ACIT5900

Master Thesis

in

Applied Computer and Information Technology (ACIT)

May 2023

Cloud-Based Services and Operations

Your head in the clouds

Towards intuitive cloud interaction through virtual reality

Martin Bang

Department of Computer Science
Faculty of Technology, Art and Design

Acknowledgement

Writing this master thesis has been an educational journey, marked by both challenges and invaluable learning experiences. Throughout the journey, it has tested my skills as a writer and encouraged me to delve into uncharted territories. I am truly grateful to the people who have supported and guided me along the way, as they have played a major role in making this achievement possible.

First and foremost, I must express my gratitude to my supervisor, Kyrre Begnum, for his guidance and support throughout this project. I am very grateful for his knowledge and expertise, which have been necessary in guiding me towards the right direction and enabling me to complete the project to the best of my abilities.

I would also like to give my gratitude to my mom, dad, and sister for supporting and believing in me through stressful periods. They are my biggest supporters and this could not have been done without them. Lastly, I would like to thank my incredible girlfriend for motivating and pushing me to never give up. Her support made me achieve the seemingly impossible, and I am very lucky to have her in my life. This journey could not have been completed without the people around me and their support.

Abstract

As cloud services continue to expand every year, their complexity in terms of user interaction also increases. For the untrained, navigating cloud tools can be confusing, especially when the interaction still relies on command line interpreters that have remained largely unchanged since their inception. While efforts have been made to streamline this process and make it more user-friendly, it can still be challenging.

Virtual reality is a technology that has been around for some time, but has recently gained popularity due to the release of more advanced headsets. It's being increasingly utilized by various industries to simplify complex concepts through simulators and other applications. Virtual Reality provides a unique and immersive perspective for user interaction, prompting the question whether it can be combined with the cloud to reduce complexity.

This project aims to explore the potential of virtual reality as an alternative to current cloud interaction methods and whether it can effectively reduce complexity. By leveraging the benefits of the immersive environment that virtual reality provides, the hope is to explore a more intuitive and user-friendly interface for cloud interaction that can benefit both experienced and inexperienced users alike.

Contents

List of Figures

1 Introduction

Digital services occupy most of our everyday lives. We are at the stage where digitalization can be said to have become part of the foundation of our society. To the point that everything we do, has some connection to a digital component. It is estimated that in 2021, 5.12 billion people are on the internet, which is equal to 64.4 percent of the global population[3]. Which is a testament to that our lives are rooted in digitalization, so much so that it's difficult to imagine a modern life without it. Digitalization has become an extension of ourselves, and we will continue to become more dependent on it as time goes.

With digital services becoming so integrated, so does their complexity. There will be more challenges to overcome and demands to fulfill[5]. Therefore, in order to sort out this growth, digital infrastructures have to be put in place to deal with it. The infrastructures need to be optimized to handle the increase in new services, as well as methods for scaling these services to new heights.

Cloud technology has become the new standard for digital services. It's one of the most important drivers of the modern tech industry, and has enabled what we have today. Through its flexibility, industries can scale services without worrying about physical limitations in hardware. Services can quickly launch new functionality without dealing with underlying infrastructure. It is its size what builds the backbone of the cloud, allowing for a shared network of millions of computers to work together. The cloud provides a seemingly endless source of computing power and creates a potent environment for technology to grow.

Even though the cloud has plenty of benefits, behind the scenes, its infrastructure is still based around the fundamental task of managing computers. Even if it's for the cloud user or administrator, the cloud architecture provides a variety of tools for making that management more achievable. Tools like Puppet or Terraform are just a few examples of the ones that are utilized, and the list can go on forever. The fact of the matter is that with the number of tools becoming even greater, it is difficult to grasp how everything operates. This way, cloud architecture can become disorienting and understanding what causes it, becomes more important.

How we operate and interact with the cloud has stagnated. With the number of tools needed to run a reliable cloud operation, it's obvious that confusion will occur due to its complexity. Most of the cloud tools operate using a Command Line Interpreter that can be difficult to manage because of the syntax and semantics of the commands involved. It feels outdated, and yet it is how we operate the cloud to this date. This way of interaction still remains unchanged and has been since the beginning of the cloud. Interaction with the cloud has room for more expansion, and could need the encouragement for more creativity within the field.

1.1 Problem statement

Virtual Reality has experienced a renaissance during the last few years. The ability to immerse users entirely inside a virtual landscape has opened new possibilities for how we can interact with technology. It has shown promise in fields such as training simulations with its ability to include the user into the learning experience. It's reasonable to assume that it can offer itself as an alternative interface to cloud operations, as it exists today, with its challenges of usability. Until now, integration of these two technologies has not yet been fully explored. Making VR able to interact with the cloud, would allow us to investigate approaches that could test VR as an alternative interface for cloud operations.

Therefore, in this project, the problem statement goes as follows: **"Design and develop a VR interface in order to explore cloud interaction with regard to reduced complexity."**

Through the process of *designing and developing a VR interface*, the objective is to create a prototype that offers an alternative to conventional cloud interfaces. The VR interface aims to simplify *cloud interaction* by harnessing the interactive aspects of VR technology. The primary focus is on *reducing complexity* and evaluating the effectiveness of this reduction through results and explanations.

2 Background

2.1 History of the cloud

The roots of cloud technology can be traced back to the 1950s[11]. The United States military was in the need of a method of sharing computing power across their internal infrastructure. They came up with the solution to develop a powerful mainframe that internal terminals could connect with and use in resource demanding tasks. This proved to be effective as it allowed more people to access powerful hardware without directly interacting with it. This was at a time when computer technology was expensive and having the ability to effectively share available computer resources was in high demand. At the time it was called non-local computing, however the term “cloud computing” and the technologies utilized today didn’t start emerging before 1996.

Compaq first used the term “cloud computing” as a way of describing the distributed computing concept. When coming up with this term, Compaq envisioned it as the future of the internet. They were right in their predictions, and “Cloud computing” was collectively accepted by professionals as the optimal approach to a new infrastructure. However, in 1996 most computers still relied on local applications to function, and the internet was not yet in the shape to embrace the “cloud”. Computers were still mostly for enthusiasts and too expensive for adopting such an idea. Going forward a couple of years, things quickly started to change, and more people could afford home computers. With cheaper computers flooding the market, so did the effort to innovate and improve the internet. Cloud became more important than ever, and companies saw the potential in removing themselves from the local computer and unto the internet. With this transformation, companies saw improvements in efficiency and reachability which created new concepts such as PaaS (Platform-as-a-Service), SaaS (Service-as-a-Service), and IaaS (Infrastructure-as-a-Service, essentially making cloud the new industry standard.

Nowadays, most sectors, either private or governmental, use some sort of cloud solution for its operation. This has created a shift in human history where knowledge has become widely accessible and opportunities for innovation have turned into the domain for individuals. The field of cloud computing is still in its early stages of life, but already has provided a lot for the global development, and will most likely continue to change into the future.

2.2 History of cloud interaction

There has been a shift in how interaction happens with the cloud. During its earliest stages, resources were only accessible through APIs (An API, also called an application programming interface, is a method for two or more computers to communicate with each other[23]). Interaction was limited to developers having the right expertise and necessary documentation to operate them. It required skills to handle, and having a graphical interface was seldom. Still, to this date, APIs are used to communicate with the cloud, but there have been some changes. More cloud providers are offering a graphical interface to interact with their services. This can be observed happening with most of the major cloud providers like AWS, Google Cloud or Azure. It's a shift where companies are simplifying the operation of the cloud. So, why did cloud providers take this step towards interfaces?

The first reason is the product range. Looking at AWS (Amazon Web Services) now, they have an assortment of 18 categories of products. Each category has subcategories like AI, VR and AR, which sums up to a lot of options[14]. With this many options, it is reasonable to understand why AWS has compiled every service within a graphical interface. If AWS still only offered an API to communicate with their services, documentation and operationality would be difficult to properly maintain. An interface allows for organizing the API in buttons and defining input methods that hinder errors that simplify usability.

The second reason is consistency[8], where the interface has the same setup regardless of which device it is being used on. This helps developers unanimously keep track of their portfolio of running services without having to worry about inconsistencies in the interface. Different teams might be responsible for various products, but since all the products are accessible through the same interface, there should be no confusion with layouts or what each of the buttons do. By having a consistent design language on an interface, teams will have better understanding of each other's portfolio without having to deal with UI (User Interface) related hindrances.

The third reason is navigation[8]. As more products are being offered by the cloud provider, more effort has to be put keeping them organized. With a graphical interface, having "search" functionalities for finding products will be essential in a large portfolio. Having the ability to search is where an interface can be beneficial, as with an API, searching, will require more effort to perform. With search being one of the examples, the other aspect of navigation is being able to use icons for simplifying usability, as well as graphical presentations to enhance visuality for the user.

The gradual change from cloud being solely API endpoints to also full-on web interfaces is becoming the go-to method for most cloud providers. These current shifts can be observed with most of them, and effort is being put in to making the user experience as easy as possible. A good example of this shift can be observed with Amazon Web Services. From their beginnings in 2006 to this date, plenty of changes have been done in how users interact is with their services.

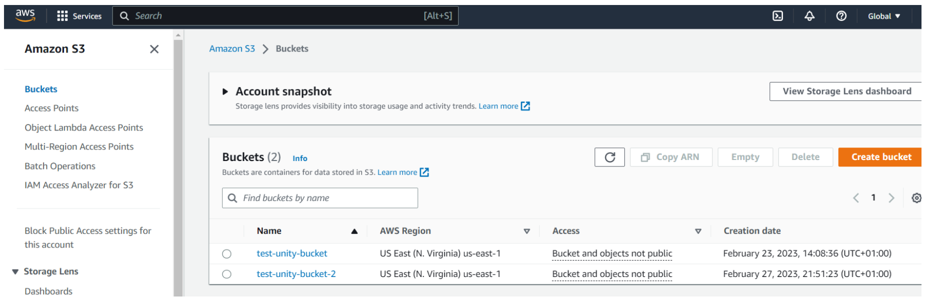
2.2.1 The early AWS API

When Amazon first launched its cloud services, Amazon Web Services or "AWS", there was a significant difference in the way it operated compared to how it operates today[13]. Initially, AWS started with a service called S3, or "simple storage service." A service that still exists today and offers customers to store and retrieve data objects on their cloud. At that time, however, the service was only accessible through an API, and not through the type of web interface they have now. During this period, web interfaces were still a novel concept, and Amazon's approach to integration via APIs was deemed more practical. However, this approach came with a learning curve that required developers to have eligible knowledge about API technology, which meant only the ones with the right skills were able to operate it. As a result, some developers were excluded from utilizing Amazon's services because of lacking the right technical skills. This, of course, was not the aim for Amazon, as they wanted a service in which any type of developer could interact with. AWS still offers an API to interact with their service, but nowadays, it is often overshadowed by the more popular web interface.

Figure 1: The AWS S3 logo.
Source: Wikimedia

2.2.2 The current AWS web interface

Since Amazon first introduced its API-based S3 service in 2006, a lot has changed in how users interact with their services. Today, Amazon offers a web interface that enables users to manage almost every aspect of their cloud services. The web interface's user-friendly design has made it accessible to a broader range of users, reducing the technical skills previously needed to operate the service. While AWS still offers an API for managing the S3 service, the current version requires interaction with the web interface to access its full functionality. Some argue that this necessity for using the web interface can be restrictive, as it limits customizability and usability for integration with custom applications. However, AWS justifies the need for the web interface as a way to intuitively organize their broad portfolio of services and make them more manageable for users.



Storage lens provides visibility into storage usage and activity trends. Learn more

Buckets (2) Info

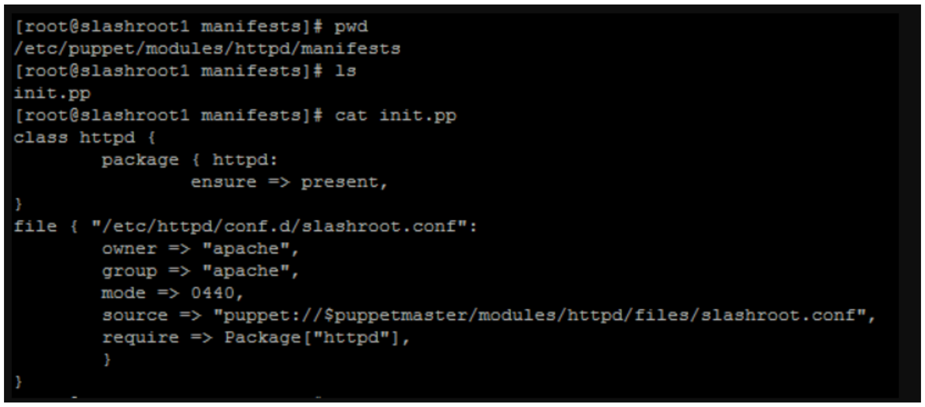
Buckets are containers for data stored in S3. Learn more

Name	AWS Region	Access	Creation date
test-unity-bucket	US East (N. Virginia) us-east-1	Bucket and objects not public	February 23, 2023, 14:08:36 (UTC+01:00)
test-unity-bucket-2	US East (N. Virginia) us-east-1	Bucket and objects not public	February 27, 2023, 21:51:23 (UTC+01:00)

Figure 2: Snapshot from the AWS web interface, showcasing the S3 service today.

2.3 The Command Line Interpreter approach

While graphical user interfaces, or GUIs, provide an intuitive way for users to interact with cloud technology, the most common method of interaction is usually through a CLI or Command Line Interpreter as seen in automation and scripting. The CLI exists on every operating system and is used for performing advanced management tasks, sometimes not possible through the normal GUI. Its primary benefit is having access to a range of functionality that is easily customizable. The default command line interpreter is presented with a minimalist interface, usually consisting of a black background with white letters. A design that hasn't changed much, since its inception, as it still maintains that retro look from the 70s.



```
[root@slashroot1 manifests]# pwd
/etc/puppet/modules/httpd/manifests
[root@slashroot1 manifests]# ls
init.pp
[root@slashroot1 manifests]# cat init.pp
class httpd {
        package { httpd:
                ensure => present,
}
file { "/etc/httpd/conf.d/slashroot.conf":
        owner => "apache",
        group => "apache",
        mode => 0440,
        source => "puppet://$puppetmaster/modules/httpd/files/slashroot.conf",
        require => Package["httpd"],
        }
}
```

Figure 3: CLI used to build a puppet manifest.
Source: Slashroot

Using the CLI, you can manipulate various elements of an OS by typing simple commands consisting of a few letters in various combinations. Though there is a learning curve involved in mastering these commands, many developers still prefer the CLI for navigating computers and servers. Allowing them to execute tasks with precision and efficiency, which otherwise would have taken longer time through a GUI. In essence, the CLI is a versatile interface that can accomplish a wide range of tasks. From managing file systems to configuring network settings, and gives the user access to the full power of an operating system. While it may not be as visually appealing as a GUI, it is an essential tool for developers, and it's the standard approach for operations.

It's important to understand the relation between CLI and cloud technology. Cloud technologies are designed and deployed on servers. When developers interact with these servers, they typically do so through SSH (Secure shell), connecting directly from their host computer to the dedicated server. Once they are inside the server's operating system, they are greeted with the CLI, which is the primary interface for managing servers. Since cloud technologies are built and deployed on servers that are accessed primarily through the CLI, developers must adapt to this format. Through CLI, developers interact with cloud technologies, building and deploying the solution to that server. This format offers no visualization, and it can be confusing to understand whether a solution works properly or not.

2.3.1 CLI is still how the cloud is operated

Despite the widespread adoption of cloud technologies, the main approach to operating and maintaining them remains largely unchanged through the use of CLI. This way of interaction is not going away anytime soon, as it has become deeply ingrained in the way of how systems are operated. While using the CLI may not be as visually appealing as GUIs, it remains popular because of its efficiency and straightforward functionality. Although there are no signs of it disappearing anytime soon, it should be worth in exploring alternative methods of interaction.

2.3.2 Any effort for change?

Is there any reason to change from this approach? It is difficult to factor out its importance in development and especially cloud operations, but it doesn't mean that something new can't be explored. In an article about the accessibility of CLI, researchers evaluated how well CLI operates with individuals having visual impermanent[10]. Their findings indicated that despite the CLI being simplistic in nature, as it is text-based and keyboard operable only, it still faces a lot of accessibility issues. These issues include "unstructured text, lack of status and progress indication, and inaccessible error messages"[10]. The researchers also found that the current CLI only fulfilled a small subset of the Web Content Accessibility Guidelines (WCAG)[2], which are a set of guidelines for making web content more accessible, primarily for people with disabilities. Therefore, it is clear that current CLI's need to make improvements to comply with these guidelines and provide a better user experience for individuals with visual impairments. One concept that tackles this lack of usability is terminal emulators. Terminal emulators are a form of graphical interface that are put on top of the CLI to make it more visually pleasing and user-friendly[9]. Some emulators even allow for mouse support and to redefine the format of the terminals to make them more useable. While there are good intentions behind terminal emulators that can improve usability for individuals with visual impairments, their prominence is still undermined by their predecessor, the standard CLI. While the CLI is still the preferred method for many, it is important that some consideration is aimed towards these accessibility issues. As well as it complies with WCAG guidelines to ensure that individuals with visual impairments can also use the CLI effectively.

2.3.3 CLI and complexity

CLI can be improved to enhance the user experience for people with disabilities. However, implementing these changes may require a shift towards a more graphical CLI or outright GUI, which could alter its operational methods. Cloud tools are typically designed to operate within a CLI environment, an environment that enables users to manage large amounts of data through specific commands. While a GUI may seem like a more user-friendly alternative, it may not be efficient enough to handle the scale and complexity of the cloud. To introduce a GUI into the CLI space, a clever design that can interact with data in a manner similar to a CLI would be necessary. This would entail exchanging the short commands and lack of visual presentations characteristic of a CLI with something that can provide a comparable user experience.

2.3.4 Capitalistic intentions behind cloud certifications and CLI superiority

When discussing the continued prevalence of CLI's as the main approach for cloud tools, it's worth considering the financial incentives at play. As the demand for cloud services grows, so does the need for expertise to maintain and operate them. For cloud developers, possessing specialized skills such as proficiency with complex CLI's can be a valuable asset in a competitive job market. As more tools and interfaces become user-friendly and accessible to a wider audience, there may be a decrease in the need for specialized skills to operate them. This shift could potentially impact the industry created around learning and handling cloud tools, which has become a significant market in its own right. It's also important to note that cloud providers offer their own certifications and courses to developers[7], often at a steep cost. This raises whether there is a financial incentive for cloud providers to keep certain aspects of cloud operations more complex and difficult to operate, in order to justify the cost of these certifications and courses. While there may be legitimate arguments for the efficiency of CLI's in certain situations, the continued use of these tools in the face of advancing technology and user-friendly interfaces raises questions about the motivations behind their prevalence.

2.3.5 Is CLI the best way to go?

Using CLI's still remains the main method of how the cloud is operated. CLI's are now so integral part of the cloud systems that introducing changes to their underlying design will be challenging. It can be argued that the CLI's straightforward design works well for its intended purposes, but there is a need to improve its accessibility, given that the format might be considered outdated. Emulators can be used to make the user experience better suited for those who need it, but their usage is not prevalent enough to make a significant impact. However, as more diversity in users enter the cloud space, there is a need to explore the way to interact with the cloud. Ultimately, it is important to strike a balance between accessibility and efficiency when developing cloud tools.

2.4 Summary

How the average user interacts with the cloud has evolved tremendously in the last decade. Every major cloud provider now offers some sort of graphical user interface for all their services, and its usage has been simplified to take on a broader audience of developers. It's a shift that has made the cloud more accessible, and it emphasizes how important it has become to the industry. Noticeable changes can be seen from how AWS has changed from its interaction beginnings, to now having transformed itself into a service with broad accessibility. The dominance of CLI's in the cloud space, however, has remained largely unchallenged, with few efforts to introduce alternative methods of interaction. CLI's have been found to present significant barriers to developers with visual impairments. To address this issue, tools like emulators have been developed to try improving the CLI experience, without much luck. While some argue that the complexity of the cloud and profit motives justify the continued use of CLI's, there should be a push towards investigating alternative methods of interacting with cloud infrastructure that are inclusive and accessible to all.

3 Reducing complexity with VR

The concept of reducing complexity is a critical aspect of creating user-friendly experiences. The primary objective is to make interactions as intuitive and straightforward as possible. This concept is implemented in various forms, and examining other examples can provide a better understanding of how it's going to be applied in virtual reality.

Johan Finstadsveen's master thesis, "If Your Webserver Was an Animal, How Would You Describe It?"[4] presents a unique approach to simplifying automated processes in virtual machines. By integrating principles from biology and computer science, Finstadsveen shows how animal behaviorism can be leveraged to reduce complexity in the management of virtual machines. This interdisciplinary approach enables the use of natural metaphors and terminologies to create a relatable and understandable connection with users, making difficult subjects easier to comprehend.

In Gaute Borgenholt's paper, "Audition: A DevOps-Oriented Service Optimization and Testing Framework for Cloud Environments,"[1] showcases another example of how complexity can be reduced. Borgenholt presents a creative approach to simplifying automated testing and quality assurance in cloud environments. Drawing inspiration from theater auditions, Borgenholt creates a model in which cloud resources present themselves as the best fit for a particular task, similar to actors auditioning for a role. This metaphorical approach makes it easier for users to comprehend the testing process and helps reduce complexity. This simple yet effective approach exemplifies how the use of metaphors can be a powerful tool in making complex concepts more understandable to users.

As demonstrated by the aforementioned examples, there are various approaches to reducing complexity. The main objective is to enhance the user experience, and achieving this goal can be done through abstract methods. The use of virtual reality to reduce complexity opens up possibilities beyond what was previously imaginable. This technology can be utilized to create landscapes tailored to serve any purpose, immersing users in three-dimensional environments that facilitate a deeper level of understanding. Virtual reality is transforming the way users can interact with technology, but how can this innovative tool be leveraged to simplify complex cloud environments?

3.1 History of VR

The concept of virtual reality has been around for quite some time, with its origins dating back to 1883 when Charles Wheatstone aimed to demonstrate how the brain processes two-dimensional images on each eye and converts them into a single three-dimensional image. Wheatstone accomplished this by creating the stereoscope, which presented a stereoscopic image consisting of two images side by side. This arrangement gave users a sense of depth and immersion when viewing the picture. The technology behind the stereoscope served as an inspiration for devices such as Google Cardboard and the more advanced Oculus Quest headsets.

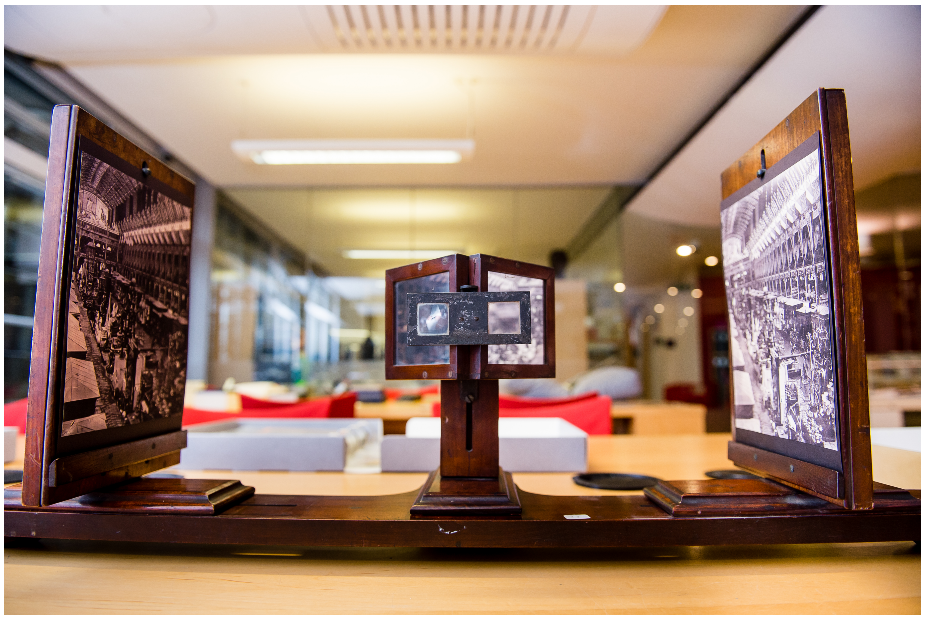


Figure 4: Wheatstone's stereoscope.
Source: King's College London

The next step in immersing virtual technology came in 1929 with the work of Edward Link. He understood the benefits of immersing users in a situation that simulated real life, and he added movement to the concept of virtual reality, even though he didn't use the concept of concealing the user's eyes within a virtual landscape. Link's invention, the "Link trainer," was the first rendition of a flight simulator. It was an electromechanical device that recreated the cockpit of a plane, adding motor function to simulate pitching and rolling, as well as turbulence and other disturbances. The link trainer was intended to train pilots in the US military in a safer environment, allowing them to be prepared before entering any real aircraft and understand the forces behind it and what it would entail. Although the link trainer was not a virtual reality headset, it paved the way for simulating reality and demonstrated that adding dynamics like movement to simulation could immerse the user even more, making something believable. This is crucial in today's VR, where movement and the response to that movement are what make something believable. Moving away from still images and adding more sensory experiences that users can experience through their usage is what makes VR what it is today.

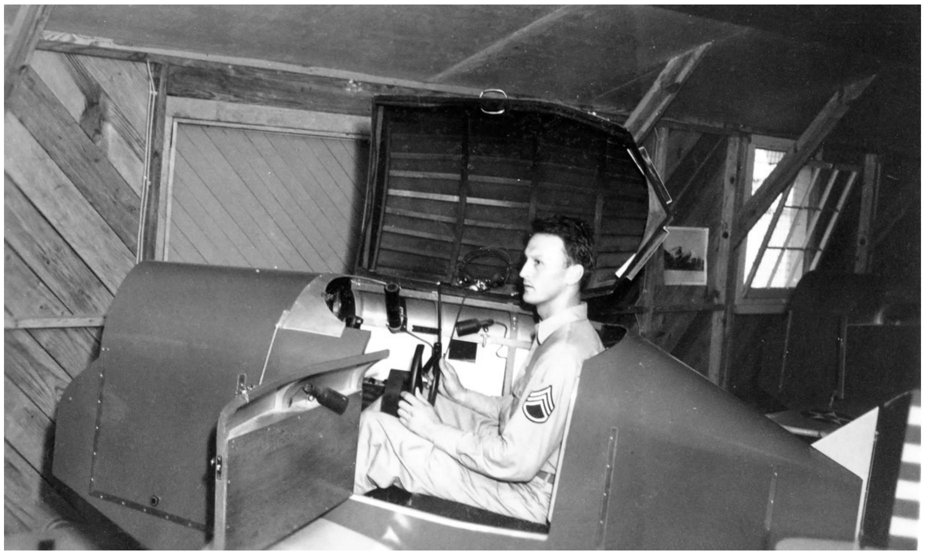


Figure 5: Soldier sitting in the "Link Trainer".
Source: Britannica

The link trainer was a significant step towards immersive experiences, but it was not until the release of the science fiction story Pygmalion's Spectacles in 1935 that the idea of virtual reality known today was first presented. The story, written by Stanley Grauman Weinbaum, features a character named Dan Burke who meets a professor named Albert Ludwig. Ludwig introduces Burke to a pair of goggles that enable the wearer to be completely immersed in a movie, with all senses covered, including sight, sound, touch, taste, and smell. The wearer becomes the main character in the story, interacting with other characters which can respond to the user's speech. The story becomes of the wearer, and it revolves around them. The main concept of being able to enter and interact with a story as the main character, is similar to modern-day VR. Pygmalion's Spectacles is an example of how fiction can inspire and shape the development of technology.

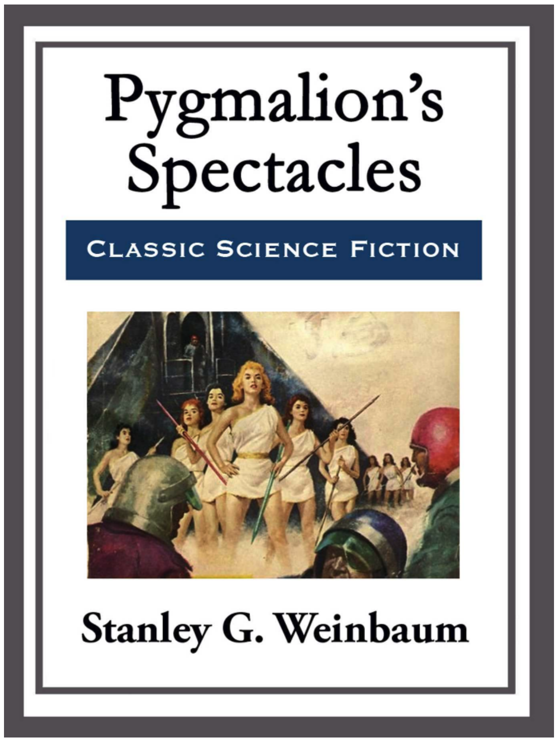


Figure 6: The cover art for Pygmalion's Spectacles.
Source: History of information

From the fiction that inspired it, to the goal of inventing new uses for VR, interest in this technology continued to grow in the following years. Individuals, governments, and companies all tried their ideas on what VR could be and how it could be utilized. From flight simulations and interactive movies to NASA using it to train astronauts, various methods of incorporating VR have been tried numerous times. Despite this, success has been mixed, and the full potential of VR was yet to be fully realized.

The gaming industry was the next to make an attempt at breaking into the VR space. In 1993, at the Consumer Electronics Show, Sega announced the prototype of a headset that wrapped around the user's head, featuring a built-in LCD screen and stereo sounds, along with head tracking capabilities to enhance the gaming experience. Sega had already developed four games for this prototype, intending to release it to the market. However, technical difficulties during development posed a challenge, and Sega was unable to bring a fully functional set to market. The company did eventually release a headset, the Sega VR-1, in 1994, but it was an arcade motion simulator that moved in accordance with the game and was not the same as the original intended prototype.

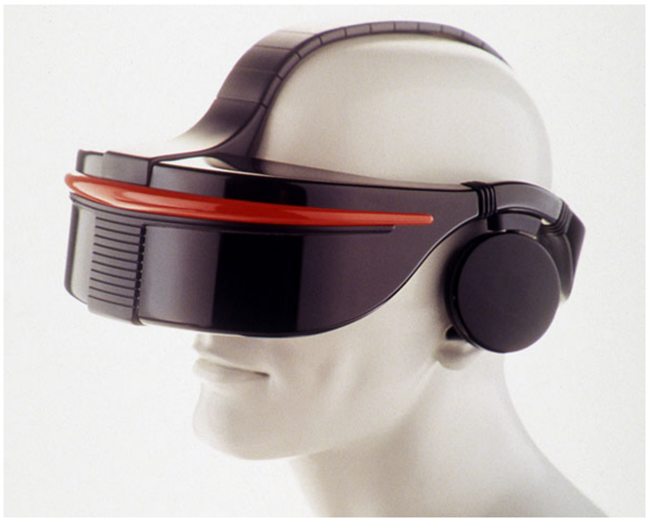


Figure 7: The Sega VR prototype.
Source: Segaretro

The gaming industry continued to experiment with VR technology, but without much success. For example, the Nintendo Virtual Boy was released in 1995 as a portable console that displayed 3D graphics, but it only rendered games in red and black and had a lack of game support. It wasn't until 2014 when Oculus was acquired by Facebook that VR saw its Renaissance, with other companies taking notice and jumping on the VR bandwagon. Headsets like the Rift, HTC Vive, and Oculus Quest were introduced to the market, offering unparalleled usability and opportunities never before seen with VR. The success of these headsets made the technology more popular, and other industries beyond gaming began to see the possibilities that VR could offer. New technology like Mixed reality is also to be explored and incorporated with headsets that can test new boundaries for what is possible with VR, and the VR space itself has only begun to adopt and create new methods of usage.

3.2 Components and technology behind VR

3.2.1 Headset and trackers

The VR headset is a sophisticated piece of technology that relies on a combination of sensors to create an immersive experience for the user. One key component of the VR headset is its trackers, which are responsible for translating the user's movements into data that is mimicked in the virtual world. Trackers can be either external or internal to the headset[6][19].

External trackers are seen used with earlier models of headsets, such as the HTC Vive and the Rift. The external trackers are small boxes, usually called base stations, that are placed at different angles in a room and communicate with the VR headset. They work by emitting a network of infrared beams that are caught by photoresistors built into the headset and controllers. By calculating the time, it takes for the beams to hit the headset and controllers, it's possible to determine the user's position and movements in real-time.

Internal trackers, on the other hand, use cameras built into the headset to track the user's position and movements. The cameras map out the surrounding environment and use this information to track the user's movements relative to the environment. This technology is used in newer VR headsets such as the Oculus Quest, making VR more portable and eliminating the need for external units. However, the downside is that the headset requires sufficient lighting, as the cameras rely on visuals to operate. Additionally, if the controllers are out of view of the headset cameras, they will not function.

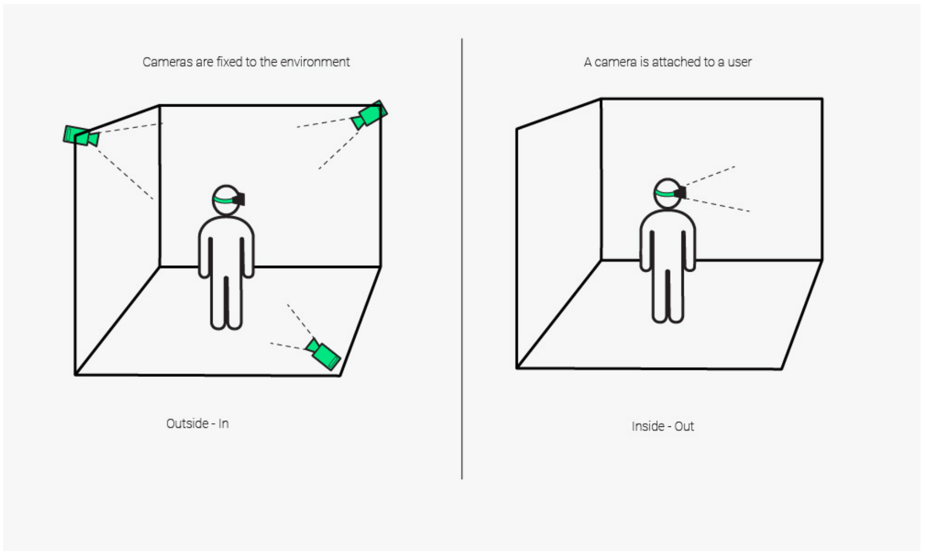

Cameras are fixed to the environment
A camera is attached to a user
Outside - In
Inside - Out

Figure 8: A comparison between external and internal trackers.
Source: VR-expert

3.2.2 Screen and lenses

The biggest factor for what makes VR unique is its lenses. VR headsets use embedded stereoscopic lenses that are positioned between a built-in LED screen. They work by utilizing two lenses, one for each eye, aimed at the LED screen which displays images for each lens. The construction of the lenses distorts the images from the LED screen in a way that creates the illusion of three dimensions. Using the built-in sensors, Six degrees of Freedom (6DoF) is achieved which allows the users to move their head in any direction and the screen will compensate, moving in conjunction with the user's movements.

3.3 Using VR for utility

Virtual reality has for the most part been considered an entertainment tool used for playing games or socializing. While this was true in the beginning, VR has recently started being utilized in more industries, bridging the real world with virtual landscapes. As more industries recognize the potential of VR, they are beginning to integrate it into their operations and processes to alter the way users interact with both real and virtual worlds. The integration of VR can be observed across various fields such as manufacturing, 3D design, and especially medicinal surgeries. It facilitates a more profound understanding of complex systems and offers unique perspectives to already complicated situations. Industries are increasingly adopting this technology because of what it can contribute to the user experience. One being a multitude of possibilities for enhancing knowledge and the reduction of complexity, making it a valuable tool in a wide range of industries.

In the industry of cloud infrastructure, dealing with complexity is inevitable. The sheer number of computing nodes, tools, and data management required to maintain a system as vast as the cloud can be overwhelming. Moreover, the fact that interacting with the cloud is often done through CLI's, without much visualization, only adds to the complexity. In such a scenario, new formats can be a breath of fresh air to tackle these challenges. The idea of using VR as a cloud management tool is relatively new, and there is some skepticism about its effectiveness for such tasks. However, with the rapid expansion of cloud technology, VR has the potential to become a necessary tool for understanding the cloud through the use of visualization. While it is true that VR may not be an effective tool for cloud management in its current form, it offers several advantages that could make it a valuable tool in the future. For instance, VR can provide a more immersive and interactive experience, enabling users to explore complex cloud environments in a more intuitive way. The creative opportunities offered by VR can help developers to design and test new cloud-based solutions more efficiently.

The use of VR technology to simplify complex tasks has a wide range of applications, and the methods employed can vary depending on the specific task. In this section, the aim is to explore two scenarios in which VR can be utilized as a utility tool. The first scenario involves the use of VR simulations, which is widely promoted by various industries. This method creates a one-to-one scenario in a virtual landscape, allowing users to perform dangerous tasks or interact with objects that do not yet exist. For example, in designing a vehicle and testing it in various conditions or running a training simulator for performing heart surgery. With VR simulations, users can engage in activities without any concern for theirs or others' safety, as they are in a controlled virtual environment. The second method of VR interaction takes a step beyond reality. This format involves the incorporation of real data into a virtual world that goes beyond reality. This approach provides the user with the freedom to interact with data in a way that is not constrained by physical laws and perspective. Users can create worlds that may not adhere to one-to-one scale and have complete control over how interactions occur in the virtual environment.

3.3.1 VR as a simulation

VR is often utilized by industries to simulate events that would be difficult to perform in real life. In these simulations, the user experiences the virtual world from a first-person perspective and moves around the environment as they would in the physical world. The objective of VR simulations is to replicate real-life scenarios to such a high degree of accuracy that it becomes indistinguishable from the physical world. This approach is often used by industries to create training platforms that allow users to train on tasks that might be too dangerous or expensive to perform in the real world. By providing a safe and controlled environment, users can perform tasks and gain valuable experience without any risks to their safety.

Realistic

Realism is a crucial factor in creating VR experiences that are believable and trustworthy. When designing VR simulations, the goal is to replicate the real world as accurately as possible to create a sense of immersion for the user. In the case of training platforms, VR has seen significant use in the healthcare sector, where various companies are developing solutions for doctors, nurses, and even paramedics to handle emergencies. One such company is Osso VR[22], which is dedicated to creating a training platform specifically for surgeons to practice a range of surgical procedures, from the easiest to the hardest. Their claim is that their platform can increase the experience of a surgeon and strengthen their skills. A claim that requires a product so realistic that it is indistinguishable from real life. Osso VR is an example of a simulation tool that uses VR technology to provide training experiences that are as realistic as possible. Their focus on realism is critical to the success of their product, as the user must be able to trust the VR environment completely. By creating a simulation that is indistinguishable from real life, Osso VR has created a tool that can significantly enhance the training experience for surgeons.

Safety

VR training platforms are often suggested as being highly effective for promoting safety in a range of industries. One of the key benefits of VR is that it allows users to be placed in scenarios where there is no danger of harm to themselves or others, providing an opportunity to practice tasks as many times as needed without any risk of dangerous failure. Industries such as construction, chemical processing, and others that deal with high-risk situations, greatly benefit from these VR tools. By training employees using VR before they face real-world situations, industries can ensure that their workers are better prepared to handle challenges safely. Additionally, because VR can offer a high degree of realism, employees can experience and learn from situations that closely resemble those they may encounter in real life. This creates an environment in which employees can gain a more profound understanding of potential hazards without compromising the elements that make real-world situations so complex and challenging.

As complex

While virtual simulations offer many opportunities that are not possible in real life, they are still bound by their own rules and limitations, which can hinder creativity. Although simulations are intended to mimic real-life scenarios, their complexity remains an issue, as users still need knowledge about the subject in order to successfully navigate them. Simulations are not there to make a situation easier by resolving it for the user, as that would defeat the purpose of the simulation. Users are limited to the same perspective and movements as before, which can be restrictive if it limits their learning potential. While simulations offer benefits in terms of providing a close approximation of reality, they are still a reimagined version of the real world, and complexity will inevitably follow as well. While simulations can be useful for helping users learn to navigate complex situations, they do not necessarily reduce complexity as a whole. In conclusion, while simulations are useful tools for learning and training, they have limitations that must be considered. They can be effective for preparing users for real-world scenarios, but they are not there for reducing complexity or increasing creativity. Ultimately, their value lies in their ability to provide a safe and controlled environment for users to practice and develop their skills.

3.3.2 VR beyond reality

Interacting with real systems through VR can go beyond a first-person simulated view. There are many methods of operating in VR, and its virtual environment can encourage experimentation beyond what is realistic. While the aim of VR simulation is to mimic real-life counterparts, this approach can limit opportunities to creatively test boundaries. Interacting with a simulation still requires complexity to maintain an accurate representation of what is being simulated. This method of VR usage is effective for helping people understand how something functions, but it doesn't necessarily reduce the overall understanding of the complexity of systems. To specifically reduce complexity, a different approach is needed. VR can be used creatively to interact with data in unique ways. Through its immersive capabilities, VR allows for developers to showcase whatever they can imagine. Real data can be projected into whatever surroundings and format that might be most beneficial for the situation at hand. This can provide a more intuitive and visually engaging approach to understanding complex data, making it easier for users to identify patterns and draw insights. This is what going beyond reality entails in VR, a real physical entity that is reimagined in VR to be whatever is needed to reduce its complexity and become more understandable.

Environment

The first benefit of going beyond reality is creative environments. When interacting with simulations in VR, the environment is typically a realistic representation of the object or system being studied. However, if the goal is to reduce complexity, the environment should be modified to better explain the concept. The key is to establish what needs to be simplified and determine the best environment to convey that simplification. For instance, a simulation of a network infrastructure with multiple routers, switches, and cables would be complex and difficult to understand. A better approach would be to represent each node as a cloud floating in the virtual space, making the infrastructure easier to grasp. Abstract representation is another effective way of simplifying complexity in VR. Abstract in its art form can be explained as "Abstract art is art that does not attempt to represent an accurate depiction of a visual reality but instead uses shapes, colors, forms and gestural marks to achieve its effect"[18]. By creating a virtual landscape that is not limited by realistic constraints, the possibilities for representation are endless. In particular, abstract art can be used as a model for simplification. Abstract art does not aim to represent a visual reality but instead uses colors, shapes, forms, and gestures to create an effect. In the same way, simple shapes and colors can be used in VR to convey vital information about a complex system. Humans tend to prefer visual imagery that is simple and colorful in nature. In creating an environment that is not bound by any coherent limitations that be in a simulation, imagination can be put in place to simplify complex systems.

Unnatural perspectives and movements

The second major advantage of VR that goes beyond reality is the ability to offer unique movement and perspective. Since the virtual environment has no physical limitations, users are not restricted to the same movements as in real life. This affords them greater freedom to explore the environment in novel ways. Instead of walking, they can fly, traversing the environment in all three dimensions. This opens up new opportunities for representing data omnidirectionally and utilizing the environment in more captivating ways.

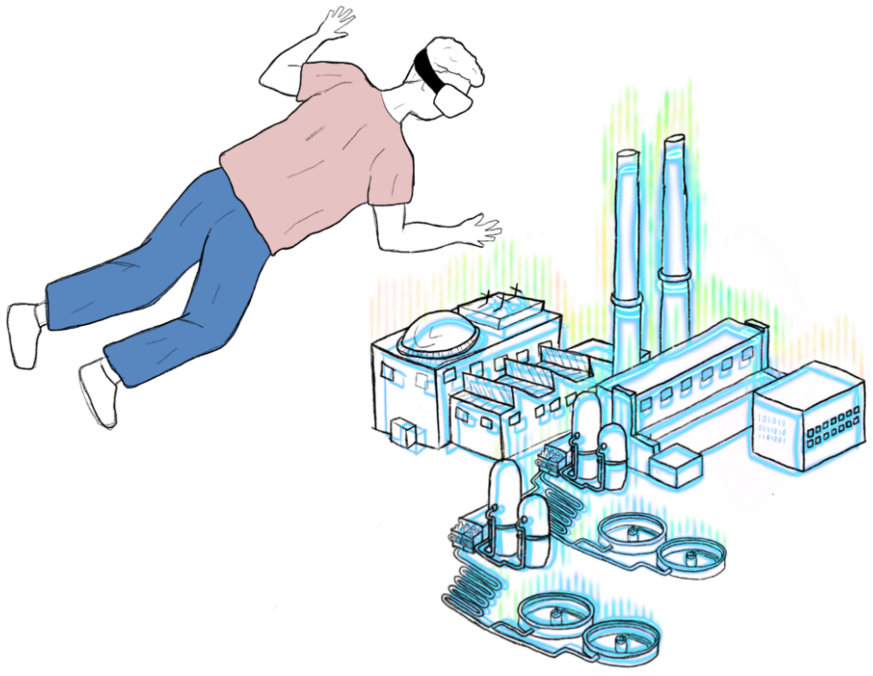


Figure 9: Man floating in VR and observing active data.
Illustration by Martine Mandt Brekne.

For example, imagine an industrial complex with numerous sensors scattered throughout a large area, producing data from all directions. Observing this data on a computer screen with graphs and numbers does not provide a sense of scale or the direct impact on the complex. However, in VR, the user can fly around the complex and observe the data being actively portrayed at the same place as the sensor. This enhances their understanding of how everything fits together and the flow of operations, simply by offering more freedom to move around.

Aside from movement, perspective is also something that can be reimagined in VR. Perspective does not need to be locked in first perspective, but can be whatever is necessary to convey information. Perspective is how the user views the world space that they are within. In most VR solutions, the perspective is in the form of a first person view, meaning that the eyes of the user are the eyes of the body in the virtual space. This creates the sense of immersion by putting you in the depth of the virtual realm, however, opportunities in perspective can go beyond that. Perspectives can be whatever beyond reasonable logic, there are no boundaries to what can be viewed.

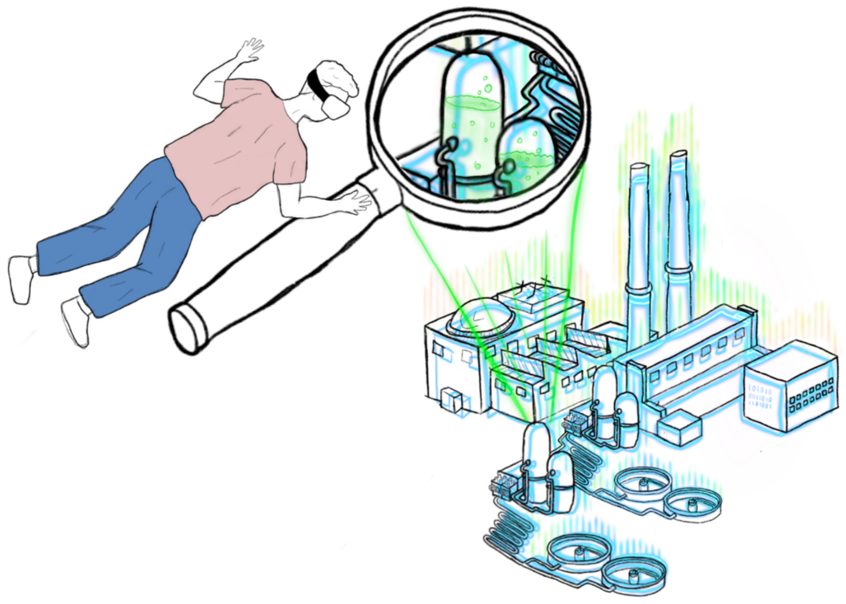


Figure 10: Man floating in VR while zooming in on a specific part.
Illustration by Martine Mandt Brekne.

Going back to the industrial complex example, with movement not having any restrictions, perspectives can be contorted to the same degree. Imagine that a user has gone to an area where a sensor is messaging about a faulty part, now to understand the issue, a better viewing angle is needed. A new perspective could, for example be, zooming into that part location and revealing the inner workings of the faulty mechanism. The users can form a detail oriented perspective, inspect the part in its entirety to understand where the sensor is complaining and quickly resolve the issue. The possibility to change perspective and viewing angles for users enables details to be revealed, which is difficult from a normal screen. It's about reducing the complexity of the infrastructure, by providing opportunities to break it into smaller pieces.

3.3.3 VR as a metaphor

“Metaphor is an expression, often found in literature, that describes a person or object by referring to something that is considered to have similar characteristics to that person or object.”- Cambridge Dictionary.

Metaphors play a vital role in helping users interact more intuitively with user interfaces[8]. A well-crafted UI metaphor conveys to the user what to do instantly, as it represents something that is universally known. In the digital world, where everything is represented in 1’s and 0’s, creating terminology that users can relate to can be a challenge. This is where the use of physical objects in a UI can be incredibly helpful. For example, the folder icon represents a place to store files and is easily relatable to an actual physical object. Similarly, the name “Windows” used by the popular operating system conveys the idea of looking through a window into the digital realm. These metaphors simplify the complexities of a computer and make UIs more understandable for humans.

Virtual Reality presents new possibilities for user interaction. Unlike traditional user interfaces, VR enables users to interact with digital environments in a more intuitive and natural way, using other methods like hand gestures, eye movement and voice control. However, since VR is a relatively new technology, it still lacks established metaphors to handle these new types of interaction. To address this issue, it’s important to be creative and explore new ways to represent the VR interfaces in a way that is easy to understand for users. One solution is to use metaphors that draw inspiration from the real world, just like in traditional user interfaces. For example, menus can be represented as every day furniture like shelves, closets, or other common household furniture. Cultural influences can also be incorporated to create more intuitive and engaging VR interactions. For example, instead of using a button to send emails in a typical email service, a stamp-based system can be implemented. In this system, users can put virtual stamps on items like images, objects, or text, and the address written on the stamp would be used to automatically send the item to the recipient. By leveraging these kinds of metaphors, it can help users understand how to interact with VR environments more easily, simplifying the complexities of this technology.

4 Approach

The approach section will detail the methodology used in this project. In the background chapter, the concept of cloud complexity and the use of CLI for cloud interaction, was discussed. From there it was established the current applications of VR and its potential as a new interface for the cloud. Based on this information, it's appropriate to address the problem statement: "Design and develop a VR interface in order to explore cloud interaction with regard to reduced complexity." This topic has not been extensively researched in literature, and it is therefore an interesting task to determine whether a VR application can be created to simplify the complexities of current cloud interaction.

4.1 Methodology

The methodology chosen for this project was not straightforward to determine, since neither a quantitative nor a qualitative method seemed suitable for the project's objectives and structure. Since specific data is not being gathered, projecting an answer based solely on data analysis would be challenging. However, if the project had taken either a quantitative or qualitative methodology approch, here are some examples of how that would have been done:

4.1.1 The quantitative approach

The quantitative method involves collecting and interpreting numerical data to derive results and analyze them to arrive at an answer or conclusion. In the context of this project, this method could be employed by conducting a focus group that would test a VR prototype. The test could involve each subject comparing the prototype with a dashboard console for cloud management, where they would be given a specific task, such as creating a VM. The time taken to complete the task on each interface would be recorded and analyzed, with the objective of determining which interface is more efficient in completing the task. The resulting data would be used to compare the two interfaces and to determine if the VR prototype is more effective than a dashboard in terms of intuitive design and usability.

4.1.2 The qualitative approach

The qualitative method is distinct from the quantitative method in that it centers on describing information that cannot be measured numerically or counted. This approach employs language to characterize items. In this project, a survey would be more suitable than a test, as it would allow the respondents to base their decisions on their user experience of the VR prototype and dashboard. The survey could include a few instances of interaction, and the test group could consist of both novice and intermediate cloud users. The resulting preference for design choices and intuitive actions with each interface could be used to establish a conclusive result.

While these methods of approach may produce a valid result that somewhat addresses the problem statement, gathering the necessary test group and organizing them within the constraints of this project's time and resources would be challenging. Moreover, the primary objective here is to showcase the potential of VR as an interface and experiment with a new perspective in the field, rather than arriving at a definitive conclusion. Therefore, an alternative approach is required to achieve the project's goal effectively.

4.1.3 The exploratory approach

Exploratory research is a methodology that aims to investigate research questions that have not been thoroughly examined before. This approach takes a more open-ended route to exploring how research should be conducted. When the collection of data is difficult, exploratory research can be a viable option when other methods fail. While there is limited literature exploring the idea of using VR to simplify complex processes, it is an essential question to address, given VR's increasing prominence in various industries. The beauty of exploratory research is that it doesn't follow a rigid framework like other methods do, and its conclusions do not depend on having predetermined results. Instead, the understanding of the final outcome can evolve throughout the project, allowing for a change in direction if new revelations emerge. This flexibility enables the possibility to pivot and adapt the approach as needed, resulting in a more comprehensive and insightful study.

4.2 Project outline

To begin the project, the first step is to create a technical map outlining how VR can be connected to the cloud. This involves determining the best tools, such as SDKs, to use for this purpose, as well as understanding the technical requirements and feasibility of integrating these two technologies.

Once the technical aspects are defined, the next step is to create a terminology that connects the worlds of cloud and VR. The goal here is to identify different cloud components and translate them into something that makes sense in VR. Having a well-defined set of terms will be important for creating a seamless user experience.

With the terminology established, the next step is to develop a set of use cases which can be performed in VR. These use cases are direct interactions that the user will perform in VR and will result in changes being made to the cloud, such as creating or deleting virtual machines.

Once the use cases are defined, it's time to build a prototype that incorporates each use case. The prototype should be thoroughly tested to ensure that it meets the project requirements. Then it's important to map out where the reduction of complexity is happening and discuss the overall process and future exploration areas.

4.2.1 Table of the project outline

To ensure that the project overview is understood, a table going over each phase is created. This table serves as a helpful guide throughout the exploratory research process, providing clarity and ensuring a satisfactory result that can be further elaborated upon in the discussion section. By using this table, it becomes easier to identify each phase of the project and ensure that all necessary steps are taken in order to achieve the desired outcome.

Phases	**Explanation**
1. SDK	Finding a compatible SDK that can work in conjunction with VR and justify that SDK choice through assessments.
2. Terminology	Bridge VR and cloud by building a terminology that defines the differences while also binding them together.
3. Use cases	Understand what the user should be able to do within the VR space to interact with the cloud.
4. Prototype	Build a prototype of the VR application, then explain the build process and the finished product.

Table 1: Project outline

5 Results

5.1 Cloud connectivity and its relation with VR

5.1.1 The availability of Software Developments Kits

What is a Software Development Kit?

A software development kit, or short, SDK, is a set of tools that enables developers to work efficiently with a specific platform. Typically, an SDK includes a framework and a set of code libraries that simplify the process of interacting with the platform, streamlining the development process. Without an SDK, developers would need to create from scratch the functionality necessary to interact with a service, which would take significant time and resources. With the pre-built functionality provided by SDKs, it allows developers to focus on building new features rather than reinventing the wheel. SDKs have simplified software development, making it more efficient and less time-consuming. One of the most widely used SDKs is the Java Development Kit (JDK), which is essential for building Java applications. The JDK includes an array of tools and libraries that work together to create runnable code that can be executed on any machine. The tools in the JDK can convert source code into a format that can be executed by the Java Runtime Environment (JRE), which is essential for running Java programs on any operating system without modifying it.

What is an API?

Every software development kit environment utilizes an application programming interface (API) to interact with services. Whether the API is used with an SDK or independently, its purpose is to enable communication between two platforms. APIs create a pathway between an application and a web server to perform a specific task. To accomplish this, an application sends a request to an API endpoint, which is typically the URL of a web server. The web server performs the requested task and sends a response back to the application. Endpoints are locations from which APIs can access the resources necessary to carry out their functions. Each endpoint performs specific tasks. One notable API is Google Maps, which enables third parties to incorporate Google's mapping technology into their applications. APIs facilitate the seamless transfer of complex functions between applications, without the need for extensive programming. In this way, APIs are a useful tool for organizations to share complex functions, to developers that require it without having the technical skills to perform it themselves.

SKD and language favoritism among cloud providers

SDKs and APIs are essential tools for cloud development, allowing applications to interact with cloud services seamlessly. Major cloud providers like Amazon, Google, and Microsoft offer SDKs for various programming languages, along with comprehensive documentation to guide developers. These SDKs have a broad scope, enabling most developers to interact with cloud services using their preferred language. However, some programming languages work better than others for specific tasks. For instance, in the context of VR, the available options for languages are limited. Additionally, language favoritism is not uncommon in these environments, with cloud providers promoting their preferred languages. For example, Microsoft promotes the use of .NET and C-sharp, while Google promotes the use of the Go language. This preference for specific programming languages can lead to limited support for other languages, potentially diminishing their usage. Although certain programming languages may be favored by cloud providers, it is worth noting that SDKs are still offered in a range of languages. While some languages with certain SDKs may work better than others, it is still possible to utilize all programming languages if the need arises.

Showcasing the simplicity of an SDK with python

To demonstrate the workings of an SDK, let's explore how Python can be used with the Amazon SDK. Python is widely known for its easy-to-understand syntax, making it an ideal language for demonstrating how an SDK works. Amazon offers different SDK formats depending on the language used, and for Python, the Amazon SDK is called Boto3. This SDK provides Python developers with the ability to write code that interacts with Amazon S3 and Amazon EC2 services. Amazon S3 is a cloud-based object storage service that can be used for various purposes, such as internet applications, backups, disaster recovery, and data archives. Amazon EC2, on the other hand, allows users to rent virtual machines or instances on the Amazon cloud to run different operating systems. By using Boto3 and Python, it's possible to write code to interact with these Amazon services, demonstrating the power and flexibility of using an SDK.

Boto3 use case

Boto3, with Python, can be simply downloaded within a Python environment or IDE. Once downloaded, all the necessary libraries are automatically configured and ready for use in a Python software. Every Amazon SDK utilizes something called AWS security credentials for making requests and accessing personal AWS data, these credentials have to be set up before being able to make calls to their services. These credentials are essential for making authorized requests to AWS services. After setting up the credentials, Boto3 can be imported within the Python code, and from there can access Amazon S3 and EC2 resources.

```
>>> import boto3
>>> s3 = boto3.resource('s3')
>>> for bucket in s3.buckets.all():
        print(bucket.name)
```

In the Python code snippet, Boto3 is showcased. Upon importing Boto3, a variable is created to collect S3 data resources from an AWS account. Since AWS credentials are already set up, accessing and retrieving resources from the service into the code is easy. With the data stored in a variable, a for loop can be used to write out each bucket or storage object from AWS. This example demonstrates the simplicity and ease of use of Python with this SDK, which provides access to vast amounts of data with just a few lines of code.

SDKs do not allow for the monitoring of systems

While SDKs are powerful tools that simplify the development process, it's important to note that they don't provide any monitoring methods. Monitoring CPU resource allocation, heat distribution, and detecting failed machines is essential for managing infrastructure and gaining insight into its performance. However, SDKs are designed to function within a cloud account and only have access to the services provided by the cloud provider. The provider controls the underlying system and infrastructure, which are confidential and part of a larger system. As such, it's understandable that normal users are not granted access to them. However, when building utility tools to simplify infrastructure management, such data becomes critical. Unfortunately, SDKs do not offer this type of access, and it's unlikely that they ever will since monitoring is the responsibility of the provider. This limitation hinders the ability of companies and organizations to fully understand how their system operates, as they only have access to the outer layer of services. Therefore, it's important to recognize the limitations of SDKs and their inability to provide access to vital monitoring data. Companies and organizations must seek alternative tools or collaborate with their cloud provider to gain insight into their system's performance.

5.1.2 VR is limited in language options

VR technology has found its primary use in the gaming industry, where it has become a popular tool for creating immersive and interactive gaming experiences. However, it's an industry which is constricted by the limitation of what programming languages they can use. As they can only utilize a selected few. C# and C++ are two of the programming languages used in game development due to their speed and efficiency. Factors, which are essential for handling the fast processing required for video games. As a result, the two largest game engines, Unity and Unreal Engine, rely on these languages for their operations.However, VR and its close association with the gaming industry presents a challenge for combining VR and the cloud. VR still relies on game engines to create new software, as there are no other foundations to build it on. This limits the range of programming languages that can be used in VR development, and restricts it to the use of two languages. This limits the potential of integrating VR with cloud technologies. Therefore, the combination of VR and cloud computing, along with the reliance on only two main programming languages, C# and C++, can make VR development a more limited endeavor. To effectively work with the cloud, it's important to carefully assess the capabilities and limitations of these languages and decide which one is best suited for this specific project.

To determine the best programming language for this VR project, it's important to investigate the opportunities provided by different game engines. Unity, for instance, relies on C# as its main language, while Unreal Engine uses C++. In order to make an informed decision, it's crucial to assess the capabilities and limitations of each language and determine which one is best suited for this project. One way to start this assessment is by evaluating the SDK availability for each language across major cloud providers such as Amazon AWS, Google Cloud, and Microsoft Azure. Cross-checking these providers can help identify whether any SDK is available for a specific language, and to what extent it can be utilized to support the project goals. Through this assessment, a more comprehensive understanding of each language's workings, as well as its accessibility and compatibility with different cloud services, can be gained. By thoroughly evaluating each language in this way, it's possible to choose the most appropriate language for this project, one that is best suited to the goals and requirements. Overall, the assessment of the SDKs across different cloud providers is a critical step in the decision-making process of selecting the best programming language for this VR project.

	C-Sharp	C++
Microsoft Azure	Supported SDK	Supported SDK
Google Cloud	Supported SDK	Supported SDK
Amazon AWS	Supported SDK	Supported SDK

The assessment aims to evaluate the level of SDK support provided by different cloud providers for two specific programming languages. Although each provider offers an SDK for both languages, the assessment reveals significant differences in the quality of their implementations. To assess the quality of the implementation, it is crucial to consider certain checkpoints that will constitute a good SDK.

The checkpoints

1. SDK presentation: Examine how the SDK is presented and whether the download process is intuitive and straightforward and the libraries are easy to obtain.

2. Documentation: Evaluate the quality of the documentation to ensure it explains the core concepts of the SDK's functionality. And to see if the documentation is centralized on the provider's website and not scattered around to the point of potentially confusing the user.

Explanation of each checkpoint

SDK presentation: refers to the steps required to download the SDK. It is essential to assess how user-friendly the process is and if the provider keeps the download within their domain or if they redirect the user to another website. The provider should simplify the process and make it quick and straightforward for the user.

Documentation: is critical for utilizing an SDK, and its quality can significantly impact the development experience. Therefore, it is important to evaluate whether the documentation provided by the provider sufficiently explains how to use the SDK effectively. The documentation should also cover the core concepts and functionality of the SDK. Moreover, having the documentation on providers' website, accessible and categorized is also a factor that needs to be considered. If the documentation is not found on their website and instead redirects the user elsewhere, it will count as a negative.

This checklist will evaluate how each cloud provider handles their SDK for each language. By analyzing the performance of the SDK in different areas, it's doable to determine the most convenient SDK for combining cloud and VR technology. Ultimately, the SDK that meets the criteria and performs the best in these areas will be considered the most convenient option.

	C++	C#
Microsoft Azure SDK	🟧 SDK presentation	🟩 SDK presentation
	🟥 Documentation	🟩 Documentation
Amazon AWS SDK	🟧 SDK presentation	🟩 SDK presentation
	🟩 Documentation	🟩 Documentation
Google Cloud SDK	🟥 SDK presentation	🟧 SDK presentation
	🟥 Documentation	🟧 Documentation

Table 5.1.2 uses three colors to indicate how well each SDK meets the criteria outlined in the checklist for each language. A green color indicates that the SDK meets the requirements and is considered good. An orange color suggests that the SDK is somewhat good, but may need improvement in certain areas. A red color indicates that the SDK does not meet the requirements and is not considered good.

C++:

Microsoft Azure SDK

When it comes to obtaining the SDK for C++, it's worth noting that the Microsoft website doesn't offer a direct download option. Instead, users are redirected to a GitHub repository through a hyperlink provided on the Azure SDK website. While this approach may seem counterintuitive and inconvenient, it is the current protocol for obtaining the SDK. This method creates an additional step for users looking to integrate the SDK into their software. Ideally, a direct download option on the Microsoft website would streamline the process and make it easier for users to access the SDK.

Documentation is not presented on the Azure SDK website. Instead, users are directed to the Azure C++ GitHub repository to access the README file for documentation. While this approach is a viable option, it may not be the most efficient or user-friendly method for SDK documentation. One downside of using the README file as the primary source of documentation is that it's static and doesn't offer advanced search capabilities or have any organized categories. This can make it difficult for users to quickly find the information they need or navigate through the documentation efficiently.

Amazon AWS SDK

Similarly to Azure, the AWS SDK website also redirects users to a GitHub repository for downloading the C++ SDK. While this method is functional, it does again add additional steps for users to obtain the necessary libraries. Ideally, the SDK should be directly accessible for download on the AWS website. Having a direct download option on the AWS website would eliminate the need for users to navigate through GitHub to obtain the SDK.

When it comes to documentation, AWS offers a comprehensive and well-organized solution directly on their website. The documentation is divided into categories and provides clear, guided steps for installation, implementation, and API reference. The categorization of the documentation also makes it easy to understand how each aspect of the SDK works, providing a thorough understanding of the SDK's capabilities.

Google Cloud SDK

Navigating the Google Cloud SDK website to find the C++ SDK turned out to be a challenging task. Unlike other cloud providers, there are no direct links to a C++ library on their official website. As a result, users are forced to search outside the official website to find the necessary information. The C++ library for Google Cloud comes in the form of a GitHub repository. However, unlike other cloud providers, Google does not present a GitHub link on their website, which may cause confusion for users and create an unfavorable user experience.

Unfortunately, the SDK website lacks documentation and fails to provide any indication of where to locate it. Users are left to search for the documentation outside the official website, which can be a frustrating experience. The only documentation that can be found is in the form of a README file on the GitHub repository, which is not categorized and lacks detailed information about implementation or API referencing.

C#

Microsoft Azure SDK

The Azure website provides a well-organized and easily accessible presentation of the C# SDK. As a Microsoft platform language, C# is prominently featured on the website, with detailed explanations and numerous resources. Direct links to download the libraries and access to their GitHub repository are readily available. Each library is also thoroughly explained, with various options for downloading them.

The Azure website offers comprehensive and easily accessible documentation for C# on their platform. From installation to implementation, and API referencing, all the necessary information is readily available. The documentation is rich with details, and each library has its own dedicated section, complete with explanations of its functionalities. The website's categorization and search capabilities make it easy to find the exact information needed, with all content thoughtfully organized for optimal accessibility.

Amazon AWS SDK

The C# SDK on AWS offers multiple options for downloading and direct access through Visual Studio IDE. The options are well-structured and provide clear instructions on how to implement the SDK into software. This flexibility in accessing the SDK makes it easier for developers to work with AWS services and integrate them into software solutions.

The AWS website provides excellent documentation that is full of information on the SDK. Each category is clearly laid out and presented with numerous coding examples, making it easy for new users to get started with the SDK. While there is a lot of information, it is presented in a structured and organized manner.

Google Cloud SDK

Google's page for the functionality of the C# SDK is clear and well-presented. However, accessing libraries for direct download can be a bit challenging. Some elements of the official website may be difficult to understand, but overall, the C# SDK website is more user-friendly compared to the C++ website.

The documentation provided by Google Cloud SDK is abundant and informative for each library. However, installation and implementation of the SDK can be challenging without in-depth exploration of the source material. While the documentation is helpful, there may be a need to spend extra time researching and troubleshooting to successfully work with the SDK.

5.1.3 The final choice for SDK and programming language

After assessing the benefits and drawbacks of various SDKs and programming languages, it was determined that some options work better than others. The assessment revealed that all major cloud providers, including Microsoft Azure, Amazon AWS, and Google Cloud, offer SDKs for both C# and C++. This is essential for interacting with Unity or Unreal game engines. However, it was necessary to further investigate which SDK works better with which language. The assessment showed that C++ had the worst SDK support across each cloud provider, and the documentation was lackluster, requiring outsourcing from other sources. On the other hand, Amazon AWS offered the best solution for SDK and documentation across the three providers, with Microsoft in second place and Google in third. Unreal Engine uses C++ for programming, which the assessment found to be the most challenging programming language to use due to the limited availability of SDK quality.

The second option was using SDK with C#, which is used by the Unity game engine for programming. Based on the assessment, C# had better support and documentation across each major provider. However, Google Cloud had the worst SDK support. Interestingly, Microsoft Azure and Amazon AWS had equally good SDK support and documentation for both languages. C# is a Microsoft native language, which explains its good support. However, Amazon

AWS had a higher score across the board with better documentation for both languages, and the setup for SDK was found to be easier. Based on the assessment, the conclusion is that C# with Amazon AWS SDK is the better option, making the Unity engine the preferred method for creating the solution in this project.

5.1.4 Unity as a game engine

Unity is a powerful, cross-platform game engine that's designed to create both 2D and 3D games, but it's not just limited to gaming. In fact, it's becoming increasingly popular in various fields, such as architecture, automotive modeling, medical training, education, and simulation training[17]. One of Unity's strengths is its versatility, which allows it to be used for a wide range of applications, including VR. Using presets, Unity provides developers with the necessary tools to create VR applications with ease.

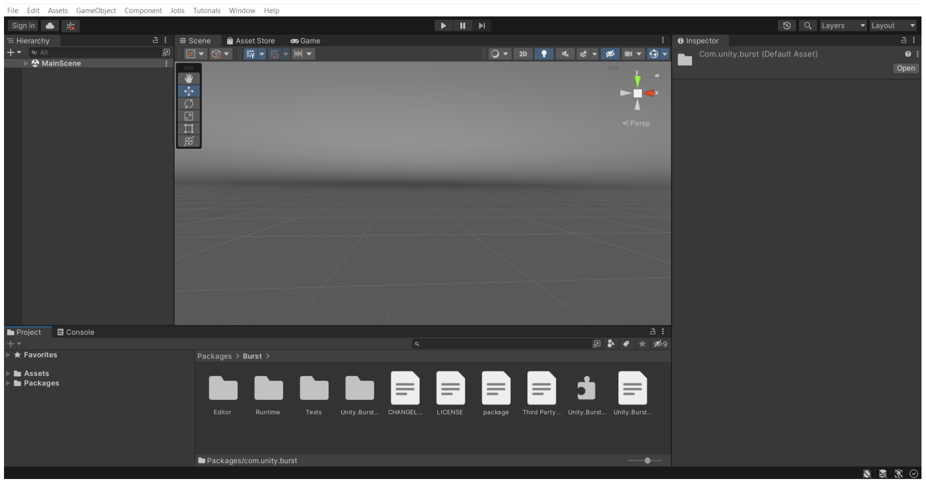


Figure 11: View of the main screen in the Unity editor.

Unity provides a comprehensive solution for game development, with many aspects of development being automatically set up. Assets can be downloaded and imported into the project, making Unity an efficient tool that is claimed to minimize unnecessary tasks already performed by the platform. Unity's primary scripting API is based on C#, allowing code to be added to individual game objects within the project to act as different components in the game's structure. This approach of writing individual scripts for each game object improves code manageability. While the code may be lengthy, it is segmented into different game objects, reducing confusion and facilitating better organization.

Scripts to game objects

In order to showcase the structure of Unity, it's important to show the most crucial part of it, which is attaching code script to game objects. Game objects are the physical models within an application that a user can interact with, like interior objects like wall floor and objects that can be lifted like swords or torches. Each game object in itself cannot do anything unless code is attached to it, this is where code scripts come in and can be attached to game objects to make them do actions.

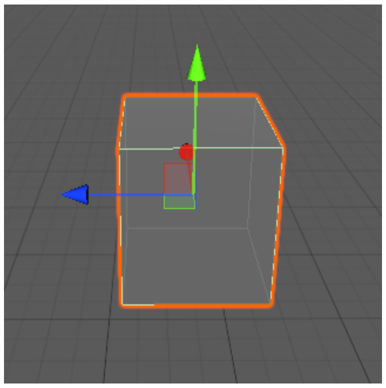


Figure 12: 3D game object of a cube.

Figure 12 depicts a typical game object in Unity, represented as a cube in this example. To associate code with this game object, one can simply attach components to it. These components may consist of various functionalities such as scripts or attributes, allowing the game object to perform a diverse range of tasks.

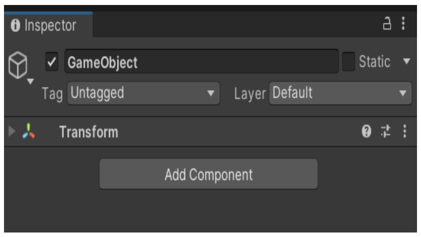


Figure 13: Before adding a script.

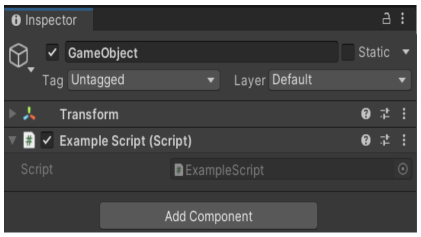


Figure 14: After adding a script.

Figure 13 depicts the component section of that game object, currently lacking any components. As of now, this game object is static and exists solely within Unity as an empty object. However, in Figure 14, a script file has been attached to the component section. This enables the game object to perform a task when the application is initiated. Therefore, by attaching the script file, the game object's functionality has been extended beyond its initial static state.

5.1.5 Unity networking

Connecting to a cloud service from Unity is a straightforward process that follows similar principles to connecting with any other service. Unity is a versatile solution that consolidates various technologies into a single framework. To integrate a Unity project with a cloud provider, developers can import the appropriate SDK into the project. Utilizing Unity's scripting API, developers can then incorporate libraries from the SDK and create code that enables communication with the cloud service, without interfering with the Unity framework or its associated applications.

5.1.6 The dilemma

After carefully considering the options, the decision was made to use the Amazon AWS SDK and C# as the programming language for this project. Unity was selected as the VR application solution, with C# as the scripting language. Unity's architecture allowed for seamless integration with the AWS SDK, resulting in a connection between VR and the cloud service. While it was discovered that communication between Unity and AWS was possible, the project's limitations had to be acknowledged. The primary objective was not just to showcase a standard AWS user's perspective, where VR is used as a replacement for the traditional AWS dashboard. Instead, the aim was to create a tool that caters to both the regular user and the system administrator. However, certain limitations arose due to the Amazon SDK's inability to provide relevant metrics from the API, which is outside the SDK's scope. This poses a significant challenge when designing a tool for the system administrator, as having access to system performance metrics is crucial.

The absence of a way to collect metrics from AWS servers or any other cloud presents a significant dilemma. Should this hinder the exploration of using VR as an administrator in the cloud? While it may seem like a roadblock, it is essential to understand the technical limitations of the SDK and accept the fact that Amazon does not offer any possible metrics for now. However, in order to continue exploring the potential of VR in this context, it is necessary to look beyond the technical limitations and focus on the possibilities within the VR space. If in the future, Amazon offers an SDK for metrics, this project's full potential can be realized. However, for now, finding alternative ways to showcase the possibility of combining the system administrator with VR in interacting with the cloud should not be let out. Since no actual data from Amazon will be used, creating a simulated environment to emulate the metrics of a server system seems like the good next option. This entails pretending that data flows in, and having parameters being modified within the VR application to simulate real metrics.

To summarize, while the Amazon SDK only supports regular cloud interaction, no actual metrics will be derived from Amazon for the administrator, and instead, the VR application will have an artificial environment to work with. This change will be reflected, and the aim is not to test technical feasibility, but to showcase what the application might look like if technical constraints didn't exist.

5.2 Bridging the gap between cloud and VR

Moving from the cloud to virtual reality involves a transformative process. The challenge lies in bridging the gap between the technicalities of buttons, icons, and text that are familiar on a two-dimensional screen and creating an immersive, dynamic, and lifelike experience in a virtual space. This requires a translation of familiar website elements into something that not only fits in VR, but also feels intuitive and natural. In other words, objects that are intuitive on a website may not necessarily translate well to VR, and therefore must be reimagined in a way that makes sense within the virtual environment.

It's important to recognize that the cloud has yet to be fully conceptualized in a 3D space, presenting a unique challenge for building interactive tools that accurately represent its functionality while also offering a fresh approach from traditional dashboards. To tackle this challenge, creative solutions must be developed that redefine the cloud environment without compromising its usability. This requires a certain level of imagination to bring all the pieces together, as there is no clear guide for navigating this uncharted territory. Above all, it's crucial to effectively be able to communicate in the 3D space without causing confusion for the users.

Originally planned as a terminology in the approach section, this definition had to change. It was not longer an actual terminology, more like a definition list of what the prototype should have in its final stage. With that in mind, this definition list seeks to identify potential points of connection between cloud technology and VR. It explores various aspects of VR and examines how they may be integrated with the cloud. By examining established VR definitions in relation to the cloud, can it be possible to identify areas where the two technologies intersect and determine whether such integration is feasible and practical? Understanding this helps build the criteria for what is required of the prototype.

5.2.1 Cloud users

Cloud users can be broadly categorized into two types: regular users and admin users. It is important to distinguish between the two as they have distinct roles and responsibilities within the cloud environment.

Regular users are the everyday users who perform standard tasks on the cloud, such as managing services. They typically use the cloud products and interfaces as intended, within the prescribed limits. These users are most familiar with the features and functions of the cloud service they are using. In the context of this project, regular users will be referred to as such and will represent one user group. They will have a user-oriented experience, and the virtual environment will need to reflect their needs.

Admin users are responsible for maintaining the entire infrastructure of a server network. They require a more in-depth understanding of the cloud architecture, tools, and services to manage and monitor the system effectively. Admin users have to work outside the boundaries of the cloud products and interfaces to solve complex issues and optimize the cloud environment. In the context of this project, admin users represent the other user group. To create a virtual world that caters to the needs of admin users, the development process must be more unconventional than that of the regular user. This is because the admin user's tasks are often complex and require specialized knowledge and tools.

5.2.2 Actions

The next important term to understand is "actions." Actions refer to the various interactions that each user can perform within their virtual world. When a user performs an action, it triggers a response, which causes something to happen on an actual cloud service. Actions thus serve as the link between the virtual world and the real world, allowing users to perform tasks and accomplish goals within the cloud environment. An example of an action is the act of pushing a button, which in turn opens a door.

5.2.3 The virtual world

Perspectives

The virtual world offers a unique experience to each user group, as they navigate it using different perspectives. These perspectives are shaped by various factors, which have been discussed in the background chapter of this project. It is crucial to reiterate the importance of perspectives in virtual reality interfaces, as they play a significant role in shaping the user experience. In essence, a *perspective* is the viewpoint from which the user perceives and interacts with the virtual landscape. It determines how they will orient themselves and navigate through the virtual world. The regular users and admin user can utilize different perspectives, which will be explored further in this paper.

Virtual environment

The term virtual environment refers to the visual representation of the world that a user will be immersed in. It encompasses various elements such as size, functionality, and aesthetics, which are crucial in creating a seamless connection between the cloud and virtual reality.

To fully understand how the virtual environment works, it is optimal to explore these factors in detail. The *size* of the virtual environment, for instance, can greatly affect the user experience. A smaller environment may be suitable for more focused tasks, while a larger one may be necessary for more complex scenarios.

Functionality is another critical aspect of the virtual environment. It refers to the various tools and features that are available to the user, such as interactive objects and communication channels. These elements help create a more immersive and engaging experience for the user.

Finally, the *aesthetics* of the virtual environment play a crucial role in creating a sense of immersion and presence. Factors such as lighting, texture, and color can greatly influence the user's perception of the environment and can make the experience more enjoyable and immersive.

5.3 Visualization approaches for VR and cloud

Visualization is key to how both regular and admin users will interact with cloud tools in a virtual landscape. This section aims to develop strategies for actualizing this landscape and determining a perspective for the different types of interaction. By consulting the established definition list and examining each user's unique needs, it's the goal to create an actual landscape that is tailored to the individual user's requirements. The landscape should be a meaningful representation of the user's needs and preferences, and that it effectively facilitates interaction with the cloud tools.

How can an optimal virtual environment for cloud management be created? It's crucial to consider the needs of two distinct user types and design settings that are most appropriate for each. Specifically, it's important to take into account the requirements of an admin user and a regular user, as defined by the definition list. The regular user typically performs basic tasks on the AWS dashboard, like creating or deleting VMs. While the admin user is responsible for monitoring and maintaining the entire infrastructure, with responsibilities that go beyond those of the regular user. To ensure the best approch for visualization, it's important to reflect these divergent tasks in the design.

5.3.1 Perspectives

Based on the definition list, the virtual world is highly driven by the perspective of the user that experiences it. The perspective can be tailored to be anything, and it's here most effort can put towards simplifying processes within the virtual world. With perspectives, it's about finding how each user will experience the virtual world to optimize the tasks that are needed to be performed. There are two perspectives that are more interesting to explore, the machinist, and god perspective. The machinist perspective is the perspective of a regular user, with the aim of showcasing the cloud dashboard in a new way by giving the user more control over the environment for easily displaying and handling data. The god-perspective is that of the admin user, since the goal is about monitoring, by giving the admin user the perspective of an all-seeing deity seems appropriate. The admin user has in the sense an all-watching eye over the infrastructure.

The machinist perspective

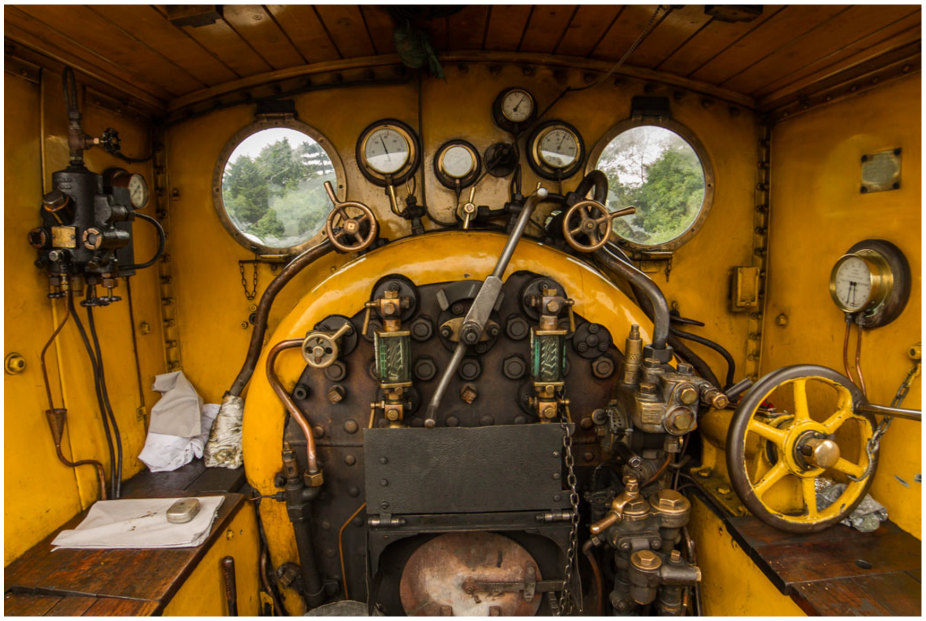


Figure 15: Inside the cabin of a steam train.
Source: NNrailway

To explain the machinist's perspective, let us use an analogy from the world of train conductors. Imagine a train conductor operating an old coal-driven steam train. The cab is equipped with a multitude of levers, gauges, and valves that the conductor needs to interact with to run the train smoothly. Each lever is designed to perform a specific task, and the placement and functionality of each lever is intuitive and easy to understand. For example, there is a lever to regulate the speed of the train, and a chain hanging from the ceiling to blow the whistle. Every mechanism in the cab has a purpose, and it's there to serve a specific function. The design is simple and intuitive, with each dial and lever placed precisely where it needs to be.

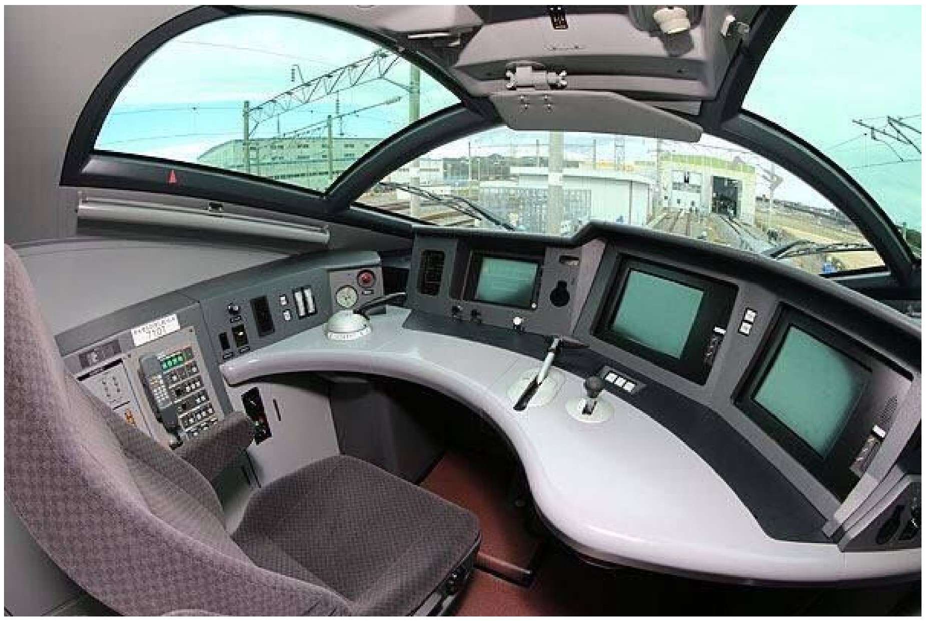


Figure 16: Inside the cabin of a bullet train.
Source: Pinterest

The difference between old steam trains and modern bullet trains like Japan's Shinkansen couldn't be more apparent. The Shinkansen is an incredible feat of modern engineering, capable of reaching speeds up to 320 km/h while claimed to provide unprecedented levels of comfort and convenience. However, this technological advancement has resulted in a loss of user interaction from the train conductor standpoint. In the past, operating an old steam train required a distinct lever for each operation, creating a tactile and immersive experience for the user. In contrast, the Shinkansen's operation has been modernized with screens and a single lever to control the entire train. While this approach may be minimal and efficient, it also leaves the user feeling somewhat disconnected from the technology. All the information and functionality is now confined to a single space, which can be less engaging than physically interacting with the train's various levers and controls of a steam train.

The romance of old steam trains lies in their simplicity and interactivity for users, despite lacking the advanced technology of modern trains. Unlike modern trains that often hide their technology behind touch screen interfaces, the steam train requires the user to physically move and interact with various aspects of the train in different ways, providing a more immersive experience. The machinist perspective in this project aims to recreate this sense of interactivity by combining cloud technology with physical objects that must be moved and manipulated in different ways. By incorporating physical objects, The machinist can provide a more engaging and dynamic experience that requires users to interact more with the environment rather than relying solely on a dashboard interface. This approach harken back to the days of old steam trains, where users could follow the train's full path and feel like they were part of the action.

The god perspective

The god perspective is distinct from the machinist perspective in that it offers a broader, more abstract view of an infrastructure. While the machinist perspective allows users to get hands-on with the virtual environment, the god perspective is intended for system administrators who need to monitor a large-scale infrastructure. In the god perspective, the role of the administrator is elevated to that of a non-physical being within the virtual world. Rather than being confined to a physical body, the administrator becomes a floating abstract entity with the power to observe changes within the virtual space. This comparison to a deity is intended to symbolize the power and control that administrators have over the system they manage. The god perspective allows administrators to understand the big picture of the system they are monitoring, and to explore its health and performance from any angle. This level of abstraction gives administrators the power to introduce changes to the system in a way that normal users cannot. Essentially, the god perspective glorifies the role of the administrator and empowers them to make informed decisions that keep the system running smoothly.

To better understand this term, let's consider the role of a fire watcher in a fire lookout tower. The tower is strategically located atop a mountain range, providing a panoramic view of the surrounding forest. From this vantage point, the fire watcher has a bird's-eye view of the entire area and can quickly detect any changes or signs of danger. For example, if smoke rises on the horizon, the fire watcher can immediately spot it and alert the fire department to prevent a forest fire. This same level of awareness and responsibility applies to an administrator's role in managing a system's infrastructure. Similar to the fire watcher, the administrator must constantly monitor the health of the system's servers and other components to ensure they are functioning optimally. If an issue arises, the administrator must be prepared to take action and call for backup to prevent the situation from escalating. Both roles require a vigilant eye and quick action to prevent potential disasters.

Figure 17: Elba fire tower in Minnesota.
Source: Wikimedia

5.3.2 Virtual worlds

Apart from perspectives, the design of the virtual world that users must interact with is also a crucial factor in virtual approaches. It is important to consider how the interface environment can be creatively designed to cater to both the admin and the regular user. One way to achieve this is by creating an immersive and exciting world that is not reminiscent of the conventional cloud or computer interfaces. The use of metaphors to transform the traditional cloud dashboard into a more captivating and engaging experience for the user is an effective way to distinguish the interface. While it is crucial to preserve the technological aspects of the cloud tools, they should be visualized in a way that explores the possibilities of how user interfaces can be represented. By utilizing metaphors and creative design, the virtual world can offer an exciting and interactive experience that enhances user engagement and optimizes task performance.

What is an environment?

When creating an engaging virtual world, the possibilities are limitless, and there is no definitive answer. However, achieving a balance between functionality and aesthetics is essential. The world should represent an interface without being too obvious and should feature intuitive designs that guide the user in the right direction, utilizing clever layouts and interactive objects. Depending on the use case, the aesthetics of the world can take any form that seems intriguing to invoke creativity and usability, but it should also be rooted in the mind of the cloud user who will be using it. This project will prioritize functionality while showcasing the range of possibilities a virtual world can offer when interacting with the cloud, inspiring the user to explore and utilize the interface to its fullest potential.

To showcase the potential of virtual worlds, it is essential to provide examples to help to understand what a creative world can be. For the regular user, the focus should be on functionality and personal cloud management. Thus, a compact virtual world is preferred as it can efficiently serve various purposes.

Figure 18: The kitchen can be a metaphor for cloud management.
Source: istockphoto

For instance, imagine a virtual house that mimics a typical real-world home, consisting of multiple rooms such as a living room, bedroom, kitchen, and bathroom. Each of these rooms serves a specific purpose, such as relaxation in the living room, sleeping in the bedroom, and cooking in the kitchen. However, when it comes to cloud management, the kitchen is the room where most comparisons can be made. This relationship between kitchen and cloud management can be translated into the metaphor of household chores. For instance, the virtual kitchen can serve as an interface for creating and deleting virtual machines. Just like cooking a dish, creating parameters and providing data to VMs can be done step-by-step in the kitchen. When a user wants to delete a VM, they can discard it in the virtual trashcan, just like discarding a finished dish. The user will know that a VM is up running by having it served on the kitchen table, indicating that it's working and ready to be deleted by throwing it into the trashcan or update it by adding more ingredients to it on the kitchen counter. This visual approch follows the perspective of the machinist where tasks are performed by the user in a first person view mimicking a real person.

And doing tasks that often look differently from what is actually being done behind the scene, like interacting with a cloud service being masqueraded as cooking up a dish.

The next example is that of the admin user and where the regular user follows a more compact virtual world, the admin user requires a broader and more expansive visual approach to manage their tasks. Admin users are responsible for overseeing server performance and maintaining infrastructure, which requires a flexible and adaptable visual world. Unlike regular users, admins work in an uncontrolled environment, which means their interface needs to accommodate a wide range of needs. Since the admin has a god perspective, their environment should reflect this level of control. The visual approach should provide an open area where the admin can observe changes in the landscape which reflect server and infrastructure behavior. The admin will act as an observer, monitoring the environment for changes that indicate problems, such as loss of power or resource allocation issues. The landscape can serve as a visual representation of these changes, allowing the admin to quickly identify and resolve issues as they arise.

Figure 19: A forest where each section of trees represents a server rack.
Source: Wallpaperflare

To help illustrate the virtual world of an admin, picture a landscape filled with trees covered in lush green leaves. From a god's perspective, the admin observes this landscape from high above. The trees are organized into sections, each representing a server rack in the real world. This virtual world is designed to demonstrate a monitoring system for server health and performance. The system works by using the color of the leaves on each tree to signal the health of the corresponding server rack. If the leaves are green, it means the server rack is functioning optimally. However, if the leaves turn yellow, it indicates that the server rack is experiencing issues such as CPU overload or poor heat distribution. This serves as a signal to the admin that actions are needed to resolve the problem. If a section of trees loses all its leaves, it means that the trees in the virtual world have died. This translates into the real world as a server rack going down, rendering the system inoperable. This world is designed to alert the admin to any issues and allow them to take corrective action before any serious problems occur. This is an example of how a virtual can be represented as an admin, by using metaphors as an unconventional approach to visualization.

5.4 Use cases

The use cases will demonstrate the range of actions that can be executed within the virtual world. The aim is to create a comprehensive list of actions that users can perform to facilitate meaningful interactions with cloud tools. Ideally, the majority of these actions will be implemented in the prototype, showcasing the full functionality of the solution.

5.4.1 In depth about AWS and its services

Before delving into the various use cases, it's important to gain a thorough understanding of AWS and its suite of services that can best fit each use case. AWS, coupled with its SDK, offers users the ability to interact with virtually all of its offerings. Selecting the right service to interact with is a crucial step in defining the project's use cases. One such service offered by AWS is the highly popular S3 (Simple Storage Service), which provides an efficient and scalable object storage solution. S3 stores objects in containers called buckets, which can hold a vast range of data for backup and archival purposes. With the AWS SDK, users can easily create and delete buckets on any account, provided they have valid credentials. S3's flexibility and ease of use make it an ideal option for testing in this project. Regarding the admin user, while the AWS SDK doesn't offer metrics related to the health of the server, in this project the plan is to simulate a monitoring environment in the virtual world using artificial data. This approach will enable the possibility to monitor the server's health and make informed decisions on how to improve its performance.

5.4.2 Actions

The chapter of "Bridging the gap between cloud and VR" outlines the various interactions between users in the virtual space and the cloud, specifically AWS. To provide a clear understanding of these actions, a comprehensive list will be created, outlining the range of activities each user can perform within the virtual space. The list will serve as a reference point for the project and will be incorporated into the prototype as an assortment of different use cases.

By creating this list, the project will have a solid foundation for designing and implementing the virtual space's various functionalities. It will also ensure that all necessary actions are included in the prototype and can be performed seamlessly by the users. Additionally, the list will provide clarity and structure to the project, allowing for a more efficient and effective development process.

5.4.3 The regular user

The regular user will focus on managing buckets on AWS from the virtual world. There are many aspects that can be changed regarding a bucket within the AWS interface, but for this project, the main focus will be on creating, listing and deleting buckets. The regular user will have three possible actions within the world, which be explained in detail. The goal here is to showcase the possibility of interacting with the cloud using VR and seeing changes happening on AWS.

Create buckets

The first action will be to create a bucket within the virtual world. Creating a bucket is a process that can be performed on either the AWS dashboard or using the AWS SDK. In this project, only the SDK will be used to create buckets, but having an understanding of how it's created within the AWS dashboard can be helpful to understand the requirements needed to create one before working with the AWS SDK.

AWS Dashboard

To create an AWS bucket, navigate to the S3 service on the AWS dashboard. Once there, it will be possible to view all currently active buckets and access information about them. To create a new bucket, click the "Create bucket" button located in the top corner of the interface. This will prompt to enter a unique name for a new bucket. While there are many options for modifying the presets of a specific bucket, having a unique name is the only requirement for creating it. This simplicity makes it easy to create buckets using the SDK, as only a name is required to perform the creation operation within the virtual world.

AWS SDK

Creating an AWS bucket using the SDK is a straightforward process, thanks to the clear and concise documentation provided by Amazon[15]. The code required to create a bucket is relatively short, and it requires only a custom name to be implemented. However, to access and activate the request within the AWS account, credentials for the account must be inserted into the code. Fortunately, inserting account information is not complicated and can be achieved from a single text file accessible by the code.

```
public static async Task<bool> CreateBucketAsync(IAmazonS3
    client, string bucketName)
{
    try
    {
        var request = new PutBucketRequest
        {
            BucketName = bucketName,
            UseClientRegion = true,
        };

        var response = await client.PutBucketAsync(request);
        return response.HttpStatusCode == System.Net.
            HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error creating bucket: '{ex.
            Message}'");
        return false;
    }
}
```

List current buckets

The next action is to retrieve all existing buckets from an AWS account and display them in the virtual environment. This can be achieved by leveraging the SDK to gather the necessary data. Once the data is collected, the virtual world should display the total number of buckets to the user and allow them to interact with the buckets from the list.

AWS Dashboard

The AWS dashboard provides a convenient way to view every bucket without requiring any command inputs. There is no need to manually list the buckets, as they are displayed as a standard view option when accessing the service on the dashboard. This feature is a useful reference when integrating similar functionality into a virtual environment. Therefore, having studied the AWS dashboard's interface before implementing a similar feature into the project, seems necessary.

AWS SDK

To list the buckets in the AWS account, a connection to the AWS API must be established[16]. Since the number of buckets can vary, an additional step is required to retrieve the names of all the buckets. This can be accomplished by utilizing a loop within the script to make multiple requests to the API and retrieve the name of each running bucket. Once the list of bucket names is obtained, it can be incorporated into the virtual world as a means of managing an active cloud service. Users will be able to interact with the list of buckets and perform actions on them as necessary.

```
class ListBuckets
{
    private static IAmazonS3 _s3Client;

    static async Task Main()
    {
        _s3Client = new AmazonS3Client();
        var response = await GetBuckets(_s3Client);
        DisplayBucketList(response.Buckets);
    }

    public static async Task<ListBucketsResponse> GetBuckets
       (IAmazonS3 client)
    {
        return await client.ListBucketsAsync();
    }

    public static void DisplayBucketList(List<S3Bucket>
       bucketList)
    {
        bucketList.ForEach(b => Console.WriteLine($"Bucket
           name: {b.BucketName}, created on: {b.CreationDate
           }"));
    }
}
```

Delete buckets

As a regular user in the virtual world, deleting a bucket is the final action that can be performed. This action involves removing existing buckets from the AWS account. These buckets are interacted with within the virtual world and, once they are destroyed in the virtual world, they will trigger an automatic deletion of the corresponding buckets from the AWS account.

AWS Dashboard

The process of deleting a bucket on the dashboard is as simple as creating one, and can be completed in just two easy steps. First, the user must select the bucket they wish to remove, and then a window will appear asking them to confirm the deletion by typing the name of the bucket. This straightforward method of deleting a bucket can also be easily translated to utilizing the SDK.

AWS SDK

To delete buckets using the SDK, a connection to the AWS account must first be established. Similar to creating buckets, the only variable needed is the name of the bucket to be deleted[12]. Unlike the dashboard method, the SDK code does not require any additional confirmation procedures to perform the deletion, making it a suitable option for this final action.

```
public static async Task<bool> DeleteBucketAsync(IAmazonS3
    client, string bucketName)
{
    var request = new DeleteBucketRequest
    {
        BucketName = bucketName,
    };

    var response = await client.DeleteBucketAsync(request);
    return response.HttpStatusCode == System.Net.
        HttpStatusCode.OK;
}
```

Virtual world

The next step is deciding how this will look within the virtual world. After understanding the SDK method for creating buckets, the aim is to translate this action into a simple and interactive task. One idea is to use interactive buttons, similar to the gauges and levers of a train conductor, to simplify the process. These buttons would be self-explanatory in their use, making it easy for the user to create buckets with just a few clicks. To avoid the need for keyboard typing, the user could be given the option to automatically generate a name for the bucket instead. This approach emphasizes an imaginative method of bucket creation, almost like using a dashboard, but with a more engaging and intuitive interface.

The second action of listing the buckets involves providing 3D game objects that the user can interact with within the virtual world. Similar to the first action of creating buckets, users can spawn these objects using a button. When the user clicks the button, physical bucket objects will be generated and can be lifted and carried around within the virtual environment. To differentiate between the different buckets, each object will be labeled with its corresponding name, making it easy for users to identify them when it comes time to delete them.

The third action involves deleting existing buckets that now exist as 3D objects within the virtual world. Unlike the previous actions, simply using a button is not sufficient in this context. The SDK requires a specific name of the bucket to be able to delete it, so a different approach is needed. One idea is to create a mechanism where users can throw the 3D bucket object into a designated area, which can trigger the deletion based on the bucket name. In a virtual setting, this could be represented as a furnace that burns the buckets, thereby destroying them both within the virtual world and on the actual cloud. This approach adds a creative element to the process of deleting buckets, while still ensuring that the necessary steps are taken to properly remove them from the user's AWS account.

The combination of these three actions forms the foundation for the virtual world of the regular user. By understanding both the AWS aspect of the build and the approach for the virtual world, users can gain a clear understanding of how the prototype might look. This knowledge will guide the development process and help ensure that the resulting virtual environment is both functional and intuitive.

5.4.4 The admin user

To gain a comprehensive understanding of the admin user use cases, it's important to consider the constraint of not being able to use actual data for the prototype. As AWS doesn't offer an SDK for monitoring server metrics, the admin user will have to rely on simulated random data to demonstrate the virtual world and its usage. The aim of this exercise is not to evaluate the technical feasibility of the approach, but rather to illustrate the potential of visualization techniques for monitoring.

A random number generator will be used to simulate the data flow. To achieve this, a script will be required to push in random numbers, which will then be incorporated into the virtual world. This approach mimics the function of a monitoring system receiving data from a cloud service and pushing it to the prototype.

Virtual world

As there is no real data to consider and implement, the focus can be directed towards constructing a virtual world for the admin user. The central concept for monitoring is to leverage the landscape and its components to present information to the user. The proposal is to utilize clouds as the primary means of monitoring a system. The admin user will navigate the virtual environment from a god's eye perspective, observing the clouds as if they were an active server infrastructure. Each cloud will represent a server rack, and its color will change based on its status. This approach enables the admin to float above the clouds and observe the infrastructure as it changes in real-time.

By utilizing randomly generated data and a virtual world to monitor infrastructure, the goal of demonstrating a use case for the admin role can be achieved. Although regular users will interact with an actual cloud system and not the admin, the primary objective of the project is to focus on visualizing a new type of user interface. The admin solution can offer insight into how the system would appear if it were technically feasible.

5.5 Prototype

This project section will demonstrate the virtual world prototype for both regular users and admin users. The structure of this part will first provide an overview of the visual world and its creation process, followed by an explanation of the user perspectives and their implementation. Finally, it will showcase how various actions and use cases were incorporated and their operational methods.

5.5.1 Virtual world: Regular user

The virtual world for the regular user was designed to be easily navigable and not overly expansive. The primary objective was to create a space where users could move between stations quickly and perform various tasks without feeling disoriented. This meant building a layout that was intended to be intuitive and allow users to have a full understanding of their surroundings without having to search too hard for the stations where tasks could be completed. To achieve this, it was necessary to keep the world's size relatively small.

Although the virtual world's theme could have been anything imaginable, for this prototype, it was central to select a theme that was down-to-earth yet still engaging and enjoyable for users. The beauty of VR is that users can be transported to worlds beyond normal appearances. The chosen theme was medieval times, and the user was to roam a castle from that era. The objects and surroundings were designed to reflect that time period, with stone walls and floors, swords, and candles adorning the environment. This design choice was intended to invoke a feeling reminiscent of classic medieval times, which would make the user more invested in the experience.

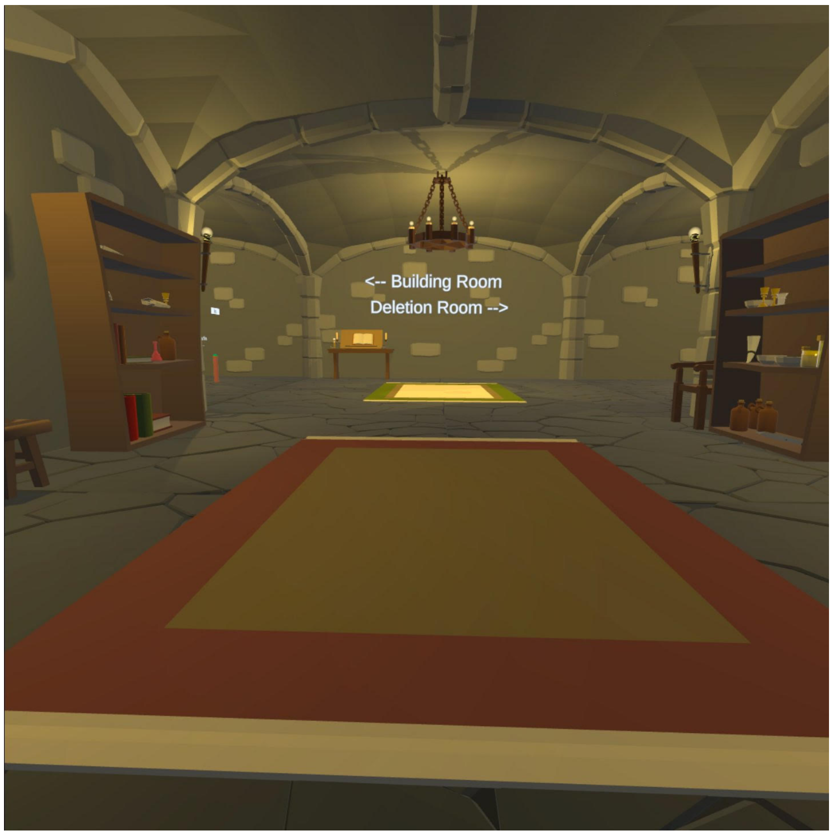


Figure 20: View from the VR headset, standing in the entrance.

The implementation of the virtual environment was done through the usage of Unity assets. Assets are downloadable 3D models that can be incorporated into Unity projects. The models often come in packs with various other models, and remove the need to do any 3D modeling as they are fully finished and rendered. For this medieval theme, a pack called Ultimate Low Poly Dungeon from Broken Vector had all the necessary models to build this world. It was a free download, as other packs on the Unity asset website often demand a payment in exchange for the assets, it came with unique items that help realize the world. Other features contained within the pack, included models with built-in light effect to illuminate the surroundings and small objects like chests, books and swords to clutter the environment to make it more engaging.

Figure 21: A picture from the Unity asset website showing the Ultimate Low Poly Dungeon assets.

After importing the assets into Unity, the next step in creating the virtual environment was to align each section, such as the walls, floor, and arches, to achieve the desired layout. Once the layout was finalized, the included models from the Ultimate Low Poly Dungeon pack were used to add clutter to the environment. Although this step was optional, it significantly enhanced the user's immersion in the world. One major advantage of this pack was that each model had a low resolution, which facilitated smooth processing by the VR headset. Using high-resolution assets intended for high-end computers could adversely affect performance when used with a less powerful VR headset.

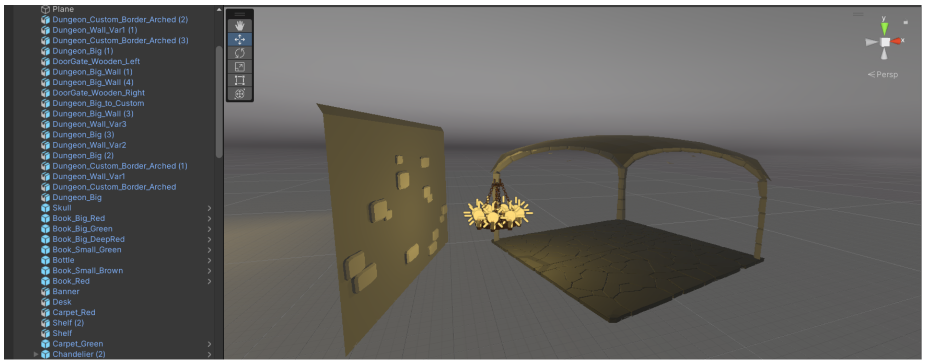


Figure 22: Showing the assets being put together to create the environment.

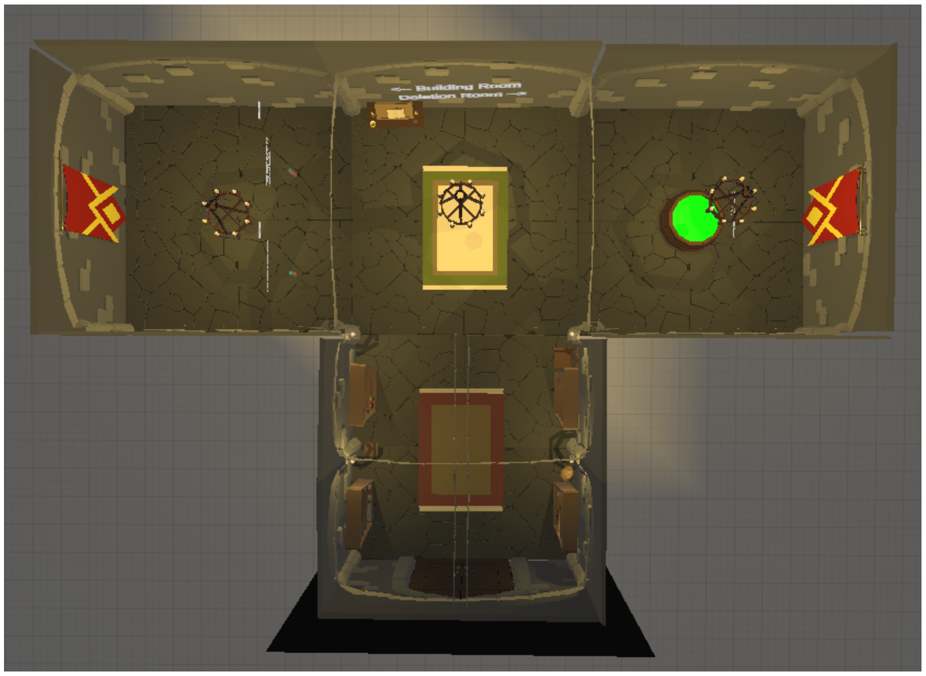


Figure 23: Top-down view of the layout.

The final layout consisted of a t-shape hall with an entrance and two interaction areas on each of the ends. The entrance does not have any purpose other than to immerse the user within the application, this is done using decorations on both sides to illustrate the theme of medievalisms. Moving forward, the user is met with a wall containing large signs explaining and pointing to each of the interaction areas. One sign saying “Building Room” points to the area for creating and listing of buckets, underneath this a sign saying “Deletion Room” which is the area for deleting buckets.

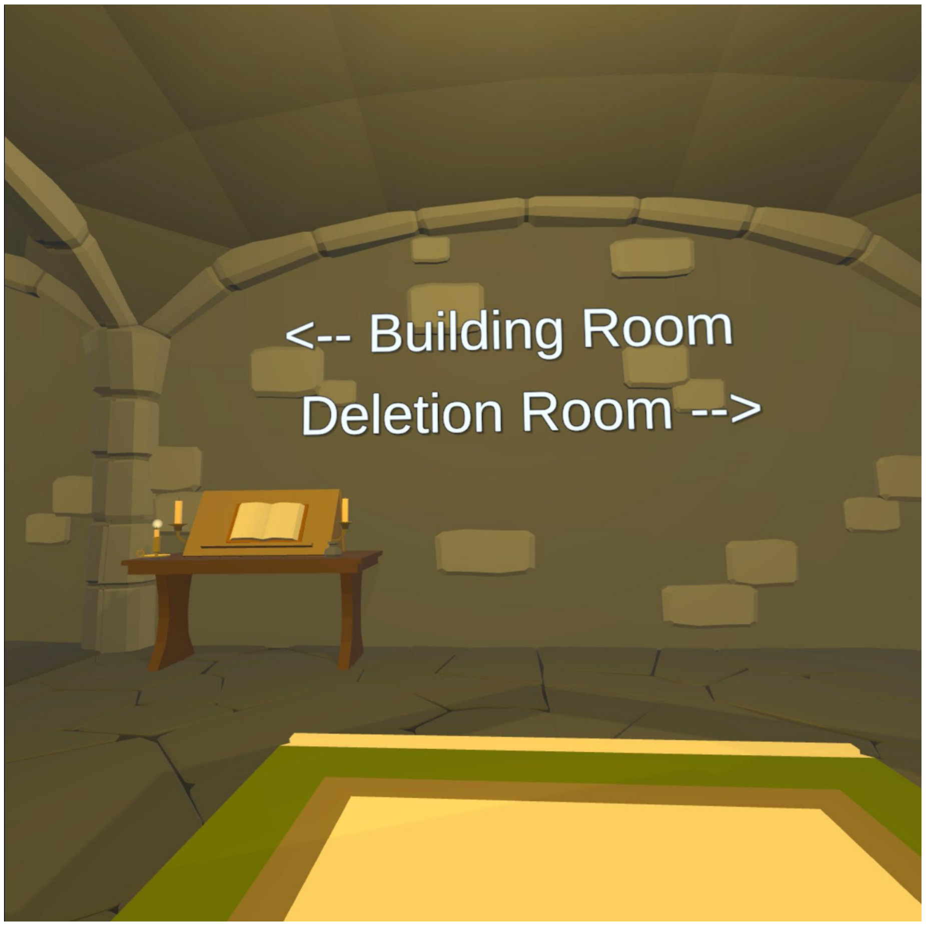


Figure 24: View from the VR headset, showing the signs for each station.

Building room

The Building Room is composed of two stations: one for creating buckets and another for listing and spawning the current number of them. The design of these stations is focused on simplicity, providing users with the necessary information to know when an action has occurred. Each station features a button on a pedestal that can be pressed to trigger the action. Behind each button, there is a floating text that changes depending on whether the button has been clicked or not. The functionality and use cases of these stations will be discussed in more detail later, for now, it's only needed to explain the structural elements of the application.

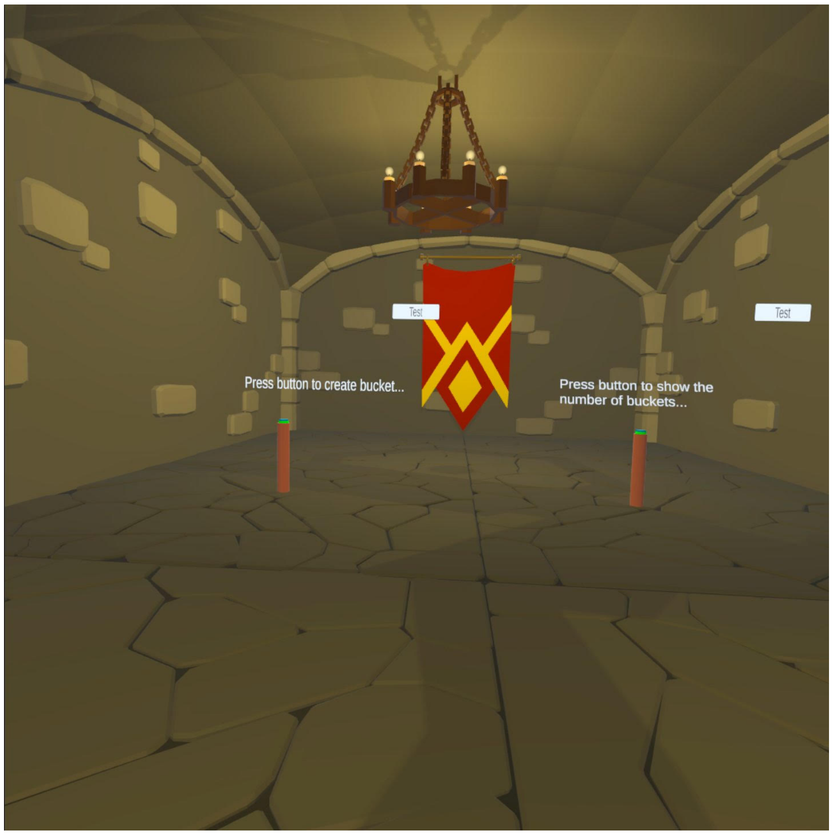

Test
Test
Press button to create bucket...
Press button to show the number of buckets...

Figure 25: View from the VR headset, showing the building room.

Deletion Room

The Deletion Room is located on the opposite side of the hall and serves as the area where users can delete buckets. The design of this room is centered around a large barrel, which is planted in the ground and features floating text above it. The purpose of this barrel is to provide users with a method for deleting buckets from both the application and AWS cloud account, but doing so in medieval fashion. To delete a bucket, users can throw the 3D representation of the bucket into the barrel. Once the bucket has been successfully deleted, the text above the barrel changes to signal to the user that the action has been completed. The use of the barrel as a deletion method also serves to differentiate it from the Building Room, which uses buttons as the primary interaction method.

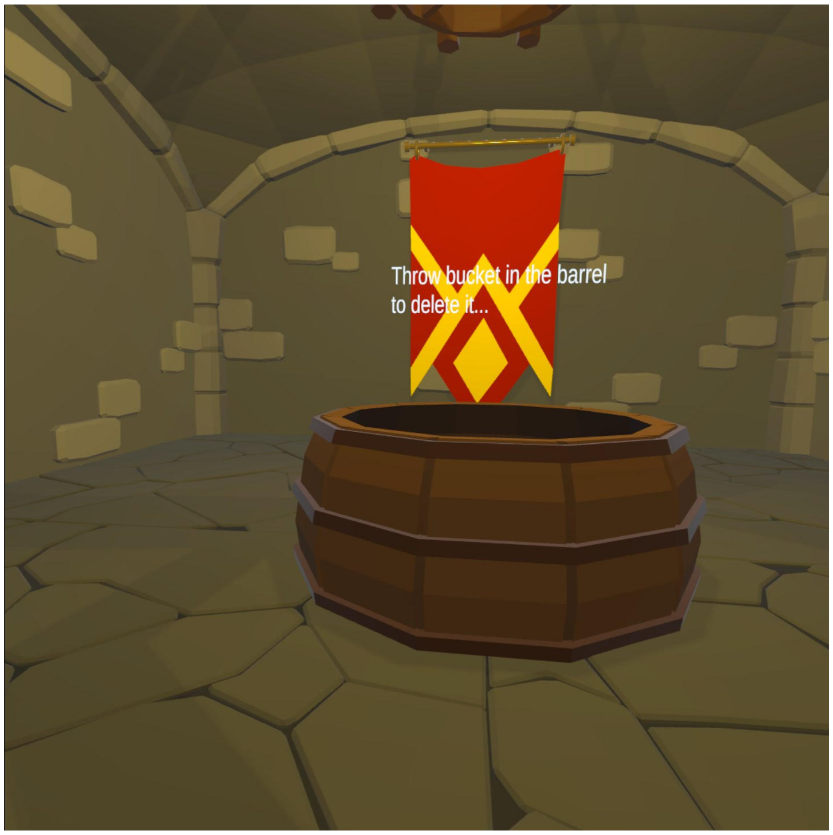


Figure 26: View from the VR headset, showing the deletion room.

The purpose of this architecture

The decision to use a medieval-themed world and layout was made to showcase a simplified version of using a cloud service. In the Building Room, the use of a pushable button that users must physically push aims to create a more immersive experience that connects the user with the world. While other interaction methods could have been used, the simplicity of the button click provides a more engaging option. Although users are restricted to simply pushing a button, the goal is to illustrate how something as seemingly complex as a cloud dashboard can be reduced to a single action. By focusing on simplicity, users can better understand how to interact with the application.

The Deletion Room design is where the metaphorical nature of the application truly shines. The act of throwing AWS storage buckets into a cauldron with a boiling substance serves as a symbol for the deletion process. This contrast between the medieval-themed world and the modern technology of AWS engages the user and makes the method of discarding items more instinctual and recognizable than clicking a button on a screen.

5.5.2 Perspective: Regular user

The perspective from which users view the virtual world plays a significant role in how they experience it. For regular users, the machinist perspective was chosen to provide a hands-on experience of interacting with tools in the virtual world through various operations. This perspective again draws parallels to the role of a train conductor, who uses different levers and gauges to drive a steam train and maintain its stability. Similarly, in the machinist perspective, users push buttons and lift objects to perform tasks on the cloud.

Movement within the virtual world

How it operates

To create a realistic movement experience for the user, it was a matter of focusing on mimicking real-life physics. The user is bound to the ground by gravity and moves through the virtual space as they would in real life. Non-realistic movements such as flying are not allowed, as the goal is to emphasize on the interaction with various objects rather than unrestricted movement in all directions.

How it functions

Unity provides a range of built-in solutions for VR development under the umbrella term "XR," which encompasses mixed, augmented, and virtual reality technologies. By incorporating XR, developers can efficiently set up controllers and viewpoints for their VR applications. To facilitate movement within the VR environment, Unity's XR offers a script called the locomotion system. This script can be attached to the player object, enabling users to move around the environment using the buttons on their VR controllers. The advantage of the locomotion system is that developers can easily add additional functionality to customize how users move. In the current project, this customization included the addition of a continuous turn provider and a continuous move provider. The continuous move provider allows users to move continuously using the joystick, allowing for seamless exploration of the VR environment. Similarly, the continuous turn provider enables users to turn or spin in a specific direction also using the joystick on their controller. These controls enable users to fully immerse themselves in the VR experience, as they can explore their environment while simultaneously moving their head.

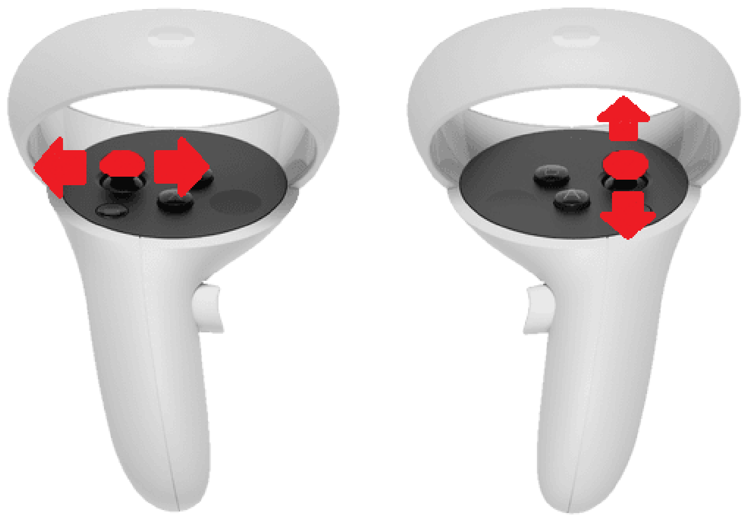
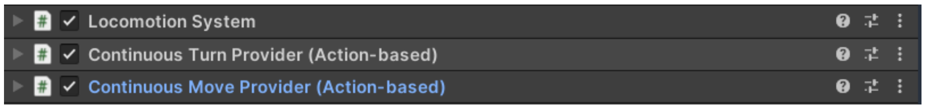


Figure 27: Illustration showcasing the operations for movement on the Oculus Quest 2 controllers.

In figure 27 the Quest controller is equipped with two joysticks that serve different functions. The right joystick can be moved up or down to move forward or backward within the virtual world. On the other hand, the left joystick is responsible for turning the user in any desired direction. By rotating the left joystick to the desired position, the users' orientation can be adjusted accordingly.



Locomotion System
Continuous Turn Provider (Action-based)
Continuous Move Provider (Action-based)

Figure 28: A snapshot of the locomotion system as seen in the Unity editor.

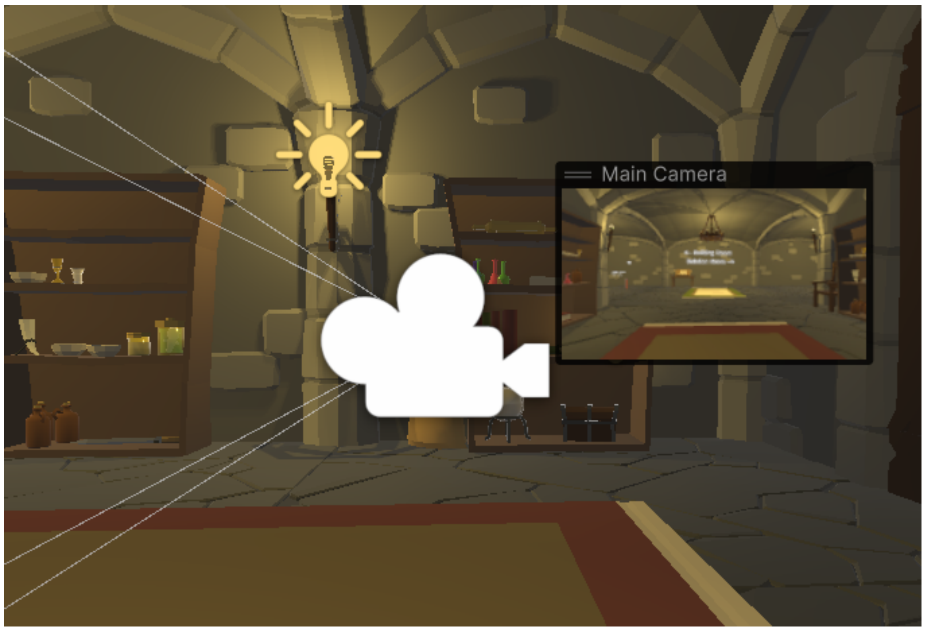


Figure 29: Showcasing the main camera of the application, or the point of view for the user.

Hands within the virtual world

How it operates

In virtual reality, the hands are the user's primary reference point for their body and are useful tools for effective interaction with the virtual world. Representing the hands in a realistic and immersive manner is a challenging task that often requires complex animation. Despite this, modern VR applications typically display animated hands to give users a sense of presence and agency within the virtual space. An alternative approach is to display the physical controllers as objects within the virtual environment. This approach is simpler to implement as it does not require complex animations or moving parts. By providing a direct representation of the user's physical controllers, this approach can enhance the sense of connection between the user's real and virtual selves. In addition to hand representation, another important aspect of VR interaction is object selection. To facilitate this, a pointing system is needed to make an understandable, yet useful selection system. A method that seemed like the most optimal for this was emitting lasers from the controllers almost like a laser pointer pen, this way the users can easily identify and select objects within the virtual world.

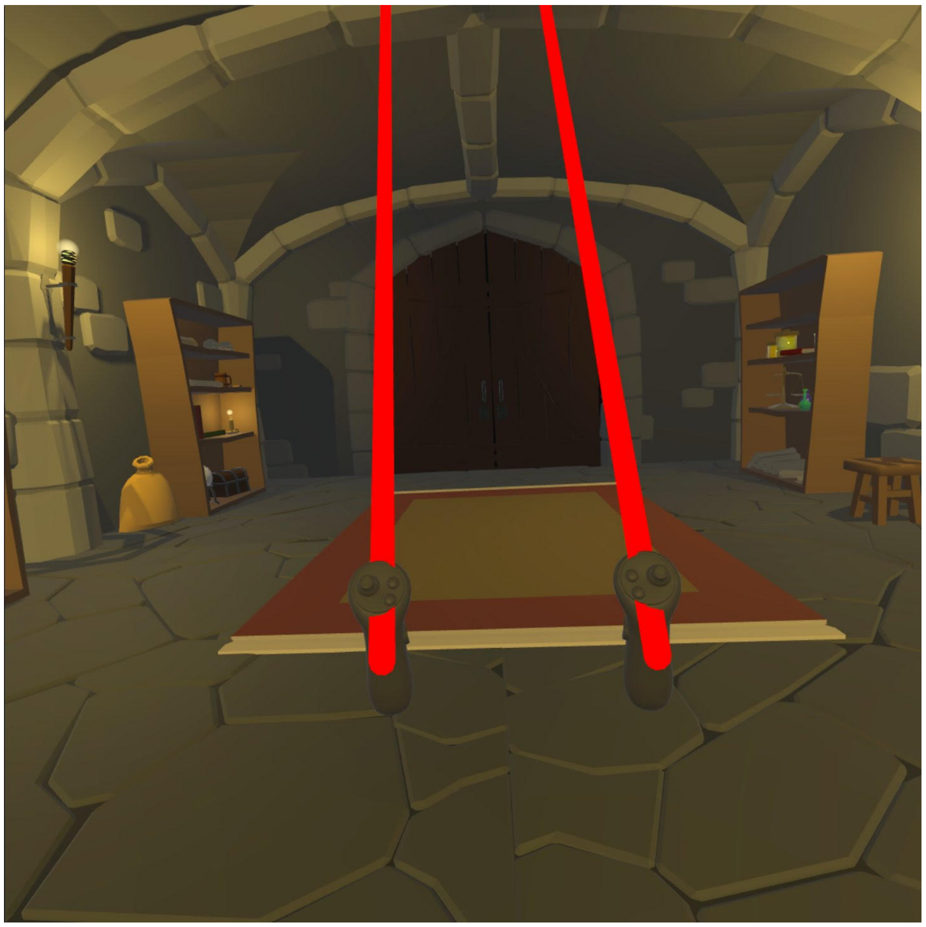


Figure 30: View from the VR headset, showing the user's hands with ray interactors.

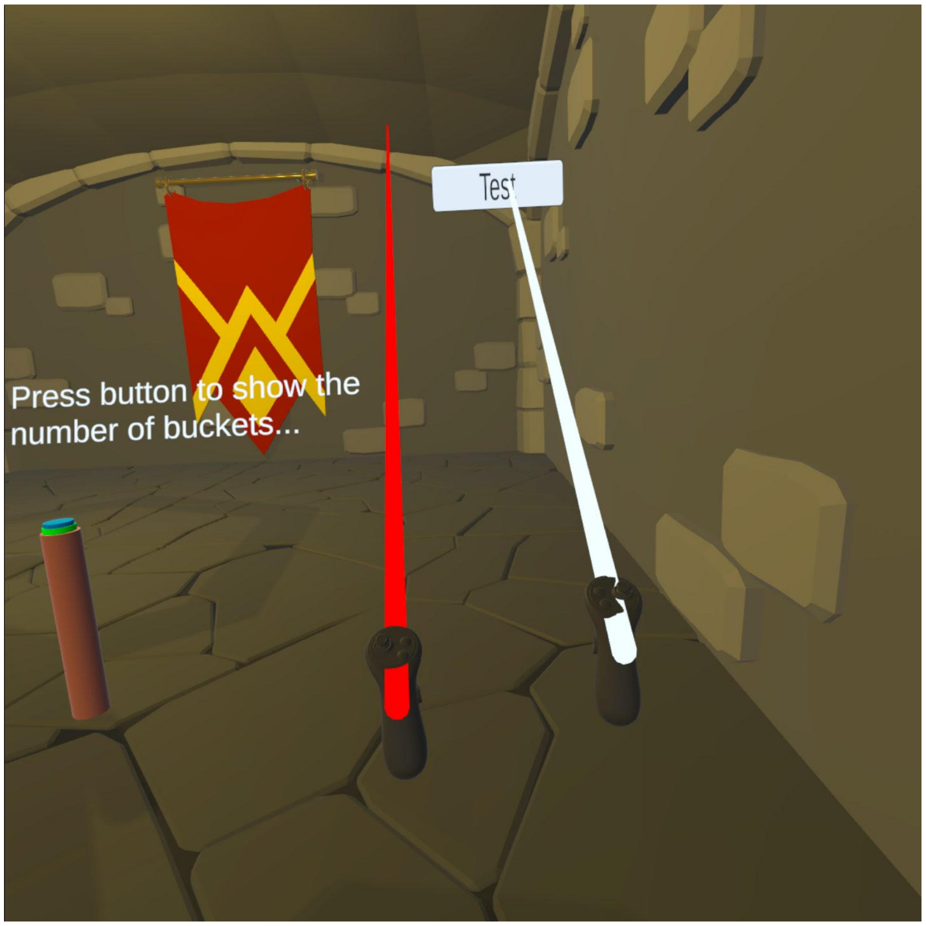


Figure 31: View from the VR headset, showing the ray interactors in action.

How it functions

To create a more immersive experience, the physical controllers were used as the basis for the virtual hands in this VR application. The controllers were represented by simple models that were added to the XR player object, allowing for intuitive interaction with the virtual environment. Unity's XR library includes a built-in system for translating real-life controller movements into the virtual world, providing a responsive experience for the user. To facilitate object selection and interaction, the Ray Interactors script was utilized. When added to the controllers, this script emits a customizable laser beam that can be set to any color. This feature was particularly useful in this application, as the laser beam changes color from red to white whenever it touches an interactive object. Additionally, the lasers are cut off upon contact with objects, providing the user with clear visual cues as to what can and cannot be interacted within the virtual world.

5.5.3 Use cases and actions: Regular user

As previously mentioned, there are three actions the user should be able to perform within the virtual world. These actions aim to facilitate the integration between the virtual space and the AWS cloud. The AWS service that the user will interact with is the S3 object storage service. S3 utilizes buckets as their storage objects, which can be created and store data on an AWS account. The three actions that the user should be able to perform within the virtual world are as follows:

1.Creating a new bucket.

2.Listing the current buckets available on AWS.

3.Deleting a specific bucket.

To enable the VR application to interact with the cloud, the AWS SDK plugins were imported into Unity. These plugins consisted of a few files that allowed for the integration of AWS libraries into the application's codebase. To access the AWS account, a file containing the account information, such as the username and password, was created and referenced by the SDK plugins. Once this setup was complete, the application was able to communicate with AWS through the SDK.

There are two stages for each action, the before interaction and after interaction, in this section, the before and after stages will be discussed and showcased.

Creating a bucket

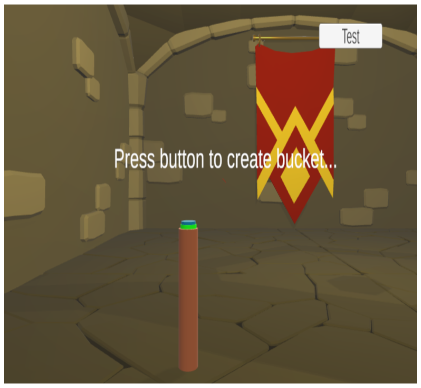


Figure 32: Before creating a bucket.

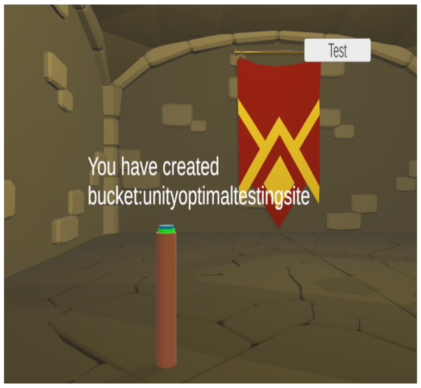


Figure 33: After creating a bucket.

Before clicking the button, the user is presented with a message indicating that pressing the button will create a new bucket on their AWS account. The button itself triggers the create bucket function through the AWS SDK when it is pressed down by the user. If the button is pressed, the script attached to it activates the function and sends the request to AWS. Once the action is completed, the user is notified with a new message indicating that the bucket has been created, along with its name. For this example, the name of the bucket sent to AWS is a static variable, which means it would need to be changed in the script if a new bucket is to be created.

Listing the current buckets available on AWS.

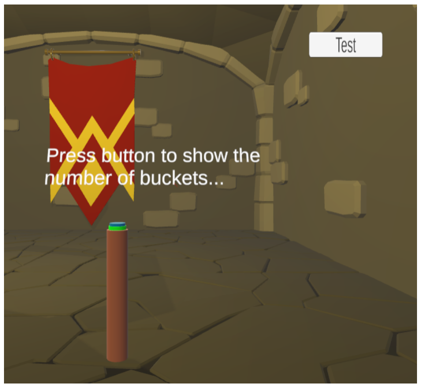


Figure 34: Before listing buckets.

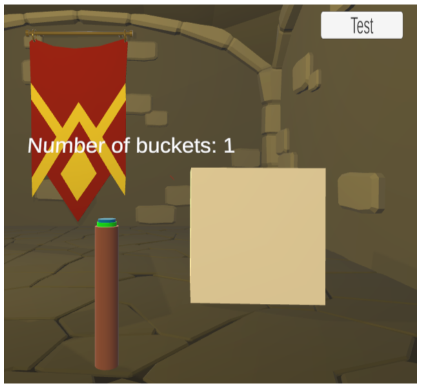


Figure 35: After listing buckets.

The next available action is to list the number of current buckets from AWS. Within the virtual environment, the user is instructed to press a button to activate the list bucket function. Upon pressing the button, the function is called, and the text displays the current number of buckets from AWS. To enhance interactivity within the virtual world, the script also spawns 3D cubes that represent the buckets in the virtual space. The number of cubes spawned corresponds to the number of current buckets on AWS. Each cube also gets the same name as what they have on the AWS account. These cubes can be interacted with – lifted, thrown around, and eventually deleted – to further engage the user in the virtual environment.

Deleting a specific bucket

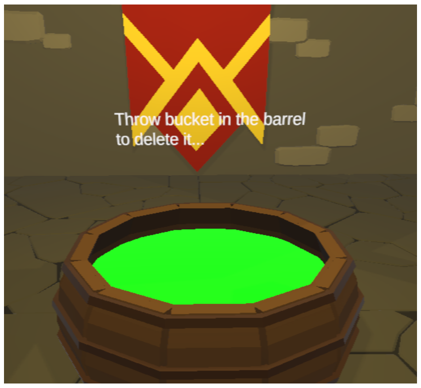


Figure 36: Before deleting a bucket.

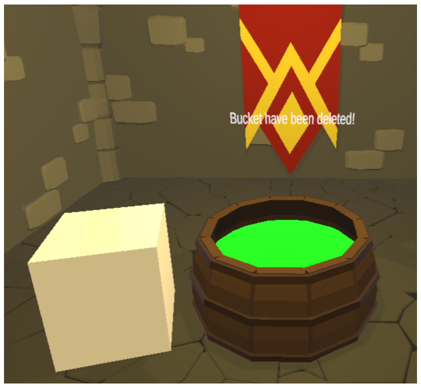


Figure 37: After deleting a bucket.

The process of deleting a bucket differs from the other two actions. Instead of a button, the user is presented with a barrel containing a green substance. To activate the deletion function, the user must pick up a 3D cube from the bucket listing station and throw it into the barrel. Upon contact with the green substance, the 3D cube is destroyed, and the deletion function is triggered on the AWS account. Since each 3D cube is named after a bucket on the AWS account, the function knows which bucket to remove. This approach adds an interactive element to the deletion action, requiring the user to physically throw the cube into the barrel. It also ensures that the correct bucket is deleted by naming the 3D cubes after their corresponding AWS buckets.

The scripts: Regular user

With an overview of each of the actions that can be performed within the solution, the next step is to examine the scripts that make them possible. There are three scripts, one for each action performed in the virtual world, namely creating a bucket, listing buckets, and deleting buckets. Although these scripts are separated into three, they share many similarities that need to be discussed in this section.

Establishing a connection with AWS

```
void Start()
{
    var chain = new CredentialProfileStoreChain();

    if (chain.TryGetAWSCredentials("unity-test", out
        awsCredentials))
    {
        Debug.Log("Loading CreationClient!");
        s3Client = new AmazonS3Client(awsCredentials,
            bucketRegion);
    }
    else
    {
        Debug.Log("An error occured with CreationClient!!");
    }

    Debug.Log("CreationClient loaded succsessfully!");

    isPressed = false;
}
```

The scripts share a common feature, which is the initial step of verifying the validity of AWS credentials and establishing a connection to the corresponding account. This is done by retrieving the credentials from a designated file, in this case the “unity-test” file, and attempting to establish a connection using those credentials by creating a new client. The code then checks whether the connection was successfully established or not and logs a debug message in the Unity console accordingly. This function serves as the entry point for communication with Amazon services and runs immediately upon starting the VR application.

The button functionality

```
private void OnTriggerEnter(Collider other)
{
    if (!isPressed)
    {
        button.transform.localPosition = new Vector3(0, 0.00
            3f, 0);
        presser = other.gameObject;
        onPress.Invoke();
        isPressed = true;
    }
}

private void OnTriggerExit(Collider other)
{
    if (other.gameObject == presser)
    {
        button.transform.localPosition = new Vector3(0, 0.01
            5f, 0);
        onPress.Invoke();
        isPressed = false;
    }
}
```

Two of the scripts, create and list buckets, share a feature: the button functionality. The button is equipped with a collider system that detects whether it has been pressed or not. The button functionality is divided into two parts: while the button is untouched, the trigger remains inactive, and when the button is pressed down the collider detects the change in the button's position from being on top to being pushed down. This change is determined by the positional status of the button, as indicated in the code snippets "button.transform.localPosition = new Vector3(0, 0.003f, 0);" and "button.transform. localPosition = new Vector3(0, 0.015f, 0);". The event is triggered when the button is down and is activated only once. Once the event is over, the button returns to its normal up position.

The delete functionality

```
private void OnCollisionEnter(Collision collision)
{
    TMP_Text txt = DeletionText.GetComponent<TMP_Text>();
    DeleteBucketAsync(collision.gameObject.name);
    Destroy(collision.gameObject);
    txt.text = "Bucket have been deleted!";
}
```

The deletion function in the deletion script also utilizes the collider system. When a 3D object, specifically the cubes representing the AWS buckets, collides with the green substance inside the barrel, the collider is triggered. This initiates the SDK function to delete the corresponding bucket on AWS, with the 3D cube's name serving as the parameter for identification. Once the deletion function is completed, the 3D cube is removed from the application and a message pops up to inform the user that the bucket has been deleted.

Listing buckets and spawning cubes

```
if (gameObjectSpawnLoopRunner == true)
{
    for (int myNum = 0; myNum < currentBuckets.Buckets.Count
        ; myNum++)
    {
        ListBucketsResponse response = await s3Client.
            ListBucketsAsync();

        GameObject cube = GameObject.CreatePrimitive(
            PrimitiveType.Cube);

        foreach (S3Bucket bucket in response.Buckets)
        {
            cube.name = bucket.BucketName;
        }

        cube.AddComponent<Rigidbody>();
        cube.AddComponent<XRGrabInteractable>();
        cube.AddComponent<Collider>();
        cube.transform.position = new Vector3(1.830845f, 0.6
            412615f, 6.638258f);
    }

    gameObjectSpawnLoopRunner = false;
}
```

The script responsible for listing buckets and spawning cubes is the more complex of the scripts. It runs a loop to check each bucket in the AWS account every time the in-application button is clicked, refreshing the list of buckets to see if any new ones have been added or removed. The primary objective of the listing script is to spawn cubes that represent the buckets, requiring the creation of an object with all the necessary components to make it interactive for the user. Each cube must also be given the same name as its corresponding bucket. The script spawns cubes equal to the number of buckets in the AWS account and can only run when the number of buckets is updated to prevent the user from spawning an infinite number of cubes by repeatedly clicking the button.

Important note!

It should be noted that the functions directly associated with the creation, listing, and deletion of buckets in AWS have been discussed in the previous use case section before the prototype chapter. Therefore, they are not reiterated here. However, it is important to highlight that the AWS SDK code has been reviewed and implemented in the application.

5.5.4 Analysis of reduction: Regular user

The regular user prototype was assessed to identify areas where complexity could be reduced. One area was the aspect of the S3 AWS dashboard being transformed into a virtual application, eliminating the need to navigate a menu. Instead, users can interact directly with buckets that are represented as 3D objects. This approach improves user comprehension by allowing them to manipulate and move objects. The most significant reduction in complexity is the delete function. Users can now throw unwanted buckets into a barrel for deletion, similar to a real life trashcan. This intuitive feature allows for quick and efficient removal of unwanted buckets that it natively understood from a user perspective.

5.5.5 Virtual world: Admin user

The virtual world for an admin is designed to provide a larger and more expansive environment than what regular users have. This allows the admin to have a comprehensive overview of the entire system and make changes on a large scale. The world is intended to represent a live infrastructure and therefore provides the admin with ample space to navigate and monitor the system as a whole. The primary purpose of this world is to enable the admin to understand the big picture and ensure the smooth functioning of the infrastructure. Unlike regular users, the admin does not interact with the world directly, but rather observes the system as a whole, allowing the world to carry out interactions on their behalf. This monitoring-centric world helps the admin to keep track of multiple events happening across different parts of the infrastructure simultaneously.

It is important to select a theme that corresponds to the magnitude of the task. A world that simulates a landscape with a river, trees, and mountains was chosen to depict the grand scale of the task. The natural setting aligns well with the concept of something colossal and offers the admin user a sense of being a part of a vast system. The natural backdrop provides the perfect setting for the admin user, while the primary monitoring aspect remains in the cloud. The proposed theme involves the admin user floating above the landscape, with clouds acting as the real-life infrastructure. Each cloud represents a server, and the clouds change color based on the health of each server. This allows the user to quickly assess the current state of each cloud and hence each server. The theme fits well with the concept of using clouds to observe the cloud.

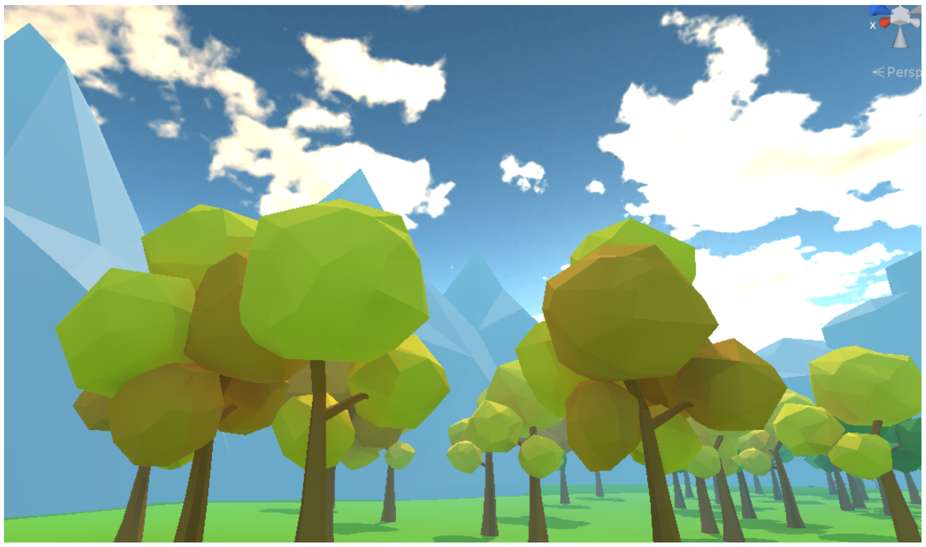


Figure 38: View from the VR headset, showing the nature landscape.

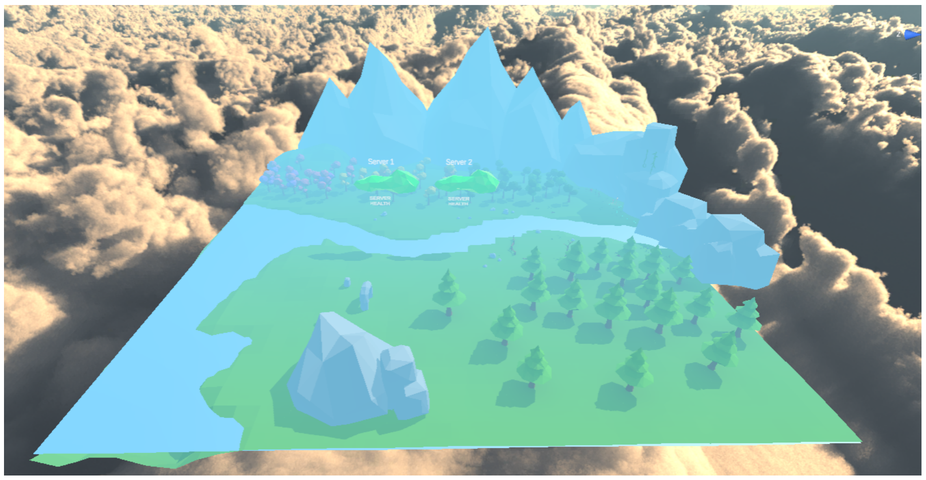


Figure 39: Top-down view of the layout.

The world was created using assets from the Simple Low Poly Nature Pack by NeutronCat. This pack includes a variety of low poly assets that have a similar design and feel to those used by the regular user. Additionally, the pack provided a pre-built landscape that fit with the requirements of the application.

The landscape had all the necessary elements to convey the feeling of a grandiose environment, making it an ideal backdrop for the admin user. To enhance the cloud theme, a skybox was implemented to give the world more character. A skybox is essentially a non-interactive background picture that creates the illusion of a sky with clouds or stars. Since it doesn't have any moving parts or 3D models, a skybox doesn't impact performance. Instead, it is a 2D image that wraps around the 3D space, creating a convincing sky that tricks the user into feeling like they're in the clouds.

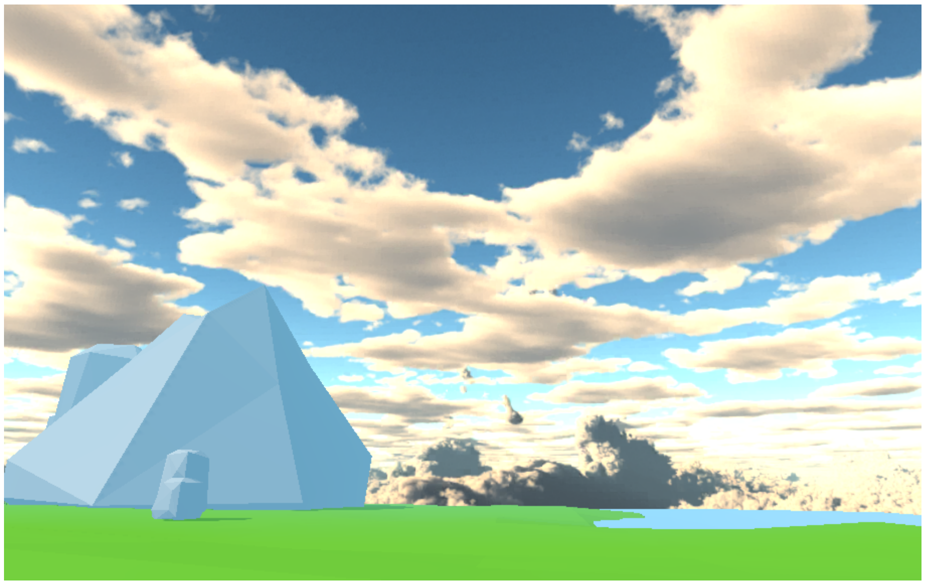


Figure 40: View from the VR headset, showing the skybox.

Upon completing the landscape, the addition of clouds served to symbolize the servers. The solution features two large clouds situated in the sky above the player, showcasing the underlying infrastructure. Each cloud displays text above and below, indicating the server name and health status. The clouds hold significance in this world, representing the monitoring aspect of the application. They aim to provide visibility into server health, ensuring optimal performance.

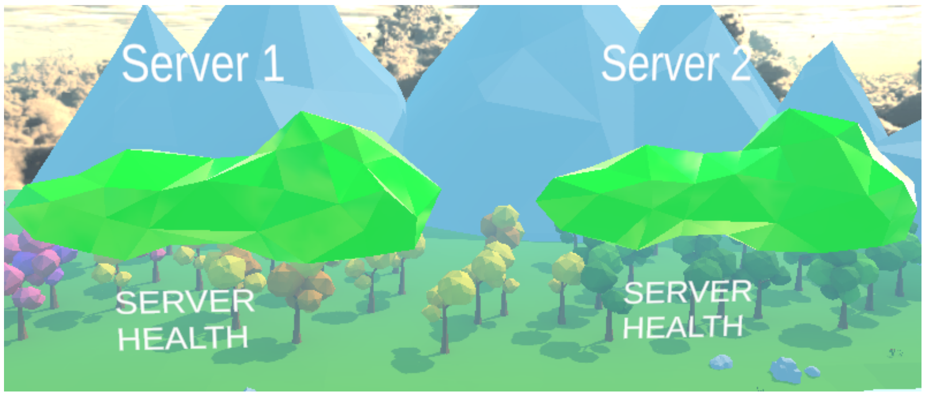


Figure 41: View from the VR headset, showing the server clouds in operation.

5.5.6 Perspective: Admin user

The admin user's perspective transcends realism and grants them a "god perspective." As previously discussed, this view provides an all-seeing eye over the world and allows the user to move in any desired direction. Unlike the regular user, the admin is not bound to the ground, creating endless possibilities for movement. This perspective enables the user to monitor the entire system from any angle they desire.

Movement within the virtual world

How it operates

To attain the god like perspective of the admin user, the movements must defy the laws of physics and allow for unrestricted movement in all directions. This entails granting the user complete freedom of motion along all axes. The user can effortlessly glide through space to reach their intended destination, and hover in midair to focus on a particular part of the world. This also requires the ability to toggle flying mode on or off, without the risk of plummeting to the ground while airborne.

How it functions

To fly in a VR environment, the player needs to use XR, to incorporate VR controllers into the application. However, unlike the regular user, there is no pre-built solution for flying in VR with XR. And creating custom flying mechanics requires a thorough understanding of programming the movement of axes, which can be a complex task. For this reason, the choice landed on a third-party script developed by the Data Visualization Research Lab, called the "XR Flying Interface script". This script can be downloaded and imported into Unity and then attached to the player object. By attaching the script to the player, the VR controller can be linked to the script to control the flight of the player. To initiate flight mode, the player needs to press a button. Once in flight mode, moving the VR controller forward makes the player fly forward, while moving it backward makes the player fly backward. To rotate the flying position, the player can move the controller clockwise or counterclockwise to set a new position. This flying mechanism is similar to that of Superman, who puts his fist in front of himself before flying. It allows the user to physically involve themselves in the flying experience, making it more immersive.

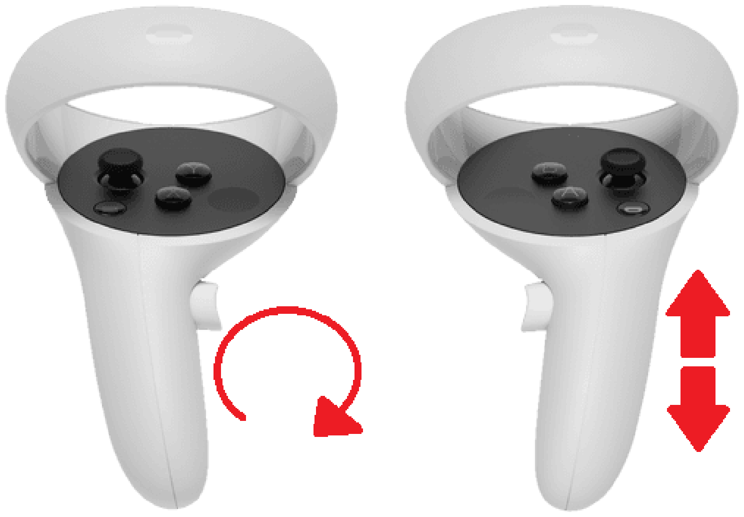


Figure 42: Illustration showcasing the operations for flying on the Oculus Quest 2 controllers.

As depicted in Figure 42, the Quest controller's themselves are utilized for flying in VR, with no buttons on the controller necessary for this function. To fly forward or backward, the user simply needs to push the right controller in the respective direction. Similarly, if they want to change their orientation, they can rotate the left controller in the desired position, before continuing flying.

Hands within the virtual world

Visible hands will not be included in this application. This decision was made because the application does not require any interactive elements that would necessitate physical interaction. Instead, the focus of the application is on observation. The user will fly around and observe the infrastructure without interacting with it directly. The primary goal is to showcase the monitoring aspect of the administrator's role, rather than other parts. As a result, the main feature will be a floating head, which serves as the user's point of view. While the absence of hands may seem unusual, it is a deliberate choice that reinforces the emphasis on observation and the administrator's monitoring role.

5.5.7 Use cases and actions: Admin user

The user assumes a monitoring role and is unable to directly manipulate any actions within the virtual landscape. Instead, the user observes as the application dynamically alters the color of two clouds. These clouds serve as visual indicators, with the color red representing poor server performance and green indicating that the server is performing well. Upon launching the application, the user will be able to see the clouds and their current color status. It is important to note that the user cannot directly alter the color of the clouds, as this action is controlled by the system. The purpose of the application is to allow the user to monitor the performance of the server, with the cloud colors providing a simple and intuitive way to gauge its status.

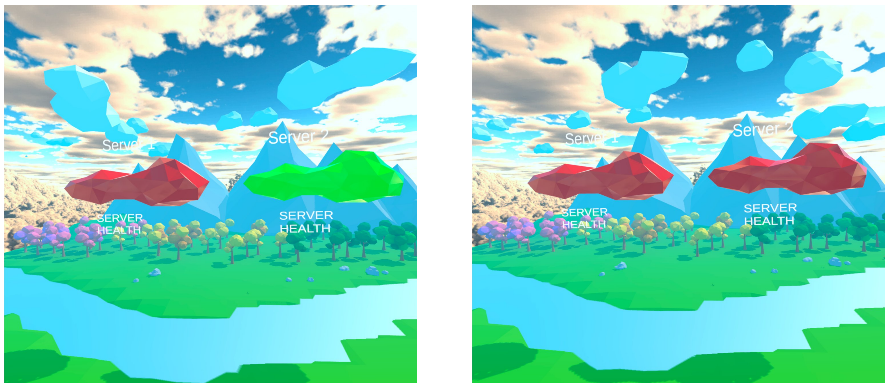


Figure 43: Clouds representing the health of a server.

To test the solution, as no actual data is coming from an SDK, a random number generator is utilized to simulate the flow of data. This generates a sequence of random numbers that mimic real data, causing the clouds to react and change accordingly, based on the values produced by the generator. Though using actual data would provide a more accurate representation of the solution's behavior, obtaining such data is challenging. Therefore, the use of a random number generator is a viable alternative for testing purposes.

The scripts: Admin user

There is a single script that controls both the random number generator and the color changing mechanism for the clouds. To understand how the script works, three distinct functions must be reviewed.

The randomizer function

```
public float Randomizer()
{
    System.Random random = new();
    float randomNumber = random.Next(0, 2);
    return randomNumber;
}
```

The randomizer function leverages the system library to generate random numbers. It assigns a float variable either a 0 or 1, and returns that number for use in other functions. While this function serves as an example of data flow simulation in cloud infrastructure, it can be made more complex to suit specific use cases. However, for the purpose of this example, simplicity has been prioritized.

The color changer function

```
public void colorchanger()
{
    value = Randomizer();

    if (value != lastValue)
    {
        float lerpValue = Mathf.InverseLerp(0f, 1f, value);
        material.color = Color.Lerp(colorA, colorB,
            lerpValue);
        lastValue = value;
    }
}
```

This function is responsible for changing the color of the clouds. The data from the randomizer is stored in a variable called “value”. This variable determines whether the “if statement” should be executed or not. The purpose of the “if statement” is to check whether the value from the randomizer has changed since the last iteration. If there has been a change, the function is called inside the “if statement”. Within the function, a float is created by mapping the value from the randomizer, which ranges from 0 to 1. This float is then used to interpolate between two colors – green and red. The resulting color is added to a cloud. Finally, the current value from the randomizer is saved so that it can be compared to in the next iteration to determine whether the "if statement" should be executed again.

The RunMethod function

```
IEnumerator RunMethod()
{
    while (true)
    {
        colorchanger();
        yield return new WaitForSeconds(changeInterval);
    }
}
```

The RunMethod function utilizes an enumerator to execute a continuous while loop throughout the application's runtime. Its primary purpose is to facilitate the periodic execution of the color changer function every two seconds, which simulates a delay between the arrival of new data. Without this delay, the color of the clouds would update with every frame, resulting in a jarring and inconsistent appearance that would render the application unusable.

5.5.8 Analysis of reduction: Admin user

The admin user prototype offers an insight into how complexities can be reduced. One example is the implementation of a color-based system to monitor server health. The cloud changes colors based on the state of the server, allowing the admin to quickly identify potential issues. Additionally, by providing the ability to fly in the virtual world, the admin can observe servers from any angle, further enhancing their ability to spot potential problems. This could be helpful when a larger system of servers is introduced and the need for a better view is necessary.

6 Discussion

This section will provide a comprehensive overview of the key learnings derived from the overall project. It will also delve into the important aspect of complexity reduction and identify the specific areas where it has happened within this project.

6.1 The concept of perceiving complex information

The human brain processes images and visuals much more quickly and efficiently than text, with images being processed 60,000 times faster than text according to research[21][20]. Despite this fact, text remains the primary mode of interaction with computer systems. This complexity can be reduced by incorporating more visualization into system design, as the brain uses pattern recognition to familiarize itself with subjects.Virtual reality can serve as a useful tool to introduce more visuals into computer systems, making them more easily understood. By using VR, complex systems like the cloud can be visualized and explained in a more intuitive manner. This approach is preferable to having people navigate through command-line interpreters, which can be difficult to understand without the proper technical education. Visuals and metaphors can be used to explain the intricate architecture of the cloud, providing an easier path for people to understand. This approach makes it easier for users to grasp complex concepts and information, enabling them to use the system more effectively.

In this project, Virtual Reality has been utilized to showcase the power of imagination in carrying out cloud operations in a more engaging and inspiring way. Initially, it was believed that conveying the intricacies of cloud technology through a visual medium would be challenging. However, the interactive nature of VR has shown itself to be a versatile tool, making it possible to create a unique and dynamic representation of the cloud environment. In such a way that one can argue that some elements of complexities are being reduced, with visualizations being used to simplify the portrayal of information.

To illustrate the power of visualization in simplifying complex tasks, consider the lock and chain metaphor. In a Docker environment, with hundreds of containers running simultaneously, a DevOps professional may need to access a specific container to address an issue. However, when working with a command-line interface, it can be challenging to determine which containers are accessible and which are not. This requires clicking on each container and attempting to access it, a time-consuming and tedious process. Now, imagine accessing the same Docker environment through a VR world. Instead of interacting with each container to determine accessibility, the VR environment can represent inaccessible containers with a lock and chain. In contrast, accessible containers would be unencumbered. This intuitive representation would allow the user to quickly understand which containers are accessible without any further investigation. With a hundred containers in the VR space, identifying inaccessible ones would be much easier than through a CLI. This example demonstrates how visualization can reduce complexity by making complex information more easily understood and navigated. By using familiar symbols and metaphors, users can quickly grasp concepts and take action, saving time and effort.

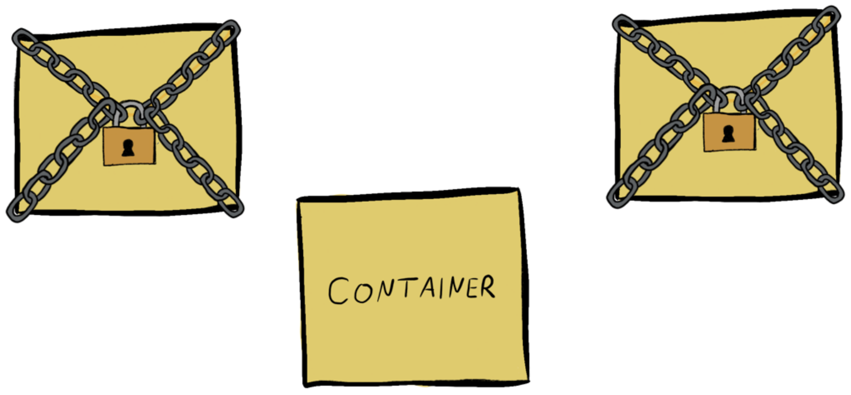


Figure 44: The lock and chain metaphor.
Illustration by Martine Mandt Brekne.

Measuring complexity is a challenging task since it concerns the ability of the human brain to comprehend information, and different individuals have varying levels of ease with certain subjects. It is up to each person to determine when a reduction in complexity has been achieved. However, one way to achieve this reduction is by using visual aids to convey complex data, as this project and the example above demonstrates. Visuals have the advantage of being immediately understandable to most people, unlike text. By using images and colors, any topic can be explained with a little less effort, as people naturally gravitate towards visual cues. This project showcases the potential of virtual reality to create immersive worlds that explain complex concepts through visuals.

It is important to acknowledge that while VR in this project may come off as a replacement for future cloud interaction, it is unlikely to replace the CLI and GUI which is commonly used today. The goal of this project is not to justify replacing the existing methods, but rather to provide a fresh perspective on how interaction can be. This new perspective may be particularly valuable for a newer generation of developers who have a different understanding of cloud technology. Developers who are accustomed to using CLI as their primary interaction method may be less inclined to accept VR as a potential alternative. This is understandable since their mental representation of the cloud is based on the CLI, where they have an in-depth understanding of command syntax and error handling. The target audience for this project could be the younger generation of developers who are still developing their mental representation of the cloud. Just as a skilled car mechanic may not need a new tool for inspecting vehicles if they already know what to look for, CLI veterans may not need to be introduced to VR since they already have a well-developed understanding of the CLI. However, the younger generation of developers who are more open to exploring new perspectives may be more receptive to the idea of VR as an interaction tool.

6.2 VR as a useful learning tool

Although VR may not be ideal for experienced users, it can still have specific applications. In this project, the main goal was to create a tool for cloud management. However, it's worth noting that VR has great potential in education, specifically for novice users. The new generation of cloud developers is more open-minded about technology, but their mental map of how to interact with the cloud is yet to be fully developed. VR can be used to teach novice developers concepts surrounding the cloud and its architecture. As seen in the background chapter, VR is a widely used simulation tool specifically for training, and it can be utilized to showcase difficult subjects using visuals to grasp concepts that are difficult to understand on paper. For instance, VR can be used to teach concepts such as Docker and Kubernetes, where a student and teacher can be together in a virtual space using interactive objects and imagery to explore these subjects. VR can also go beyond cloud and be used to teach general computer science concepts, such as object-oriented programming. It may seem surprising that this use case for VR in education wasn't considered earlier, but with an understanding of cloud technology, it's natural to investigate how to improve already existing technologies, which can make it easy to overlook the already good case that VR can be used for in teaching. This new approach to education shows a good example of how reducing complexity is crucial when trying to teach something new, and it's where this project may be the most useful. While this may not have been initially considered in the project, it opens up room for further exploration into the realm of VR and education, which could be a project in itself.

6.3 Towards intuitive cloud interactions through VR

The goal of this VR project is not to replace traditional methods of interacting with the cloud such as CLI or GUI, but it is important to acknowledge that VR has the potential to provide a new perspective on cloud interaction through its immersive visualization. As showcased with the machinist and the god perspective, they offer a unique way to immerse users in a virtual environment and convey complex data through unconventional means. Through its visual capabilities, VR enables users to experience data in a more direct and interactive way, allowing them to engage with the underlying components of the data in a more intuitive and natural manner than traditional methods allow.

In addition, it is plausible that VR could simplify complex tasks that may be more challenging to execute through a CLI or GUI, due to the more direct and intuitive nature of VR interaction. For example, consider the key and lock analogy. By enabling the visual aspects of VR, it is possible to communicate complex concepts more efficiently and effectively than relying solely on text-based information of the CLI. With images and colors, information can be portrayed in a way that users can naturally understand and process. When moving beyond a text-based CLI into a visualized environment, complex ideas and data can be conveyed more naturally, making it easier for users to comprehend.

Similar efforts can be seen in other literature related to cloud management, such as in the works of Johan Findstadsveen[4] and Gaute Borgenholt[1]. By leveraging the familiar and relatable concepts of animal herds and theater, respectively, these authors simplify various aspects of cloud environments in a manner that is similar to this project. This approach is effective because it compares something familiar with something challenging to understand, making it more accessible to a wider audience. In their own way, they are able to reduce complexity in a way that feels intuitive to most readers.

While it is currently challenging to confirm if VR is the optimal way to interact with the cloud, exploring this topic is worthwhile as there is limited literature on the combination of VR and the cloud. Even though, some examples exist on reducing cloud complexity in general, more can be said about reducing cloud complexity utilizing VR. Having explored this project and showcased that combining these two fields is feasible. Proves that this topic should be allowed the opportunity for further investigation, as it's undoubtedly certain that VR and it visualization capabilities can enhance aspects of cloud interaction. Thus, this project presents an opportunity to gain further insights into the potential benefits and limitations of using VR for cloud interaction.

The objective of this project was to address the problem statement: "Design and develop a VR interface in order to explore cloud interaction with regard to reduced complexity." Which required multiple stages of exploration. The first stage involved identifying a suitable cloud SDK that could be integrated with VR. This stage included testing different cloud providers and experimenting with trial and error to determine a feasible candidate. Once a suitable SDK was identified, the next step was to develop a definition list that could bridge the concepts of VR and cloud computing. This helped to determine the appropriate integration points for creating the prototypes. Two prototypes were developed by combining the knowledge gained from SDK testing with Unity to create workable models that accurately reflected the definition list. After confirming that the prototypes were technically sound, the next step was to discuss the original problem statement and identify areas where complexity is being reduced. This was a challenging journey that required several steps to be taken, culminating in a point where the problem statement was thoroughly addressed, and opportunities for further development could be explored.

6.4 The project journey

Writing an exploratory thesis can be challenging, as it often involves exploring a problem statement without a clear, definitive path. The uncertainty of not having a conclusive answer can make exploratory research daunting. However, the beauty of exploratory research lies in the journey of discovery, which can lead to new insights and ideas. For instance, in this project, exploring the connection between VR and the cloud was a challenging task due to the vast differences between the two subjects. Nevertheless, the process of discovering how these subjects could be connected was both tedious and necessary for the success of the project. Ultimately, the ability to reflect on and adapt to new knowledge made this project an exciting and rewarding experience.

The key to unlocking an understanding of this alignment was gained through researching the perspectives of users. By taking into consideration how the user would experience the virtual world, it became clear how they would interact with the cloud using VR. This shift in focus allowed for a better understanding of how the various pieces of the project fit together, making it easier to envision how the entire application would function. With this newfound perspective, the pieces of the puzzle fell into place, enabling the opportunity to concentrate on making the transformation a reality.

When planning the project, the goal was to create a tool that would simplify the way cloud users interact with their systems. This objective remained unchanged from start to finish, as the focus was on developing a cloud management tool that would do that simplification of interaction. However, as the project was discussed further, a paradigm shift occurred, and it became clear that the creation had evolved into something different, a learning tool. This new direction was an interesting revelation that took almost the entire duration of the project to uncover. Given more time to test the project, the new path of pursuing the potential of a learning tool would have been explored. The shift from a cloud management tool to a learning tool was a significant step, and it opened up a field of possibilities. This experience taught the importance of remaining open to new ideas and opportunities, even when they deviate from the original plans, as they can lead to unexpected outcomes.

One of the most exciting aspects of the exploratory method is that it can create a journey from one starting point to an answer that is entirely different from what is expected. This process of exploration and discovery is what makes this method so valuable. The journey itself is essential because it allows the exploration of different options and possibilities, which ultimately leads to finding the best outcome. Without this journey, the answer may have never been realized, and the opportunity for growth and learning would have been missed.

7 Conclusion

In conclusion, this project aimed to investigate the potential of Virtual Reality as an alternative interface for cloud interaction, with the objective of simplifying the complexity surrounding cloud interaction. By studying the evolution of cloud interaction over the years and comprehending the capabilities of VR technology in conjunction with the cloud, a foundation was established to explore the integration of these two technologies. Developing an appropriate approach was crucial in determining the optimal path for combining VR and the cloud. Through careful consideration, an exploratory research path was chosen, offering a more flexible and open-ended approach. Once the approach was established, the next step involved identifying the most effective method for integrating the cloud with VR. This entailed selecting a cloud provider with the best-suited software development kit for connecting VR to the cloud. The choice of an appropriate SDK was pivotal in ensuring a connection between VR and cloud services.

Significant time was dedicated to the technical preparations and exploration of the project. However, a considerable amount of time was also invested in comprehending the metaphorical connection required to unify the realms of VR and the cloud. The brainstorming process played a significant role in achieving an understanding of how these two worlds could be integrated, ultimately transforming the VR application into a cloud tool. Determining the most effective perspectives to employ and creating an immersive environment for users were critical steps in establishing the desired connection. As a result of these efforts, prototypes could finally be developed.

The path of understanding the intricacies of the game engine and programming language proved to be a formidable challenge. The objective was to develop a VR application that could communicate with the cloud in real-time, which demanded ample time and effort to achieve a complete realization. Integrating these two contrasting technologies seamlessly was essential for their practicality. The resulting prototypes strived to create an immersive world, leveraging creative elements and visualization to streamline cloud interaction. It became evident that this approach reduced certain complexities along the way.

However, while further discussing complexity reduction, the prototype underwent a significant transformation. Initially conceived as a means of utilizing VR for cloud management, it became apparent that this direction was less desirable. Instead, the potential of VR as a powerful learning tool, particularly for challenging subjects within computer science and cloud technology, came to the forefront. This realization was both captivating and revelatory, and it was only by undertaking this project that such potential could be understood. As the project unfolded, the fusion of VR and cloud technology emerged as a viable and exciting technical achievement. The project effectively demonstrated how VR and its visualization capabilities could simplify complex concepts, making them more accessible and understandable. Despite successfully showcasing the technical aspects and unveiling its newfound potential as a learning tool, there is still an ongoing journey to fully explore the vast territory of VR and the cloud. By continuing to delve into this realm, further discoveries and advancements can be made.

7.1 Future work

The new-found knowledge of the learning tool path suggests exciting opportunities for further exploration. The availability of VR-enabled connections to cloud services opens up vast potential for enriching the educational journey of students studying cloud subjects. By harnessing the power of visualization and real-time data interaction, complex concepts can be rendered more easily understandable. A key future endeavor would therefore involve investigating the efficacy of VR in the learning process and developing learning environments for students to interact in. Through testing, it could be possible to ascertain whether VR can truly enhance students' learning experience. The creation of a VR-based learning tool stands as an intriguing project for the future, offering more profound insights and potentially becoming an important aspect of future education.

References

[1] Gaute Borgenholt, Kyrre Begnum, and Paal E. Engelstad. "Audition: a DevOps-oriented service optimization and testing framework for cloud environments". 2013. URL: https://oda.oslomet.no/oda-xmlui/bitstream/handle/10642/1944/1104071.pdf?sequence=1&isAllowed=y.

[2] World Wide Web Consortium. *Web Content Accessibility Guidelines (WCAG)*. 2008. URL: https://www.w3.org/WAI/standards-guidelines/wcag/.

[3] Datareportal. *Digital 2021: Global Overview Report*. 2021. URL: https://datareportal.com/global-digital-overview.

[4] Johan Findstadsveen. "If your Webserver was an Animal, how would you describe it?" May 2023. URL: https://www.duo.uio.no/bitstream/handle/10852/8960/Finstadsveen.pdf?sequence=1&isAllowed=y.

[5] GlobeNewswire. *Demand for Global Cloud Computing Market Size & Share to Surpass USD 1025.7 Bn by 2028, Exhibit a CAGR of 15.80% — Cloud Computing Industry Trends, Dynamics, Growth, Value Analysis, Forecast*. June 2022. URL: https://www.globenewswire.com/en/news-release/2022/06/22/2467017/0/en/Demand-for-Global-Cloud-Computing-Market-Size-Share-to-Surpass-USD-1025-7-Bn-by-2028-Exhibit-a-CAGR-of-15-80-Cloud-Computing-Industry-Trends-Dynamics-Growth-Value-Analysis-Forecast.html.

[6] Shreya Mattoo. *Virtual Reality: The Promising Future of Immersive Technology*. 2023. URL: https://www.g2.com/articles/virtual-reality.

[7] Microsoft. *AZ-400: Designing and Implementing Microsoft DevOps Solutions*. 2023. URL: https://learn.microsoft.com/en-us/certifications/exams/az-400/.

[8] David Mytton. *Why does the interface matter? Cloud provider consoles compared*. May 2017. URL: https://www.infoworld.com/article/3193339/why-does-the-interface-matter-cloud-provider-consoles-compared.html.

[9] PuttyGen. *Best Windows Terminal Emulators in 2022*. https://www.puttygen.com/windows-terminal-emulators. 2022.

[10] Harini Sampath, Alice Merrick, and Andrew Macvean. “Accessibility of Command Line Interfaces”. In: (May 2021). URL: https://dl.acm.org/doi/fullHtml/10.1145/3411764.3445544.

[11] Scality. *The History of Cloud Computing*. 2020. URL: https://solved.scality.com/solved/the-history-of-cloud-computing/.

[12] Amazon Web Services. *Amazon S3 Documentation: Deleting an Amazon S3 Bucket*. 2023. URL: https://docs.aws.amazon.com/AmazonS3/latest/userguide/example_s3_DeleteBucket_section.html.

[13] Amazon Web Services. *Amazon Simple Storage Service Developer Guide*. 2006. URL: https://cdn.awstatic.com/pub/s3-developer-guide.pdf.

[14] Amazon Web Services. *AWS Products*. 2023. URL: https://aws.amazon.com/products/?aws-products-all.sort-by=item.additionalFields.productNameLowercase&aws-products-all.sort-order=asc&awsf.re%5C%3AInvent=*all&awsf.Free%5C%20Tier%5C%20Type=*all&awsf.tech-category=*all.

[15] Amazon Web Services. *Create an Amazon S3 bucket using an AWS SDK*. 2023. URL: https://docs.aws.amazon.com/AmazonS3/latest/userguide/example_s3_CreateBucket_section.html.

[16] Amazon Web Services. *List Amazon S3 buckets using an AWS SDK*. 2023. URL: https://docs.aws.amazon.com/AmazonS3/latest/userguide/example_s3_ListBuckets_section.html.

[17] Alex Sterrantino. “The Many Industries that Use Unity”. In: (Dec. 2020). URL: https://medium.com/@alexsterrantino/the-many-industries-that-use-unity-dd5663f8bc77.

[18] Tate. *Abstract art*. 2023. URL: https://www.tate.org.uk/art/art-terms/a/abstract-art.

[19] XR Today. *How Do Virtual Reality Headsets Work?* 2023. URL: https://www.xrtoday.com/vr/how-do-virtual-reality-headsets-work/.

[20] Anne Trafton. *In the blink of an eye*. 2014. URL: https://news.mit.edu/2014/in-the-blink-of-an-eye-0116.

[21] Rhys Ulerich. *Visuals 60,000 times faster*. 2011. URL: https://rhdeepexploration.wordpress.com/2011/12/05/visuals-60000-times-faster/.

[22] OSSO VR. *How it works*. 2023. URL: https://www.ossovr.com/how-it-works.

[23] Wikipedia. *API*. 2023. URL: https://en.wikipedia.org/wiki/API.

8 Appendix

8.1 Prototype scripts: Regular user

8.1.1 CreateBucket.cs

```
using Amazon.Runtime.CredentialManagement;
using Amazon.Runtime;
using Amazon.S3;
using UnityEngine;
using Amazon.S3.Model;
using System;
using Amazon;
using UnityEngine.Events;
using TMPro;
using UnityEngine.UI;

public class CreateBucket : MonoBehaviour
{
    private AWSCredentials awsCredentials;
    private const string bucketName = "
        unityoptimaltestingsite";
    private static readonly RegionEndpoint bucketRegion =
        RegionEndpoint.USEast1;
    private static AmazonS3Client s3Client;
    public GameObject button;
    public UnityEvent onPress;
    public UnityEvent onRelease;
    GameObject presser;
    bool isPressed;
    public Button yourButton;
    public TMP_Text yourText;

    void Start()
    {
        Button button = yourButton.GetComponent<Button>();
        button.onClick.AddListener(TaskOnClick);

        var chain = new CredentialProfileStoreChain();

        if (chain.TryGetAWSCredentials("unity-test", out awsCredentials))
        {
            Debug.Log("Loading CreationClient!");
            s3Client = new AmazonS3Client(awsCredentials, bucketRegion);
        }
        else
        {
            Debug.Log("An error occured with CreationClient !!");
        }

        Debug.Log("CreationClient loaded succsessfully!");

        isPressed = false;
    }

    private void OnTriggerEnter(Collider other)
    {
        if (!isPressed)
        {
            button.transform.localPosition = new Vector3(0, 0.003f, 0);
            presser = other.gameObject;
            onPress.Invoke();
            isPressed = true;
        }
    }

    private void OnTriggerExit(Collider other)
    {
        if (other.gameObject == presser)
        {
            button.transform.localPosition = new Vector3(0, 0.015f, 0);
            onPress.Invoke();
            isPressed = false;
        }
    }

    public void TaskOnClick()
    {
        CreateBucketAsync();
    }

    public async void CreateBucketAsync()
    {
        try
        {

            var putBucketRequest = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true
            };

            PutBucketResponse putBucketResponse = await s3
                Client.PutBucketAsync(putBucketRequest);

            TMP_Text txt = yourText.GetComponent<TMP_Text>()
                ;
            txt.text = "You have created bucket:" +
                bucketName;

            Debug.Log("Bucket was created:" + bucketName);
        }
        catch (AmazonS3Exception e)
        {
            Debug.Log($"Error encountered on server when
                writing an object '{ e.Message}'");
        }
        catch (Exception e)
        {
            Debug.Log($"Unknown encountered on server when
                writing an object '{ e.Message}'");
        }
    }
}
```

8.1.2 ListBucket.cs

```
using UnityEngine;
using Amazon.S3;
using TMPro;
using Amazon.Runtime.CredentialManagement;
using Amazon.Runtime;
using UnityEngine.UI;
using UnityEngine.XR.Interaction.Toolkit;
using Amazon.S3.Model;
using Amazon;
using System;
using UnityEngine.Events;

public class ListBucket : MonoBehaviour
{
    private AmazonS3Client s3Client;
    private AWSCredentials awsCredentials;
    public Button yourButton;
    public TMP_Text yourText;
    public GameObject button;
    public UnityEvent onPress;
    public UnityEvent onRelease;
    GameObject presser;
    bool isPressed;
    private bool gameObjectSpawnLoopRunner = true;

    void Start()
    {
        var chain = new CredentialProfileStoreChain();

        if (chain.TryGetAWSCredentials("unity-test", out awsCredentials))
        {
            Debug.Log("Loading ListClient!");
            s3Client = new AmazonS3Client(awsCredentials);
        }
        else
        {
            Debug.Log("An error occured with ListClient!");
        }

        Debug.Log("ListClient loaded succsessfully!");

        isPressed = false;

        Button button = yourButton.GetComponent<Button>();
        button.onClick.AddListener(TaskOnClick);
    }

    private void OnTriggerEnter(Collider other)
    {
        if (!isPressed)
        {
            button.transform.localPosition = new Vector3(0, 0.003f, 0);
            presser = other.gameObject;
            onPress.Invoke();
            isPressed = true;
        }
    }

    private void OnTriggerExit(Collider other)
    {
        if (other.gameObject == presser)
        {
            button.transform.localPosition = new Vector3(0, 0.015f, 0);
            onPress.Invoke();
            isPressed = false;
        }
    }

     public void TaskOnClick()
    {
        ListBucketAsync();
    }

    public async void ListBucketAsync()
    {
        TMP_Text txt = yourText.GetComponent<TMP_Text>();

        var currentBuckets = await s3Client.ListBucketsAsync
            ();
        Debug.Log("Number of buckets: " + currentBuckets.
            Buckets.Count);

        txt.text = "Number of buckets: " + currentBuckets.
            Buckets.Count.ToString();

        if (gameObjectSpawnLoopRunner == true)
        {

            for (int myNum = 0; myNum < currentBuckets.
                Buckets.Count; myNum++)
            {
                ListBucketsResponse response = await s3
                    Client.ListBucketsAsync();

                GameObject cube = GameObject.CreatePrimitive
                    (PrimitiveType.Cube);

                foreach (S3Bucket bucket in response.Buckets
                    )
                {
                    cube.name = bucket.BucketName;
                }

                cube.AddComponent<Rigidbody>();
                cube.AddComponent<XRGrabInteractable>();
                cube.AddComponent<Collider>();
                cube.transform.position = new Vector3(1.8308
                    45f, 0.6412615f, 6.638258f);
            }

            gameObjectSpawnLoopRunner = false;
        }
        else
        {
            Debug.Log("All buckets have been spawned!");
        }

    }
}
```

8.1.3 DeleteBucket.cs

```
using Amazon.S3.Model;
using Amazon.S3;
using System;
using UnityEngine;
using Amazon.Runtime.CredentialManagement;
using Amazon.Runtime;
using Amazon;
using TMPro;

public class DeleteBucket : MonoBehaviour
{
    private AWSCredentials awsCredentials;
    private string bucketName;
    private static readonly RegionEndpoint bucketRegion =
        RegionEndpoint.USEast1;
    private static AmazonS3Client s3Client;
    public TMP_Text DeletionText;

    void Start()
    {
        var chain = new CredentialProfileStoreChain();

        if (chain.TryGetAWSCredentials("unity-test", out
            awsCredentials))
        {
            Debug.Log("Loading DeleteClient!");
            s3Client = new AmazonS3Client(awsCredentials,
                bucketRegion);
        }
        else
        {
            Debug.Log("An error occured with DeleteClient!!"
                );
        }

        Debug.Log("DeleteClient loaded succsessfully!");
    }

    private void OnCollisionEnter(Collision collision)
    {
        TMP_Text txt = DeletionText.GetComponent<TMP_Text>()
            ;
        DeleteBucketAsync(collision.gameObject.name);
        Destroy(collision.gameObject);
        txt.text = "Bucket have been deleted!";

    }

    public async void DeleteBucketAsync(string bucketName)
    {
        try
        {
            var request = new DeleteBucketRequest
            {
                BucketName = bucketName,
            };
```

```

                var response = await s3Client.DeleteBucketAsync(
                    request);
            }
            catch (AmazonS3Exception e)
            {
                Debug.Log($"Error encountered on server when
                    deleting object '{e.Message}'");
            }
            catch (Exception e)
            {
                Debug.Log($"Unknown encountered on server when
                    deleting object '{e.Message}'");
            }
        }
}
```

8.2 Prototype scripts: Admin user

8.2.1 ColorChanger.cs

```
using System;
using System.Collections;
using System.Threading;
using UnityEngine;

public class ColorChanger : MonoBehaviour
{
    public Material material;
    public float value;
    public Color colorA;
    public Color colorB;
    private float lastValue;
    public float changeInterval;

    void Start()
    {
        StartCoroutine(RunMethod());
        lastValue = value;
    }

    IEnumerator RunMethod()
    {
        while (true)
        {
            colorchanger();
            yield return new WaitForSeconds(changeInterval);
        }
    }

    public float Randomizer()
    {
        System.Random random = new();
        float randomNumber = random.Next(0, 2);
        return randomNumber;
    }

    public void colorchanger()
    {
        value = Randomizer();

        if (value != lastValue)
        {
            float lerpValue = Mathf.InverseLerp(0f, 1f, value);
            material.color = Color.Lerp(colorA, colorB, lerpValue);
            lastValue = value;
        }
    }
}
```