooting out of the water hose. According to Grabowski The scenario designed for the VR Headset was more difficult compared to the scenario designed for the CAVE, as the scenario for the VR Headset aims to train the use of a water hose with assistance of another firefighter and is aimed at trainees already familiar with the handling of the tools like the water hose. Overall, the cadets being part of the study found the VR Headset compared to the CAVE both slightly more immersive and slightly more useful [28].

Figure 2.4: CAVE Setup for training scenario by Grabowski [28].

Professional VR Fire Training Applications

FLAIM[2] is a commercial immersive Virtual Reality Training System developed in Australia, coming in two different versions. One is a fire extinguisher training for companies using a stand-alone VR-Headset (Vive Focus 3 at the time of writing this thesis) and the other is a training for professional firefighters, using a backpack PC, a Vive Pro non-stand-alone VR Headset , a fire proximity heat suit, simulating heat, a mask capturing biometric data and a replica water hose. FLAIM offers multiple different training scenarios and regularly receives new ones via updates. Since December 2022 FLAIM also offers some multi-user scenarios [2].

FLAIM is not only used by fire brigades in Australia but also in other countries including but not limited to Brazil [9] and the US [7].

For German-speaking countries in Europe there is FireFighterVR VR training available developed by Northdocks GmbH in Germany[4] backed by the Werkfeuerwehrverband

[2]https://flaimsystems.com/
[4]https://www.firefightervr.de/

Figure 2.5: Image of the setup for professional fire training for FLAIM. [3]

Deutschland also known as WFVD. FireFighterVR is an online platform where different scenarios can be downloaded and then run on a non-stand-alone VR Headset connected to a Computer. FireFighterVR requires users to have a VR Headset (on the level of Oculus Quest, or HTC Vive Pro) and a PC with capable hardware and does not come with any hardware delivered contrary to FLAIM. FireFighterVR does not seem to have multi-user functionality, and there is no information on how widespread its usage in German-speaking countries is.

British company RiVR (Reality in Virtual Reality Limited) developed RiVR Investigate, a VR application not focused on effectively fighting fires but on investigating virtual environments to find the original cause of fire or explosions. The RiVR Investigate setup needs a PC or Laptop, a VIVE or Oculus (with Linkcable) VR Headset. RiVR allows multi-user trainings over the internet, with the possibility to communicate and pass items between users. To keep virtual environments as realistic as possible, each scene provided in the application is based on a real fire, for which a real environment is built before being burnt by a controlled fire. To make sure that a level of realism is high when digitalizing the environment, photogrammetry is used. RiVR has a cooperation with Reality Capture, an industry-leading photogrammetry software, which was acquired by Epic Games, the developer of the Unreal Engine, in 2021.

All these professional systems have one thing in common: they are using PC connected VR Headsets. While PC connected VR Headsets allows for higher visual fidelity as the computations and rendering are handled by a PC, it comes literally at a cost. Each client not only needs a VR Headset but also needs to be connected to an adequately powered PC, to be able to run the training at all. This is one of the gaps this work tries to address, reducing the overall system cost by only needing one PC or Notebook as

Figure 2.6: Example of environments in RiVR investigate

server and one stand-alone mobile VR Headset for each client, thus effectively reducing the level of entry. As mobile VR Headsets have computing power around (high-end) cellphones and are only running on a battery instead of a stationary power supply it is necessary to keep the performance of an application in mind when building applications to reach a certain level of visual fidelity while still being able to reach relatively high frame rates around 72 frames per seconds (FPS) or even more.

2.4 Fire Simulation

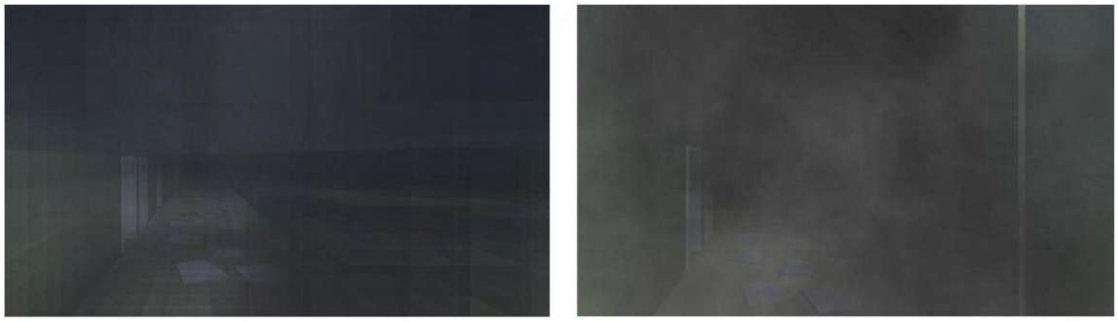
A physically accurate spreading of fire is dependent on many physical factors such as wind, the materials that are burning, temperature to name only a few. As applications targeting VR Headsets need to be performant to stereoscopically render two images at the same time, one for each eye, at high frame rates most of the maritime VR fire fighting research considered by Vukelic et al. relied on particle effects to visualize and simulate fire in a performant manner [45]. One exception was the research by Cha et al. [26] from 2012, which implemented a fire and smoke visualization in the OGRE[5] open source 3d graphics engine based on computational fluid dynamics (CFD) data computed by the Fire Dynamics Simulator (FDS)[6], which was developed by the National institute of standards and technology.

Lu et al. tried to model accurate smoke behavior in the Unity 3D engine for a post-earthquake scenario [33]. They compared particle effects with simulated CFD data from the FDS and deemed that the initial stage of smoke build up shortly after the fire breaks out deviates too much from the CFD simulation in indoor scenarios, while in the later stages when rooms are quite heavily filled with smoke and visibility is overall low particle

[5]https://www.ogre3d.org/
[6]https://pages.nist.gov/fds-smv/

Figure 2.7: VR prototype of Cha et al. [26]

effects are sufficient. A comparison of the two approaches for a room filled with a lot of smoke can be seen in figure 2.8. Volume rendering based on CFD data while more realistic is also more computational expensive, to tackle this problem Lu et al. proposed a hybrid solution where a Volume Rendering Approach is used for the smoke in the initial stages which is then switched out with particle systems during runtime when the rooms are filled with smoke for better performance. For the visualization of fire, Lu et al. relied on particle effects only instead of CFD Data.



(a) Volume rendering (b) Particle system

Figure 2.8: Comparison of Volume Rendering (a) and Particle Effects (b) for Smoke Rendering by Lu et al. in later stages where the room is filled with a large amount of smoke [33]

Lorusso et al. used CFD Data from FDS by creating animations out of the CFD Data by using the 3D Plot format[7], importing it into Paraview[8] and exporting each frame

[7] https://www.grc.nasa.gov/WWW/wind/valid/plot3d.html
[8] https://www.paraview.org/

of simulation as separate 3D object [32]. The 3D objects were merged in Blender into a single OBJ sequence with the OBJsequence plugin (Note: no reference was provided by Lorusso et al., thus the exact plugin used is unknown) and then converted into an alembic file using the Sverchok plugin[9], which can be imported into Unity. An example of this can be seen in 2.9.

Figure 2.9: Smoke example of Lorusso et al. [32]

For all the aforementioned approaches it is important to note that the computational fluid dynamics data was simulated and pre-processed beforehand, thus not being "interactive" as the pre-processed CFD data can't react to changes in the VE during real time training e.g., the extinguishing of fire or the addition of new objects to the VE during runtime. To counter this problem, Grabowski tried to simulate different scenarios via CFD depending on possible actions of the trainees beforehand and blends the different scenarios then accordingly during runtime [28].

One Fire model being able to react to such changes is the model proposed by Moreno et al. [34]. Moreno et al. divided the VE in a grid with a cell size of 3 by 3 meter, where each cell has a type (e.g., grass, tall trees, water, road), a state (e.g., not burning, burning, burnt) and some other variables like wind and slope that define its behavior. The algorithm was designed with urban and wildfires in mind and also allows fire to jump over "barrier" cells and burn a tree across water cells, if wind blows in the according direction, for example.

All the presented research was using a capable (at least for the time of publishing of the respective papers) PCs to simulate and visualize the fire and smoke, to the best of my knowledge no publication as yet was targeting a mobile stand-alone VR Headset, where fire and smoke have to be rendered on the mobile VR Headset instead of a capable PC, thus requiring a stronger focus on performant implementations of features.

2.5 Performance Optimization

Mobile VR Devices have come a long way, but the hardware gap between a mobile VR device and current PC hardware are quite large. UploadVR used the GFXBench Aztec

[9] https://sverchok.readthedocs.io/en/latest/main.html

ruins to compare the hardware of the Quest 2 with an Nvidia GTX 1060 and the Quest 2 fared around 6.13 times worse than the GTX 1060 as can be seen in figure 2.10.

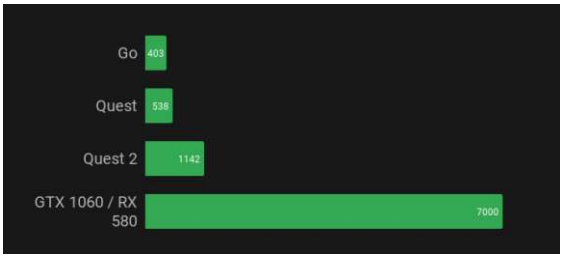
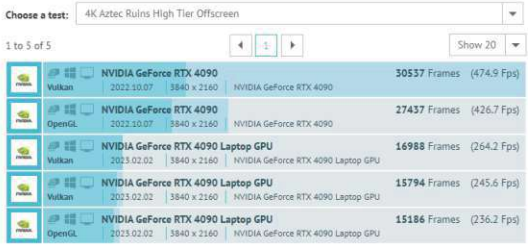


Figure 2.10: GPU comparison between Quest 2 and GTX1060

I tried to side load GFXBench on a Quest 3 to get a comparable score, but sadly the benchmark tools don't seem to run properly on Quest 3 and no score is achieved as a result. Other devices with the same 740 GPU as the Quest 3 achieve scores between 3217 and 4414 according to the GFXBench Database[10] as seen in figure 2.11a. Compared to an up-to-date high-end graphics Card the NVIDIA RTX 4090 which reaches score between 15186 and 16988 for the weaker notebook variant or 27437 and 30537 for the PC variant depending on the rendering API (OpenGL or Vulkan) used as can be seen in figure 2.11b, showing that the 740 GPU is around 3.5 to 4.73 times weaker than a RTX 4090 notebook variant and around 6.21 to 9.14 times weaker than a RTX 4090 PC variant. Since it can be seen that even devices with the same GPU can have significantly different scores, these values should only be considered as a rough estimate for the GPU power of the Quest 3 as the performance also depends on different factors like clock levels of the GPU and available memory in the device.

(a) 740 GPU scores of other devices between 3217 and 4414 for OpenGL and Vulkan on GFXBench



(b) Nvidia RTX 4090 scores between 15186 and 30537 for OpenGL and Vulkan on GFXBench

Figure 2.11

According to Meta itself, the Quest 3 is approximately as powerful as a "Rift min-spec machine" which Meta lists from a GPU perspective as an NVIDIA GTX 960 or alternatively a GTX 1050 on the minimum requirements page for the Rift [8], which reach a score between 1435 and 1767 on GFXBench as seen in figure 2.13a and figure

[10] https://gfxbench.com/result.jsp

2.13b respectively, opening up the possibility that the Quest 3 GPU may be up to around 19.12 times weaker than an RTX 4090 desktop variant.

This may suggest that the 740 may be clocked significantly lower in the Quest 3 than for example in the Samsung Galaxy S23 mobile phone used in the GFXBench comparison. And indeed, in the "The Future of Meta Quest, Mixed Reality, AI and More" presentation [3] held on the 28th of September 2023 it is stated the GPU clock of the Quest 3 depending on the used mode is between 492 MHz and 599 MHz, as seen in Fig 2.12, while the GPU clock of the Samsung Galaxy S23 is 720MHz [4].

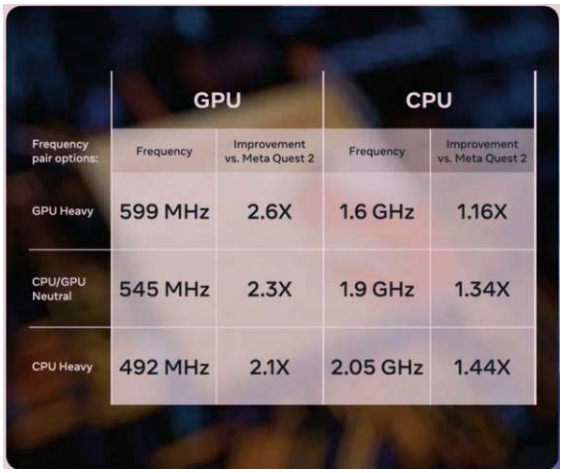
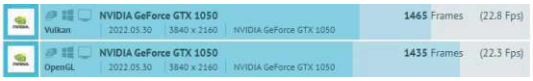


Figure 2.12: GPU and CPU Clocks of the Quest 3

(a) Nvidia GTX 960 scores between 1767 and 1850 for OpenGL and Vulkan on GFXBench



(b) Nvidia GTX 1050 scores between 1435 and 1465 for OpenGL and Vulkan on GFXBench

Figure 2.13

Research on performance optimization for stand-alone VR Headsets specifically is scarce. Hosny et al. tried to optimize an Unreal Engine 4 building blocks application for the Meta Quest [40]. After optimizing the draw calls, the game thread and GPU thread via GPU instancing, disabling of Tick, the Unreal equivalent to Unity's Update, instead using an event based updating which is only called on changes instead of every frame, and an overall reduction of polygons which need to be drawn the application was able to display 12100 blocks instead of 120 initially before dropping frames, showing that optimization is a crucial step when porting applications from a VR Headset connected to a PC to a stand-alone VR Headset.

Singh et al. tested a range of non VR specific Unity optimization techniques and compiled them and their influence on performance [41]. While the paper gives reason to critique that neither the specs of used hardware nor Unity version are explicitly mentioned, it is assumed that some findings should be generally applicable. Singh et al. grouped and compiled their findings in three categories, API Optimizations 2.1, Memory optimizations 2.2 and GPU optimizations 2.3. While there are specific test cases listed which were used to achieve the API and GPU optimization findings, it is not clear how the Memory optimization suggestions were acquired. Nevertheless, since they mostly align with the Garbage Collection best practices of Unity itself [12], they can still be considered as valid suggestions. The tables were taken over from Singh et al. and reworded based on their performance tests and the Garbage Collection Best Practices of Unity. Some details were added to the table entries as some descriptions were not specific enough, e.g., "Use StringBuilder class to build string, it build (sic!) string without allocation"[41]. As soon as the toString() method is used on a StringBuilder there is an allocation to the heap, which would not be sufficiently clear without further clarification. In The GPU Table 2.3 some entries were omitted, as they were neither explicitly tested in the paper of Singh et al. nor sufficiently explained to be deemed useful, e.g. "Optimize Image Effects with different settings" [41].

Nusrat et al. studied performance optimizations on 45 open source VR projects and identified 183 performance relevant changes via static code analysis and derived overall nine findings from it. Nusrat et al. compiled the changes in four groups, with examples for each. The groups and examples are ordered by number of appearance, meaning the first group had the most observed findings overall in the identified changes and the first example for each group was the most observed optimization belonging to this group, while the last example was observed the least.

Graphics Simplification - Reducing the complexity of shaders and the polygon count of 3D models, reducing resolution, simplifying particle effects, removing game objects from scenes.
Rendering Optimizations - Draw call batching (reducing number of draw calls), rendering setting changes (turning on occlusion culling or light baking), disabling game objects to avoid the calling of unity life cycle methods.
Heap Avoidance - Using field (member) references instead of initializing objects multiple times, avoidance of language features which create temporary objects, using primitive types instead of "boxed" data.
Other categories - Replacing deprecated or non performant API calls with more efficient API calls, replacing foreach with for loops, caching, using conditions to avoid unnecessary calculations, moving code from Unity Update to Start().

Many of the changes compiled by Nusrat et al. are also part of the findings of Singh et al. showing that non VR specific optimizations most likely are also effective for VR projects. Since the analysis by Nusrat et al. was only done via static code analysis and not tested and bench-marked, there is no explicit proof that performance actually improved.

API	Optimization Techniques
Update()/ FixedUpdate()	Replacing the use of Update/FixedUpdate per game object instance with a single Update call which calls a custom function on each game object. Singh et al. described the performance test case as "1000 instances populated with a custom function, custom function updated by single Update() function" [41]. While an interesting statement, the implementation is not described in more detail in the paper.
Loop in Update()	Reducing the execution of loops in Update by using conditions
Use Caching	What Singh et al. mean with caching is getting and storing relevant game components of game objects (like the Renderer or custom scripts) in the Start() function in a field (member) variable as reference instead of getting it each Update().
SendMessage()	Using components on the same game object as references and calling functions directly instead of triggering functions of other game objects via SendMessage()
Find()	Using components as a reference instead of using the Find() method
Transform .localposition	Transform.position gets recalculated every time it is used, so it should be replaced with Transform.localposition where possible.
Vector3.sqrMagnitude	Using sqrMagnitude() instead of Vector3.Distance() for distance checks if possible. The exact implementation or examples are not provided in the paper.
Camera.main	Using and storing the main camera as reference instead of calling Camera.main

Table 2.1: Table of API optimizations gathered by Singh. et al.[41] reworded and with some additions

2.6 Other Notable Applications

In this section some plugins and programs used will be shortly discussed together with some alternatives. A more in depth look in the specific workflows used can be found in the Implementation Chapter 4.

QGis

QGis is an open source software to view, create and edit geographic information system data like map data or digital elevation models. A commercial alternative to QGis would be the ArcGis Desktop software.

Gaia Pro

Gaia is a commercial procedural terrain generation tool for Unity 3D developed by Procedural Worlds. It allows of the creation of a generated terrain through a step-by-step assistant, which allows the configuration of the terrain and also asks if Unity project

Memory	Optimization Techniques
GC trigger frequency	Reducing the frequency of how often the Garbage Collection is triggered by avoiding allocations on the heap space.
GC trigger time	Triggering the Garbage Collection manually at non-critical moments regarding performance via GC.Collect() [12].
Caching	Since caching reuses objects instead of creating new ones, no new heap space is allocated when accessing reference fields. Every code using "new" or Instantiate allocates space on the heap, including the creation of lists and arrays. [6].
Per frame allocation	Heap space should be allocated only during Start() or loading of levels, avoid operations allocating heap space in Update() or LateUpdate().
New collection	Creating and initialising Collections in Start() and reusing them with the .Clear() method, which empties the collection but keeps the memory reserved.
Object Pool	Instead of redundantly creating and destroying game objects (e.g. for missile or bullet prefabs) object pooling should be used. Unity also has its own implementation of an object [13] which can be used.
StringBuilder	Using StringBuilder when a String gets changed multiple times in a single update as each change of the String would allocate new heap space since Strings are immutable. By using Stringbuilder only the final toString() call allocates on the heap memory.
String Concatenate	Avoid using string concatenation by separating texts if possible e.g. for Score: 44 create a fixed textfield with Score and the value separately instead of updating the textfield like score+value.toString() [12].
Debug.Log()	Removing all Debug.Log(), which are not needed for builds
GameObject .CompareTag()	Using GameObject.CompareTag() instead of directly comparing gameobject.tag with a string via double equal signs (==).

Table 2.2: Table of Memory optimizations gathered by Singh. et al.[41] reworded and with some additions

settings should be adapted to optimal settings for the usage of Gaia. Additionally, Gaia comes with a range of assets like shader, models, and textures delivered and supports all three render pipelines of Unity 3D (Standard, Universal, High Definition). Gaia also provides some performance optimizing methods like using tiled terrains, terrain streaming and automatic replacement of far away terrains with so-called lower fidelity “impostors”, basically being an implementation of the “Level-of-Detail” optimization method. Gaia uses a non-destructive workflow, meaning all operations which are done on the terrain (like changing textures, or modifying the topology of the terrain) are stored in a session manager and can be reverted. It is also possible to create Terrains from grayscale images or height maps, and all Gaia tools can also be used for terrains created with the standard Unity terrain creation workflow. For the implementation of the thesis Gaia Pro 2021 3.4.1 was used, but in the meantime a newer version Gaia Pro 2023 is available.

Rendering Processes	Optimization Technique	Optimization Examples
Batching	Reducing the number of draw calls through batching	Using the same material on different objects Consider static batching for non-moving objects, but keep memory usage in mind as Unity uses the memory to store the combined Mesh [14] Singh et al. suggest using dynamic batching, with consideration of the used memory [41]. These suggestions can be ignored for most projects as dynamic batching only works for meshes with less than 300 vertices and Unity explicitly warns that the using of dynamic batching is designed for old low end-devices and may even use more CPU resources than the overhead of a draw call [10]. Using Sprite atlases for UI elements which are shown at the same time Using texture atlases if possible for objects shown at the same time Disabling renderer components and cameras not in use
SetPass call & Batching	Reduce amount of rendered objects	Decreasing Camera Draw Distance Using occlusion culling to avoid rendering covered objects
SetPass call	Decreasing draw calls per object	Avoiding dynamic lights Using baked light and shadows Tweaking real time shadow settings like shadow quality Avoiding reflection probes
Fill rate	Decreasing number of pixels that GPU has to render each second	Decreasing the rendering resolution Avoiding complex fragment shaders Minimizing overdraw
Memory Bandwidth	Decreasing texture memory needed	lowering texture quality settings reducing the texture size Using compressed textures. Unity provides recommendations for each platform [11] Using Mip Maps for objects which are not close to the camera all the time
Vertex processing	Reducing number of vertices and vertex operations performed per frame	Lowering the number of vertices by reducing poly count of 3D models Using normal maps instead of models with high poly count for higher detail Disabling vertex tangents for meshes without normal maps Using Level of Detail (LOD) variants for models Reduce vertex shader complexity

Table 2.3: Example of some GPU optimizations described by Singh. et al.[41]

GeNa Pro

GeNa is also developed by Procedural Worlds and can be used for the creation of streets, rivers or for placing game objects with splines and other methods. Additionally, Gena can also be used to procedurally spawn objects in a scene. Since Gaia and Gena are both developed by Procedural Worlds, they have some interconnected functionality e.g. Gena can automatically split created streets on the seams of terrain tiles to be able to unload them together with the terrain when terrain streaming is used without needing much additional setup. The used version for the implementation was GeNa Pro 3.5.6.

GIS Terrain Loader, GIS Downloader & Terrain Streaming System V2

The GIS Terrain Loader, GIS Downloader and the Terain Streaming System V2 are commercial Unity plugins developed by GisTech[11]. The GIS Terrain Loader can directly create Terrains from GIS Data like GeoTiff files but also from height maps and grayscale images. Additionally, it can import GIS Vector Lines for roads and even trees. GIS Terrain Loader has no terrain streaming functionality built in. The GIS Downloader allows to directly download Map Data from multiple services like ArcGIS, Bings Maps or Mapbox. The GIS Downloader is also able to create DEMs by "collecting elevation points from different servers, focusing on free high resolution datasets with a spatial resolution of 30 meters and below."[5]. It is unclear which services are used exactly, and if DEMs for all countries are covered by that functionality. Data downloaded from the GIS Downloader can be directly used in the GIS Terrain Loader. Lastly, there's the Terrain Streaming System V2 which comes bundled with the GIS Downloader and enables terrain streaming for the downloaded terrain. The Terrain Streaming System comes bundled with the GIS Downloader, but the GIS Downloader can also be acquired separately.

[11] https://www.gistech.org/

CHAPTER 3

Design

This chapter aims to give an overview of the overall system design and a short overview of relevant existing VROnSite modules for this thesis. Since each component in VROnSite is modular, it was important that fire, smoke, and the terrain were also designed and implemented in a modular way, without having dependencies on each other or other functionalities, like seen in figure 3.1. For the overview I chose to group the modules from a software implementation perspective, which is not necessarily reflected from a user perspective e.g., the "core" functionality of VROnSite includes the "scenario editor" but from a code perspective, the scenario editor is encapsulated in such a way that it can be seen as a separate module. The only dependency additional modules are "allowed" to have is the dependency on the "core" module, which contains all the basic functionality. In practice, this also means that already existing VROnSite modules have to avoid dependencies to these new functionalities and still be working without any, with any or with all the other modules imported. This interoperability requirement made it necessary to also adapt already existing modules, as detailed in Chapter 4.

Core

The Core module of VROnSite contains the base functionality including prepared multi-user scenarios, where the operator can place additional predefined objects during runtime, the networking and trainee behavior like locomotion with controller. The operator acts as the host or server of the application and is usually running on a PC or Notebook, while for each trainee the application is running on a mobile VR Headset worn by the trainee. Additionally, the core module contains the following submodules: Interaction, Distribution and Networking and Visualization. These modules, while theoretically separate, are tightly coupled regarding functionality and structure.
The interaction submodule handles the interaction between trainees and different objects, e.g., giving trainees the possibility to drag some objects or injured persons or even virtual keyboards to use web views of web pages.

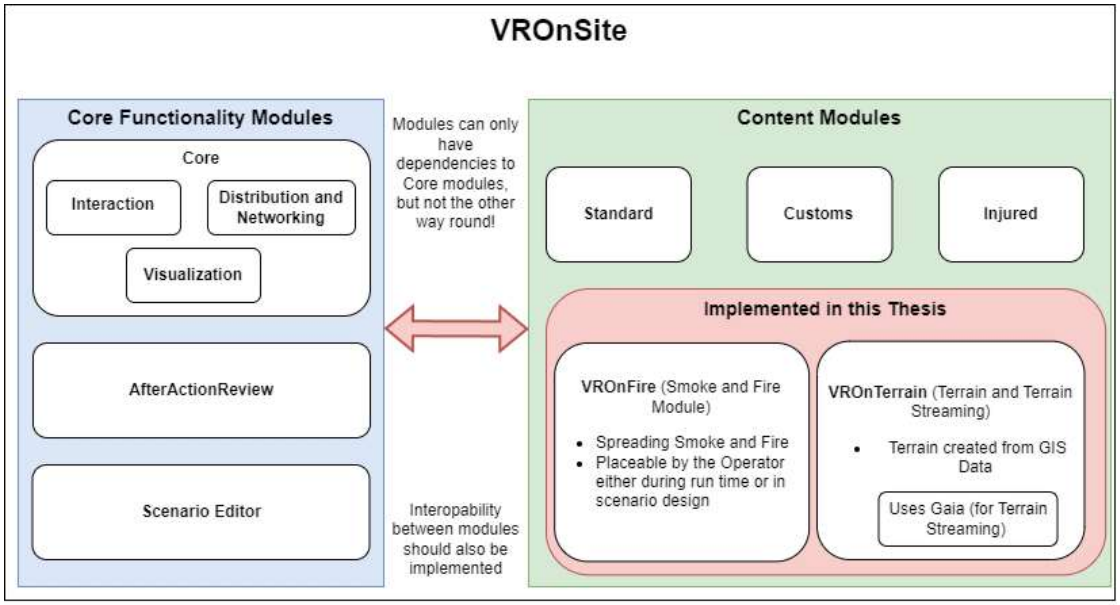


Figure 3.1: VROnSite Module Overview

The distribution and networking submodule is based on the Unity Netcode taking care of the distribution and synchronization of trainees, objects, and events like the spawning of an injured person for all trainees during runtime but also setting the scene up when trainees get connected or a new scene is loaded making sure that the training scene is the same for all trainees. Additionally, the networking submodule provides convenience methods to be used for other modules for spawning and deleting objects correctly. Lastly, there is the visualization submodule, which is using the default Unity Rendering Engine with different camera related settings and visualizations for trainees and operators, performance related graphic settings and some specific settings only relevant for VR Headsets like the activation of Foveated Rendering.

Scenario Editor

The Scenario Editor allows editing and preparing existing predefined environments, by adding spawn points of trainees and the operator and a range of predefined objects, which will be automatically spawned when starting training in the prepared scene. In the “Scenario Editor” it is also possible to mark these predefined objects as “operator spawnable” which simply means, that marked objects can also be spawned during runtime by the operator. Scenarios created by the scenario editor can then be used for training sessions without any limitations compared to predefined scenarios.

AfterActionReview Module

The AfterActionReview module records the training sessions and allows reviewing them, with the possibility to pause at any moment, to evaluate the training and give feedback to trainees. The recording contains the voice of the operator, the movement of trainees (as seen by the operator, but also from the first person view of the trainee) and the spawning and de-spawning of objects, in addition to the movement and transformation changes of objects.

Standard

The standard content module includes interactive and non-interactive assets used for the creation and adaption of scenes like buildings but also models and animations for first responder and fire brigades, including vehicles and audio files.

Customs

The customs submodule contains assets for special use cases like a virtual environment for an airport or custom controls including models for the in and outside of an airport, planes, and custom tools like a fingerprint scanner.

Injured

The injured submodule contains models, animations, and audio for persons of different ages with a range of different injuries.

Fire and Smoke Module

The Fire and Smoke Module handles the visualization and behavior of fire and smoke, suitable for mobile stand-alone VR devices. The fire had to be integrated in the core module and scenario editor, as it needs to be placeable (and deletable) by the operator during runtime as well as from the scenario editor. Additionally, the fire needed to be implemented in the AfterActionReview Module to not only be visible during replay, but also to have the same spread behavior as during a training session. The spawning (and control) of fire and smoke should be handled by the Server Application, while the rendering should be handled directly on the device it is shown on, specifically the mobile VR-Headset for Trainees, PC or Notebook for operator. For the spread of the fire, a simple spread-over-time algorithm was used. The fire is growing over time with a configurable rate until a configurable maximum size is reached. As soon as it touches another “burnable” object, the other objects also start burning, which in turn again starts a small fire, which grows over time and is able to “burn” other objects in reach, thus effectively spreading over an area via “burnable” objects. The algorithm makes sure that an object can only start burning once. The operator has the ability to “extinguish” each single spawned fire individually, thus hindering the spread. Additionally, there’s the possibility to write “burn down” behavior for different objects. For the prototype, only

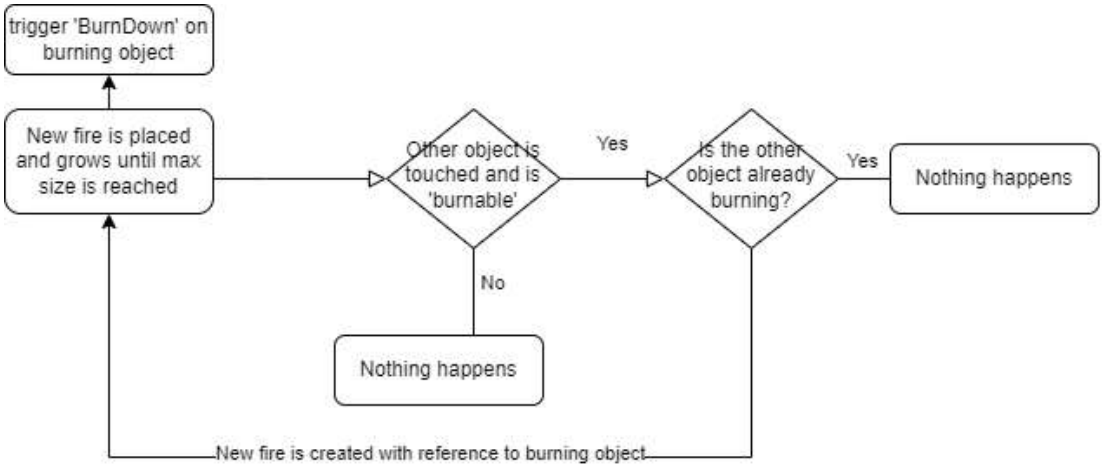


Figure 3.2: Basic behavior of the fire spread algorithm

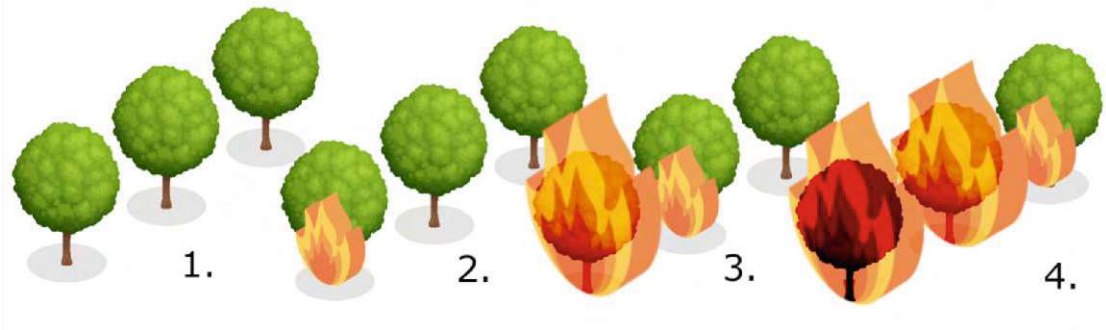


Figure 3.3: Visualized explanation of the fire spread algorithm

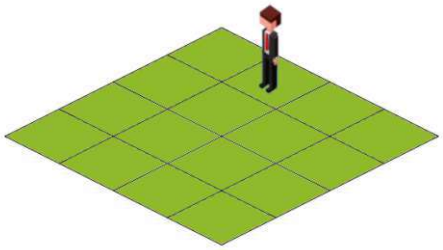
the tree burn down was implemented, where the model gets changed to a burnt down tree model after some time or if the fire gets extinguished, whichever happens first. A simplified flow chart of the algorithm can be seen in 3.2.

A visualized explanation of the basic behavior of the fire spread algorithm can be seen in figure 3.3, where 1. is the set-up of the scene with no fire yet. In 2. a fire gets placed - in case of VROnSite by the operator. In 3. The fire grows big enough to touch and ignite an additional tree and finally, in 4. the first tree model changed to a burnt tree model, while the fire of the second tree ignited the third tree but did not add another fire to the first tree.

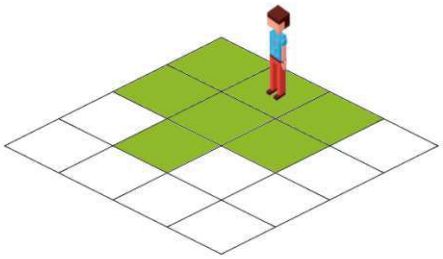
Terrain Module

The Terrain Module handles the visualization of a terrain in the Core module and the Scene Editor module, as well as in the AfterActionReview module. This includes the loading and unloading depending on the position of the trainee (Streaming), as well as separate default settings for the operator and trainees and the ability to adapt these

settings, as the application used by the operator is usually run on a notebook or PC and thus can handle a higher load than a mobile stand-alone VR Headset could. A basic visualization for the loading behavior of the terrain can be seen in figure 3.4a for the operator, where all tiles are loaded and in figure 3.4b for the trainee, where tiles are only getting loaded within a certain distance to the player position, to free up memory and thus reduce the impact on performance.



(a) For the operator, all terrain tiles are loaded (green) and none unloaded, as the operator is usually run on capable hardware.



(b) For the trainee, only tiles within a certain distance to the player position are loaded (green) while the rest are unloaded for performance reasons (white).

Figure 3.4: Character images were taken from `https://www.vecteezy.com/vector-art/987953-isometric-people-character-set`

The implementation actually resulted in two artifacts - one, reusable objects (prefabs) to be used in Scenes with terrains to set up VROnSite and Gaia accordingly without the need for additional custom code, and two - a scenario of a designated part of Stammersdorf, usable in the scenario editor for the creation of custom scenarios.

CHAPTER 4

Implementation

This chapter aims to provide a detailed and easily reproducible description of the implementation of the Terrain creation, the spreading fire and smoke and the performance optimization for mobile VR devices in the Unity engine. The VROnSite project provided used the Unity LTS 2021.3 version, thus this was also the version used for the following feature implementations. All implementations are also possible on the latest Unity version to date (2023.2) as of writing this thesis. For the sake of completeness, it should be mentioned that a Windows computer and the Windows distribution of Unity was used for development. Additionally, if multiple approaches were considered for a feature implementation, the alternative approaches will be mentioned together with an explanation about the reasons which contributed for choosing one over the other.

4.1 Terrain

A terrain or Virtual Environment for Virtual Reality consist of one or multiple 3D textured meshes. In training applications user are usually placed in a first person-view in the Virtual Environment, having the freedom to look any way they like, moving through or on the VE by the implemented locomotion method e.g., walking on the terrain via controller or teleport through it. In VROnSite the user can move on VEs by walking on them with a controller or by driving on them with prepared vehicles. Movement and Collisions are handled via Unity Physics System and so-called colliders, which can be described as simplified geometry to represent complex meshes and their interactions in a physical based layer. To be able for the user to walk on the terrain, both the player and the terrain need a collider. Unity provides a special “Terrain Collider” for Unity Terrains, which while simplified, deliver an accurate representation in the physics simulations. This introduces some requirements for the creation of a terrain based on a real environment in VROnSite and other first-person VR applications: Since movement is physics based, the terrain should be as accurate as possible regarding elevation and smoothness, to

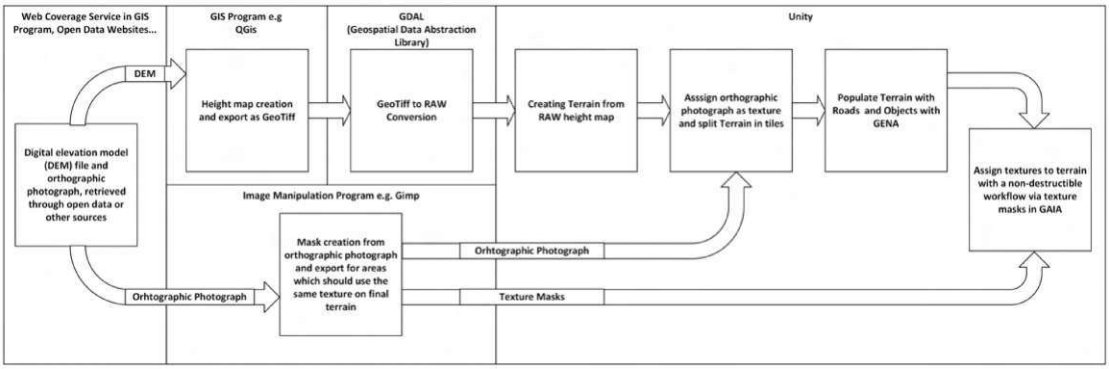


Figure 4.1: Visualized terrain creation workflow

make sure all important areas are accessible and not blocked by steep slopes, walls, or rifts not existing in the real counterpart. Additionally, the placement of objects like buildings or trees according to an orthographic photo, also requires faithfulness to the real environment to a certain degree, especially if buildings should be accessible, while still being not too taxing on the intended hardware.

The approach taken for the implementation is visualized in 4.1 and can be roughly broken down in the following steps:

1. Creating a height map from digital elevation models
2. Creating a Unity terrain from the height map
3. Terrain tiling and streaming
4. Populating terrain with objects
5. Texturing Terrain with texture masks created from an orthographic photograph

Height map Creation from Digital Elevation Models

For the implementation of the terrain, a digital elevation model (DEM, German: Digitales Geländemodell or DGM) of the target area Stammersdorf was used. DEMs for Austria can be found on the open source government platform data.gv.at. DEMs can be either provided directly as GeoTiff files or acquired via a geographic information systems (GIS) program (e.g., QGIS, ArcGis) if the source provides a web coverage service (WCS) interface for the relevant area.

To get suitable DEM data of Vienna, the public GeoDaten-Viewer[1] of the Stadtvermessung can be used.

[1] https://www.wien.gv.at/ma41datenviewer/public/

The DEM files in the Geo-Daten Viewer Vienna are provided as tiles with resolution of 1 meter and a size of 2500m by 2500m. They can be downloaded by clicking on a tile and selecting the wished format in a pop-up, which in my case is a TIFF file.

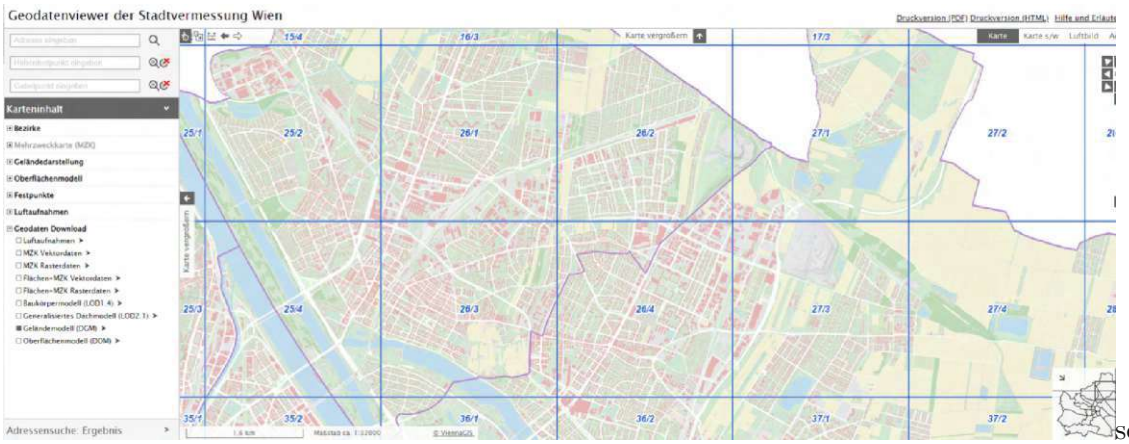


Figure 4.2: Screenshot of the Vienna Geodaten-Viewer

While the city of Vienna also provides a Web Map Service (WMS), a Web Map Tile Service (WMTS) and a Web Feature Service (WFS) only the WMS provided actually contains an elevation layer allowing to visualize the elevation in a GIS viewer. It is important to note that WMS only provide image data which may look identical to actual elevation data in a GIS viewer but instead of providing height data only provide lower bit depth image data, which results in a loss of accuracy when being used as a height map compared to a GeoTiff and is therefore not an optimal choice. To be precise, the image provided by the WMS service is a greyscale PNG image with a depth of 8 bit, meaning that the possible height values will be limited between 0 and 255, while GeoTiff allows to store higher bit information. For the purpose of Unity terrain creation usually a 16 bit height map is used, but GeoTiff in general allows a bit depth of up to 64 bit depending on the program used for reading or creating the GeoTiff e.g. GDAL[2]).

Usually, it is necessary to match the area contained in the DEMs to the designated target area. This can be achieved with a GIS program, for this thesis QGis 3.26 was used.

First, for easier visual navigation, an orthographic photograph of Vienna from the year 2022 was imported. This was done by adding the WMTS[3] provided by the city of Vienna as a source in QGIS and loading the orthographic photograph as a layer. Then, the aforementioned DEMs with the GeoTiff format were loaded into QGIS, as seen in Fig. 4.3.

Since each file is imported as a separate layer, the layers were merged via the raster merge function of QGis. Afterward, a quadratic shape layer in the size of the designated target

[2]https://gdal.org/drivers/raster/gtiff.html
[3]https://data.wien.gv.at/daten/wmts/1.0.0/WMTSCapabilities.xml

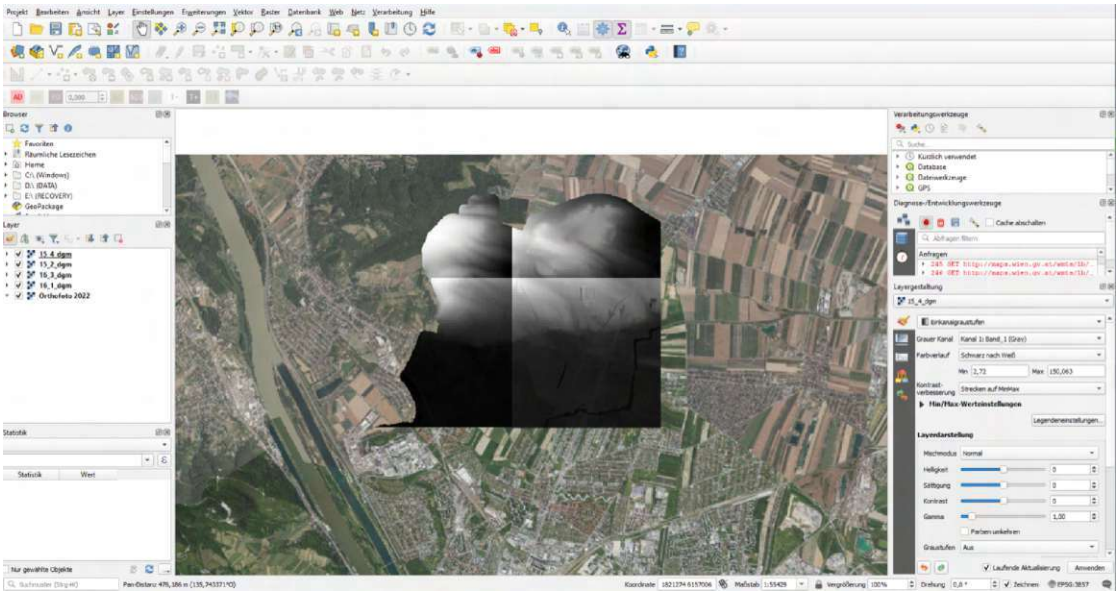


Figure 4.3: Screenshot of imported orthographic photograph and digital elevation models (DEMs)

area with 1730 meter by 1730 meter was created and used as a boundary for exporting the merged layer. Here, the minimum and maximum value of the resulting layer, which can be found in the layer properties as seen in figure 4.4 are of interest, as they will be used on two occasions in the set-up process:

1. For converting the final GeoTiff to a Unity compliant RAW file.

2. As a Y (height) value for the terrain inside of Unity.

For our designated area, the minimum was 7.193 meter and the maximum 93.317 meter, resulting in a difference of 86.124 meter.

Figure 4.4: Showing the minimum and maximum value of the merged layer in the layer properties

Depending on the coordinate reference system (CRS, German: Koordinatenbezugssystem or KBS) used (or by mismatching CRs of layers) it may happen that the measurements of the shape layer may not be accurate, therefore measuring the shape file manually with a measuring tool is strongly advised at this step.

Finally, the matched layer can be exported as GeoTiff and converted with gdal (Geospatial Data Abstraction Library), which is automatically installed together with QGis, to a Unity conform RAW in the command line with the following command :

```
gdal_translate.exe -of [file format] -ot [data format]
-scale [input min value] [input max value]
[output min value] [output max value]
-outsize [outsize resolution] [outsize resolution]
[name of input file] [name of output file]
```

Exemplary, the RAW conversion of Stammersdorf was done with the following command:

```
gdal_translate.exe  -of ENVI -ot UInt16
-scale 7.1963 93.317 0 65535
-outsize 4097 4097
height_map_1m.tif heightmap_1m_4096.raw
```

An explanation of the parameter used can be found here, note that obvious parameters like “name” have been left out. Other available parameters can be found in the official gdal documentation. [4]

- file format: ENVI - has to be used for .raw files, additionally creates a .hdr and raw.aux.xml file. The .hdr file and raw.aux.xml file contain meta information, while the .raw file contains the image data. For Unity height map purposes, only the .raw file is needed.
- scale: scales the incoming values according to the output values, e.g. 7.1963 -> 0, 93.317 -> 65535
- data format: UInt16 - Unity expects a .raw file in a 16 bit format.
- outsize resolution: resolution of raw file - (power of 2) + 1 e.g. 257, 513. Note: If the terrain is created with the Unity Terrain Toolbox plugin instead, an exact power of 2 resolution is expected.

In the following section, two workflows for creating terrains from .RAWS will be presented, as both were used initially to compare ease of use and overall functionality:

[4] https://gdal.org/programs/gdal_translate.html

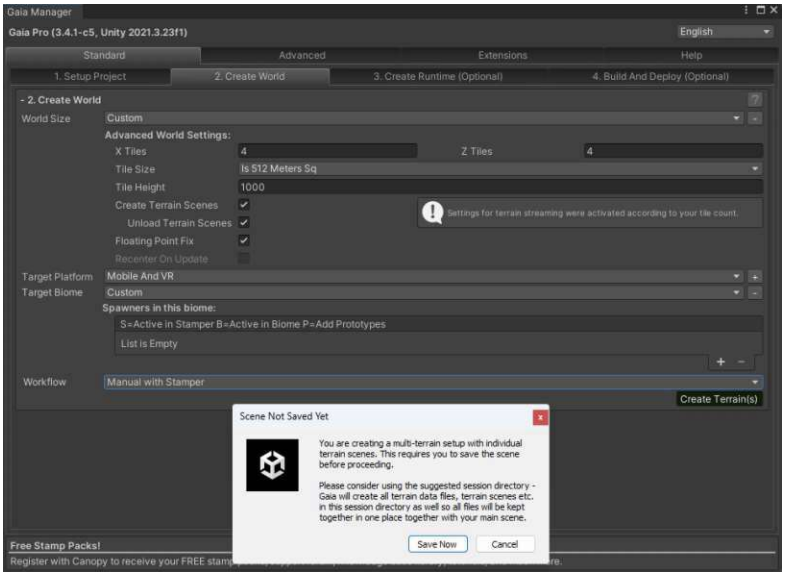


Figure 4.5: GaiaManager

1. Terrain Creation with Gaia
2. Standard Unity Terrain Creation Process

4.1.1 Workflow 1: Terrain Creation with Gaia

The Gaia plugin also allows creation of terrains from height maps, which then can be easily integrated in the Gaia Pipeline for rendering and streaming functionalities. Compared to Unity, the source height map can also be a grayscale image, but leads to a loss of detail compared to RAW files as mentioned above.

To create Terrains in Gaia the Gaia Manager has to be used. In figure 4.5 the settings I used for setting up the terrain can be seen. When creating a Terrain with Gaia it is automatically checked if Texture Streaming and Incremental Garbage Collection are activated in the Unity Project Settings to avoid performance spikes when loading terrains together with a recommendation to activate and the option to activate either of these settings immediately. Incremental Garbage Collection is actually activated by default when creating a new Unity Project, but was not activated in the already existing VROnSite Unity Project, which was used for implementing the Terrain creation.

Initially, Gaia creates a “random” Terrain in the Scene with the values defined in the set-up. To adapt Gaia Terrains, so-called “Stamper” have to be used. Stampers are, like the name implies, Gaia specific stamps which can be used to work on the terrain with different operations like raising or lowering height. Stampers can be created via another Gaia Component called the Scanner, which can create Stamper from Textures, Meshes, other Terrains or RAW height maps. The Scanner is also activated via the Gaia

Manager, which then expects one of the aforementioned files to be dragged and dropped in it as seen in figure 4.6 and saved as a scan. Then a Stamper has to be created via the Gaia Manager and the newly created scan selected via Stamps -> Exported Scans. The scale of the created stamp is seemingly not adhering to real world measurements, thus special care needs to be taken of the height value for the stamp. In my case, setting the Y-Axis of the stamp to the difference between lowest and highest point of the height map (the 86.124 meters calculated above) did result in a vastly different overall height for the stamp. One workaround to still set the stamp to a nearly accurate height, is by creating a 3d Object in Unity, setting the calculated height to the object and adapt the Y-scale of the stamp until it matches the height of the 3D object as close as possibly. A tenth of the calculated Y-value (8.6124) provided a good approximated value to start, but was still not completely accurate. Additionally, it should be noted that the Stamper Preview is also not completely accurate, and the resulting terrain after the stamp will not be completely in line with the preview.

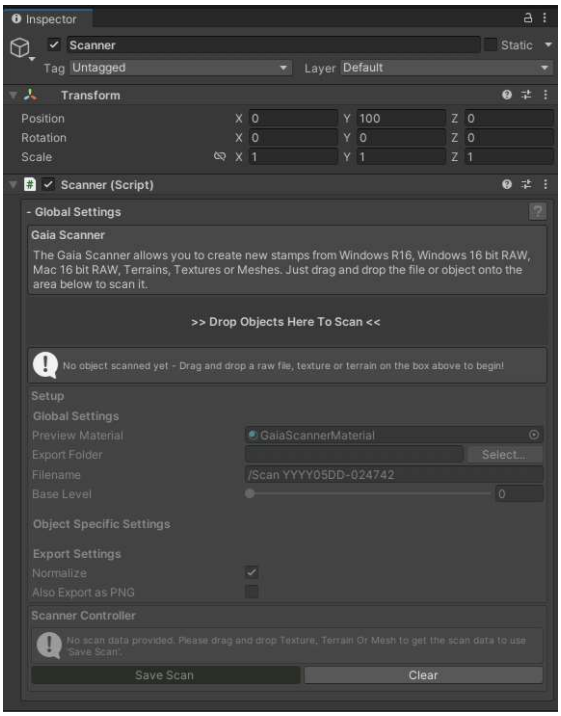


Figure 4.6: Screenshot of the Gaia Scanner

The resulting terrain was already split in tiles by Gaia for performance according to the settings chosen in the set-up, has already set up a working Terrain Loader automatically, which handles Terrain Streaming for the multiple tiles and can be further configured via Terrain Loader Settings. Nevertheless, after comparing the Gaia terrain creation options with the integrated Unity terrain option, ultimately, the native Unity terrain generation process was chosen to be used for the final implementation. This decision leads to some difficulties when integrating the terrain in the Gaia set up for using Gaia functionalities

like the terrain loading and streaming. The deciding factor to use the standard Unity Terrain creation pipeline was the fact that Gaia does not allow to generate terrain tiles in arbitrary sizes, but only in 8 predefined sizes 256 meters to 16384 meters in power of 2 steps (e.g., 512, 1024 etc.) per tile. Specifically, this would mean 36 (6 times 6) tiles with 256 times 256 meters each for a terrain of 1778 meters times 1778 meter or 16 (4 times 4) tiles with 512 times 512 meters each for a terrain of 2048 times 2048 meters. This, combined with the above-mentioned height value inaccuracy, lead to continuing with the standard Unity terrain creation approach, which will be explained in the following section.

4.1.2 Workflow 2: Creating Unity Terrain from Height Map

Depending on the size of the terrain, it is recommended to split the terrain in multiple tiles and implement terrain loading (also known as streaming) which loads terrains only if the viewer is close enough to them. With Gaia this was handled during terrain creation, for the Unity approach this needs to be handled separately. Detailed comparisons and the impact on performance depending on number of terrains, terrain size and the usage of level of detail (LOD) will be described in the Evaluation chapter.

There are multiple ways to achieve the tiling of a large terrain, like directly exporting multiple smaller RAW files with the process explained in the section ***Preparing and creating a height map for Unity***. When this approach is used, the min and max height values and the height difference of every tile need to be used for creating the corresponding terrain tile in unity.

Another possibility would be to use an image processing tool allowing the processing of RAW files like ImageMagick to tile the resulting RAW file containing the whole area into smaller tiles, resulting in one .RAW file for each terrain tile.

One straightforward solution to create terrain tiles is using the Unity Terrain Tools package, which is supported starting with Unity 2021.1, as there's no need to handle separate tile height values or manual assignment of textures for each tile should any be used. For Unity versions below 2021.1 the Terrain Tools exists as a preview package starting from version 2019.1.

To use a RAW file for Terrain creation, first a new Unity Terrain object needs to be created. There in the inspector under Texture Resolution (On Terrain Data) the Import Raw... button can be found.

The following settings have to be used for height maps created by this workflow:

- Depth: Bit 16
- Set Resolution manually, if not taken automatically
- Byte Order: Windows
- Flip Vertically

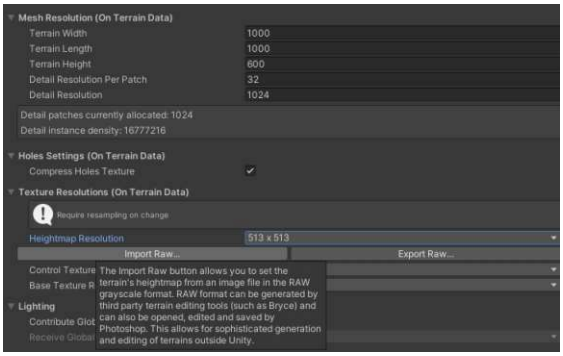


Figure 4.7: Import Raw Button in Inspector of Unity Terrains

4.1.3 Setting up Terrain Streaming with Gaia

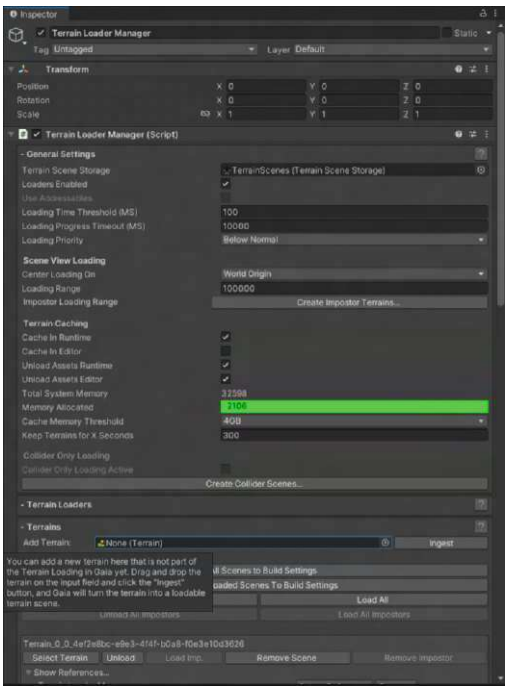
In the approach that Gaia uses for Terrain Streaming, each terrain tile is contained in its own scene, allowing to unload and load terrain scenes as additive scenes separately. Since loading and unloading is a costly operation, Gaia also implements a Cache, which keeps the Terrain in memory by only deactivating the root GameObject of the terrain scene for a certain time if the player reaches a configured distance before actually unloading it.

In case the terrain was created by Gaia directly, the set-up happens automatically, but it is also possible to use Gaia's Terrain Streaming functionality for Terrains created with the standard Unity approach.

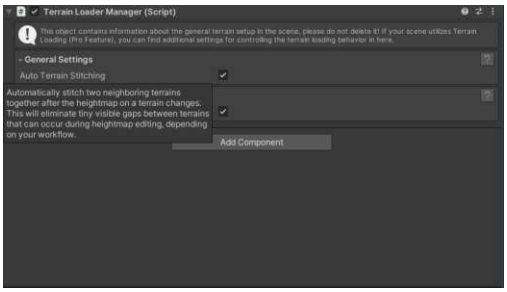
If a Terrain Loader Manager is added to a scene with terrain created with the Unity workflow, it is missing all options compared to a Terrain Loader Manager set up by Gaia automatically, as seen in figure 4.8b.

The reason for this behavior is the Gaia Session Storage, where each change in the terrain (made with Gaia Tools) is recorded to enable a non-destructive workflow. A non-destructive workflow allows reverting to each individual working step, compared to a destructive workflow where only the last state is stored. The creation of the Terrain Loader Manager component also created a Gaia Session Manager found under the Gaia Tools Objects. There, a field named "Session Data" can be found, which leads to the Path where Gaia stores its sessions (*\Assets\Gaia User Data\Sessions*). Inside there is a separate folder for each session (e.g. GS-20240324 - 165359 -> GS-Date - Time) containing a TerrainScenes.asset, which is the so-called Terrain Scenes Storage. This is used by the Gaia Loader to determine which and how many scenes are part of the corresponding main scene and can be loaded in both, the editor and during runtime. After manually creating a Terrain Loader Manager, the List of Terrain Scenes here is as empty as seen in figure 4.9. After increasing the Terrain Scenes List size by pressing the + button and filling out the data of one of the terrains to be streamed, the Terrain Loader Manager shows all the options needed to set it up.

Another option is to create a temporary Terrain with Gaia, and then use the Add Terrain Field scene in figure 4.9. Here, one terrain at a time can be added by either using



(a) TerrainLoaderManager created by Gaia



(b) TerrainLoaderManager created manually

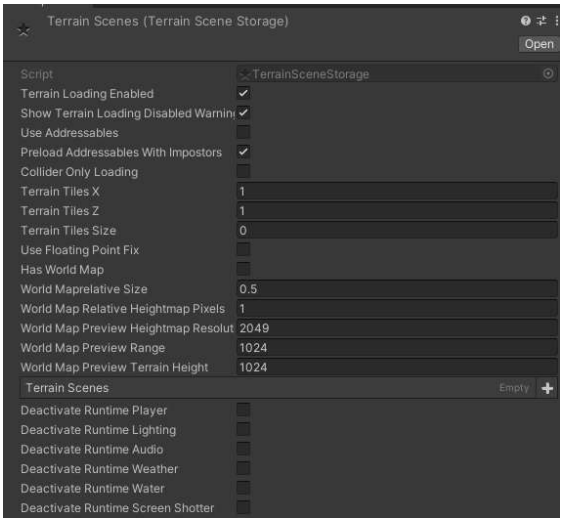


Figure 4.9: TerrainScenes with empty Terrain Scene list

drag and drop or searching for the correct terrain data and then confirming with the "Ingest" button. "Ingesting" terrains configures the terrain not only for the Terrain Loader Manager but also in the Session Storage.

Since both of these workflows are rather tedious, I created a helper Unity Editor script as part of the thesis, which allows for a faster set up. When Gaia Ingests a terrain, it is moved to a newly created separate scene and added to the Session Storage. The solution I created is basically a copy of the relevant code triggered by the "Ingest" button but instead of being executed for only one terrain, the script executes the code for each selected terrain in the Scene Editor instead. Since ingesting is setting up terrain tiles correctly in both Terrain Loader Manager and Session Storage, no further tinkering with either of them is needed after execution of the script. When opening the "main" scene in the editor, the Gaia Terrain Loader now handles loading all terrain tiles as additive scenes in the editor. Terrain Loading is now functional in the editor and also during runtime in case a Gaia Player is used. Since VROnSite uses its own "player" objects, or to be more precise objects having a camera and thus serving as "window" into the VE for the user, it is necessary to set them up accordingly as well, which will be explained in the following section.

4.1.4 Integrating Terrain Loading in VROnSite

Gaia splits the scene setup in one main scene containing necessary scripts for Gaia in Editor and during Runtime (Terrain Loader Manager and Session Manager) and multiple additional scenes each containing a terrain tile. The Terrain Loader Manager handles loading and unloading of the Terrains in the Editor, but since terrain loading during runtime should be dependent on the position of camera, it is necessary to equip each potential "player" game object containing camera on it or in its hierarchy with a Terrain Loader. VROnSite has four such objects, stored as Prefabs: the "Operator" prefab, the "Trainee" prefab, the "EditorManager" prefab and the "Replay" prefab. The "Operator" prefab is spawned on the server, meaning there's only one in the scene, and is usually run on a notebook or pc. The "Operator" prefab has all the necessary operator scripts like for example the operator camera, operator movement handling and the operator spawn manager, which allows spawning new objects, visible for all connected instances, into the VE during runtime. The "Trainee" prefab is spawned for each trainee (usually using a VR headset) connected to the server and contains all the necessary trainee scripts and objects like for example the camera, the movement handling and network objects. The "EditorManager" prefab is used as "player" object, when the scenario editor is used, where new scenarios can be either created or existing scenarios edited. Lastly, the "Replay" prefab is used in the AfterActionReplay, where scenario replays can be watched.

Since each of these prefabs is not made up of a single object, but instead a hierarchy of multiple game objects, it is necessary to make sure that the terrain loading is set up on an object which actually reflects the correct position of the "player object". To illustrate the problem with an example: If the terrain loader would be placed at the root object called "Player", but the movement and thus the translation is actually happening on a

child game object called “Trainee” the terrain loading would actually not work correctly, as the unchanging translation of the root object “Player” would be used for calculating the relevant terrains and since no movement is happening on the root “Player” object but only on the child, the root “Player” object always stays at the same position. The safest way to set up the terrain loader is finding the game object containing the camera and attaching it to said game object. Even in the case that the game object containing the camera is not moved directly, but instead a parent object is moved, the correct translation will be used as the global position instead of the local position is used for calculating the relevant terrains for loading or unloading.

The obvious solution would now be simply placing the Terrain Loader script provided by Gaia in the prefab on the object containing the camera. That, however, would break one of the principles established in the design chapter. By placing the Terrain Loader directly in these prefabs, a dependency from a functionality to my VROnTerrain module be established and only dependencies from content modules to functionality are allowed, not the other way round. To tackle this I created a GaiaSetup prefab, which can be placed in a Terrain scenario and sets itself during runtime, dependent on the mode started, and allows setting terrain loading boundaries for both operator and trainees.

During Update, the GaiaSetup script scans for one of relevant player game objects in the following order:

1. Check if the VROnSite ClientManager has an instance defined and get the gameobject in case it is - ClientManager.Instance returns the trainee object containing the relevant trainee game object for the connected client if executed as trainee or an operator game object if executed as operator.

2. Check if the LobbyManager has an instance defined and that the instance has a reference to the EditorManager set, in case it is get said game object.

3. Check if no game object was found yet and a game object with the name "ReplayCamera" is existing and get the associated game object.

4. If a game object was found, add a Gaia TerrainLoader, set a custom terrain loading mode called RuntimeAlwaysVrOnSite and set it up with predefined bounds, which are used to check if terrains should be loaded or unloaded and set the setup variable, so the checks are not getting executed anymore. Should nothing be found, try again in the next update. A more detailed explanation of TerrainLoaderModes can be found below.

As a side note: Finding game objects by name is a costly command and should be avoided if possible, since it’s only happening during the loading of the scene as the GaiaSetup script is only in the terrain scene itself, not in the Lobby, which works as a hub for the operator. For trainees on mobile VR headsets the find GameObject should not be called at all, as the ClientManager should already be available, thus performance impact should

be negligible, as the execution of the logic above is not repeated as soon as a game object was set.

Gaia provides some predefined TerrainLoaderModes: Disabled, EditorSelected, EditorAlways and RuntimeAlways [18]. Each of these provide the possibility to set boundaries i.e., the range in which terrain tiles should be loader or unloaded and the RuntimeAlways mode additionally, allows setting a minimum distance and maximum distance and time values for each distance how often the need to load or unload terrains should be checked. In VROnSite there should be different boundaries, since the Operator is usually run on a PC or Notebook the possibility to set larger boundaries should be given. Thus, I rewrote the Terrain Loader script provided by Gaia to offer an additional Mode called RuntimeAlwaysVROnSite in which it is possible to set distinct trainee and operator boundaries. Since the Scene Editor and the AfterActionReplay are also usually run on a Notebook or PC, the code for the RuntimeAlwaysVROnSite mode checks with the VROnSite ClientManager instance if the player mode is trainee, which then results in setting of the trainee boundaries or if either no player mode is set or set to operator, then the operator boundaries are set.

4.1.5 Terrain Objects

Unity standard terrain texturing uses the same approach as is used for terrain sculpting, adding trees and other details like grass, brushes. These brushes can be scaled up to a size of 500, which would correspond to 500 meters with default Unity settings. One option would be to take an orthographic map as reference and paint textures accordingly onto the terrain. Another option is to use Gaia's non-destructive workflow and apply textures according to certain spawn rules, like using a texture on the terrain that has a defined distance to a certain point on the terrain or if a slope has a certain angle or for a certain height. These spawn rules can be used in combination for a single texture, but can also be configured separately for multiple different textures. I used an Image Mask spawn rule, where black areas in the image should be textured on the terrain, and transparent areas ignored. Gaia gives multiple possibilities how the mask can be set up, it would also be possible to paint the transparent areas and ignore the black ones, or use one of the r (red), g (green), b (blue) channels of the image as a mask. I created the mask in an image editing software (GIMP) based on the landscape map of the area via an automatic selection tool and then filled the selected areas with black and deleted the rest. An example of such a mask can be seen in figure 4.10.

The masks were saved with 1730 times 1730 pixel, corresponding to the terrain size of 1730 by 1730 meter. The rules are applied in sequential order, meaning that if the first rule draws only over parts of terrain, and the second rule draws over the whole terrain, then the results of the first rule will be overdrawn by the second rule. If the order of the rules will be swapped, then the first rule paints the whole terrain and the second will only draw over the parts, thus having a combination of the two rules as a result. Overall, I defined four texture rules, each with a different texture: A Grass rule without any mask set, thus painting the whole terrain with a grass texture, a Fields rule and two different

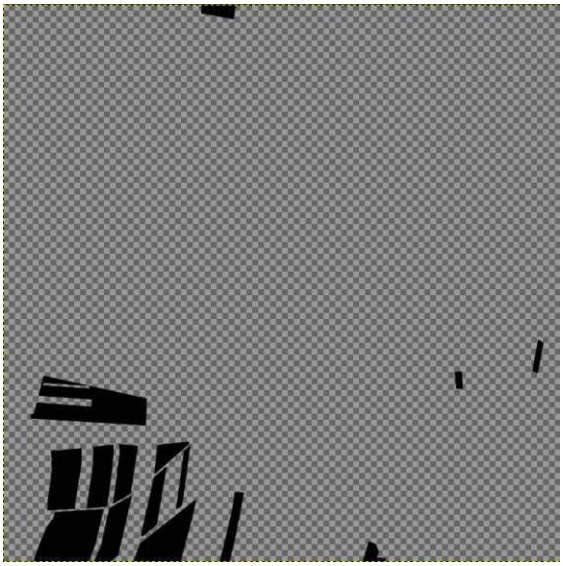


Figure 4.10: Example of a mask created for fields used for terrain texturing

Grapevine rules which all use a different mask to texture fields and two different types of grapevine soils over the whole terrain according to the masks. At this point it should be noted that only four terrain textures (per terrain tile) were used for performance reasons and that in general Spawners should only contain textures which will be used on the terrain, as even set up textures which are unused have an impact on performance. A more detailed explanation will be given in the Evaluation chapter. figure 4.11 shows the Spawner settings used, the list of the Spawn rules and a preview of the areas which will be textured marked by a color related to a spawning rule. A Spawner provides two buttons, “Fit to Terrain” and “Fit to World”. “Fit to Terrain” sets the range value in the spawner to match the size of the Spawner to a single terrain Tile, while “Fit to World” matches the size of the Spawner to the Gaia World, which includes all terrain tiles. In the Stammersdorf scenario, when using “Fit to World” the range is set to 865 as the range is measured from the center of the terrain (or terrain tile) and thus is set to half of the terrain size, which 865 (half of 1730) meters in my case.

To set up an Image Mask in the Spawner, Mask Settings needs to be added, by clicking on the "+’-button of a rule and opening extended settings. For Stammersdorf I used an Image Mask with the Alpha Channel as Filter Mode, as seen in figure 4.12. All the other settings have been left on default.

The areas of interest to be used for training are the main square of Stammersdorf and a parking area in the Senderstraße, where a forest is also located. Thus, it was necessary to remodel these areas with more detail than the rest of the terrain by placing additional objects like trees and buildings in the scene. The two areas are marked in the terrain from a top-down perspective as seen in figure 4.13, where the top left area is the parking area and the area in the bottom right is the main square.

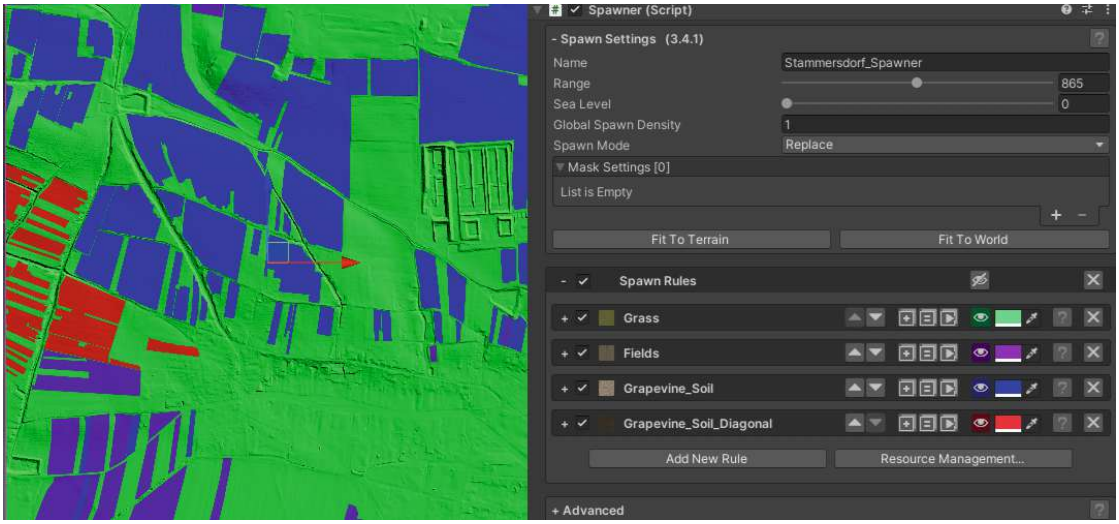


Figure 4.11: Settings used for the Spawner and overview of the Spawn rules. Each color to the right of the rule is the preview color used on the terrain and shows the area which will be textured after applying said rules.

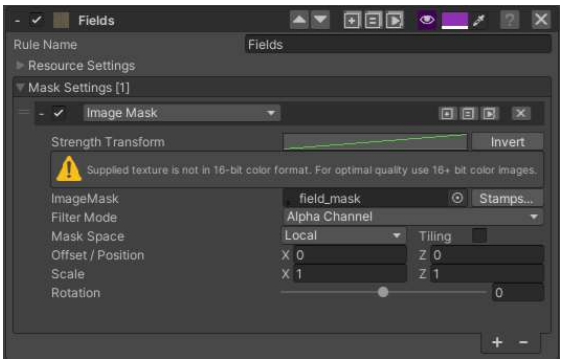


Figure 4.12: Settings used for the Image Mask for the fields rule

Additionally, the road network should be remodeled for the whole terrain, to allow travel between the two points of interest. The Road network was modeled via Procedural Worlds GeNa, which allows placing different kind of models either via splines or other methods of placement like brushes or procedural generation. One advantage of using GeNa is, that it automatically adheres to the terrain tiling of Gaia, meaning that if a street (or any other model placed via spline) crosses over the boundary of one terrain tile into another the resulting mesh is automatically split there, allowing to unload also only parts of a street related to a certain terrain when unloading terrain tiles instead of having to unload (or load) the whole street mesh. Another advantage is the function to "flatten" (or "raise") the terrain under the street — since the height map has "only" an accuracy of 1 meter, the streets intersected the ground in several places. This can be achieved by the GeNa Carve Extension, which is automatically available when a GeNa

Figure 4.13: Top-Down view of the textured terrain, with the Senderstraße parking lot marked in the top left and the main square of Stammersdorf in the bottom right

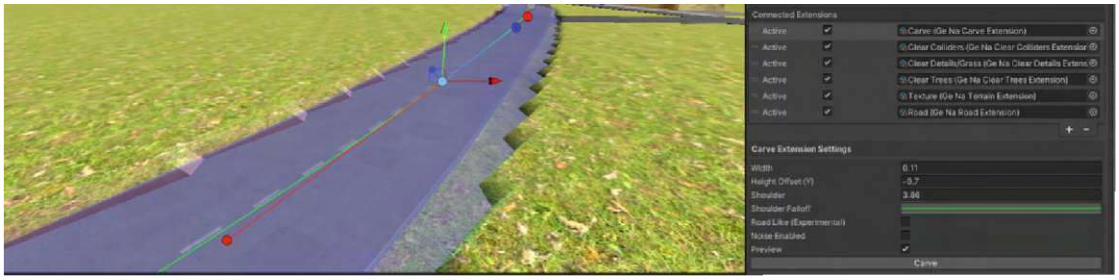


Figure 4.14: Example of flattening terrain under a street with the Gena Carve Extension

road is placed in the scene. The extension provides some parameters like width, shoulder and noise, where the shoulder depending on the "shoulder fall off curve" can provide a fall-off e.g., a slope, at the edges of the defined width and noise can provide some random raised areas to break symmetry and add more uniqueness to the pathway. An example of the Geva Carve Extension in preview mode can be seen in figure4.14.

To adapt the visualization of the street to the application's needs, the GeNa Road Extension has to be used, as seen in figure 4.15. The road extension has settings like the width, the street having a certain layer or a collider and the so-called "Road Profile". A "Road Profile" is a GeNA specific scriptable object having settings for shader and materials and textures.

It is advised to create a separate "Road Profile" for each type of road that should be used in the scene, to avoid accidental changes. In theory, it should be possible to prepare a GeNa Spawner with a sidewalk mesh, add a GeNa Spawner Extension with an offset and

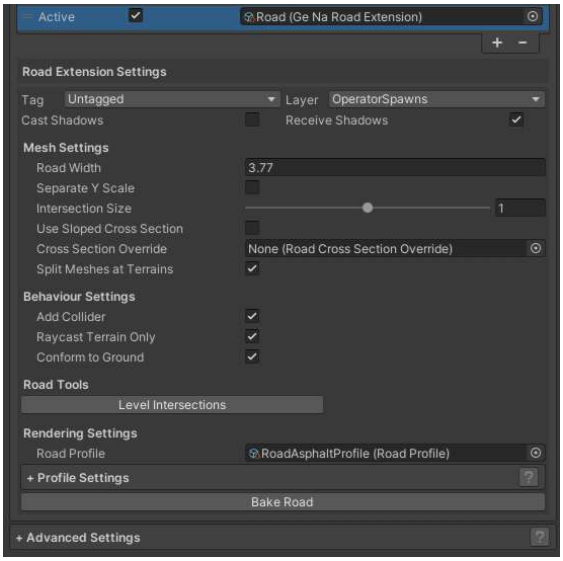


Figure 4.15: Overview of the Gena Road Extension settings

Figure 4.16: Example of possible mesh misalignment of road barriers introduced by the GeNa Spawner Extension. Image is taken from the official "GeNa Pro And Gaia Pro - Level Design Example" Youtube video [15].

spawn multiple sidewalk meshes automatically with an offset, for some unknown reason, I couldn't get this to work as no meshes spawned at all, even with example Spawners provided by GeNa. Nevertheless, while the Spawner approach may be automated to a degree, it leads to other disadvantages due to the fact that a sidewalk would then be built up from multiple smaller meshes, introducing the possibility of misalignment of these smaller meshes relatively to each other and bigger performance impact than a single longer sidewalk mesh. An example of such misalignment can be seen in figure 4.16. Thus, in the end, sidewalks were also placed manually.

The disadvantage of this approach is the need to manually place the streets and that each node of the spline has the same width, thus changing width in streets can not be handled accurately. As a workaround, I created multiple splines having a separate thickness each, to be able to model the road network more accurately, although still not perfectly accurate. Modeling the road network with multiple streets introduces a new problem. Road Creation in GeNa is handled in two steps. First, the splines are placed and adapted with the extensions and road profile as mentioned above. In a second step, a road needs to be "baked". In this step the spline previews are getting converted to a real mesh, where all settings like the collider and layer but also the split of the mesh along terrain tile borders are getting applied to the final mesh. Out of the box, GeNa only allows baking a single road via a "Bake Road" button in the GeNa Road Extensions. For Stammersdorf the road network was modelled with 44 different Splines, baking each Road manually would be a tedious task, thus I created a helper editor script which allows the baking of multiple roads at once. The script gets the spline component of each selected gameobject in the editor and checks if the spline component is active and enabled, then the road extension of each such spline is gotten and if available the Bake method provided by GenaRoadExtension objects is called.

There are also external Unity plugins allowing to import road networks from Open Street Map exported streets. Sadly, the roads of open street map did not accurately match the GIS maps provided by the city of Vienna at all, thus not being a suitable alternative for modelling the road network.

The trees were placed via a GeNa brush with random size, rotation, and color modifier so that each tree looks slightly different even though the same model was used. In the current VE are 138 trees placed for the forest and the parking area and 50 additional trees for the main square area. The tree model used is s SpeedTree model with 672 triangles with the default SpeedTree shader used.

The buildings and the contents of the building were provided and placed by Bytewood e.U[5]. In some spaces the terrain clipped through buildings, which are accessible for trainees, making the terrain visible inside the buildings, as the terrain data provided by the city of Vienna was not "flattened" under buildings. To flatten the terrains under the building, an editor script created by the user "ZeroCool" and adapted by the GitHub user "KurtDekker" was further adapted by me and used on trainee accessible buildings [16, 17]. In the version of "KurtDekker" the script was searching for any terrain object in the scene if none was provided via drag and drop. Since in the Stammersdorf scenstio there are multiple terrain tiles and therefore multiple terrains this approach led to no usable results, thus, I rewrote the script to fire a ray cast either below or above the game object to which the script is attached to check for terrains. If a terrain is found it is set as terrain to be used for flattening and a debug message showing the name of the found terrain and the information that the height will be applied after pressing the button again is shown. The reason for this two-step implementation is, that some accessible buildings

[5] https://bytewood.com/

are actually covering multiple terrains, because they are placed at the borders of two tiles being placed next to each other. This step allows checking if the found terrain is the expected terrain to be flattened and in case it is, now allows dragging and dropping the correct one in the terrain field to be used before continuing with the flattening step.

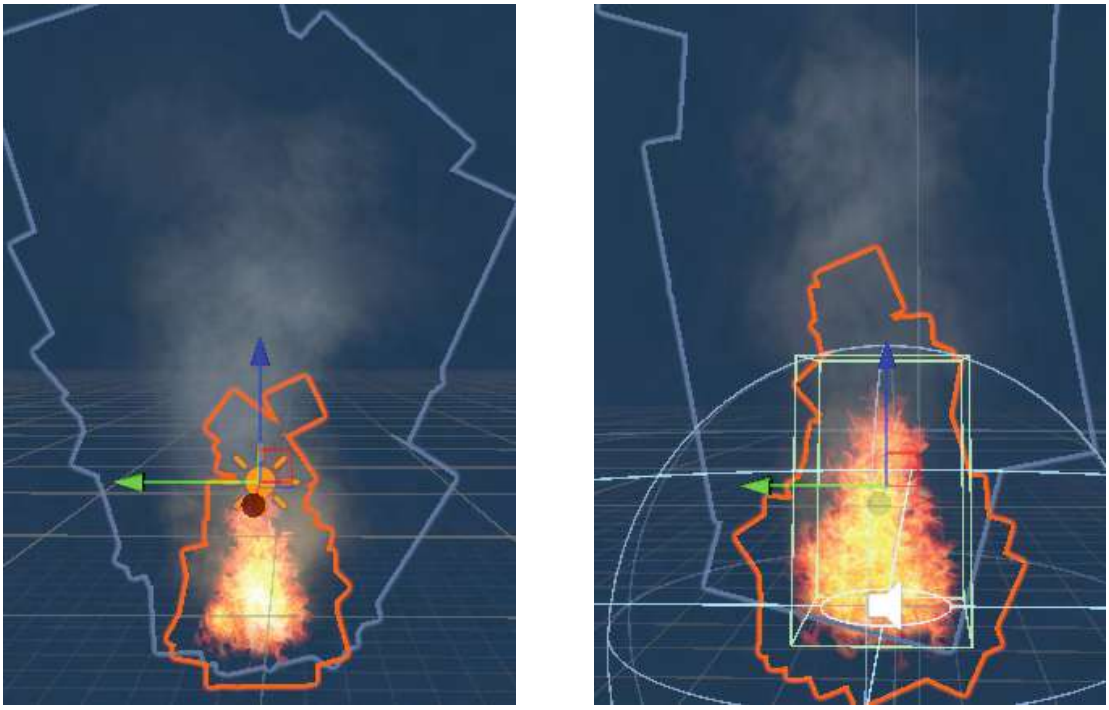
4.2 Fire and Smoke

Rendering realistic simulated fire and smoke is computationally expensive. As mentioned in the state-of-the-art chapter 2, the most realistic behavior at present can be achieved by using computational fluid dynamics combined with volume rendering as implemented by e.g. Lu et al. [33] or Lorusso et al. [32]. One problem with such implementations is the lack of interactivity during run time, since CFDs have to be pre-computed to be usable for real time simulations. Grabowski tried to improve this by preparing multiple simulations, which were then blended accordingly to reflect user actions [28]. Still, all these implementations were designed for VR headsets in combination with a PC, not a stand-alone mobile VR headset, which performance wise is more comparable to smartphones than computers. This combined with the fact, that smoke should be visible on a distance for wide area simulation and thus can't be culled to reduce the performance impact particle effects were chosen for the implementation of smoke and fire.

As a base a fire variant already prepared for mobile hardware from The "Ultra Realistic Fire and Smoke"[6] Unity Package was used and adapted. The first step was to reduce the number of non-essential particle emitters. Since each and every tree in the scene should be able to burn, and there are 138 trees in the parking lot area and 50 trees in the main square, reducing the number of emitters and particles is essential, to have a performant visualization of fire and smoke. The mobile fire was made up of three different emitters for fire and one additional emitter for smoke. One emitter was handling the "main fire", one as a slightly different colored center fire, and one was creating sparks. As the center fire emitter and the spark emitter only added details to the main fire, they were completely removed, thus reducing the worst case emitter count by 376 possible emitters. Additionally, the maximum allowed particles per emitter were reduced from 1000 to 30 for the fire and reduced to 5 for the smoke. It should be noted, that the mobile fire of the unity package also never surpassed 30 particles for the main emitter in my tests, as particle lifetime was configured, so that particles usually died before reaching more than 30 particles at a single time for the main emitter. All three fire emitters combined reach a maximum particle amount between 130 and 140, while the smoke capped out at around 50 particles but sometimes hit 51, resulting in around 190 particles worst case per fire in the original mobile fire prefab. Since fewer particles also mean that e.g., less smoke is rendered, I balanced this out by increasing the particle size of each smoke particle and decreasing the lifetime of particles to still give the illusion of a continuous stream of smoke being present all the time while the fire is burning. The

[6] https://assetstore.unity.com/packages/vfx/particles/fire-explosions/ultra-realistic-fire-and-smoke-13549

second step was to remove the light source attached to the fire. While an additional light source adds realism, it also adds a burden to the performance, let alone 188 additional light sources in the worst case. A visual comparison between the original fire and the adapted fire can be seen in figure 4.17.



(a) Original fire prefab for mobile from the "Ultra Realistic Fire and Smoke" Unity package

(b) Adapted fire with only one fire emitter and particle count for fire reduced to 30 and for smoke to 5

Figure 4.17

The spreading of the fire was handled by a simple scale-over-time algorithm, where the growth is not handled by increasing the amount and/or lifetime of particles, but by increasing the scale of the fire and smoke emitter. For scaling to work on particle emitters the Emitter Scaling Mode has to be set to Hierarchical, otherwise the scale values of the transform of the gameobject are ignored. Each fire starts to spread (grow) the moment it is set (either by the operator or already in the scene) up to a configurable maximum size. For the Stammersdorf scenario, I used a maximum size of 12 meter and the spread is set to 10 centimeters per second. To not only grow but spread to other objects, each fire has a collider (with the is trigger flag activated), which checks for collisions with other objects: If another objects has the “burnable” tag, the other object also starts to burn. An example of the fire spread and a burnt tree can be seen in 4.18 All this behavior is handled in a single script called Firebehavior. After some time, or when the fire gets extinguished by the operator the model of the tree should change from the normal tree to a burned down tree, this is handled in a “BurnDown” method in a Treebehavior script,

Figure 4.18: In-application screenshot from operator perspective of spreading fire and a burnt tree.

which gets called by Firebehavior.

In case an object would be hit by multiple growing fires, it would start to burn multiple times as seen in figure 4.19a with a separate fire each time. In turn, the new fires would also spawn fires to the “original” burning trees, which in turn can again spawn new fires on the other tree, leading to an infinite fire spawn loop as visualized in figure 4.19b. This is unwanted behavior, as the impact of multiple fires per object on performance would be immense compared to the arguably negligible use for trainees. To avoid this additional to a Firebehavior script a FireController script is introduced, which has a ConcurrentDictionary in which each burning gameobject will be stored as value, with the unique InstanceId available from the Unity Object class as key. Addtionally, the instanceId is set to the FireSpread, so each fire knows which tree it belongs to and can trigger the "burnDown" on the correct tree. If the fire was parented to the tree, this could have been solved easier via a GetComponentInParents call, but for some reason when trying this approach a major mismatch between the fire positions of the client and the server occurred. The FireController offers some convenience methods to check if a specific tree is (or was) already burning and to get or add burning Trees to the controller. It should be noted, that the InstanceId is not persisted between different executions of scenes, meaning that a tree may have a different InstanceId between two sessions. For my implementation, this does not lead to problems, but should be considered for implementations relying on the need of persisted Unitys InstanceIds.

So, while in this scene the fire spread is only set for trees, as only trees have a burnable tag added, the fire spread should also work with other objects with slight adaptions of the script by taking the size of the burning objects in account to define a more accurate individual maximum spread size.

(a) Exemplary visualization of one tree getting ignited twice by different fires

(b) Exemplary visualization of infinite fire loop

Fire and Smoke Integration in VROnSite

Since VROnSite is a distributed system with clients and a server, some additional measures need to be taken for the implementation. The spawning, the deletion, and the size of fire should be handled by the operator to avoid inconsistencies and to keep the state of the application in a single place, to also allow for the connection of clients after the application started or in case of possible re-connects. Thus, the FireSpread scripts checks if the ClientManagerInstance provided by VROnSite is an operator and only then executes the logic. A new helper function was introduced in VROnSites ClientManager called SpawnNetworkFirePrefabServerRpc which additionally handles the setting of the clientInstanceId of the burning object the fire belongs to in the FireSpread component of the newly spawned fire. A NetworkTransform was added to the fire prefab and set, so that the scale of the object gets distributed from the operator to the clients.

Changing a model in the RendererComponent is not distributed via standard Unity networking, thus I had to implement a workaround. Instead of changing the model, the normal tree model gets despawned and the burnt tree model then gets spawned instead. Both spawning and despawning are done via a helper function provided by VROnSites ClientManager, which not only destroys or spawns the network object on all clients and the server but also handles the necessary steps to record the despawn or spawn for the AfterActionReview functionality. Before an object is spawned over the network, it is instantiated via the standard Unity prefab instantiate method. While the standard instantiate allows for setting a position and a rotation, it is not possible to set a scale. As mentioned before, each tree has a different scale, this scale needs to be taken over to the burnt tree, to make sure that the different sizes of the trees are still accounted for even if they are burnt down. For this reason, the helper method SpawnNetworkPrefabServerRpc of the ClientManager of VROnSite had to be adapted. The method was enhanced with an optional parameter Vector3 scale=default. By assigning a default value, the other invocations in the existing VROnSite code could still compile without throwing an error. Additionally, the logic was adapted to set the scale, if it is not the default value. By

setting the scale after instantiating the object but before triggering the network spawn, the scale values are also reflected on the spawned objects on client side. In figure 4.20 an example image can be seen of a relatively progressed fire already integrated in the VROnSite application.

Figure 4.20: Screenshot from VROnSite of forest fire with multiple ignited and burnt trees.

When testing the functionality in the AfterActionReview I found that VROnSite can't handle multiple events happening in the same frame. The reason for this is, that the last entry in a frame, simply overwrites the one before it, making it impossible to store multiple events per frame as of now. To avoid this issue, I added a little delay to make sure that all spawns and despawns happen in different frames. In fact, this means that there is a delay before spawning the new burnt tree and before despawning the old normal tree. The reason that a delay is needed also before spawning a burnt tree is that the change may also have been triggered by the operator extinguishing a fire. Extinguishing a fire also triggers a distributed despawning of the fire and registers in the AfterActionReview. Thus, to make sure that the despawning of fire is not getting overwritten by the spawning of the tree, a delay was added here as well.

CHAPTER 5

Evaluation

The evaluation chapter will be divided into two sections. The first section will be an explanation of the setup for the user study setup, including the evaluation. The second section is the performance evaluation of different settings on the mobile VR hardware and an explanation which approach was used for measuring the performance.

5.1 User Study Evaluation

To answer the first research question, Q1 "How well do expert users assess and accept first responder training of wide area disaster events on a mobile stand-alone VR system ?" I conducted a user study with a total of eight people who are active members in a voluntary fire brigade. The user study was conducted in the voluntary fire brigade of Langelebarn, Lower Austria. The setup consisted of one Notebook, which ran the server and was handled by an expert operator, and four Meta Quest 3 prepared on a swivel chair to be used by the expert users for the study. With the swivel chair, it was possible to turn around for the participants and facing the same direction they are moving in VR, without needing to stand for the whole test. The VR part was split in two separate scenarios: First, the trainees were introduced to the Quest 3 and the controls in a simple scenario, where they got an explanation on how to move and how to interact with objects. This introduction took around 10 to 15 minutes per group. After the warm-up, the second scenario containing the VE of Stammersdorf was loaded, where the spreading fire with visible smoke was prepared. In this scenario, the expert users were informed of smoke rising from the forest near the parking lot of Stammersdorf and to react to the situation accordingly. One of the users in VR was asked to take the role of the leader, handling coordination with the control center and giving audible commands to the other three VR participants. A car was placed in the scene, which was also driveable by the participants. Additionally, the leader had the ability to contact a control center — which was played by the system operator — to call for back up and other information. The

scenario was played out in such a way, that the users also have the opportunity to see the fire from a distance and not only up close. The scenario overall took between 30 and 45 minutes per group.

An image taken during the study can be seen in figure 5.1

Figure 5.1: Image taken during the study, showing the study setup with four participants on the swivel chairs and the operator in the back.

The setup of the user study was as follows:

- Pre-Training Questionnaire: A questionnaire before the users participate in the VR Training to get an overview of the age, gender, current physical wellness with the classic simulator sickness questionnaire [39] of the users and some general questions about their experience with training scenarios and technical affinity.
- Warm-Up: Giving a status report of the situation that will be encountered in the scenario and explaining the basic controls in a basic training scenario to get people accustomed to the VR experience.
- Playing through the scenario in VR in a group of four participants and an experienced operator, which was not part of the study participants, to trigger events like starting the forest fire in the VE. This was done twice with four different participants, each time.
- Post-Training Questionnaire: Asking for the well-being of the user, to see if any kind of cybersickness was introduced because of the training, asking questions about their experience with the terrain, fire, smoke and general questions regarding the usability of the system with a system usability scale questionnaire [23].

Participants had the possibility to ask questions if one or more of the questions were unclear, but none asked for clarification, thus it should be fair to assume, that the questions were generally understandable. All questions, besides the well-being questions, were answered on a typical 5 point Likert scale from not agreeing at all to agreeing strongly. Additionally, users had the possibility to provide additional information in three open questions should they wish to add something. It should be noted that the questionnaire originally was conducted in German but is translated to English for the sake of consistency in the thesis. The German questionnaires used for the study can be found in the appendix.

Pre-Training Questionnaire Results

All eight of the participants were male and seven of them between the age of 23 and 49, with a median value of 38, the eighth user either forgot or refused to fill out the age. Since all the participants identified as male, the pronoun "he" will be used where appropriate in the result chapter. Seven of the users were active in a leadership position, and six of those seven are also active in an educational function. Two participants had no role in education, one of which also had no leadership role. Six participants regularly have trainings prepared for them. Two participants are having one to three trainings prepared for them per year, two four to seven trainings, two more than seven trainings, leaving two participants for which no trainings are regularly prepared. Five of the participants also prepare trainings for other people in leadership roles, two prepare more than seven, one between four and seven and two participants prepare between one and three training for leadership positions per year.

Six of the participants also regularly take part in simulation exercises, two participate in between one and three simulation exercises, three in between four and seven and one participates in more than seven per year. Most of the participates rate themselves as fit, one as very fit, five as moderately fit and two as averagely fit.

Overall, seven participants had prior experience with VR, although the experience level with VR was mixed. Four of the participants have an average experience level with VR, two users have little experience with VR and one user each had a high level of experience and non respectively. One of the attendees had no prior experience with Virtual Reality, two had slight experience, four average experience and one had a large amount of experience with Virtual Reality. While seven participants had experience with VR, only six participants had experience with a gamepad, although with an overall higher experience level compared to the VR experience results. Three of the participants stated a high level of experience with gamepads, two stated an average level of experience, and the last one a low level of experience.

In the cybersickness pre-evaluation, three of the participants felt absolutely healthy with no symptoms stated. Five of the participants were sweating, three slightly and two moderately. Additionally, one of the five participants also felt slightly tired. It should be

1. Please indicate how many training exercises are prepared for you on average per year.

None	1-3	4-7	More than 7
2 (25%)	2 (25%)	2 (25%)	2 (25%)

2. Please indicate how many training exercises you prepare on average per year for other persons in leadership roles.

None	1-3	4-7	More than 7
3 (37.5%)	2 (25%)	1 (12.5%)	2 (25%)

3. Please indicate how many simulation exercises you participate in on average per year.

None	1-3	4-7	More than 7
2 (25%)	2 (25%)	3 (37.5%)	1 (12.5%)

4. How would you rate your personal fitness level?

Not fit	Slightly fit	Averagely fit	Moderately fit	Very fit
		2 (25%)	5 (62.5%)	1 (12.5%)

5. Do you have prior experience with Virtual Reality?

None	Low	Average	Moderate	High
1 (12.5%)	2 (25%)	4 (50%)		1 (12.5%)

6. Do you have experience or prior practice in using a gamepad?

None	Low	Average	Moderate	High
2 (12.5%)	1 (25%)	2 (50%)		3 (12.5%)

Table 5.1: Results for pre study questionnaire

noted that on the day of the evaluation the temperate was up to 34 degrees Celsius and the room prepared for the training had no cooling measures installed.

Post-Training Questionnaire Results

According to the cybersickness post-evaluation, the amount of participants who were experience sweating were increasing from five to seven, with four slightly sweating and three moderately sweating. The one participant, who was slightly tired before the training, did experience no increase in fatigue and was still slightly tired after the training. No further symptoms were experienced by any of the participants. Again, it should be noted, that it was a hot day in Langenlebarn with a temperature of up to 34 degrees Celsius on the day of study. There was no air conditioning or other cooling devices in the room, leading to an overall hotter room as more people stayed in the room, which resulted in the increased sweating of some participants, which can also be seen when comparing the pre questionnaires of the first and the second training group. For the first training group, only one person experienced slight sweating before the training, while for the second training group all four of the participants experienced at least slight sweating.

General Questions

All the participants agree (six strongly), that the forest fire scenario would help them to prepare for forest fire operations, resulting in a mean value of $\mu = 4.75$ and a standard deviation of $\sigma = 0.433012702$. The expectations that the participants have regarding forest fires scenario were met for seven out of eight participants (six strongly) and one participant was neutral regarding his expectations, leading to a $\mu = 4.625$ and $\sigma = 0.695970545$. Again, all of the participants agreed (five strongly) with the statement, that the VR forest fire training could be a useful addition to traditional training methods ($\mu = 4.625$ and $\sigma = 0.484122918$).

Thus, the results strongly indicate that the implementation of forest fires may be a suitable and well received alternative to additional training methods, would they be used in training for professional firefighters.

Six of the participants did not (four) or only slightly (two) perceive any kind of delay or stuttering during the VR training. The other two, however, have perceived stuttering or delays, resulting in a $\mu = 2.25$ and $\sigma = 1.08972473$. The aforementioned results can also be seen in table 5.2. Interestingly, the two participants stating to notice stuttering or delays fairly, were using the same Meta Quest 3 in the first and second run respectively, which may be indicating some issue with the device itself. Combined with the fact that the results of the post cybersickness questionnaire did not indicate any worsening of the well-being of the participants, it should be safe to assume that the perceived stuttering or delays were not happening very frequently, as these usually lead an increase in dizziness or nauseousness.

1. The VR forest fire scenario would help me prepare for operations involving forest fires

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
			2 (25%)	6 (75%)

2. The VR forest fire scenario meets my expectations for virtual training on forest fires

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
		1 (12.5%)	1 (12.5%)	6 (75%)

3. I would find the VR forest fire training useful as an additional training method alongside traditional training methods

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
			3 (37.5%)	5 (62.5%)

4. I noticed unexpected stuttering or delays during the training

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
2 (25%)	4 (50%)		2 (25%)	

Table 5.2: Results for general questions about the training

Fire and Smoke Questions

The visualization of fire was perceived as suitable by seven participants, of which six strongly agreed, and as neutral by the last participant, resulting in μ= 4.625 and σ = 0.695970545. The fire was also perceivable from a distance according to all the participants (seven strongly agreed, one agreed) with a μ= 4.875 and σ = 0.330718914. Most participants also found the behavior of the fire suitable for the training (five strongly agreed, two agreed), with only one user finding it unsuitable (μ= 4.375 and σ = 0.992156742). This suggests that the fire implementation was overall accepted and deemed suitable by the expert firefighters participating in the study.

Even though the visualization was deemed suitable by seven participants as well, the suitability was overall rated slightly worse with four participants agreeing strongly and three agreeing (μ= 4.375 and σ = 0,695970545) compared to six strongly agreeing and one agreeing (μ= 4.625 and σ = 0.695970545) regarding suitability of the visualization for smoke and fire, respectively. A possible explanation for the difference in perceived suitability can be found in the open free text question, where three of the participants stated that the smoke should be black instead of white, before the fire is extinguished with water. Nevertheless, the smoke could be noticed as well from a distance as the fire according to the participants, with seven agreeing strongly with the statement regarding perceptibility of smoke from a distance and one agreeing (μ= 4.875 and σ = 0.330718914). The behavior of the smoke was also regarded as slightly worse than the behavior of fire when compared, while six participants stated that the fire was suitable (five agreed

strongly, one agreed), one user was neutral regarding the suitability of the smoke behavior and the last one even disagreed, although not strongly (μ= 4,25 and σ = 1,089724736).

During the implementation, the focus lied on being able to visualize fire and smoke without perceiving noticeable stutters, diminishing the experience and possibly leading to cybersickness symptoms. The justified critic regarding the behavior of smoke and in a lesser form also for fire, was to be expected and leaves room for future work to optimize and focus on the behavior specifically. Interestingly, the training was still perceived as useful for preparing for forest fires and generally met the expectations of the users (as seen in Table 5.2), which could possibly indicate a certain leniency or lowered expectation for realistic behavior of fire and smoke in VR forest fire training because of its novelty for the participants. Overall, the fire and smoke section was the one with the strongest fill rate for the open question. Six participants wrote comments, of which four indicated, that they liked the fire and smoke ("very realistic", "liked it very much", "strongly modeled") and three mentioned the issue with the color of the smoke. The aforementioned results can also be seen in table 5.3.

It should also be noted that the participant stating Disagree for the behavior of fire and smoke, asked questions before the study regarding behavior of fire and smoke, e.g., if the fire was incorporating material properties and weather in its behavior, or if smoke reacted on wind conditions. This was not implemented as part of this study and may have had an impact on his answers. Sadly, the participant did not fill out the open questions, making it hard to identify the problems perceived by this participant.

Terrain Questions

The level of familiarity with Stammersdorf of the participants was equally distributed, as four participants were familiar with the area of Stammersdorf (two strongly), and four not or only slightly familiar with Stammersdorf (two slightly). Seven of the participants believe (six strongly) that VEs can convey knowledge about real world locations, while one participant felt neutral about it. Most participants were confident in their ability to identify parts of Stammersdorf recreated in the VE, as seven of the participants agreed (five strongly) that they can identify the locations depicted in the training in the real world, while one was very certain (strongly disagreed) that he could not, leading to a μ= 4.125 and σ = 1.363589014. Identical answers were given by each participant for the question, if the participants think they could navigate better through the simulated area of Stammersdorf in the real world (μ= 4.125 and σ = 1.363589014).

If the results are grouped by familiarity with Stammersdorf, three of four participants who were not or only slightly familiar with Stammersdorf were rather confident in their ability to recognize the visualized parts of the VE in the real environment of Stammersdorf and as confident in their ability to navigate through Stammersdorf better than before the training. The two participants who were not at all familiar with Stammersdorf

1. The visualization of the fire was suitable for the training.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
		1(12.5%)	1 (12.5%)	6 (75%)

2. I could perceive the fire from a distance.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
			1 (12.5%)	7 (87.5%)

3. The behavior of the fire was suitable for the training.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
	1 (12.5%)		2 (37.5%)	5 (62.5%)

4. The visualization of the smoke was suitable for the training.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
		1 (12.5%)	3 (37.5%)	4 (50%)

5. I could also perceive the smoke from a distance.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
			1 (12.5%)	7 (87.5%)

6. The behavior of the smoke was suitable for the training.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
	1 (12.5%)	1 (12.5%)	1 (12.5%)	5 (62.5%)

Table 5.3: Results for questions focused on fire and smoke

agreed to both questions (one strongly) that they feel like they could recognize parts of Stammersdorf and even navigate better through Stammersdorf than before. One of the two participants who was not familiar with Stammersdorf felt "Neutral" about the improvement of his ability to recognize and navigate through the real Stammersdorf, while the other was confident (strongly agreed) that he can now recognize and navigate better through Stammersdorf.

The four participants who were familiar with Stammersdorf were for the most part also confident in their ability to recognize parts of the VE in the real world, as three of four strongly agreed with the statement that they could recognize parts of the VE in reality and gave the same answer regarding navigation. The last participant felt like he can not recognize parts of the VE at all (strongly disagree) in reality and also gave the same regarding the navigation statement. Sadly, the free text question was not filled out by said participant, leaving the meaning up to interpretation.

The visual representation of Stammersdorf was perceived as suitable by seven of the

participants (six strongly agree). The last participant was neutral regarding the suitability of the visualization, resulting in μ= 4,625 and $\sigma = 0.695970545$. Three of four participants who were familiar (two familiar, one strongly familiar) with Stammersdorf agreed strongly with the suitability of the visualization, while the last one, who was also strongly familiar with Stammersdorf, felt neutral about the visualization.

All participants agreed (seven strongly) that large areas are important for leadership trainings in forest fire scenarios (μ= 4.875 and $\sigma = 0.330718914$).

Also, seven of the participants agreed (five strongly) that the structure of the virtual environment has an influence on the decision-making of leaders in VR forest fire scenarios, while one was neutral regarding that statement (μ= 4.5 and $\sigma = 0.707106781$).

Overall, all participants agreed that the VE of Stammersdorf was suitable for the training, six even strongly leading to a μ= 4.75 and $\sigma = 0.433012702$. The aforementioned results can also be found in table 5.4.

Now, with taking a look at the other answers provided by the participant who strongly disagreed with being better in navigating in the real Stammersdorf after the training could be explained in two ways: One, the person knows Stammersdorf extremely well and is very confident in his local knowledge of the area that he felt like that the presented visualization was not detailed enough for the participant to learn anything new. Two, the person felt that the environment was quite lacking in its visualization and structure of Stammersdorf. While the second option is a valid conclusion, it should be noted that the participant regarded the visual representation of the environment as "Neutral" instead of disagreeing or even strongly disagreeing with the suitability of the visualization of the VE and even agreed regarding the overall suitability of the VE for the training. Still, the participant also stated that he thinks he can not recognize any part of the VE in the real environment, which rather supports explanation one.

Usability Questions

The usability questions were standard system usability scale question and will be evaluated following the standard system usability scale formula, where the sum of each average for every odd question is subtracted by the sum of the average for each even question, then increased by 20 and multiplied by 2.5 [23].

$$score = 2.5(20 + \sum(Q1, Q3, Q5, Q7, Q9) - \sum(Q2, Q4, Q6, Q8, Q10))$$

Overall, the answers were rather positive with seven participants, who would use the VR system for forest fires frequently (five strongly agree) and none of the participants finding the system unnecessarily complex (all strongly disagree). Seven participants regarded the system easy to use and easy to learn for most people (six strongly agreed

1. I was already familiar with the area of Stammersdorf, Vienna, before the warm-up.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
2 (25%)	2 (25%)		2 (25%)	2 (25%)

2. I believe that virtual environments can convey knowledge of real-world locations.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
		1 (12.5%)	1 (12.5%)	6 (87.5%)

3. I think I can recognize parts of the virtual environment (Stammersdorf) in reality.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1 (12.5%)		1 (12.5%)	1 (12.5%)	5 (62.5%)

4. I think I can now navigate the area of Stammersdorf shown in the training better in the real world.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1 (12.5%)		1 (12.5%)	1 (12.5%)	5 (62.5%)

5. The visual representation of the virtual environment was suitable for the training.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
		1 (12.5%)	1 (12.5%)	6 (87.5%)

6. I think the simulation of large areas is important for leadership training in forest fire scenarios.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
			1 (12.5%)	7 (87.5%)

7. I think the structure of the virtual environment influences the decision-making of leaders in VR forest fire scenarios.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
		1 (12.5%)	2 (25%)	5 (62.5%)

8. I think the virtual environment was suitable for the training.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
			2 (25%)	6 (75%)

Table 5.4: Results for questions focused on the terrain of Stammersdorf

for both statements), but still only five think could use the system without support of a technical person, two would need support and one felt neutral regarding that statement. Interestingly, even though two participants felt like they needed technical support, all participants stated that they did not need to learn a lot to use the system. This could imply, that the technical support person may actually be needed for set up purposes like when putting on the VR headset or getting the controllers while having the headset already on instead of interacting with the VR application for the training. Nevertheless, all user felt confident in their use of the VR system for forest fires (six agreed strongly).

Seven users strongly disagreed with the statement regarding the cumbersomeness of the application, while one strongly agreed. The free text questions, sadly, give no further insight regarding that statement for said participant. As the participant deemed the system easy to learn for other people (strongly agreed) and also felt very confident in his usage of the system (strongly agreed), it should be fair to assume, that this simply may have been a mistake.

The numerical evaluation with the answers provided by the participants (as seen in Table 5.5) leads to:

$$88.125 = 2.5(20 + \sum(4.5, 4.625, 4.625, 4.75, 4.75) - \sum(1, 2.625, 1.75, 1.5, 1.125))$$

The SUS formula results in an above average score of 88.125, which implies that the VR Training application was perceived as rather usable by the participants. According to research of Bangor et al., who tried to map SUS scores to adjectives, a score above 85.5 can be interpreted as excellent [20].

5.2 Performance Evaluation

The Unity Profile, an out of the box available tool inside of Unity, gives a lot of insight in overall performance of an application. Sadly, the output is limited to the last 300 to 2000 frames, depending on configuration. If we assume an average frame rate of 120 fps, this would only be around 16 seconds of material. It is possible to save the complete profiler data in its own file by activating the binary log via script and defining a file path:

```
UnityEngine.Profiling.Profiler.enabled = true;
UnityEngine.Profiling.Profiler.enableBinaryLog = true;
string filepath = Application.persistentDataPath + "/"
+ _fileName + SystemInfo.deviceModel + ".data";
UnityEngine.Profiling.Profiler.logFile = filepath;
```

1. I think that I would like to use this VR system for forest fires frequently.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
		1 (12.5%)	2 (25%)	5 (62.5%)

2. I found the VR system for forest fires unnecessarily complex.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
8 (100%)				

3. I thought the VR system for forest fires was easy to use.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
		1 (12.5%)	1 (12.5%)	6 (75%)

4. I think that I would need the support of a technical person to be able to use this VR system for forest fires.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1 (12.5%)	4 (50%)	1 (12.5%)	1 (12.5%)	1 (12.5%)

5. I found the various functions in this VR system for forest fires were well integrated.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
		1 (12.5%)	1 (12.5%)	6 (75%)

6. I thought there was too much inconsistency in this VR system for forest fires.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
4 (50%)	2 (25%)	2 (25%)		

7. I would imagine that most people would learn to use this VR system for forest fires very quickly.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
			2 (25%)	6 (75%)

8. I found the VR system for forest fires very cumbersome to use.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
7 (87.5%)				1 (25%)

9. I felt very confident using the VR system for forest fires.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
			2 (25%)	6 (75%)

10. I needed to learn a lot of things before I could get going with this VR system for forest fires.

Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
7 (87.5%)	1 (12.5%)			

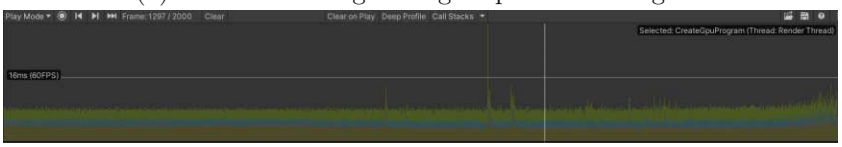
Table 5.5: Results for general questions about the training

This approach works independent of targeted platform, no matter if executed on Windows or Android. One problem is, that the resulting files need a lot of space, resulting in multiple gigabyte used per file. In theory this files can then be loaded in the Unity Profiler and analyzed in more detail, in practice the loading of larger Profiler files was not possible and resulted in Unity shutting down.

Benchmarking on the Quest bring additional difficulties for performance comparisons. One problem is that the performance of the Quest is not static, meaning that the same scenario can have different results depending on certain conditions like the temperature of the Quest. The Quest then either provides more or less hardware power through higher or lower clocks of the CPU and GPU, which is handled through so-called CPU or GPU levels. While Unity theoretically allows setting these levels per code, the values set weren't reflected on the Quest, suggesting that this functionality may not work correctly. Another problem is that the Quest can't run at uncapped frame rates but in the case but only in predefined caps. For the Quest 3 the allowed frame rate caps are 60fps, 72fps, 90fps and 120fps. This makes performance testing more difficult, as only scenarios and settings which are resulting in dropped frames are actually comparable, at least by frame rate alone.

The final approach to compare settings and their impact on performance was setting the Meta Quest to the highest FPS cap possible (120fps for the Quest 3) and running the test scenarios in the setup used for trainings with the application, meaning the performance tests were done in the Stammersdorf scene and the application running in client on the meta quest and server mode on a PC. The real setup provides a solid base load, making the performance impact visible faster than when an empty scene would be used, especially combined with the frame rate cap set to 120fps, when the application was initially designed to run with 72 fps, which is the default setting when creating a Quest 3 build with Unity. To change the frame rate cap, the Unity.XR.Oculus.Performance.TrySetDisplayRefreshRate(frame rate) can be used in a scrip, where frame rate is the targeted number of frames as a float. The test scenarios were designed to be short enough, so they can still be extracted from the Profiler set to 2000 frames, which in the case of Quest 3 for 120fps means that a test can take at most 16.67 seconds. To make the player move automatically, the Cinemachine package of Unity was used. This was achieved by creating a Cinemachine DollyTrack with a DollyCart. Dolly track and dolly cam are terms used in film making, where the dolly cam is a camera on rails, the dolly tracks. This behavior is replicated by Cinemachine, where the dolly track defines the "rails" the cart should follow the form of a curve made from two or more points. The dolly cart can then either be controlled by script or set to automatically move along the track. To function in a networked setting, it is necessary to add a Networkbehavior and NetworkTransform component to the dolly cart and make sure that the code executed in the Unity Update function of the CinemachineDollyCart script is only executed for the server. For this reason, I created a DollyCartVROnSite script which inherits from CinemachineDollyCart, where I simply copied the code from CinemachineDollyCart and added a condition that checks if the executing entity is the

(a) Profiler showing the right spike as the highest



(b) Profiler showing the left spike as the highest

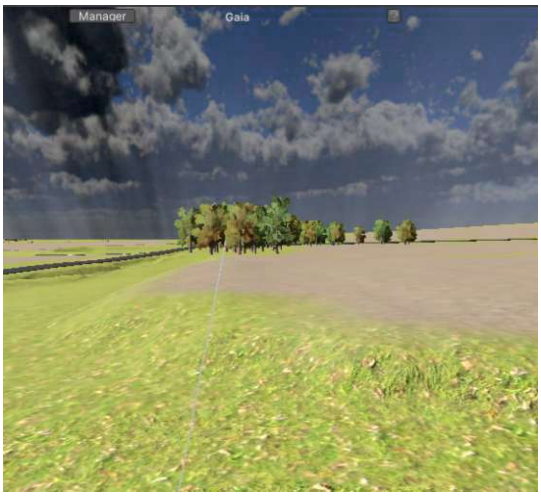
Figure 5.2: Profiler showing different (misleading) visualizations of spikes depending on width of window for the same data set

operator (server). Would the code also be executed on the client, the position of the dolly cart on the client would be constantly getting set back to the original position, thus resulting in the dolly cart not moving at all. To move the player, I added another script to the Trainee game object called Automatic Movement, which disables the normal trainee movement behavior and sets the position and rotation of the trainee game object to the position and rotation of the dolly cart in every frame. Since the server and client setup is used. An additional script was added to the cart, so that once the dolly cart reaches the end of the dolly track. At first, an automated process to trigger tests in the build was implemented, but the setup time for setting up the scene and networking among other things, did not consistently need the same time between tests, thus often resulting in frame spikes in the profiler, which were not part of the actual functionality which should be tested. To counter this, I decided to start the tests manually by pressing the Space Key after the profiler has stabilized. To make sure that the recorded frames of testing actually measure the same behavior and resulting in comparable performance tests, at the start and end of the test so-called Profile Marker are used. Profile Marker can be placed in the Unity Update function and then have an extra entry in the Profiler, allowing to make certain code identifiable in the profiler. As an example, how it was used in the first test cased: One profile marker is set when space is pressed to start the test (and the cart is not moving yet), while the other is set once the cart reached the end of the track. Thus giving a clear start and end for the relevant section of frames, which can then be used to make an appropriate comparison.

To actually compare different frame sets of the profiler the Unity Profiler Analyzer Package was used, which can compare two profiler data sets and returns some overall stats like median, average, min, and max frame rate. At this point I'd like to mention, that there seems to be a bug in the Profiler, which doesn't accurately visualize spikes depending on the width of the profiler window.

The first test scene was the player moving closer to the forest with 138 trees as used in the Stammersdorf training scenario from a distance of around 170 meters, once with

Tree LODs on and once with Tree LODs off. For the Tree LOD on setting, each tree had an LOD Group with 4 Different Level of Detail states: Full Quality with 672 triangles, Medium Quality with 560 Triangles, Low Quality with 220 Triangles and finally a single plane with a texture, called a bill board. The distance to the forest is set, so that initially the lowest LOD (bill board) is active and that all different LODs are activated when the cart moves closer.



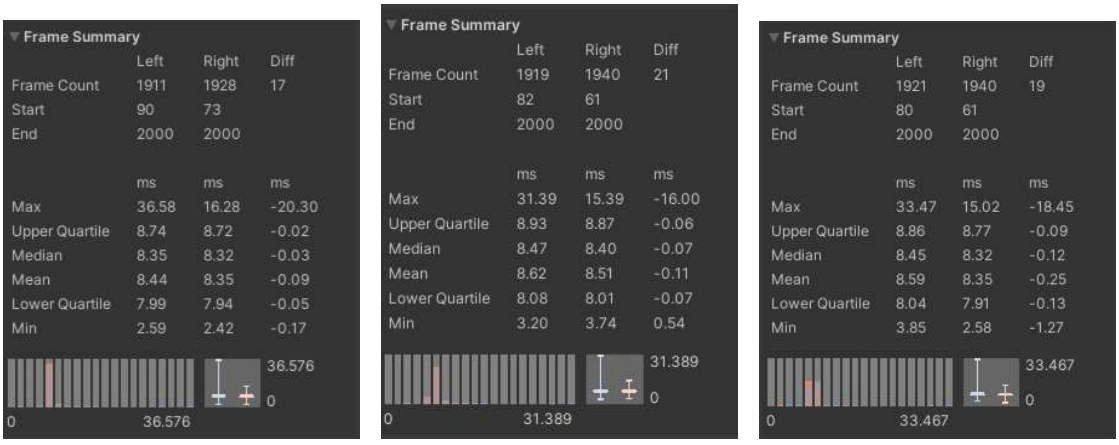
(a) Initial position in the LOD test.

(b) Final position in the LOD test activated.

The first LOD off Test was completed in 1923 frames, with an average frame time of 8.35 ms, while the first LOD on test was completed in 1911 frames, with an average frame time of 8.44 ms as seen in figure 5.4 and figure 5.5a. Completing the test with less recorded frames, actually means having worse performance, than completing the test in more frames, as the test was running for a fixed time and achieving to render more frames in a fixed time budget effectively means a higher frame rate.

Figure 5.4: Showing the relevant frame areas for LOD Test 1. The upper blue graph is for LOD on and the lower orange graph for LOD off.

The time budget for a single frame when aiming for 120 fps is 8.33 ms. Unexpectedly, the test with LODs off could keep closer to the frame rate target with an average of 119.76 frames, while the Test with LODs had an average frame rate of 118.34 frames per seconds. The test was repeated two additional times, and each time the LOD off run was faster, as seen in figure 5.5, where all the statistics of all three runs are posted. This indicates that the test scene geometry complexity was overall not very taxing on the Meta Quest 3. While interesting, that the overhead of activating LODs can introduce worse performance than having no LODs at all, a more interesting fact that can be seen



(a) First run (b) Second run (c) Third run

Figure 5.5: Summary of the three test runs. “Left” are the statistics for LOD on runs, while “Right” are the statistics for the “LOD” off run.

is that in all three test the “Max” stat (the highest amount of time a frame needed to be processed) is over twice as high.5.6. The biggest differentiating factor in favour of LOD off is Shader.CreateGPUProgram, which is only called when LODs are activated, while all others also appear for LOD off. “Create GPU Program” is called when a (new) shader is loaded onto the GPU, which is triggered because each LOD uses a shader variant. Nevertheless, rendering itself took around 0.23 ms longer in average when LODs were off, showing the positive impact on rendering performance when LODs are enabled in reduced rendering times as seen in figure 5.7. Additionally, in 5.4 some performance spikes can be seen for the LOD on graph in blue. The hierarchy view of the Profiler confirms that the Shader.CreateGPUProgram call was mostly responsible for the spikes, as can be seen in 5.8 for the left spike.

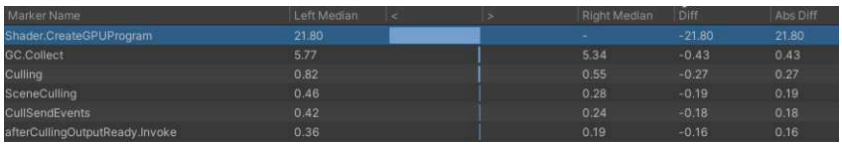
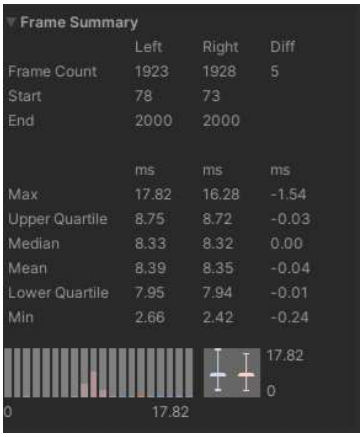


Figure 5.6: Overview of biggest performance differences in favour of LOD off

Figure 5.7: Overview of biggest performance differences in favour of LOD on

Unity allows to “pre-warm” shader, which enables that shader are not uploaded to the GPU in the frame they are first needed, but already once the first scene is loaded. The setting to activate preloading can be found in the graphics settings. When activated,

Figure 5.8: Hierarchy view of the first spike in the Unity Profiler

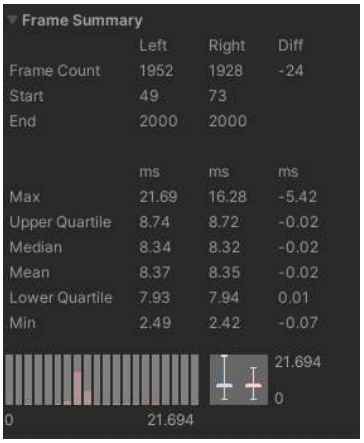


Frame Summary

	Left	Right	Diff
Frame Count	1923	1928	5
Start	78	73	
End	2000	2000	
	ms	ms	ms
Max	17.82	16.28	-1.54
Upper Quartile	8.75	8.72	-0.03
Median	8.33	8.32	0.00
Mean	8.39	8.35	-0.04
Lower Quartile	7.95	7.94	-0.01
Min	2.66	2.42	-0.24

17.82 0 0 17.82

(a) First LOD on run with pre-warmed shader

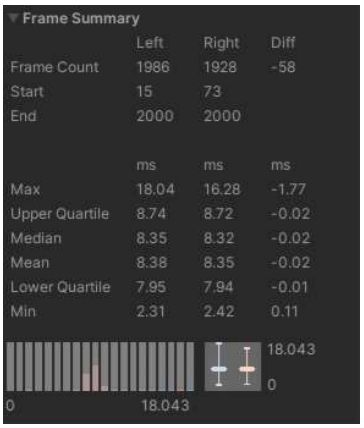


Frame Summary

	Left	Right	Diff
Frame Count	1952	1928	-24
Start	49	73	
End	2000	2000	
	ms	ms	ms
Max	21.69	16.28	-5.42
Upper Quartile	8.74	8.72	-0.02
Median	8.34	8.32	-0.02
Mean	8.37	8.35	-0.02
Lower Quartile	7.93	7.94	0.01
Min	2.49	2.42	-0.07

21.694 0 0 21.694

(b) Second LOD on run with pre-warmed shader



Frame Summary

	Left	Right	Diff
Frame Count	1986	1928	-58
Start	15	73	
End	2000	2000	
	ms	ms	ms
Max	18.04	16.28	-1.77
Upper Quartile	8.74	8.72	-0.02
Median	8.35	8.32	-0.02
Mean	8.38	8.35	-0.02
Lower Quartile	7.95	7.94	-0.01
Min	2.31	2.42	0.11

18.043 0 0 18.043

(c) Third LOD on run with pre-warmed shader

Figure 5.9: 'Left" is the LOD on run with pre-warmed shaders, “Right” is the first LOD off run

the performance gap between LOD on and LOD off is nearly closed, as can be seen in figure [**?**]. The differences in performance are comparable to the test before: culling was faster for LOD off runs, while drawing geometry was faster for LOD on runs. This poses the questions if Unity is using its culling code to handle the visibility of LODs, as these would mean more objects to handle compared to LOD off. Although, since the Unity Engine is not open source, this suspicion can't be confirmed.

In the second test was the player moving closer to 100 fire particle systems positioned in a ten by ten grid with a maximum particle count of thirty each, once with particles as billboards and once with particles as meshes with GPU instancing enabled. Smoke was disabled in both instances for this test. A screenshot of the 100 billboard particle systems can be seen in figure 5.10. The billboard test runs needed between 8.49 ms and 8.53ms resulting in around 117 fps and were between 3.31 and 3.71 milliseconds faster in average than the GPU instancing test runs with 11.84ms up to 12.20ms average frame time resulting in around 83fps, as can be seen in figure 5.11

The biggest difference stems from the garbage collection according to the Profiler Analyzer, as seen in figure 5.12. Although on closer inspection, the garbage collection was only triggered in a single frame in both runs, thus garbage collection may introduce spikes but can not be made responsible for the general difference in performance. The second-biggest difference is the PlayerLoop marker also doesn't give much insight, as the PlayerLoop is the overarching update loop on the main thread running all other Unity update functions. EarlyUpdate.XRUpdate and specifically OculusRuntime.WaitToBeginFrame, which is

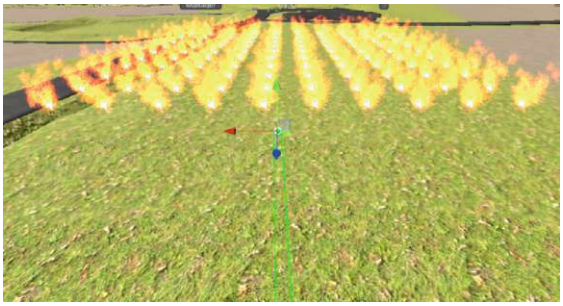
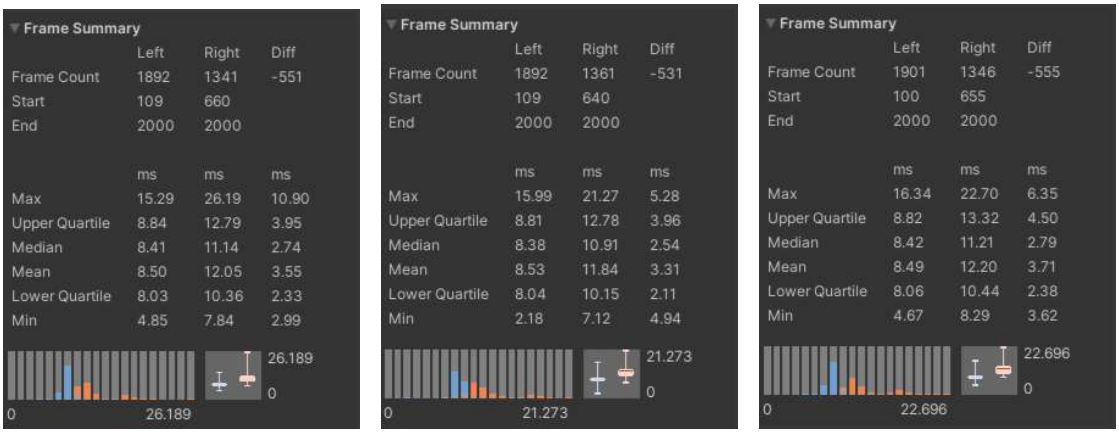


Figure 5.10: Fire particle systems with billboard fires in 10 by 10 grid



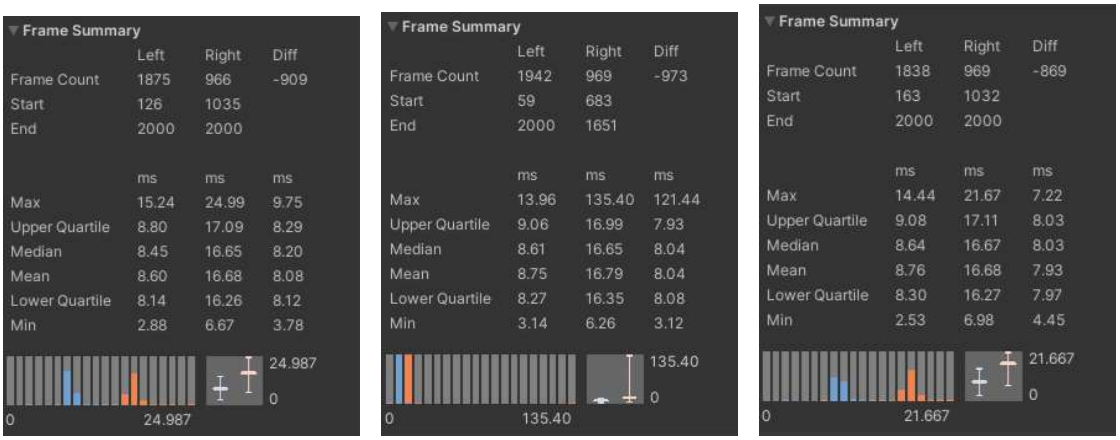
(a) First run (b) Second run (c) Third run

Figure 5.11: 'Left" fire particles are billboards, “Right” fire particles are meshes with GPU instancing

part of EarlyUpdate.XRUpdate, are of more relevance as they indicate that frames were either rendered “too fast” and need to be stalled to be not transmitted too early to the Quest 3 for the targeted frame rate, or frame times were missed and need to be stalled for transmission of the next frame. Since neither of the tests ran not at 120 fps it should be fair to assume, that frame target were missed more often than not, especially in the case of the GPU instanced test. Most of the other markers were in favor of billboard rendering, although slightly, as can be seen in the differences for Camera.Render (most other rendering related markers are part of Camera.Render) and managing the particle system, in figure 5.12. This suggests, that the differences although slight taken individually are summing up overall leading to missing the targeted frame time more often for GPU instanced particle systems when compared to billboard particle systems.

In the third test the impact of the maximum number of particles per particle system was tested with the same ten by ten particle system grid once using the adapted fire with only one emitter and a maximum of 30 particles per fire and once using to the original mobile fire, consisting of three separate emitters emitting between 130 and 140 particles.

Figure 5.12: Summary of interesting performance markers for the second test.



(a) First run (b) Second run (c) Third run

Figure 5.13: 'Left" is the adapted 30 particle fire, “Right” is the original 130 particle fire

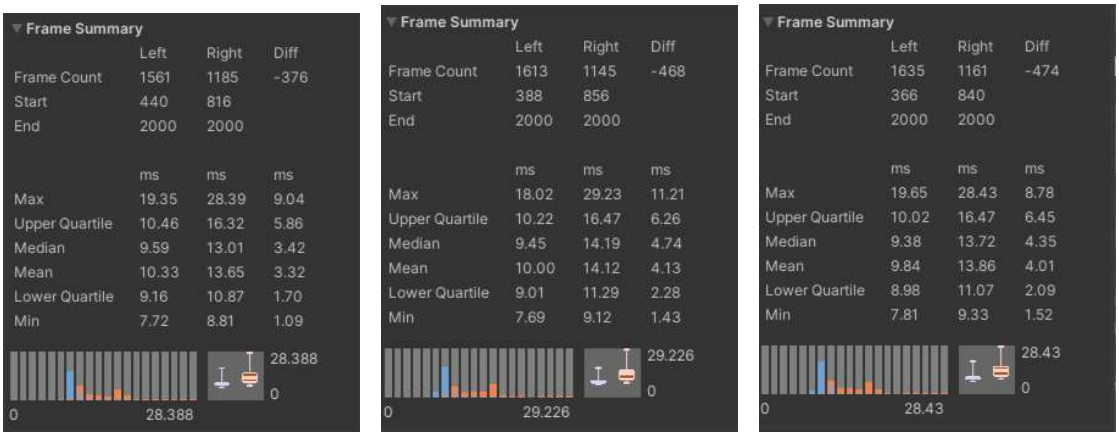
Figure 5.14: Example of “fire wiggler” script frame time

As before, the camera was starting around 170 meters away and smoke was disabled for both instances in this test. This results in around 130 to 140 particles less per fire, or 13000 to 14000 particles less over the whole grid. The average frame time of the adapted fire particle system was between 8.6 and 8.76 ms resulting in around 116 frames, while the average frame time of the original particle system was between 16.68ms and 16.79ms resulting in around 60 frames, as seen in figure 5.13. In figure 5.15 it can be seen that rendering was in average 1.58 ms faster, while updating the particle systems was around 0.42 ms faster. The original fire particle system had an additional “fire wiggler” script attached. This script randomized the emission rate, particle lifetime, size and speed of each fire, which also added up to 1.5 ms per frame, as seen in figure 5.14. Additionally, it can be seen that frames often had to be stalled because frame times were missed, adding another 4.68 ms in average.

In the fourth test, smoke was added to the adapted fire with fewer particles of the third test. Here, adapted smoke was tested using only five particles in comparison to the original 50 particles “mobile” smoke provided by the unity package. In practice that means 45 particles less per emitter, resulting in 4500 particles less over the ten by ten

Marker Name	Left Median	<	>	Right Media	Diff	Abs Diff
PlayerLoop	8.42			16.62	8.20	8.20
OculusRuntime.WaitToBeginFrame	0.11			4.79	4.68	4.68
EarlyUpdate.XRUpdate	0.41			5.07	4.67	4.67
Camera.Render	2.89			4.47	1.58	1.58
PostLateUpdate.FinishFrameRendering	2.97			4.55	1.58	1.58
Update.ScriptRunBehaviourUpdate	1.02			2.54	1.52	1.52
BehaviourUpdate	1.02			2.54	1.52	1.52
Assembly-CSharp.dll!::fire_wiggle.Update() [Invoke]	-			1.25	1.25	1.25
Drawing	1.83			3.06	1.24	1.24
Render.TransparentGeometry	0.72			1.85	1.13	1.13
RenderForwardAlpha.Render	0.65			1.76	1.11	1.11
RenderForward.RenderLoopJob	1.30			2.32	1.02	1.02
BatchRenderer.Flush	0.56			1.08	0.52	0.52
PreLateUpdate.ParticleSystemBeginUpdateAll	0.19			0.62	0.42	0.42
ParticleSystem.Update	0.19			0.61	0.42	0.42

Figure 5.15: Summary of performance markers for the third test.



(a) First run (b) Second run (c) Third run

Figure 5.16: 'Left" is the adapted 5 particle smoke, “Right” is the original 50 particle smoke

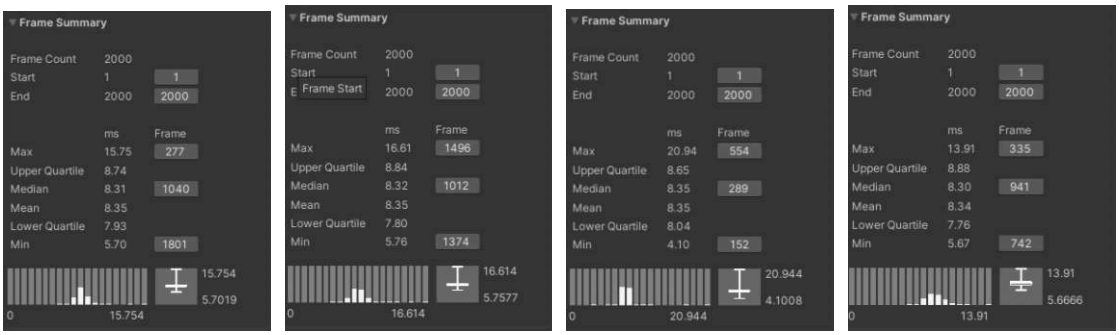
grid. The adapted fire and smoke test ran with an average of 9.38 to 9.59 ms, which is around 106 fps, while the adapted fire with original smoke ran with an average frame time of 13.75 to 14.12 ms, which is around 71 fps, as seen in figure 5.16. The biggest notable difference can be found in Camera.render with 0.57 ms and ParticleSystem.Update with 0.18ms, as seen in figure. In average 2.08ms were stalled, because of missed frame times.

An additional observation made was, that in all particle tests two, three, and four it could be seen that getting closer to the grid of the particle system lead to consistently lower frame rate. Some possible explanations could be that higher resolution textures were used when getting closer to the fires, or that the so-called overdraw is at fault here. This observation was not closer investigated in this thesis.

In the fifth test, the influence of terrain streaming and the number of terrain tiles was measured. Here, the Stammersdorf terrain was loaded in different setups: as full terrain with one single terrain tile with the size of 1730 meter by 1730 meter, as 4 by 4 terrain

Marker Name	Left Median	<	>	Right Median	Diff
PlayerLoop	9.56			12.42	2.86
OculusRuntime.WaitToBeginFrame	0.08			2.19	2.11
EarlyUpdate.XRUpdate	0.38			2.46	2.08
PostLateUpdate.FinishFrameRendering	3.55			4.13	0.58
Camera.Render	3.48			4.05	0.57
Drawing	2.29			2.67	0.37
RenderForward.RenderLoopJob	1.63			1.89	0.25
Render.TransparentGeometry	1.19			1.42	0.23
RenderForwardAlpha.Render	1.10			1.32	0.22
WaitForJobGroupID	0.90			1.08	0.18
PreLateUpdate.ParticleSystemBeginUpdateAll	0.33			0.51	0.18
ParticleSystem.Update	0.33			0.51	0.18

Figure 5.17: Summary of performance markers for Test 4.



(a) Single terrain tile (b) 4 by 4 terrain tiles (c) 8 by 8 terrain tiles (d) 16 by 16 terrain tiles

Figure 5.18: Performance of terrain tile configurations without any additional spawned objects

tiles with a size of 432.5 meter by 432.5 meter, as 8 by 8 terrain tiles with a size of 216.25 by 216.25 meter and as 16 by 16 terrain tiles with a size of 108.125 meter by 108.125 meter. For each test, the terrain loading was set to follow the transform of the player object and load terrains with a bounding box of 200 meter centered on the player. The player was positioned in the center of the scene, to load the maximum possible number of terrain tiles within the bounding box restriction. After the scene was fully loaded, multiple rows with 400 trees were spawned row by row up to a maximum of 16 rows, beginning on the row of tiles farthest away from the player in its initial view direction.

The full terrain loaded as a single terrain tile before spawning any additional objects had an average frame time of 8.35 ms, resulting in an average frame rate of 119,8 fps, staying fairly close to the 120 fps cap. None of the other terrain configurations had noteworthy differences in performance, with an average of 8.35 ms for four by four and eight by eight tiles and an average of 8.34 ms for 16 by 16 tiles. While the max and min frame time show larger gaps, they were not reliably different from each other on further tests.

For the full terrain the average frame time was 8.33 ms or around 120fps for spawning up to two rows (800 trees) and dropped to an average of 17.28ms or around 57 fps after having spawned 3 rows (1200 trees) as seen in figure 5.19.

After spawning an additional row (1600 trees) the frame time increased further to an

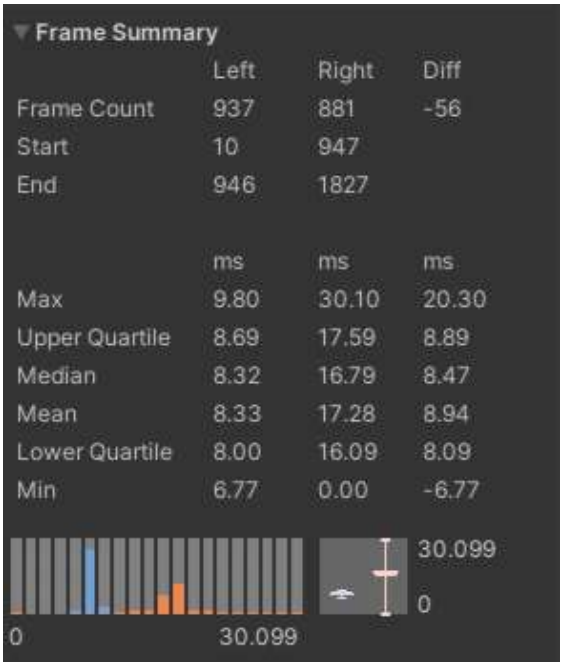


Figure 5.19: Full Terrain: “Left” performance for up to three rows. “Right” performance for four rows

average of 31.44 ms or a frame rate of around 32 fps, as seen in figure 5.20. Note that the listed start and end range are not completely identical between figure 5.19 “Right” and 5.20 "Left’, the reason is that the range of frames can not be explicitly defined but have to be selected by dragging a rectangle with a mouse, which proves to be not the most accurate of selection methods and sometimes even doesn’t allow the selection of a single frame but only a range of frames as start or end.

For the four by four terrain, the frame time dropped to around an average of 15.88ms or around 62 frames after spawning six rows (2400 trees) and to an average of 24.92ms or around 40 frames after seven rows (2800 trees), as seen in figure 5.21a and figure 5.21b respectively. Meaning that twice as many objects could be spawned before the frame rate was affected negatively compared to the full terrain, where no terrain streaming was used. Additionally, the spawn of the seventh row dropped to “only” 40 fps compared to the fourth row spawn for the full terrain, which dropped to 30 fps.

The results for all the tiled terrains were nearly identical again, independent of the configuration for the amount of tiles used (4x4,8x8,16x16) all had the same impact on frame rate in this test case. Six rows of trees (2400 trees) reduced the frame rate to around 60 fps and 7 rows of trees (2800 trees) to around 40 fps.

The Profile Analyzer showed similar behavior between all frame rate jumps: Camera.Drawing increased between frame rate jumps and a lot of frames were missed and had to be stalled. A more interesting finding was on the render thread: All terrain

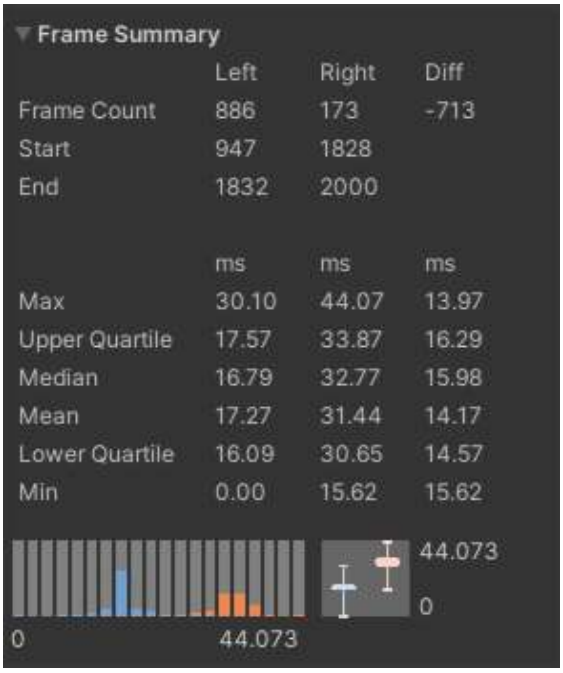
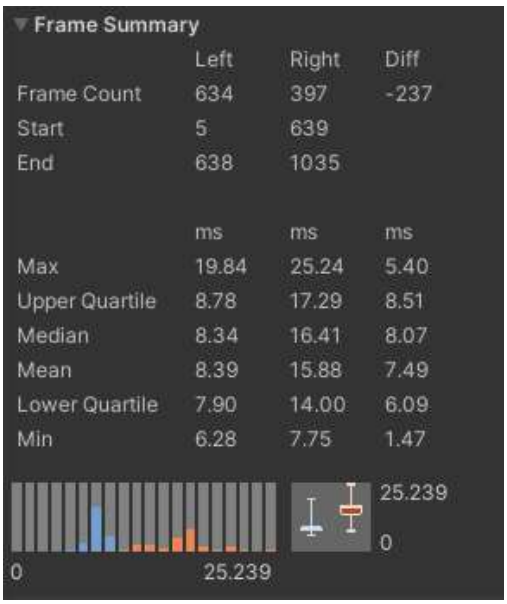
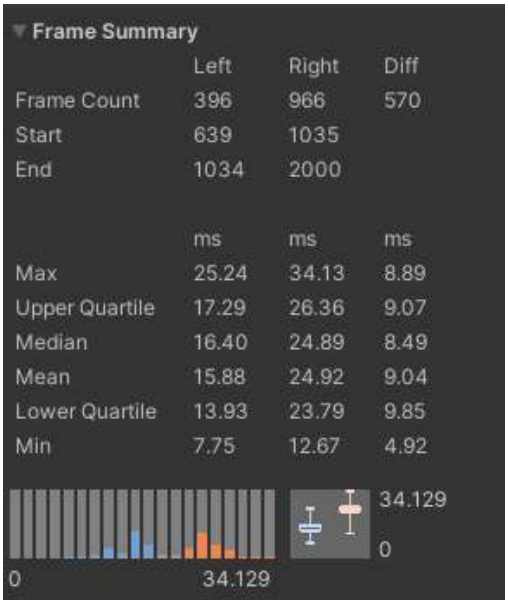


Figure 5.20: Full Terrain: "Left" performance for four rows. "Right" performance for five rows

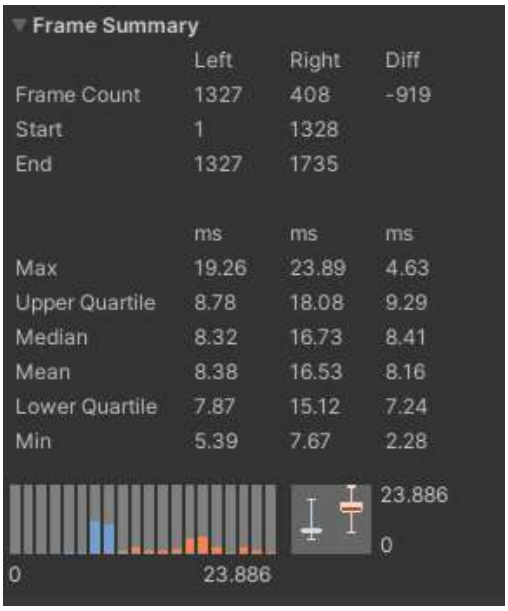


(a) 4 by 4: "Left" performance for up to five rows. "Right" performance for six rows

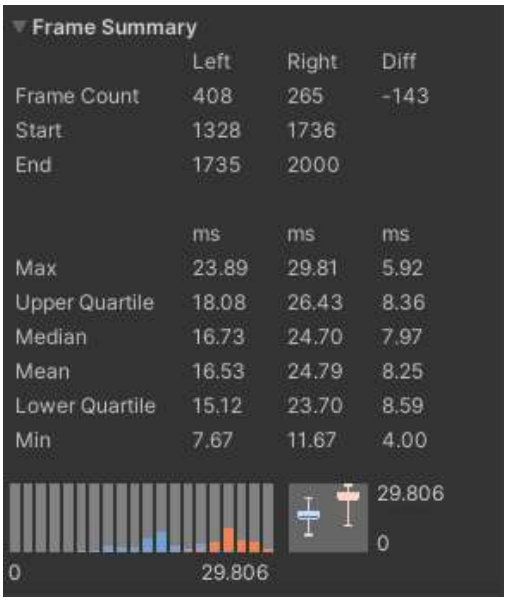


(b) 4 by 4: "Left" performance for six rows. "Right" performance for seven rows

Figure 5.21

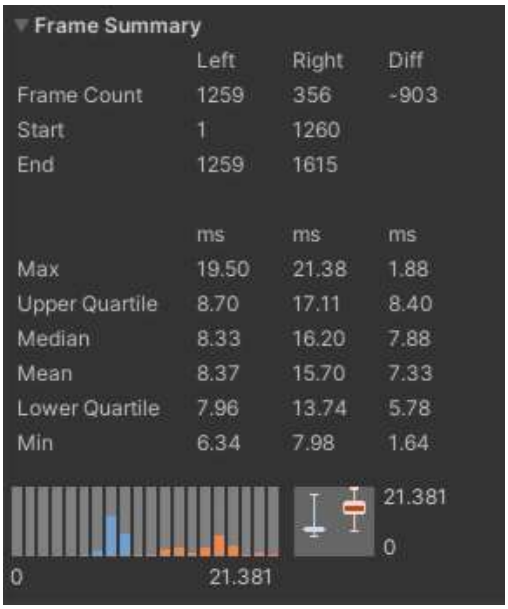


(a) 8 by 8: "Left" performance for up to five rows. "Right" performance for six rows



(b) 8 by 8: "Left" performance for six rows. "Right" performance for seven rows

Figure 5.22

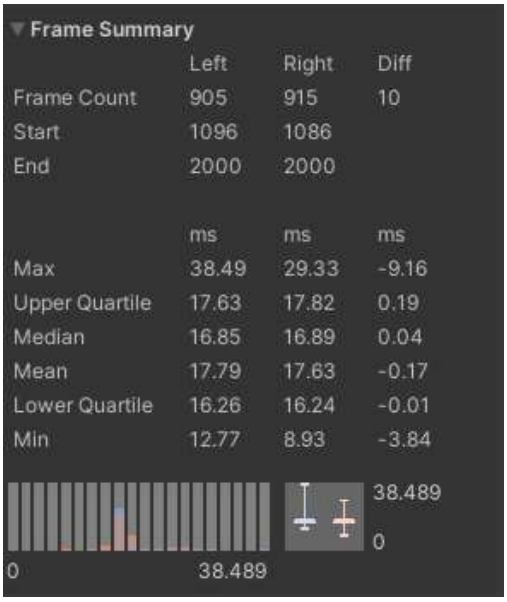


(a) 16 by 16: "Left" performance for up to five rows. "Right" performance for six rows

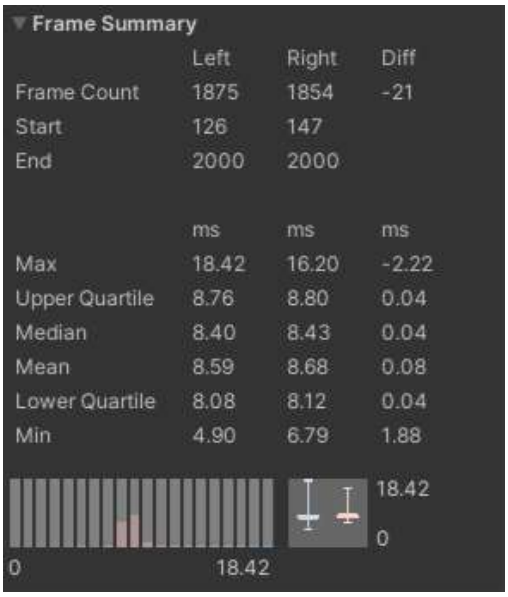
(b) 16 by 16: "Left" performance for six rows. "Right" performance for seven rows

Figure 5.23

Figure 5.24: GFx.WaitForGfxCommandFromMainThread on Render Thread



(a) Original Fire: "Left" Foveated Rendering off, "Right" Foveated Rendering on



(b) Adapted Smoke: "Left" Foveated Rendering off, "Right" Foveated on

Figure 5.25

test scenarios waited a lot of time on commands from the main thread, as seen on 5.24. According to official Unity Documentation [44] this may indicate a bottleneck on the main thread, while Gfx.WaitForPresentOnGfxThread may indicate a GPU bottleneck. Gfx.WaitForPresentOnGfxThread was not seen in any of the terrain test configurations, indicating that in this case it may indeed be a CPU and not GPU related performance bottleneck.

In the sixth test, foveated Rendering was tested. Foveated Rendering renders outer parts of the screen in lower resolution, mimicking the natural blurriness of vision, where objects in the center of the human field of view appear sharper than those close to the border. Foveated on and off was tested with the original high particle fire of test three with the "fire wiggle" script deactivated, and an additional run with the adapted smoke of test four. There was no overall noteworthy impact on performance, as seen in either first or second test setup in 5.25, indicating that the increase in frame time is most likely CPU based instead of GPU based, as foveated Rendering should decrease the GPU load.

CHAPTER 6

Discussion

The first prototype was overall well received and accepted by the expert users in the expert study. All attendees agreed that the VR forest fire training would be a useful addition to traditional training methods, and all participants besides one felt that their expectation for VR forest fire training were met. Overall, 87.5% were positive, they would use the VR forest fire training frequently. The terrain itself was deemed as suitable, with no additional comments for either improvements or pain points. While also received positively, most free text comments were received for fire and smoke, where one participant did not find the behavior of smoke and fire very suitable and others mentioned that the color of the smoke should be darker instead of white. The SUS score of 88.125, which can be seen as excellent according to Bangor et al.[20], also supports this by showing a high usability value and acceptance of the expert users participating in the study. The SUS score of 88.125 is comparable to the SUS score of 84.58 received by Grabowski for VR cadets in the study of a VR fire training prototype [28]. Grabowski's results for experienced firefighters were lower with a score of 67.67, which is to be expected since the training was aimed at cadets and not experienced firefighters.

The results of the user study also should be regarded cautiously, as three of eight participants had either none or low experience with VR, which could have lead to more positive answers because of overall the novelty of the experience. Also, none of the participants identified as female while in 2023 around 9.5% of firefighters were female according to the Austrian "Bundesfeuerwehr Verband" [1], thus the participants are not a perfect representation of the overall target group. During the tests, some network latency was seen in the initial connection stage, which was not encountered before in a controlled test environment. It is not clear what exactly introduced the extended connection times, but once connected the latency seemed fine and should have had no impact on the results.

Summarized and applied to the first research question, Q1: "How well do expert users assess and accept first responder training of wide area disaster events on a mobile stand-alone VR system?" it is fair to say, that the first prototype used by expert users was

assessed and accepted excellently. Acceptance was very high, as seven out of eight users would use the system frequently, and all eight users agreed that it would be a meaningful addition to standard training methods. Seven out of eight participants also believe, that virtual environments can convey knowledge of real world locations and six participants felt like their navigation skills for the real counterpart of the areas visualized in the training improved. Additionally, seven of the users think that the structure of the virtual environment is important and has an influence on decision-making for leaders, showing a need for training scenarios, which are built closely mirroring reality through virtual environments based on real locations. This, combined with the SUS score of 88.125, shows that wide area forest fire and more generally wide area disaster trainings should prove to be a feasible, well-accepted and useful possibility to extend the current training received by first responders. Especially, when taking in consideration that more occurrences of natural disasters like floods and forest fires are expected because of climate change [24] [30].

Regarding performance, it was evident that optimization methods are a necessary step to reduce computation power on mobile stand-alone VR devices and keep frame rates at an acceptable level, as seen especially on the particle effect tests (test 2, test 3, test 4) nearly doubling frame rate for test 3 and an increased frame rate of 30 to 40 frames for test 2 and 4. Also, a tiled terrain with terrain streaming allowed to spawn twice the amount of objects (2400 trees) compared to using a single full size terrain tile before reducing the frame rate. Interestingly, the amount of tiles used, did not have a significant influence on the performance. LODs and foveated rendering unexpectedly did not have a noticeable impact on performance. This can most likely be explained, that the maximum renderable triangle count of the Quest 3 was not exceeded in test case 1, thus using LODs which effectively use reduced complexity models with lower polygon count and lower resolution textures, had not the expected effect. Also, the LOD test showed that performance optimizations often just shift the complexity from one area to another: While the polygon count and the rendering time was indeed reduced, albeit slightly, new performance spikes were introduced because of additional shaders that needed to be loaded for each LOD, when shown for the first time. This problem can also be often seen in modern video games, and is known as “shader compilation stutter” [25]. None of the tests seemingly were GPU limited, thus foveated rendering did not show any noticeable performance improvements. Nevertheless, this shows that preemptive optimization should not be used for every optimization method available. Foveated rendering has a noticeable impact on image quality, as the outside areas on the screens get rendered in lower resolution. By using foveated rendering preemptively, image quality may suffer even when there may be no gain in performance at all. In comparison, the negative visual impact of LODs should be negligible, as well configured LODs change on distances that in an optimal scenario changes should be unperceivable, or at least hardly perceivable, to users and thus may be considered to be used pre-emptively under the assumption that pre-warming shaders keeps shader stutter spikes from appearing even when used for a larger number of different models.

Summarized and applied to the second research question Q2: "How well can optimization-methods reduce the computation power needed on a mobile stand-alone VR device for the simulation of wide area disaster events while maintaining the visual realism necessary?" it was shown that optimization are not only working well, when used right, but even absolutely necessary, when simulating wide areas. The optimization of particle effects and terrain streaming was proving to be effective for reducing the performance load on the Quest 3, allowing for higher frame rates when used. The LOD test was not showing the expected results, for one, because LODs can introduce performance spikes when shader prewarming is not configured, and for two, because the prepared test scene did not put enough strain on the Quest 3 as the scene was most likely not rendering enough triangles simultaneously to make the frame rate suffer.

The implementation of smoke and fire were tested on trees only, while the fire spread script was written in a way that generally every object can start burning in theory, some additional script changes would be necessary to make the fire spread according to the length and height of the object, instead of a predefined fixed height and width to properly work. Another crucial feature to improve the realism of the fire and smoke behavior is that the particle systems of fire and smoke react to wind. In theory this is already enabled, but neither performance nor behavior were tested yet with wind in the Unity Scene as part of this thesis.

Extensive performance tests and measurements in Unity seem to be not optimal yet. The Unity profiler can only track up to 2000 frames and while there is a possibility to log more frames by saving them to a file via scripts, Unity crashed as a whole when trying to import bigger files. It should also be noted, that the profiler itself introduces some performance overhead, which further increases when increasing the number of tracked frames from the default setting of 300 frames tracked. Thus, the measured performance may also not be completely in line with a release client, which is usually shipped without profiling. Additionally, performance can also vary between Unity versions, while I expect that the general influence of settings on performance is mostly the same (i.e., option A is faster than option B) the rate of how much these settings impact performance may differ.

The VROnSite project is using the Unity built-in render pipeline, so all tests and findings are only applicable for the built-in render pipeline and do not automatically apply to other render pipelines availabe in Unity like the Universal Render Pipeline (URP) or High Definition Render Pipeline (HDRP). Also, since the tests were only executed on a Meta Quest 3, performance may vastly differ for other stand-alone VR headsets like the Meta Quest 2 or the Pico 4[1]. Thus, the generalizability of the performance results is certainly limited without further testing. In the Unity Engine road map on the 20th September 2024 Unity announced that the built-in render engine will be deprecated starting with Unity 6 and completely removed with Unity 7 to be replaced by a Unified Renderer, which is a combination of the URP and the HDRP, thus, the findings have, to be frank, an expiry date. Nevertheless, it is common practice to use Long-Term-Support

[1] https://www.picoxr.com/de/products/pico4

versions and instead of the latest most up-to-date version to avoid bugs and experimental features, thus, the findings still provide value even after the initial release of Unity 7.

CHAPTER 7

Summary and Future Work

In the Design chapter 3 and Implementation chapter 4, a terrain creation workflow using real height data as a base was proposed and implemented for a roughly 3 km² area of Stammersdorf in the Unity 3D Engine, together with an efficient fire and smoke implementation, both being able to run on stand-alone VR headsets at a mostly stable frame rate of 72fps. These functionalities were then implemented in the existing VR training application VROnSite and used for both a user study with expert users and a performance evaluation in the Evaluation chapters 5.1 and 5.2 respectively.

For the user study, eight expert users participated in a predefined VR training scenario of a forest fire in the area of Stammersdorf. The training was overall received very well, with a SUS score of 88.125, which can be regarded as excellent [20] and all participants being able to see the fire and smoke from a distance. Still, the participants identified room for improvement in the visualization of the smoke, and one participant for the behavior of fire and smoke.

For the performance evaluation, six tests were defined, implemented and evaluated. The results show that performance optimization is essential when developing for stand-alone VR headsets and can increase frame rates significantly but should be used cautiously to pre-optimize as e.g., optimizations like foveated rendering have a negative impact on image quality but no performance benefit if the performance bottleneck is introduced by the CPU side of the application.

This thesis opens up the possibilities for future work in different directions. The creation of the terrain still involves a lot of manual steps, which could be analyzed and see which steps can be automatized in future work to reduce the time needed for creation of terrains based on GIS data. Another possibility is to introduce physical properties like type of wood, slope, and wind in the fire simulation and adapt spreading behavior for fire and smoke accordingly. As already mentioned in the Discussion chapter 6 the built-in render pipeline will be deprecated and removed in the future, thus future work could use the

Universal Render Pipeline instead and repeat the performance evaluation there with an additional focus on where the CPU limitations seen in the tests stem from. Another topic in this direction could be trying to incorporate the Unity Data-Oriented Technology and the Entity Component System, which should boost performance in CPU limited scenarios [46]. A more general but not less useful topic would be the creation of a reusable performance testing process for Unity for VR applications, which could prove useful in a multitude of Unity VR projects.

List of Figures

[1] https://flaimsystems.com/products/trainer

List of Tables

Acronyms

CAVE Cave Automatic Virtual Environment. 7

HDRP High Definition Render Pipeline. 77

URP Universal Render Pipeline. 77

VE Virtual Environment. 4

VR Virtual Reality. 1

WFS Web Feature Service. 27

WMS Web Map Service. 27

WMTS Web Map Tile Service. 27

Bibliography

[1] Feuerwehr statistik 2023. https://www.bundesfeuerwehrverband.at/2024/02/02/feuerwehr-statistik-2023-34-000-einsaetze-mehr-klimawandel-zeigt-seine-auswirkungen/. Accessed: 2024-10-05.

[2] Flaim trainer software update and new scenarios – r2 2022flaim trainer software update and new scenarios – r2 2022. https://docs.unity3d.com/Manual/class-TextureImporterOverride.html. Accessed: 2024-04-24.

[3] The future of meta quest, mixed reality, ai and more. https://www.facebook.com/MetaforDevelopers/videos/847258756776961. Timestamp around 35:20 ; Accessed: 2024-04-24.

[4] Galaxy s23 series thrashes the s22 in gpu benchmark. https://www.sammobile.com/news/galaxy-s23-series-thrashes-the-s22-in-gpu-benchmark/. Accessed: 2024-04-24.

[5] Gis data downloader asset description. https://assetstore.unity.com/packages/tools/integration/gis-data-downloader-199112?fbclid=IwAR2dVOdf-vIJp3fO8QGz6Gcepo4_cL0rp144cSUAwGXfFdLr8LFE7QqzcUA#description17. Accessed: 2024-04-24.

[6] Illogika studio | how to avoid garbage. https://illogika-studio.gitbooks.io/unity-best-practices/content/how-to-avoid-garbage.html. Accessed: 2024-04-24.

[7] Memphis fire department the first us metro to adopt flaim trainer. https://flaimsystems.com/case-studies/memphis-fire-department-the-first-us-metro-to-adopt-flaim-trainer. Accessed: 2024-04-24.

[8] Meta | oculus rift s minimum requirements. https://www.meta.com/de-de/help/quest/articles/headsets-and-accessories/oculus-rift-s/rift-s-minimum-requirements/. Accessed: 2024-04-24.

[9] Paraná fire department rolls out first of its kind immersive firefighter training in latin america. https://flaimsystems.com/case-studies/parana-fire-department-rolls-out-first-of-its-kind-immersive-firefighter-training-in-latin-america. Accessed: 2024-04-24.

[10] Unity documentation | dynamic batching. https://docs.unity3d.com/Manual/dynamic-batching.html. Accessed: 2024-04-24.

[11] Unity documentation | dynamic batching. https://docs.unity3d.com/Manual/class-TextureImporterOverride.html. Accessed: 2024-04-24.

[12] Unity documentation | garbage collection best practices. https://docs.unity3d.com/Manual/performance-garbage-collection-best-practices.html. Accessed: 2024-04-24.

[13] Unity documentation | objectpool. https://docs.unity3d.com/2021.2/Documentation/ScriptReference/Pool.ObjectPool_1.html. Accessed: 2024-04-24.

[14] Unity documentation | static batching. https://docs.unity3d.com/Manual/static-batching.html. Accessed: 2024-04-24.

[15] Gena pro and gaia pro - level design example. https://www.youtube.com/watch?v=cQ9odT9gtvY, December 2020. Accessed: 2024-06-02.

[16] Unity forum | terrain leveling. https://forum.unity.com/threads/terrain-leveling.926483/], July 2020. Accessed: 2024-06-02.

[17] Github gist | terrain leveling adapted by kurtdekker. https://gist.github.com/kurtdekker/f9e7b0bdf4b2f9c0455a99f7f0f4a77a, August 2021. Accessed: 2024-06-02.

[18] Terrain loading i& streaming in gaia pro i& gaia pro 2021. https://canopy.procedural-worlds.com/library/tools/gaia-pro-2021/written-articles/creating_runtime/2-terrain-loading-streaming-in-gaia-pro-gaia-pro-2021-r64/, December 22, 2021. Accessed: 2024-06-02.

[19] J. Bailenson. *Experience on Demand: What Virtual Reality Is, How It Works, and What It Can Do*. W. W. Norton, 2018.

[20] A. Bangor, P. Kortum, and J. Miller. Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, 4(3):114–123, 2009.

[21] M. Bellemans, D. Lamrnens, J. De Sloover, T. De Vleeschauwer, E. Schoofs, W. Jordens, B. Van Steenhuyse, J. Mangelschots, S. Selleri, C. Hamesse, T. Fréville, and R. Haeltermani. Training firefighters in virtual reality. In *2020 International Conference on 3D Immersion (IC3D)*, pages 01–06, 2020.

[22] P. Braun, M. Grafelmann, F. Gill, H. Stolz, J. Hinckeldeyn, and A.-K. Lange. Virtual reality for immersive multi-user firefighter-training scenarios. *Virtual Reality I& Intelligent Hardware*, 4(5):406–417, 2022. Computer graphics for metaverse.

[23] J. Brooke. Sus: A quick and dirty usability scale. *Usability Eval. Ind.*, 189, 11 1995.

[24] C. Burton, S. Lampe, D. I. Kelley, W. Thiery, S. Hantson, N. Christidis, L. Gudmundsson, M. Forrest, E. Burke, J. Chang, H. Huang, A. Ito, S. Kou-Giesbrecht, G. Lasslop, W. Li, L. Nieradzik, F. Li, Y. Chen, J. Randerson, C. P. O. Reyer, and M. Mengel. Global burned area increasingly explained by climate change. *Nature Climate Change*, Oct 2024.

[25] S. Butler. What is shader compilation and why does it make pc games stutter? `https://www.howtogeek.com/846514/what-is-shader-compilation-and-why-does-it-make-pc-games-stutter/`, Nov 2022. Accessed: 2024-10-25.

[26] M. Cha, S. Han, J. Lee, and B. Choi. A virtual reality based fire training simulator integrated with fire dynamics data. *Fire Safety Journal*, 50:12–24, 2012.

[27] Y. Fu and Q. Li. A virtual reality–based serious game for fire safety behavioral skills training. *International Journal of Human–Computer Interaction*, 0(0):1–17, 2023.

[28] A. Grabowski. Practical skills training in enclosure fires: An experimental study with cadets and firefighters using cave and hmd-based virtual training simulators. *Fire safety journal*, 125:103440, 2021.

[29] R. V. Hoang, M. R. Sgambati, T. J. Brown, D. S. Coming, and F. C. Harris. Vfire: Immersive wildfire simulation and visualization. *Computers & Graphics*, 34(6):655–664, 2010. Graphics for Serious Games Computer Graphics in Spain: a Selection of Papers from CEIG 2009 Selected Papers from the SIGGRAPH Asia Education Program.

[30] J. Kimutai, R. Vautard, M. Zachariah, R. Tolasz, V. Šustková, C. Cassou, B. Clarke, K. Haslinger, M. Vahlberg, R. Singh, E. Stephens, H. Cloke, E. Raju, N. Baumgart, L. Thalheimer, F. Otto, G. Koren, S. Philip, S. Kew, P. Haro, J. Vibert, and A. Von Weissenberg. *Climate change and high exposure increased costs and disruption to lives and livelihoods from flooding associated with exceptionally heavy rainfall in Central Europe*. Sept. 2024.

[31] S.-C. Lee, C.-Y. Lin, and Y.-J. Chuang. The Study of Alternative Fire Commanders' Training Program during the COVID-19 Pandemic Situation in New Taipei City, Taiwan. *IJERPH*, 19(11):1–22, May 2022.

[32] P. Lorusso, M. De Iuliis, S. Marasco, M. Domaneschi, G. P. Cimellaro, and V. Villa. Fire emergency evacuation from a school building using an evolutionary virtual reality platform. *Buildings*, 12(2), 2022.

[33] X. Lu, Z. Yang, Z. Xu, and C. Xiong. Scenario simulation of indoor post-earthquake fire rescue based on building information model and virtual reality. *Advances in Engineering Software*, 143:102792, 2020.

[34] A. Moreno, J. Posada, Álvaro Segura, A. Arbelaiz, and A. García-Alonso. Interactive fire spread simulations with extinguishment support for virtual reality training tools. *Fire Safety Journal*, 64:48–60, 2014.

[35] A. Mossel, C. Schönauer, M. Froeschl, A. Peer, J. Göllner, and H. Kaufmann. Immersive training of first responder squad leaders in untethered virtual reality. *Virtual Reality*, 204:1–15, Dec. 2020.

[36] M. A. Nasaruddin, N. H. Azmi, N. Ibrahim, E. M. Saari, and N. A. Ahmad. Development of virtual reality fire extinguisher game for safety training. *AIP Conference Proceedings*, 2750(1):040026, 06 2023.

[37] S. Ooi, A. Kikuchi, T. Goto, and M. Sano. Development and verification of mixed disaster training system in virtual reality based on experience learning. In *2021 10th International Conference on Educational and Information Technology (ICEIT)*, pages 29–33, 2021.

[38] B. Pairet. Vrfftu. `https://xrlab.rma.ac.be/vrfftu/`. Accessed: 2024-04-24.

[39] K. S. B. Robert S. Kennedy, Norman E. Lane and M. G. Lilienthal. Simulator sickness questionnaire: An enhanced method for quantifying simulator sickness. *The International Journal of Aviation Psychology*, 3(3):203–220, 1993.

[40] Y. S. Sadek Hosny, M. A.-M. Salem, and A. Wahby. Performance optimization for standalone virtual reality headsets. In *2020 IEEE Graphics and Multimedia (GAME)*, pages 13–18, 2020.

[41] N. P. Singh, B. Sharma, and A. Sharma. Performance analysis and optimization techniques in unity 3d. In *2022 3rd International Conference on Smart Electronics and Communication (ICOSEC)*, pages 245–252, 2022.

[42] R. Tao, H.-x. Ren, and X.-q. Peng. Ship fire-fighting training system based on virtual reality technique. In M. S. Mohamed Ali, H. Wahid, N. A. Mohd Subha, S. Sahlan, M. A. Md. Yunus, and A. R. Wahap, editors, *Modeling, Design and Simulation of Systems*, pages 249–260, Singapore, 2017. Springer Singapore.

[43] D. Tate, L. Sibert, and T. King. Virtual environments for shipboard firefighting training. pages 61–68, 215, 04 1997.

[44] U. Technologies. Common profiler markers. `https://docs.unity3d.com/Manual/profiler-markers.html#:~:text=Gfx.WaitForCommands,bottleneck%20on%20the%20main%20thread.` Accessed: 2024-10-25.

[45] G. Vukelic, D. Ogrizovic, D. Bernecic, D. Glujic, and G. Vizentin. Application of vr technology for maritime firefighting and evacuation training—a review. *Journal of Marine Science and Engineering*, 11(9), 2023.

[46] H. Zahran. Boosting performance with unity dots and ecs. https://www.linkedin.com/pulse/boosting-performance-unity-dots-ecs-hamam-zahran-clrbf#:~:text=Performance%3A%20DOTS%20excels%20in%20CPU,cleaner%20code%20and%20easier%20maintenance., Mar 2024. Accessed: 2024-10-05.

Appendix

TeilnehmerIn #: _____

Allgemeine Informationen

1. **Alter:** _____ (Jahre)

2. **Geschlecht:** [ ] männlich [ ] weiblich [ ] divers
3. **Sind Sie mit einer Führungsaufgabe betraut?** [ ] Ja [ ] Nein
4. **Sind Sie im Rahmen der Ausbildung tätig?** [ ] Ja [ ] Nein

5. **Bitte geben Sie an, wie viele Übungen im Schnitt pro Jahr für Sie vorbereitet werden.**

Keine	1-3	4-7	Mehr als 7

6. **Bitte geben Sie an wie viele Übungen Sie im Schnitt pro Jahr für andere Führungskräfte vorbereiten.**

Keine	1-3	4-7	Mehr als 7

7. **Bitte geben Sie an, an wie vielen Planspielen Sie im Schnitt pro Jahr teilnehmen.**

Keine	1-3	4-7	Mehr als 7

8. **Wie würden Sie Ihren persönlichen Fitness-Level einschätzen?**

Keine Fitness	Geringe Fitness	Durchschnittlich	Moderate Fitness	Starke Fitness
1	2	3	4	5

9. **Haben Sie Vorerfahrung mit Virtual Reality?**

Keine	Gering	Durchschnitt	Moderat	Viel
1	2	3	4	5

10. **Haben Sie Übung / Vorerfahrung in der Benutzung eines Gamepads?**

Keine	Gering	Durchschnitt	Moderat	Viel
1	2	3	4	5

Prä-Exposition Simulatorkrankheit

Bitte markieren Sie, welche der folgenden Symptome aktuell auf Sie zutrifft. Die gleiche Fragestellung wird Ihnen nach dem Experiment nochmals gestellt.

1.	Allgemeines Unwohlsein	Kein	Gering	Moderat	Stark
2.	Müdigkeit	Keine	Gering	Moderat	Stark
3.	Kopfweh	Kein	Gering	Moderat	Stark
4.	Schwitzen	Kein	Gering	Moderat	Stark
5.	Übelkeit	Kein	Gering	Moderat	Stark
6.	Unscharfes Sehen	Nein	Ja (Gering	Moderat	Stark)
7.	Schwindel	Nein	Ja (Gering	Moderat	Stark)
8.	Verwirrtheit	Nein	Ja (Gering	Moderat	Stark)

FRAGEBOGEN TRAINING

TeilnehmerIn #: _____

Post-Exposition Simulatorkrankheit

Bitte markieren Sie, welche der folgenden Symptome aktuell auf Sie zutreffen.

1.	Allgemeines Unwohlsein	Kein	Gering	Moderat	Stark
2.	Müdigkeit	Keine	Gering	Moderat	Stark
3.	Kopfweh	Kein	Gering	Moderat	Stark
4.	Schwitzen	Kein	Gering	Moderat	Stark
5.	Übelkeit	Kein	Gering	Moderat	Stark
6.	Unscharfes Sehen	Nein	Ja (Gering	Moderat	Stark)
7.	Schwindel	Nein	Ja (Gering	Moderat	Stark)
8.	Verwirrtheit	Nein	Ja (Gering	Moderat	Stark)

Generelle Fragen

1. Das VR Waldbrand Szenario würde mir helfen mich für Einsätze mit Waldbränden vorzubereiten

Trifft nicht zu	Trifft eher nicht zu	Unentschieden	Trifft eher zu	Trifft zu
1	2	3	4	5

2. Das VR Waldbrand Szenario erfüllt meine Erwartungen an ein virtuelles Training für Waldbrände

Trifft nicht zu	Trifft eher nicht zu	Unentschieden	Trifft eher zu	Trifft zu
1	2	3	4	5

3. Ich fände das VR Waldbrand Training nützlich als zusätzliches Training zu herkömmlichen Trainings Methoden

Trifft nicht zu	Trifft eher nicht zu	Unentschieden	Trifft eher zu	Trifft zu
1	2	3	4	5

4. Ich konnte während des Trainings unerwartetes Ruckeln oder Verzögerungen wahrnehmen

Trifft nicht zu	Trifft eher nicht zu	Unentschieden	Trifft eher zu	Trifft zu
1	2	3	4	5

Feuer und Rauch

1. Die Visualisierung des Feuers war geeignet für das Training

Gar nicht	Kaum	Okay	Gut	Sehr Gut
1	2	3	4	5

2. Ich konnte das Feuer auch aus der Ferne wahrnehmen.

Sehr schlecht	Schlecht	Unentschieden	Gut	Sehr gut
1	2	3	4	5

3. Das Verhalten des Feuers war geeignet für das Training

Trifft nicht zu	Trifft eher nicht zu	Unentschieden	Trifft eher zu	Trifft zu
1	2	3	4	5

4. Die Visualisierung des Rauchs war geeignet für das Training

Sehr schlecht	Schlecht	Unentschieden	Gut	Sehr gut
1	2	3	4	5

5. Ich konnte den Rauch auch aus der Ferne wahrnehmen.

Sehr schlecht	Schlecht	Unentschieden	Gut	Sehr gut
1	2	3	4	5

6. Das Verhalten des Rauchs war geeignet für das Training

Sehr schlecht	Schlecht	Unentschieden	Gut	Sehr gut
1	2	3	4	5

Bitte beschreiben Sie kurz Ihre Erfahrung mit dem Feuer und Rauch in dem Trainingsszenario (was mochten Sie besonders, was sollte verbessert werden, was hat Ihnen gefehlt...) :

Terrain

1. Ich hatte bereits vor dem 'Warm-up' Ortskenntnisse von Stammersdorf, Wien

Trifft nicht zu	Trifft eher nicht zu	Unentschieden	Trifft eher zu	Trifft zu
1	2	3	4	5

2. Ich glaube, dass Virtuelle Umgebungen Ortskenntnisse für die echte Welt vermitteln können

Trifft nicht zu	Trifft eher nicht zu	Unentschieden	Trifft eher zu	Trifft zu
1	2	3	4	5

3. Ich denke, dass ich Teile aus der Virtuellen Umgebung (Stammersdorf) in der echten Umgebung wiedererkennen kann

Trifft nicht zu	Trifft eher nicht zu	Unentschieden	Trifft eher zu	Trifft zu
1	2	3	4	5

4. Ich denke, dass ich mich in dem Training dargestellten Teil von Stammersdorf nun auch in der echten Welt dort besser zurechtfinden kann

Trifft nicht zu	Trifft eher nicht zu	Unentschieden	Trifft eher zu	Trifft zu
1	2	3	4	5

5. Die visuelle Darstellung der Virtuellen Umgebung war geeignet für das Training

Trifft nicht zu	Trifft eher nicht zu	Unentschieden	Trifft eher zu	Trifft zu
1	2	3	4	5

6. Ich denke, die Simulation von weitläufigen Gebieten ist wichtig für Führungskräftetrainings in Waldbrandzsenarien

Trifft nicht zu	Trifft eher nicht zu	Unentschieden	Trifft eher zu	Trifft zu
1	2	3	4	5

7. Ich denke, dass der Aufbau der virtuellen Umgebung Einfluss auf die Entscheidungsfindung von Führungskräften in VR-Waldbrandszenarien hat

Trifft nicht zu	Trifft eher nicht zu	Unentschieden	Trifft eher zu	Trifft zu
1	2	3	4	5

8. Ich denke, dass die Virtuelle Umgebung geeignet war für das Training

Trifft nicht zu	Trifft eher nicht zu	Unentschieden	Trifft eher zu	Trifft zu
1	2	3	4	5

Falls es Dinge gab, die Ihnen bei der Visualisierung oder dem Aufbau des Geländes gut oder weniger gut gefallen haben, erwähnen sie diese bitte kurz mit Erläuterung was Sie daran gestört oder ihnen gefallen hat:

Fragen zur Benutzerfreundlichkeit des VR-Systems für Waldbrand

1. Ich denke, ich würde das **VR-System für Waldbrand Training** regelmäßig nutzen.

Trifft nicht zu	Trifft eher nicht zu	Unentschieden	Trifft eher zu	Trifft zu
1	2	3	4	5

2. Ich habe das **VR-System für Waldbrand Training** als unnötig komplex empfunden.

Trifft nicht zu	Trifft eher nicht zu	Unentschieden	Trifft eher zu	Trifft zu
1	2	3	4	5

3. Ich denke, das **VR-System für Waldbrand Training** war leicht zu benutzen.

Trifft nicht zu	Trifft eher nicht zu	Unentschieden	Trifft eher zu	Trifft zu
1	2	3	4	5

4. Ich denke, ich würde Unterstützung einer technisch versierten Person benötigen, um das **VR-System für Waldbrand Training** zu verwenden.

Trifft nicht zu	Trifft eher nicht zu	Unentschieden	Trifft eher zu	Trifft zu
1	2	3	4	5

5. Ich fand, dass die unterschiedlichen Funktionen (speziell Gelände, Feuer, Rauch) gut in das **VR-System für Waldbrand Training** integriert waren.

Trifft nicht zu	Trifft eher nicht zu	Unentschieden	Trifft eher zu	Trifft zu
1	2	3	4	5

6. Ich finde, dass es zu viele (technische) Inkonsistenzen im **VR-System für Waldbrand Training** gab.

Trifft nicht zu	Trifft eher nicht zu	Unentschieden	Trifft eher zu	Trifft zu
1	2	3	4	5

7. Ich könnte mir vorstellen, dass die meisten Leute schnell lernen würden, wie man das **VR-System für Waldbrand Training** verwendet.

Trifft nicht zu	Trifft eher nicht zu	Unentschieden	Trifft eher zu	Trifft zu
1	2	3	4	5

8. Ich fand das **VR-System für Waldbrand Training** sehr umständlich zu benutzen.

Trifft nicht zu	Trifft eher nicht zu	Unentschieden	Trifft eher zu	Trifft zu
1	2	3	4	5

9. Ich war sicher, das **VR-System für Waldbrand Training** richtig zu benutzen.

Trifft nicht zu	Trifft eher nicht zu	Unentschieden	Trifft eher zu	Trifft zu
1	2	3	4	5

10. Ich musste eine Menge Dinge lernen, bevor ich das **VR-System für Waldbrand Training** nutzen konnte.

Trifft nicht zu	Trifft eher nicht zu	Unentschieden	Trifft eher zu	Trifft zu
1	2	3	4	5

Bitte beschreiben Sie kurz Ihre Erfahrung mit dem VR Waldbrand Training (was mochten Sie besonders, was sollte verbessert werden, was hat Ihnen gefehlt...) :