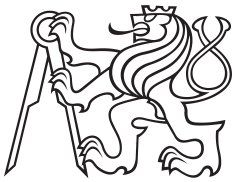
Master Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Radioelectronics

Simulation and Evaluation of Image Impairment Impact on Quality of Experience in Virtual Reality Environments

Bc. Jakub Špaňár

Supervisor: Ing. Karel Fliegel, Ph.D.
January 2024

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Špaňár Jakub** Personal ID number: **474522**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Circuit Theory**

Study program: **Medical Electronics and Bioinformatics**

Specialisation: **Signal processing**

II. Master’s thesis details

Master’s thesis title in English:

Simulation and Evaluation of Image Impairment Impact on Quality of Experience in Virtual Reality Environments

Master’s thesis title in Czech:

Simulace a hodnocení vlivu zkreslení obrazu na kvalitu zážitku v systémech pro virtuální realitu

Guidelines:

Provide a state-of-the-art review of simulation and evaluation of various image impairments that impact the Quality of Experience (QoE) in virtual reality environments, including existing software packages. For an available virtual reality imaging system, develop software tools to perform simulations of selected image impairments, and tools to automate subjective experiments, including their evaluation. Verify the functionality of the implemented tools in pilot experiments with a group of observers and compare them with published techniques.

Bibliography / sources:

[1] Brunnström, K., Dima, E., Qureshi, T., Johanson, M., Andersson, M., Sjöström, M., Latency impact on Quality of Experience in a virtual reality simulator for remote control of machines, Signal Processing: Image Communication, 89, 2020.
[2] Huang, M., Shen, Q., Ma, Z., Bovik, A. C., Gupta, P., Zhou, R., Cao, X., Modeling the Perceptual Quality of Immersive Images Rendered on Head Mounted Displays: Resolution and Compression, IEEE Transactions on Image Processing, 27 (12), 2018.
[3] Vlahovic, S., Suznjevic, M., Skorin-Kapov, L., A survey of challenges and methods for Quality of Experience assessment of interactive VR applications, Journal on Multimodal User Interfaces, 16 (3), 2022.

Name and workplace of master’s thesis supervisor:

Ing. Karel Fliegel, Ph.D. Department of Radioelectronics FEE

Name and workplace of second master’s thesis supervisor or consultant:

Date of master’s thesis assignment: **13.02.2023** Deadline for master's thesis submission: **09.01.2024**

Assignment valid until: **22.09.2024**

Ing. Karel Fliegel, Ph.D.	doc. Ing. Radoslav Bortel, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the master’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master’s thesis, the author must state the names of consultants and include a list of references.

. | Date of assignment receipt | Student’s signature

Acknowledgements

My deepest thanks go to my supervisor, Ing. Karel Fliegel, Ph.D., whose expertise and encouragement were instrumental in the development of this work.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with methodical instructions for observing the ethical principles in the preparation of university theses.

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

In Prague, 8. January 2024

Abstract

This diploma thesis develops software for investigating the impact of diverse image impairments on Virtual Reality (VR) Quality of Experience (QoE). Featuring an innovative Unity application for flexible simulation customization, the study explores technical and visual distortions. By employing various features and methods, the application was able to simulate latency, lag, image noise, color vision deficiency, and blind spots. As a proof of concept pilot experiment was conducted on the mentioned application. Pilot experiments with 25 participants yield subjective QoE scores and eye movement data, facilitating detailed motion analyses and heatmaps. Additionally, statistical analyses are conducted to compare the effects of impairments on QoE.

Keywords: Virtual reality (VR), Quality of Experience (QoE), image impairment, Unity, latency, lag, noise, flicker, color vision deficiency, blind spots

Supervisor: Ing. Karel Fliegel, Ph.D.

Abstrakt

V této diplomové práci byl vytvořen software na zkoumání vlivů různých zkreslení obrazu na kvalitu zážitku (QoE) ve virtuální realitě (VR). S využitím inovativní aplikace v Unity pro flexibilní přizpůsobení simulací, studie zkoumá technické a vizuální zkreslení. Použitím různých prvků a metod byla aplikace schopná simulovat zpoždění, zásek, šum v obraze, poruchy vnímání barev a slepá místa. Jako důkaz o funkčnosti softwaru byl proveden pilotní experiment. Pilotní experimenty s 25 účastníky poskytují subjektivní hodnocení QoE a data o pohybu očí, umožnující detailní analýzy očního pohybu a heatmapy. Kromě toho jsou provedeny statistické analýzy k porovnání efektů zkreslení na QoE.

Klíčová slova: Virtuální realita (VR), Kvalita zážitku (QoE), zkreslení obrazu, Unity, zpoždění, zásek, šum, blikání, porucha vnímaní barev, slepé skvrny

Překlad názvu: Simulace a hodnocení vlivu zkreslení obrazu na kvalitu zážitku v systémech pro virtuální realitu

Contents

Figures

Tables

Chapter 1

Introduction

Virtual Reality (VR) stands as a pivotal advancement in interactive entertainment technology, rapidly evolving with the emergence of unique applications tailored exclusively for VR use. As VR becomes increasingly dominant in realms of enjoyment and education, a profound understanding of the diverse factors shaping user experiences becomes imperative. This thesis embarks on the development of software tools to perform simulations of image impairments and to automate subjective experiments.

The initial focus is on the user's enjoyment and Quality of Experience (QoE) within VR. Beyond purely technical aspects, numerous factors intricately contribute to the overall QoE. Chapter 2 describes the essence of QoE and delves into various metrics employed to measure QoE, accompanied by recommendations for conducting subjective tests.

Virtual reality refers to a complex technology that ranges from using projection on a wall to haptic feedback gloves. Therefore, Chapter 3 explores VR technology itself, highlighting the myriad forms of digital environment display, including alternatives like augmented reality. The ways users observe and interact with these rendered environments are also provided. While VR technology caters to complex needs, from interactive lessons to virtual surgery, it is not without challenges that demand attention.

Chapters 4 and 5 provide comprehensive insights into image impairments. They delve into the explanation of what these impairments are and their origins. Additionally, a deliberate decision was made to encompass image impairments that do not originate from the VR system, such as eye defects. The impact of these impairments on users is further examined in Chapter 5 through an exploration of relevant research studies.

In Chapter 6, a comprehensive exploration of the Unity software and its utilized features is provided. The main mechanics of the system, coupled with an expected timeline, are thoroughly explained, offering a clear roadmap of the project. A significant portion of the chapter delves into elucidating the system architecture, detailing the role of each subsystem within the overall framework. Special emphasis is placed on clarifying the methods employed to simulate various image impairments, a crucial aspect of the system's

functionality. Furthermore, the chapter candidly addresses the limitations inherent in the proposed approach, providing a well-rounded understanding of the system's capabilities and constraints.

Chapter 7 provides a detailed account of the pilot experiment, explaining the methods employed to design and execute the study. This encompasses insights into the tested sequence, the procedural instructions conveyed to participants, and the characteristics of the participants themselves. The concluding section of the chapter is dedicated to the comprehensive analysis of the data collected during the experiment.

Chapter 2

Quality of Experience

The sudden growth in digital technology in the last century led to an increase in applications and services that people use daily. However, not all of these applications and services provide the same level of value to one's life. Thus various applications focusing on the same goal can cause distinctive user experiences in terms of satisfaction, enjoyment, etc.

To quantify the subjective experience of the user, the Quality of Experience (QoE) concept has been created. Although there are several similar definitions of QoE, this thesis will follow this definition: " The overall acceptability of an application or service, as perceived subjectively by the end-user " as defined in [33]. The study also notes that the QoE is influenced by all aspects of the end-to-end system, while overall acceptability can be influenced by user expectations and context.

A similar concept to QoE exists and that is Quality of Service (QoS). QoS is the quality of a given service to satisfy the stated needs of the user. This domain has been popular for over 20 years and service providers are accustomed to supporting it [46]. It is important to note that QoE is an extension of QoS, not a replacement.

QoE is a complex measurement of many influencing factors (IF). IF refers to anything that can affect users' perception of quality. However, these IF cannot be considered independent, as they can have any sort of connection between them, thus affecting their impact on a particular outcome. They can be grouped into three categories [9, 50].

Human IFs are complex and highly connected. They are related to the mental, emotional, and physical state of the user. These characteristics can be static like gender, age, and visual and auditory acuity, or dynamic such as the user's mood, motivation, and attention level. Systems cannot be created for a single user, so users are typically categorized into categories that are important for the evaluation.

System IFs refer to properties of the technology used. System IFs can be subdivided into four categories content-related, Media-related, Network-related, and Device-delated. That includes media configuration, system

specifications, device capabilities, etc. Factors are often specified in advance, resulting in documentation listing the features.

Context IFs are situational attributes, that describe a user's environment. These include any attributes that might influence outcomes like acoustic and lighting conditions, the way of interacting with a system, and the time of day. Context IFs can be divided into smaller categories, such as task context, social context, technical and informational context, temporal context, and economic context.

The unpredictable nature of human IF on MOS is problematic but some studies focus on minimizing the impact. The idea of not using MOS but instead QoE distribution was introduced. The study [39] was based on a framework of multi-nominal distribution. The metric of QoE fairness was created to capture the scale of participants rating the same category to rating uniformly. Exemplary evaluations based on QoE distribution were done and showcased advantages over normal MOS.

In study [19] the researchers proposed the idea of using machine learning to improve QoE. They proposed quality of experience data fusion (QoEDF). QoEDF uses a two-layer data fusion algorithm for predicting MOS. This method had higher accuracy than traditional methods. QoEDF also self-updates to dynamically change the flow of multimedia Internet of Things (MIoT) network. That works on adjusting the routing and allocation bandwidth to achieve an optimized average MOS. The method QoEDF is capable of improving QoE in terms of MOS and the authors expect an average increase of MOS by 0.5. This paper could lead to a better user experience for MIoT applications and services such as video streaming, audio playback, and real-time communication. Additionally, there is the opportunity for the method to be applied outside of the MIoT.

The studies that delve into the impact of various factors on QoE are further explored in Chapter 5.

2.1 Objective Measurement

To measure the effect of factors and evaluate QoE subjective and objective metrics are used. The objective methods can be quantified with measurement tools, usually involving the reference as a comparison. Examples of tools commonly used in the field of image and video processing for quantifying distortion are Peak Signal-to-Noise Ratio, Video Quality Metric, and Structural Similarity. The subjective methods are based on collecting data from users by directly asking or filling up any kind of surveys etc. The standard measurement for subjective metric became the Mean Opinion Score (MOS), the user rate their experience on the five-point discrete scale, as they see fit. However, user bias must be considered because mood swings or prolonged testing can shift the scale [23].

Similarly, images can undergo various forms of distortion, such as color distortion. In evaluating the QoE for color distortion, there are several metrics available. Numerous studies present comparisons of these metrics, as demonstrated by [14]. In one such study, nine distinct color difference measures were assessed, and CIEDE2000 consistently yielded the highest values. Specifically engineered for discerning small color differences, CIEDE2000 excels in accurately measuring color discrepancies.

2.2 Subjective Tests

When evaluating quality, it is crucial to recognize its inherently subjective nature. While opinions on quality may vary, certain features tend to be universally more pleasing, and we can quantify this through subjective testing. This section will provide a concise overview of the recommendations outlined in ITU-T P.919 for subjective test methodologies applied to 360-degree images in VR [16].

To ensure a comfortable viewing experience, the presented images or videos should refrain from incorporating adult or violent imagery. The optimal viewing duration is suggested to be 10 seconds, with the entire continuous testing session lasting less than 25 minutes and the overall testing period capped at 1.5 hours.

Subjects should be allotted sufficient time to assess the sequence, typically utilizing a commonly employed five-level rating scale (ranging from excellent to bad).

Simulator sickness is an undesirable outcome. Therefore, it is advisable to conduct sickness assessments both before and after the subjective test. Additional assessments can be incorporated as needed, with the evaluation primarily relying on the simulator sickness questionnaire, elaborated upon in the subsequent simulator sickness section (7.3.1).

The testing environment should foster a calm and non-distracting atmosphere, with the test moderator remaining present to address any potential simulator sickness incidents.

Including dummy or reference sequences becomes imperative to ensure comprehensive score coverage. These sequences should be presented in a pseudo-random order, and the testing pattern should follow a sequence of play, pause for rating, and repeat.

Subjects participating in the experiment should be well-informed about its fundamentals, and instructions should be tailored to the specific requirements of the test.

Chapter 3

Virtual Reality Technology

In this chapter, our goal is to cultivate a comprehensive understanding of the technological foundations that support immersive experiences. Considering that virtual reality stands as a more advanced form of immersive technology in comparison to conventional flat screens, it becomes crucial to delve into its intricacies. This exploration will cover various forms of enhanced reality and their intricate interactions with users. The chapter aims to elucidate these technologies, shedding light on their applications, limitations, and the challenges they present.

3.1 Virtuality

To experience the world around us we count on our senses to give us an accurate understanding of our surroundings. In past decades researchers managed to build upon the reality we see by adding context or creating new ones. To describe the spectrum of immersive technologies we use the name extended reality (ER). ER includes virtual is augmented, mixed, and mediated reality [29].

Virtual reality (VR) is the first ER that comes to mind when talking about immersive technologies. The premise of VR started in the 1950s and it aims to replace the real world with an artificial one. For this to happen the outside world is essentially blocked and the intended environment is shown via a head-mounted device with goggles or projections of stereoscopic images on walls around the user, so-called Cave. The real-life position of the user's head is translated to the virtual camera duo gyroscopes in head-mounted gears. The environment can be generated or previously captured like 360-degree images or video [8, 29].

Augmented reality (AR) similar to VR showcases the user images and videos other than reality but in the case of AR the outside world is not shut off. It uses intelligent display technology to insert generated assets into the user's view. This effect can be done through the usage of wearable, mobile, or spatial AR [28].

Another breach of ER is mixed reality (MR). Although the similarity to AR and VR, MR exists as a slider on the continuum between the real and virtual. The slider can be referred to as the "X-axis", therefore MR is sometimes called X-reality. To do so MR maps the virtual assets to the real world in real time. The user doesn't interact with the MR environment with a controller, but instead by using natural communications needs [29, 37].

Finally, there is Mediated Reality (MeR). MeR is a technology that modifies the user's vision of reality. This can be done deliberately or accidentally. The deliberate MeR was used in the study by wearing goggles, which flip the user's vision upside down. Accidental MeR happens for example when using some see-through AR, the simple fact that additional context is overlaying other results in unintentionally modified reality [38, 29].

3.2 Immersion and Presence

Another term often used in VR technology is immersion. It is described as a sensation of being surrounded by other reality, which takes our whole perception. There are said to be two key parts to immersion and that is an individual's psychological state and an objective property of a system [4].

Psychological immersion refers to the player's absorption into the world. However, a higher level of immersion doesn't necessarily correlate to a higher level of technical aspects used to create the world. For example, reading a book can be as or more immersive than some virtual reality. That is because immersion is achieved when people concentrate their attention, thoughts, and objectives on the current medium. The researchers [4] managed to find some attributes that help with psychological immersion. One of which is a multi-sensory simulation or sensory immersion, the idea behind it is that players get more involved the more their senses are engaged in overpowering the real environment's sensory information. Secondly, narrative immersion can play a vital role, because the player shifts attention to the event in the story which invokes the player's curiosity and emotional involvement. Lastly is the idea of a challenge-based immersion. By introducing the challenges of strategic planning, motor skill, or thinking players get actively engaged in completing the task ahead. However, some speculate that challenge-based immersion happens when players accept the rules of a game system while rejecting the real world, thus accepting the existence of magic for example. The other less popular idea of immersion is system immersion. Its description rejects the idea that immersion is a subjective experience and sees immersion as purely objective. The approach believes that only system characteristics determine a player's immersion. This means the more accurate and faithful representation of the real world through all sensory modalities leads to more immersion experience.

Immersion is often misunderstood for Presence. Despite the terms being highly correlated there is a difference. Presence is a sense of a player leaving

a real location and being transported into a virtual environment, in short, it is the illusion of being there. Some individuals can achieve a stronger sense of Presence. Those individuals exhibit high levels of openness, neuroticism, absorption, and extraversion, however, that could be biased at the response level. Based on the description stated above about psychological immersion, it can be perceived as being the same, however, one can exist without the other and vice versa. For example, a player, who plays an abstract game like Tetris can be immersed in gameplay but hardly feel the Presence of being in an environment where blocks fall from the sky. The other example can be listening to recorded restaurant sounds. That could create a sense of Presence but the record alone would not be sufficient enough for immersion [4, 51].

3.3 Technical aspect of VR

Users can experience a new reality in different ways, and virtual reality can take various forms, from common practices to more experimental approaches. Therefore, this section will offer information about the hardware capabilities of current VR systems, how users engage with the VR environment, and how the environment can be experienced.

3.3.1 Visual Output

The main aspect of current VR technology is to provide stereoscopic images, therefore the system needs a display.

Mobile VR is the most basic form of design, to begin with. To use mobile VR, a phone must be mounted within a customized case with lenses; this case could be as basic as cardboard or more ergonomically designed (see figure 3.1). The advantages include its cost, mobility, and lack of additional technical requirements such as a PC or wires, but it is limited in the services it can offer, which is why mobile VR typically just shows 360-degree movie images. [6].

Figure 3.1: Image of Google Cardboard[1].

A more advanced system is a Head-Mounted Display (HMD). The design works the same by displaying two different images to each eye, however, HMD

[1]Image source: `https://arvr.google.com/cardboard/`

is mounted to the head via strap or helmet, etc. HMD offers a higher quality resolution, higher frame rate, and shorter latency. Based on the provider of HMD some HMDs provide a see-through display for AR[2], and other eye-tracking capabilities such as VIVE Pro Eye (see figure 3.2). Consequently, VIVE Pro Eye is the VR used in this study. HMDs are connected to systems like PCs, gaming consoles of workstations. The connection can be wired or wireless. Although wireless HMD offers a lower technical aspect researchers found that it didn't worsen the sense of Presence and cybersickness [6, 15].

Figure 3.2: VIVE Pro Eye HMD, with base stations and controllers [3].

For a more immersive experience, researchers are still developing Haptic Feedback technology. Haptic Feedback refers to tactile sensations to users in response to their interactions with digital systems. This includes applying a force, temperature, or stimulating nerve via electrodes on the skin. Combinations of such would lead to sensations of hot/cold or the resistance of an object when holding it in VR. However, creating effective haptic feedback systems is not without its challenges, as it requires the development of specialized hardware and software that can accurately replicate real-world sensations. The main challenges of Haptic Feedback lie in miniaturization, the density of actuators, and the material properties [20].

3.3.2 Interaction with Digital Environment

The second part of VR technology is the various ways to let the system know of our actions. The main input category is controllers (see figure 3.2). In the past, the early stages of VR relied on a mouse and keyboard, but nowadays

[2]Example: `https://www.ayes.cz/en/products/microsoft-hololens-2/`
[3]Image source: `https://www.vive.com/us/product/vive-pro-eye/overview/`

handheld controllers are becoming increasingly popular. They may be wired or wireless. Controllers provide buttons and often a joystick or a touchpad. Most VR packages come with a pair of controllers, one for each hand. It is also possible to track user input via sensors in Haptic Feedback equipment. Such controllers would provide both input and output [6].

Some VR headsets already have the feature of eye-tracking. With eye-tracking integrated into the HMD, it allows the system to capture the user's gaze direction. Calibration ensures the precision of eye-tracking. It involves showing target points in different locations on the screen to calculate the subject's gaze. It is done with simple geometric calculations. This tool can be used for the analysis of where the user looked during a time, which objects were the most *interesting* etc. Of course, this system has flaws. For example, individuals wearing glasses cannot always participate, the calibration requires additional time for each subject and subjects cannot move the HMD after the calibration is done, therefore they need to sit comfortably. The researchers believe that eye tracking will allow for answering the question about human cognition and behavior. This feature has the potential to enrich gameplay by allowing users to interact more directly with objects of interest in their view [11]. VR developers can also use the knowledge of eye-tracking to implement Foveated Rendering (see Section 5.1).

Fairly new VR headsets offer hand tracking[4]. For instance, The Meta Quest is a virtual reality headset that offers hand recognition through an inside-out camera. This camera detects the position, configuration, and orientation of hands and fingers, and an algorithm is used to track movement. Hand recognition allows for basic gestures such as pinch to select, pinch and drag to scroll both horizontally and vertically, and pinch with palm facing toward the user to bring out a menu.

Although voice recognition existed for several decades already, its usage in VR has been limited so far. However, researchers have developed VR games with voice recognition. Its premise is to help children with autistic spectrum disorder with social skills without negative social consequences. The player can interact with a shopkeeper via speech and the shopkeeper answers. Further research is needed to determine whether this technology can effectively support social skill development, particularly through large-scale user studies. However, the current development of AI chatbots could lead to better and more human-like interactions with the player, thus creating a more suitable environment for social skill development [43].

Lastly is a brain-computer interface (BCI). BCI allows users to control VR environments using their brain signals, bypassing traditional input devices. This method of input is still in the early stages of development. One study carried out an experiment to train motor imagery (neural activity imagining

[4]Vive hand tracking (Jan 2024): `https://www.vive.com/au/support/focus3/category_howto/hand-tracking.html`; Meta Quest (Jan 2024): `https://www.meta.com/help/quest/articles/headsets-and-accessories/controllers-and-hand-tracking/hand-tracking/`

bodily movement) using gamification and VR embodiment. Their results showed that the method used effectively trained the participants for a basic level of motor imagery BCI [41].

3.4 Applications of VR

VR has become a popular tool for entertainment among the general public. It offers a unique way for users to experience various types of media. Gaming has become one of the most significant applications of VR, to the point that many users buy the VR system just to play VR games. As accessibility for headsets grows so will the market and the gaming aspect of VR will surely grow in the future. Other forms of media are also noteworthy such as watching videos, viewing 360-degree images, and even VR tourism duo to virtual museum visits, and Google Street View [17].

As mentioned in the previous section VR offers the possibility of education. Namely, the start of the COVID-19 pandemic forced students to participate in online meetings. VR online classes would allow the students to be present in an immersive environment to keep them engaged with the lecture. A study was also conducted on university students where they compared virtual recorded lectures. The virtual lecture featured simulated abilities that of a normal lecture together with a virtual instructor. The system can simulate questions asked by an instructor or virtual student. The questions were associated with slides and were answered by the instructor or the system would display possible answers and the user would choose, in this case, the system would also reveal the right answer. Also, the user can ask any time during the virtual lecture by typing. If the question is expected the system will provide the correct answer, if not the system will use a chatbot or find the answer online. The participating students were then given a quiz based on the material in the lecture. The result showed better performance for virtual lectures, especially for low-grade students. Also, all students found virtual lectures more enjoyable. This form of learning would also offer more freedom for students such as picking their own time and place to view lectures, the possibility to stop, skip, or go backward, and even help deaf students by providing subtitles [45, 17].

Virtual surgery, the use of VR technology to simulate surgical procedures, is gaining prominence in the medical field due to its ability to reduce the time and risk associated with traditional surgical procedures. VR is particularly useful for telesurgery, allowing surgeons in different locations to collaborate and perform surgical procedures. With the help of virtual 3D models, surgeons can successfully plan and practice surgical procedures, which provides a better way of communication among medical professionals. VR technology also allows for the development of a precise image of the patient during surgery, and surgical treatments can be interacted with using the latest virtual glasses in a 3D environment. VR technology is a powerful diagnostic tool for accurate diagnosis carried out by doctors and physicians and quickly identifies the

signs and symptoms of diseases, eliminating the requirement of computerized tomography/magnetic resonance imaging scans [21].VR also helps in physical therapy. There is a change in the way how people exercise with VR. It also plays a significant part in the routine of exercise and additionally, it showed that VR exposure therapy helps with post-traumatic stress disorder of veterans [7]. Amputee patients were similarly treated with virtual mirror therapy with tactile stimuli. Mirror therapy works by tricking the brain with the reflected mirror view of a person's missing limb, thus reducing pain after practicing. However, the effectiveness of VR technology for treating phantom limb pain had a small effect and will require further investigation [52]. Pain management trials were also conducted. The patient, who viewed VR experience designed for reducing pain reported lower pain [44]. In terms of medical training trainees demonstrated improvements in surgical skills through immersive VR in comparison to traditional training. The VR trainee group demonstrated faster completion times along with greater scores and improved accuracy. It has been debated that the most applicable use for VR training would be for residents, but experts can still benefit whenever new techniques arrive. It also helps students with confidence but self-reported improvements may contain bias. Moreover, the affordability of VR headsets compared to traditional training equipment is a major advantage. However, the most discussed limitation of such technology is the lack of realistic tactile feedback. However, it has been shown that auditory stimuli could be used as a supplement when bone drilling is involved. This technology has the potential to remodel the methods used for surgical training [30].

3.5 Challenges of VR

Throughout the VR development over the years, some challenges were solved some were discovered but one persistent challenge is everpresent. That challenge is cybersickness. Cybersickness is a subset of motion sickness in a virtual environment that users can experience while engaged in VR. It is often created by the movement in a VR environment while the real body stays stationary. The typical symptoms of cybersickness are nausea, eye strain, and dizziness but it could also manifest in blurred vision and headache. Everyone is affected by cybersickness at different levels thus it is dependent on other factors such as age, gender, the field of view, and posture. These factors can also be created by the system such as lag, flickering, or wrong calibration. Lag especially can be crucial, in a high-definition environment with many complex objects the time for computing is prolonged. To minimize the risk of cybersickness researchers have found some prevention measures. Such preventions are shrinking the field of view, building up resistance if repeated sessions of VR, shortening the sessions, and having breaks when experiencing VR for longer periods. Although the solution for cybersickness is progressing the creation of the objective measure would help in future development [13, 36].

Similarly to physical discomfort, VR also causes negative emotional consequences, which will persist beyond the gaming experience. A study was conducted when comparing negative emotional responses while playing the same context either on PC or in VR. It showcased that VR interactions lead to bigger negative emotional responses, which even lasted long after the session was over. This is also supported by the VR users survey. This finding highlighted the potential harm, which VR technology can have on the user when the user is required to engage in morally egregious behavior. Therefore VR firms are advised to mitigate harm in their design and the nature of the gameplay [25].

VR is also highly dependent on processing power both CPU and GPU. While the CPU handles tracking events and gets input from HMD and controller, the GPU takes care of display and calculations of shading, lighting, and special effects. The requirement for a good CPU depends on the purpose they will serve, for viewing 3D videos and images core such as an Intel Core i5 is sufficient, for playing VR games CPUs such as an Intel Core i7 or AMD Ryzen 7 and it is recommended that VR developers choose CPUs with even higher processing power. Unlike regular PC games, VR requires two screens to ensure a 3D view and a high refresh rate for the fluid feeling when moving. In the first releases of HMD, each eye had a resolution of 1080×1200 with 90 Hz (see figure 3.3). Some headsets also require the rendering of off-screen areas, further requiring a strong GPU. For GPU NVIDIA's GeForce RTX is sufficient, anything with higher performance will be sufficient enough but some may require an adapter. Memory also depends on the needs of the user but 8GB is considered the bare minimum. As newer VR titles are becoming increasingly complex and demanding, the recommended system requirements are constantly evolving, therefore 32GB is recommended for most systems, and for developers sometimes 64GB is needed. As the primary drive SSD is recommended, inside the VR experience the difference won't be noticeable but how fast the VR title launches depends on the speed of the drive. Since the amount of space occupied by different applications varies from 1 GB to 100 GB the storage should be above 500 GB but 1 or 2 TB is a safer choice [2].

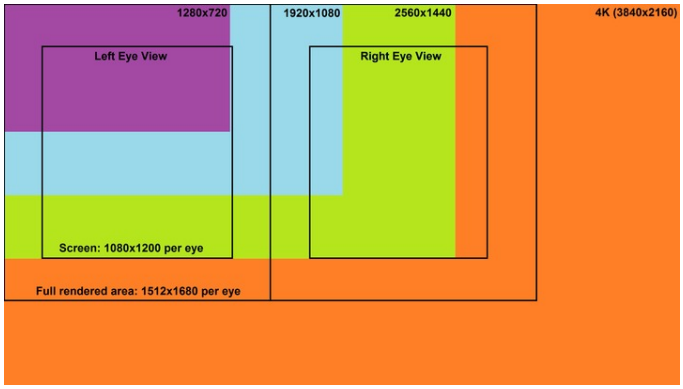


Figure 3.3: Example resolution of first-generation HTC Vive compared to common monitor resolutions (image taken from [2]).

As mentioned above, the system requirement is not budget-friendly and that excludes the headset itself and the purchasable application. This and the fact that VR is a relatively recent entertainment system results in only 2 % of Steam users having any VR system. Another disadvantage is the space needed for the full experience VR can offer since body and hand movement contribute to immersion and presence. Also, the VR market is still in its early stages meaning not many applications are developed, because of the lower number of VR users and on the other hand, not many VR users because of the lack of possible applications. However, this could likely change as companies are developing better and better HMDs like Meta Quest 2 and more games with high development costs like Half-Life: Alyx or Horizon Call of the Mountain are being created.

When it comes to ensuring a high-quality user experience in VR, several challenges need to be addressed. While QoE is essential for any VR application, some factors, as mentioned in the QoE section (2), are more relevant than others. Identifying key factors and examining their influence can help researchers adjust their study design and collect data for QoE modeling. Researchers should consider VR-induced discomfort and explore symptoms beyond cybersickness, such as digital eye strain, ergonomic factors, and cognitive aftereffects. While ethical considerations, prolonged exposure to VR would provide vital information which could be used to adapt the VR system, some researchers argue. There is a need for diversity in participants because university students are over-represented. However, individuals who are more sensitive like children and the elderly would require additional research and methodology guidelines. In terms of measurement, the subjective measurements should be more specific to VR and new technologies like eye-tracking can be used for objective measurement of user experience. Conducting research in a more realistic setting is likely to result in valuable insights and greater external validity, as opposed to a sterile laboratory environment [47].

Chapter 4

Image Impairments

This chapter aims to explain the various types of image impairments that will be simulated in the system. The term *image impairment* encompasses any factor that influences the visual perception of viewing an image. These factors will be briefly expounded upon, clarifying their nature and sources.

To facilitate a clearer distinction between impairments, they have been categorized separately for simplicity. The first section, technical impairments, will address issues stemming from inadequate technical parameters. These factors may arise due to insufficient optimization, hardware limitations, or variations in display quality.

The second section, visual impairments, delves into limitations arising from natural or health-related factors. Primarily, these encompass disorders affecting the human eye.

4.1 Technical Impairments

In this section, we delve into the technical aspects that significantly impact visual enjoyment, focusing on three key factors: latency, frame rate, and image quality. The subsequent discussion will elucidate how these factors influence the overall viewing experience, offering a comprehensive understanding of their nature.

4.1.1 Latency

Latency refers to the duration between issuing a command and witnessing the corresponding response, encompassing the time taken to transmit the command, the time for it to travel, and the time required for task processing.

Latency is highly dependent on several factors which contribute to the overall latency. Firstly, is the hardware. The processing power, graphics card, and other parts all affect the speed at which the system can render and display images. Secondly the design of the application. Of course, every action done

in VR will require a certain level of processing power but optimizing the design will avoid redundant computing power and thus guarantee minimal possible delay. Also, the more complex the environment is the longer the latency will be [22]. In the case of online VR applications network conditions also play a role. The network bandwidth, latency, and packet loss all contribute to increased latency. The HMD design also contributes to latency. For example, if the HMD is wireless the transfer of data will again take longer time. Similar to HMDs are the input devices. One example can be seen in the comparison between buttons and joysticks, both frequently integrated into VR controllers. Joysticks, offering continuous input signals and functioning in multiple directions, necessitate the interpretation of signals from various sensors. In contrast, buttons generate discrete signals, simplifying the processing task. On top of that more advanced input devices like voice or gesture recognition could contribute to overall system latency due to their higher computing power in comparison to simple button triggers. Lastly, many forms of environmental interference impact latency in a negative way.

4.1.2 Frame Rate

Frame rate refers to the number of still images, or frames, that are displayed per second in a video or animation. A higher frame rate typically results in smoother and more realistic motion, while a lower frame rate may result in choppy or stuttering motion. Frame rate is measured in frames per second (fps) and can vary depending on the medium and application. For example, movies are typically shot and displayed at 24 fps, while video games may run at 60 fps or higher for smoother gameplay. Higher frame rates also typically require more processing power and storage capacity. Frame rate is however somewhat connected to latency. That is there always will be a minimal delay that happens between two frames. In VR newer HMDs have usually 90 fps which is roughly 11 ms [35]. This delay can be divided into two parts. Firstly screen refresh rate is fixed and it is the rate at which the display receives and updates the image, Secondly, the response time is the time the physical pixels take to change color. In VR display the developers also utilize the double buffer sometimes referred to as double buffering. While one buffer is being shown the other is being calculated. This prevents the user from seeing unfinished images. This however creates the inherent delay that occurs. In frame 0 the input is registered, calculation happens during frame 1, and frame 2 is the result being displayed. This created an additional delay of 1 or 2 frames.

In some systems, the computer's frame rate is much higher than the display's refresh rate. In that case, the screen may display parts of multiple frames at the same time, resulting in a visual artifact called screen tearing (see figure 5.2). Vertical synchronization (VSync) solves this by delaying the rendering process until the screen is ready to display the next frame. This ensures that each frame is a single undistorted image. However, using VSync has a large effect on total delay since the frame calculation is held back by

the display's refresh rate [35].

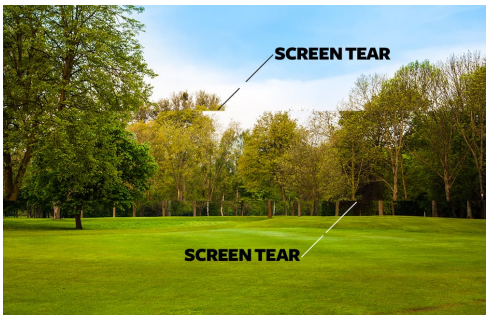


Figure 4.1: Illustration depicting screen tearing with two visible tears. The upper, middle, and lower sections of the image were rendered in separate frames[1].

4.1.3 Image Quality

One factor to consider is the quality of the image. However, in this context, it pertains not to the *beauty* of the image but rather its resolution.

The term resolution refers to the number of distinct pixels on the screen. It is described using the width and height of the screen in pixels. The higher the number of pixels the more details the screen can display. Common resolutions are 1920×x1080 (Full HD) or 3840×2160 (4k Ultra HD). When it comes to resolution there are two different kinds: image and display resolution. Image resolution is the number of pixels in the image file and display resolution is the number of pixels that a particular display device can showcase. In case that image is displayed at a lower resolution than its intended resolution, some distortions such as blur, blockiness, or pixelation can occur, thus removing the details.

Due to the often extensive space requirements of images, particularly when dealing with 360-degree images or videos, there is a way to reduce file size through compression. 360-degree images possess unique properties compared to regular images, presenting both new challenges and benefits. Motion estimation, commonly employed, encounters challenges in 360-degree videos due to non-linear motions such as rotation and zooming. Another challenge involves sampling density correction, where non-uniform sampling density causes distortion and increased bitrate. These issues are addressed through downsampling or quantization for compression. Re-projection, aimed at enhancing compression efficiency, can be achieved through rotation in the spherical domain or by exploring novel projection types. Lastly, systems can leverage the fact that only approximately 12.5% of the image is visible. By employing perceptual compression, the system can prioritize the quality of the viewport or region of interest [53].

[1]Image source (Jan 2024): `https://www.it4nextgen.com/what-is-screen-tearing/`

4.2 Visual Impairments

The perception of picture quality is subjective and varies among individuals. This section will spotlight certain factors that can influence this perception, with a particular focus on the photoreceptor cells in the retina. Abnormal cells play a vital role in color vision along with blind spots in the visual field.

4.2.1 Color Vision Deficiency

The human eye's capability to perceive colors is attributed to three distinct types of retinal photoreceptors, each characterized by different peak sensitivities. These photoreceptors, known as L, M, and S cones, exhibit peak sensitivity at large, medium, and small wavelengths, respectively (refer to figure 4.2). However, under various natural conditions, the sensitivity of these cones can be altered, leading to a phenomenon commonly recognized as Color Vision Deficiency (CVD) (see Section 6.4.4 for practical usage).

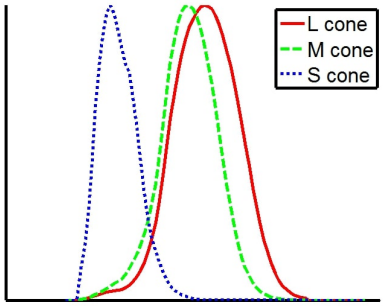


Figure 4.2: Cone spectral sensitivity functions for an average normal color vision (image taken from[27]).

Globally, the prevalence of color blindness is estimated to affect approximately 300 million people. This condition manifests in around 8 % of men and 0.5 % of women[2]. CVD encompasses various types, with trichromacy representing normal color vision—defined by the ability to perceive light through all three types of cones. Anomalous trichromacy, a deviation from normal trichromacy, occurs when one type of cone perceives light differently. This anomaly is categorized into protanomaly, deuteranomaly, and tritanomaly, affecting the sensitivity to red, green, and blue light, respectively. Dichromacy arises when one type of cone is functionally absent, resulting in protanopia, deuteranopia, or tritanopia, each associated with a deficiency in perceiving red, green, or blue light. Achromatopsia denotes a complete inability to perceive any color, while achromatomaly represents a milder form, allowing color perception under specific conditions.

[2]Data source (Jan 2024): `https://www.colourblindawareness.org/`

To illustrate such deficiencies in individuals with normal color vision, M. Machado's paper [27] introduces a model grounded in the stage theory. This theory posits that light absorbed by photopigments is processed through three opposition channels: an achromatic channel (white-black) and two chromatic channels (red-green and yellow-blue). Utilizing the altered spectral sensitivity function and applying principles of linear algebra, the model can generate a 3×3 transformation matrix. This matrix aids in seamlessly converting the standard color space into a color space influenced by CVD.

4.2.2 Other Visual Impairments

The human visual system is susceptible to the occurrence of blind spots, medically termed as scotomas (see figure 4.3). These scotomas often arise due to abnormalities within the retina, optic nerve, or visual center of the brain. Notably, they can serve as symptomatic indicators across a spectrum of conditions, encompassing age-related macular degeneration to diabetic retinopathy. It is noteworthy that scotomas maintain a fixed position relative to the central field of vision and are further categorized based on the specific areas they occupy as illustrated in 4.4 [3].

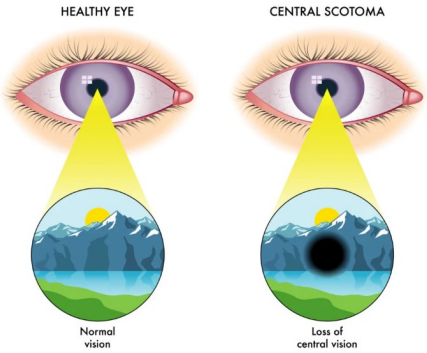


Figure 4.3: Example of vision with scotoma (image taken from [3]).

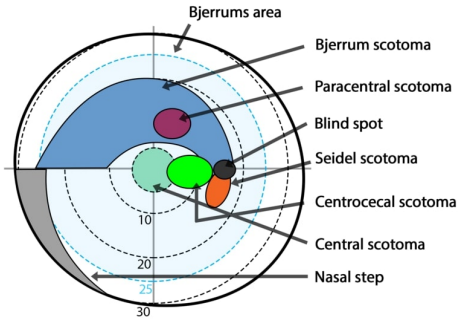


Figure 4.4: Vizualization of different types of scotoma shapes and locations (image taken from [3]).

Akin to scotomas, similar visual anomalies can be induced by entities referred to as floaters. However, floaters exhibit a less conspicuous presence and tend to gradually drift away from the primary line of vision. If floaters persist, it could indicate the presence of more severe eye conditions, such as retinal tears or retinal detachment[3].

Other visual impairments, such as myopia (nearsightedness) and hyperopia (farsightedness), affect vision by causing blurriness in objects based on their distance from the person's eye. These particular impairments were not included in the simulation, as they do not require complex simulations; glasses with specific lenses are typically sufficient. On the other hand, more intricate visual defects, like age-related macular degeneration mentioned in the previous paragraph, manifest through a combination of blind spots, partial blurring, and other smaller defects. While these impairments can be simulated, they would necessitate the incorporation of blind spot simulation with others. Therefore, the blind spot simulation developed in this study could be applied to address more complex conditions in future studies.

[3]Data taken from: `https://www.mayoclinic.org/diseases-conditions/eye-floaters/symptoms-causes/syc-20372346`

Chapter 5

Related Work

Several factors influence the quality of visual stimuli. This chapter will provide a concise overview of elements relevant to this thesis and their corresponding current research. Several studies have directly addressed the influence of specific factors in VR on the QoE. However, certain factors, such as CVDS, have received little to no attention in the literature. For these neglected factors, we have gathered simulations in VR or studies with a nature similar to QoE.

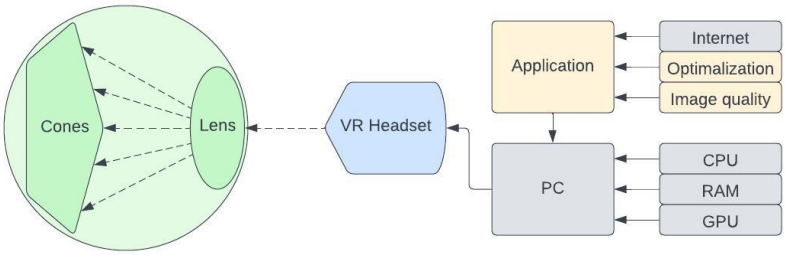


Figure 5.1: The diagram depicts potential sources of image impairments. On the left side is a representation of the human eye, portrayed in green. In the middle, the VR headset transforms digital signals into light from the screen. On the right side, the computer aspect is shown in gray. Developer factors, influenced by the developer's skill, are presented in orange.

Figure 5.1 illustrates potential causes of image impairment. Visual impairment encompasses issues such as color blindness (refer to Section 5.4), incorrect refraction, missing cones (refer to Section 5.5), and lens faults. The VR headset may also introduce potential impairments, including faulty gyroscopes, incorrect eye calibration, and others (an example, such as a faulty display, is discussed in Section 7.1.2). Both computer and developer factors have the potential to introduce impairments, such as latency (refer to Section 5.1), decreased frame rate (refer to Section 5.2), or even crashes if not adequately addressed.

5.1 Latency

In the world of computing, latency is an important and often unavoidable aspect of system performance. Latency refers to the delay between an action or input and the resulting output or feedback. To clarify the meaning of latency, this paper will refer to latency as end-to-end latency. Meaning, the amount of time it takes for the pixels in a head-mounted display to update and reflect a real-world movement that has occurred. There are many studies measuring the end-to-end latency of different HMDs. There were several measurements of the HMD HTC Vive in which all findings point to an end-to-end latency of 22 ms [31]. Additionally, in a study by [22] the researchers noted, that their VR environment consisted of single polygons and the complexity of rendered geometry does affect the end-to-end latency.

Latency is a critical factor in determining the quality of the user's experience. Low latency can contribute to the creation of a more seamless, smooth, and responsive experience. In contrast, high levels of latency are noticeable and negatively impact the user's immersion and sense of Presence, which can lead to cybersickness. Cybersickness decreases the quality of VR even more and can make the VR experience completely unusable. Latency can differ between different systems and the majority of research between latency and cybersickness only test the effect of static latency added, which shows an increase in cybersickness. However, latency spikes and latency patterns also contribute to cybersickness. Therefore is important to understand the implications of time-variant latency [42].

The impact of added static latency on QoE is well documented. A study by K. Brunnström et alii that simulated latency to measure the impact on QoE in VR was conducted. The VR simulated a crane for loading logs onto a truck. The simulation could affect the display update delay and the crane's control delay separately. Afterward, participants were given subjective surveys to fill out. The result demonstrates that display delay had a strong influence on the feeling of sickness and some participants preferred to stop the session. Therefore display latency should be avoided or minimized to less than 30 ms. The controller delay didn't impact the quality significantly to 200 ms, weak effects were found for 400 ms and 800 ms had a strong effect. This led the researchers to the conclusion that controller latency has a very limited impact and is tolerable if under 0.5 seconds. Studies like this help to set guidelines for VR application development to maximize quality [10].

Sudden head movement in HMD can result in delay or latency in visual feedback. Modern VR systems have a countermeasure and that is the motion-prediction algorithm. This however complicates the correct measurement of latency since the next display has begun calculating before the corresponding movement has been made. Using system-independent, high-speed camera-based latency measurement techniques to measure real and virtual movement allowed the researcher to measure how latency changes during movement. They applied this technique to several VR systems. At the sudden start of

head movement, their mean latencies were 21 to 42 ms. After 25 to 58 ms the motion-prediction algorithm reduced the latency to 2 to 13 ms [49].

One technique that can indirectly reduce the latency is Foveated Rendering. This technique aims to reduce the computation cost of rendering high-resolution images and thus reduce latency. This is achieved by rendering high-resolution images only in the center of the user's vision, where the human eye has the highest visual acuity. Outside of the center, the visual acuity is lower, and lower-quality images can be used. Instead of rendering the full image in high quality, the method renders only part in high quality, therefore reducing the rendering cost. This technique however requires the knowledge of where the gaze is. This method also has downsides, firstly eyes need to be tracked which requires special hardware and computational power. Secondly, rapid eye movements called *saccades* involve abruptly changing the point of fixation in the visual field, which can lead to the perception of low-resolution peripheral images if there is significant latency in rendering the high-resolution foveal images during these movements. Luckily during a short period before, during, and after the saccadic eye movement, saccadic omission prevents some visual input from being processed, therefore the requirement for latency can be relaxed. A study [5] researched the latency requirement for Foveated Rendering. They tested three possible methods to ensure Foveated Rendering: Gaussian Blur, Subsampling, and Foveated contrast preserving sampling (fCPS). They found that the fCPS technique allowed for more foveation than the Subsampling method even at 10 degrees eccentricity (distance from the center of vision). Both fCPS and Gaussian Blur allowed for much greater foveation at 20 degrees eccentricity. They also tested the impact of increasing eye-tracking latency. The results showed that all techniques are negatively impacted by added latency. From their result, they argue that eye-to-image latency of 50 to 70 ms is tolerable for Foveated Rendering.

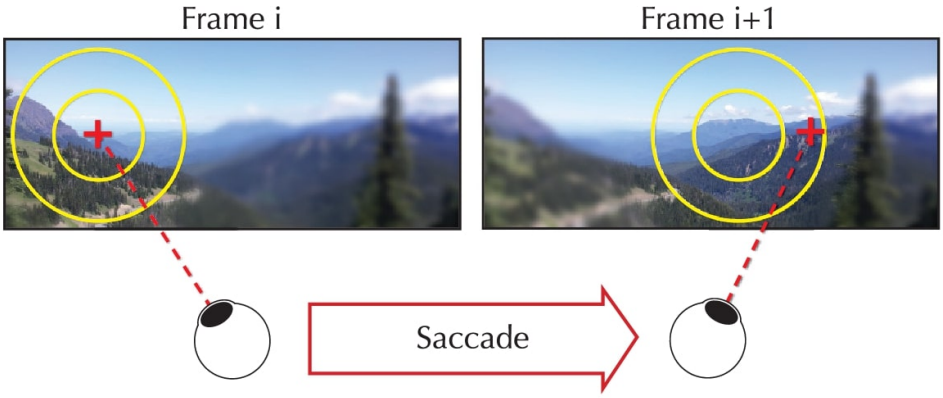


Figure 5.2: Eye-tracking with foveated rendering and the impact of high latency from eye tracking with saccades (image taken from [5]).

5.2 Frame Rate

When a movie filmed at 24 fps is adapted to television with a higher frame rate of 60 fps, the movie needs to be upsampled to meet the television's criteria. This process involves repeating some frames two times and others three times, which can cause judder, or visible artifacts due to differences in frame persistence. To address this issue, a low frame rate, low persistence technique can be used, which inserts black frames to reduce motion blur and judder. To evaluate the effect of image persistence on user experience, study [55] designed tasks to compare high frame rate, low frame rate low persistence, and low frame rate high persistence in VR. The tasks included a selection task, where participants pointed at appearing circles with a ray-casting controller, and a navigation task, where participants steered using a joystick and moved forward by squeezing a trigger. The study found that tasks done at high frame rates were completed faster, but there was no significant difference between low frame rates. A higher frame rate provided a greater sense of presence but resulted in slightly worse SSQ scores, and some participants reported severe nausea only at high frame rates. In terms of overall preference, 22 % of participants ranked high frame rate last, while only 17 % rated low persistence higher than high persistence.

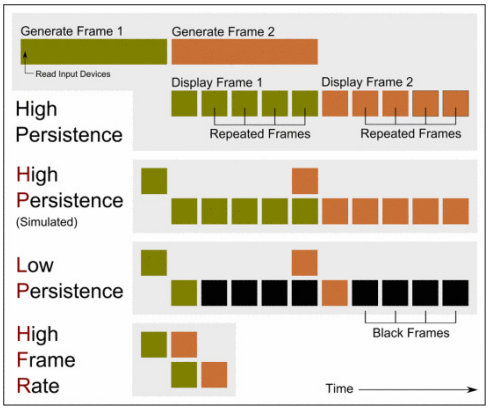


Figure 5.3: Timeline of frame generation and display. Showcasing the difference in high and low persistence frame rate (image taken from [55]).

5.3 Image Quality

The desire to have the highest resolution often clashes with the need for lower storage space for an image. Researchers have conducted a study that examines the impact of resolution and quality factors of compression on perceptual quality in head-mounted displays. The researcher [18] used original 360-degree images in 4K resolution and downsampled them to 2K, 1K, and 720p. After a short training session, human subjects viewed randomly shuffled images and rated them on a scale of 0 (bad) to 100 (excellent). Following that MOS

was calculated. The result showed higher MOS with higher-resolution images, as anticipated. Also, MOS decreased together with a quality factor but it was content-dependent, meaning some images tend to "look worse" in lower resolution. It is important to note that the degradation occurred at a slower rate for higher spatial resolutions, such as 2K and 4K, whereas it was faster for lower spatial resolutions, such as 1K and 720p. The same goes for the quality factor. It is worth noting that users are unlikely to perceive any differences between images compressed with a quality factor of 60 versus 100.

Similar study explores the correlation between video quality in VR and QoE through subjective assessments and objective methods. Starting with the highest resolution video, three lower video quality conditions were created by linearly degrading the resolution. The videos were then presented to 32 participants, who provided post-test feedback via questionnaires covering subjective opinions, sensory immersion, and other aspects. Simultaneously, EEG recordings with 64 electrodes were taken. Results indicate that lower video quality adversely affects sensory immersion in the VR environment, while narrative immersion remains unaffected. This nuanced outcome suggests that the experience of VR presence is intricately multifaceted. Notably, heightened simulator sickness was observed only in response to the lowest video quality, emphasizing the impact of video quality on the overall user experience in VR [54].

5.4 Color Vision Deficiency

The real world predominantly caters to individuals with traditional trichromatic vision. Given that color is a crucial medium for conveying information, those affected by CVD may face disadvantages. Over the years, various technologies and tools have been developed to aid color-blind users, such as special glasses or lenses aimed at improving color sensitivity, the ability to customize color displays, enhance contrast, and the potential consideration of gene therapy in the future. Nevertheless, designers can play a role in supporting individuals with CVD. For instance, numerous websites provide color correction transformations to simulate CVD experiences [12].

The quality of a color is often overlooked but it also plays a role in overall quality. When images of similar or even the same scene are captured, these images can differ in color or brightness. This is caused by several factors like time of day, light intensity and position, the mean of capturing, and so on. To address this, the researchers have proposed a new metric to assess the quality of color correction by combining color contrast similarity and color value difference. The authors conducted experiments using benchmark datasets and compared the results with other existing metrics. Through subjective assessments from 126 volunteers with normal or corrected vision, the study demonstrates that the proposed metric outperforms 17 state-of-the-art image quality assessment methods in terms of consistency with users' subjective evaluations [32].

Several websites provide image recoloring options based on selected CVD, as illustrated in [1]. However, these platforms primarily display image examples instead of modifying the entire color spectrum for the complete field of view. Therefore, it does not represent the true vision of a person affected by color blindness.

5.5 Other Visual Impairments

Given the impact of blind spots on vision, individuals experiencing such impairments tend to adapt to their condition in a certain manner. A study conducted by V. Walsh [48] delves into this adaptation process among individuals with normal eyesight. The investigation involved assessing the time taken by subjects to locate the correct characters within an image. Initially, subjects engaged in a search without encountering any blind spots, allowing them to familiarize themselves with the test procedure. Subsequently, the participants were divided into two groups. One group underwent the first search block with a center scotoma characterized by sharp edges, followed by a transition to a center scotoma with gradually fading edges. Conversely, the second group followed the opposite sequence. The presence of scotomas compelled subjects to employ their "side vision" during the search, yet they demonstrated an ability to learn and enhance their performance in the task. This adaptation suggested the development of a novel visual routine to effectively navigate simulated field loss.

A study conducted by J. Lewis and colleagues [26] delved into the simulation of visual impairments within VR. Collaborating with an expert in visual impairment awareness, they developed a mod for a game in the Unreal Engine, utilizing the game's development kit to reconstruct a restaurant as the VR environment. The mod successfully implemented five distinct impairments: glaucoma, macular degeneration, cataracts, hemianopia, myopia, and hyperopia. The simulator underwent testing on 21 students who had the option to activate and deactivate impairments at will. Through the analysis of pre and post-questionnaires, the study confirmed that the simulator increased understanding of the nature of visual impairment. This heightened awareness could potentially lead to an improved level of empathy, particularly among individuals who are actively engaged with the visually impaired.

Similarly, one thesis [24] delved into the simulation of image impairments in both AR and VR. The primary aim was to foster a better understanding of visual impairments and quantify their effects. Researchers created numerous simulations in VR and AR, incorporating various effects to replicate diverse eye disease patterns. In a VR simulation task involving 30 participants, two objectives were assessed: identifying when an escape-route sign becomes recognizable and navigating through an escape route during a simulated emergency. The findings indicated that the conventional placement of escape-route signs is inadequate in areas with a higher number of visually impaired individuals. Moreover, it was noted that the emergency simulation did not

encompass additional environmental conditions such as flickering, smoke, and haze. Additionally, she examined lighting conditions to assess accessibility for individuals with various visual impairments.

One can also come across websites that showcase various vision defects. Essentially, there are two types of simulations for vision impairments. The first type[1] displays altered static images. The other type[2], actually simulates impairments by altering brightness, incorporating masks that block vision, and so on. These websites are effective only for demonstration purposes. They do not capture the true experience particularly because these types of vision impairments are linked to the eyes. For instance, when simulating diabetic retinopathy, the mask remains static on the screen, allowing users to focus on the portion of the mask within their peripheral vision, which would be impossible for anyone suffering from such a defect.

[1]Discoveryeye, Vision Simulations (Jan 2024): `https://discoveryeye.org/resources/vision-simulations/`

[2]Versanthealth, Vision Simulator (Jan 2024): `https://versanthealth.com/vision-simulator/`

Chapter 6

System overview

The main objective of this thesis was to create a system that would enable the simulation of variable image impairments. This chapter will elucidate the development process, the fundamental logic underlying the created system, as well as its capabilities and limitations.

To comprehend the logic behind the system, Section 6.1 will initially explain the environment in which it was built. Numerous software, primarily game engines facilitate the development of VR applications. The chosen software will be overviewed, focusing on the core features that are predominantly utilized.

Following this, the architecture will be explained. It encompasses several subsystems that collaborate to create a functional automatic subjective experiment. Their design is concentrated on being as independent of each other as possible, thus enabling the improvement or replacement of any subsystem with minimal knowledge about the system as a whole. This will be discussed in Section 6.4, particularly in Subsection 6.4.4, where the simulation methods will be presented.

To substantiate the functionality of the system, a pilot experiment was conducted in Chapter 7.

Instructions on how to download the project and its subsequent setup are located in Appendix B.

6.1 Development Environment

Firstly, it is crucial to elaborate on the software that facilitated the initial development stages. This section will consequently clarify the engine employed, detailing its capabilities and highlighting specific features utilized during the development process.

6.1.1 Unity Engine Overview

At present, various software options are available for the development of VR applications, with primary emphasis placed on exploring game engines. Prominent engines in this domain include Unity, Unreal Engine, and Godot. After careful consideration, the choice was made to utilize Unity due to prior familiarity with the environment and its extensive online user community. Unity is distinguished as a widely adopted and free game engine, well-regarded for its versatility in creating interactive applications across 2D, 3D, and VR platforms. Renowned for its comprehensive toolkit encompassing rendering, physics simulations, scripting, and asset management, Unity emerges as a favored platform for developers aspiring to construct engaging and immersive digital experiences.

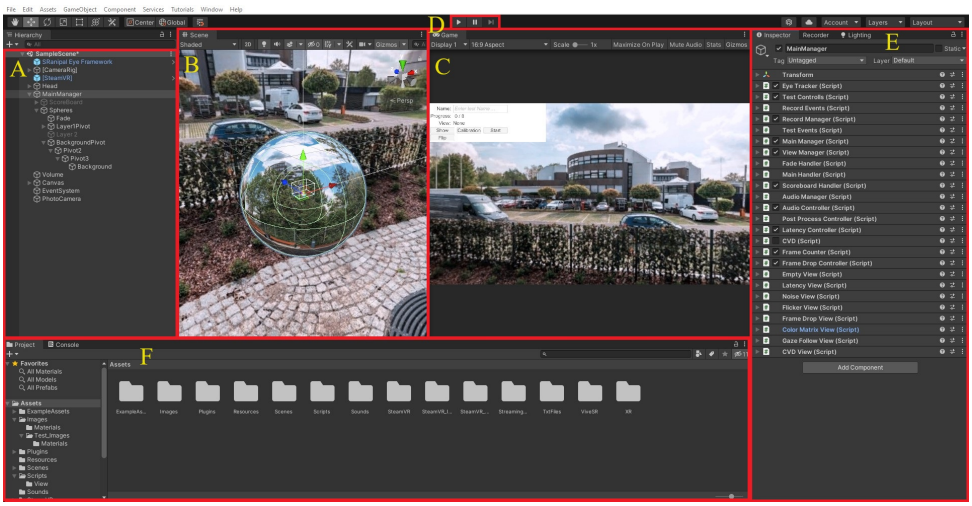


Figure 6.1: Screenshot of Unity environment with highlighted sections; (A) Hierarchy window, (B) Scene window, (C) Game Window, (D) Play button, (E) Inspector window, (F) Project window.

Once a Unity project is created, Unity presents an overview of the project, as illustrated in figure 6.1. The primary components include:

(A) The Hierarchy window serves as a comprehensive list depicting every GameObject and its structural hierarchy within the Scene. Here, developers can arrange objects by establishing parent-child relationships, selecting specific objects, and organizing the overall structure according to their requirements.

(B) The Scene window provides a visual representation of GameObjects within the Scene, facilitating navigation and editing activities. This might involve observing objects from various angles, relocating them, or adjusting their size, rotation, and other properties.

(C) The Game window shows the rendered image view from the main camera.

(D) The Play button positioned in the top middle of the screen, encompasses start/stop/pause functionalities within Play mode. Play mode allows testing the current scene without the requirement to build an executable file.

(E) The Inspector window facilitates the viewing and manipulation of GameObject properties. As the properties define the object's functionality.

(F) The Project window reveals the library of Assets available in a Project. It encompasses all files, including scripts, textures, models, audio, and more.

GameObjects

GameObjects constitute a fundamental element within the Unity framework, residing prominently within both the Scene and the Hierarchy window. Each GameObject possesses a versatile array of properties, referred to as components, delineated within the Inspector window. The Transform component, inherent to every GameObject, serves as an initial configuration encompassing parameters such as position, rotation, and scale. This foundational component is integral to defining the spatial characteristics of the associated GameObject.

The diversity of components becomes evident when considering a specific example, as illustrated in figure 6.1. In this instance, the selected MainManager GameObject is highlighted, revealing not only its Transform component but also a large number of script components. This exemplifies the nuanced and specialized nature of components, each contributing uniquely to the functionality and behavior of the respective GameObjects in the Unity framework.

Scripts

For the implementation of custom behaviors, one can craft tailored script components capable of dynamically modifying properties and functionalities based on specific requirements. These scripts are scripted in C#, inheriting the properties of the MonoBehaviour class optimized for seamless integration with Unity. MonoBehaviour comes equipped with default methods, including the essential Awake(), triggered during play mode initialization, and Start(), executed immediately after Awake(). The pivotal Update() method, invoked in every frame of play mode, plays a crucial role, complemented by variations such as LateUpdate(), specifically called at the conclusion of frame calculations.

Custom scripts were technically implemented in each subsystem outlined in Section 6.4. Their specific purposes will be described individually within each subsystem.

Internal Communication

Given the complexity of the desired application, effective communication among numerous subsystems was imperative. Various methods were employed to facilitate communication between custom scripts.

Reference stands out as a straightforward communication method. By establishing a reference, a script gains the ability to invoke public methods and edit public variables within components. This reference can be conveniently set in the Inspector window through a drag-and-drop action, linking the desired component to the script (see figure 6.1). Alternatively, references can be obtained by requesting them through functions like GameObject.GetComponent<*DesiredComponent*>(). While the former ensures a clear overview of connections, it can be time-consuming, especially when dealing with repeated references across different scripts.

Events presents a slightly more intricate communication approach, providing enhanced flexibility. Functioning as a broadcast, events allow different scripts to subscribe. The notable advantages include the caller's independence from knowledge about subscribing listeners and the ability to trigger multiple listeners with a single statement.

6.1.2 Steam VR

Steam VR is a virtual reality platform developed by Valve Corporation, designed to enable users to experience immersive virtual environments. It provides a range of tools and features for developers to create VR experiences that can be accessed through the Steam platform. Steam VR supports a wide range of VR headsets, including the HTC Vive, Oculus Rift, and Windows Mixed Reality devices, making it accessible to a broad audience. The platform also includes a variety of resources and documentation for developers, including a Unity plugin and a software development kit, to facilitate the creation of VR content. The Steam VR platform has contributed significantly to the growth and adoption of VR technology and remains a prominent player in the VR market.

The Steam VR package provides a variety of valuable development tools for VR applications. Among these tools are prefabs, or pre-fabricated GameObjects, which can be readily incorporated into a project. By dragging these prefabs from the project window to the scene or hierarchy window, developers can effortlessly instantiate predefined GameObjects.

One noteworthy prefab offered by Steam VR is *CameraRig*, which was utilized in the project. This prefab encompasses a camera equipped with a script that mirrors the actual headset position onto the virtual camera. Additionally, *CameraRig* includes models for controllers, which dynamically replicate the real-time positions of the physical controllers in the virtual environment.

6.2 Core Mechanism

Displaying a 360-degree image in VR offers multiple approaches. The most straightforward method involves transforming the given 360-degree image into a skybox material and applying it universally in the scene. However, due to the specific nature of the chosen image impairments for this test, an alternative approach was necessary.

The selected solution involves simulating the skybox by enclosing the camera within a sphere with a skybox material. This approach posed challenges, given that default 3D objects have outward-facing surfaces. If the camera were inside such an object, it would either be invisible or result in distorted artifacts. To address this, a script named *FlipSpheres* was implemented to invert the normals of the required 3D object.

In this setup, three inverted spheres surround the camera at any given time. The first is the Fade sphere, ensuring a smooth transition (see 6.4.5). The second is the Background sphere, featuring a skybox material to create the illusion of a skybox. Lastly, the Layer sphere serves as an intermediary layer between the Background and the camera. Moreover, the Background and Layer spheres are placed as children of several Pivot GameObjects, enabling the stacking of independent rotations. It utilizes a standard material, allowing transparency in specific sections, a functionality explained further in 7.1.2.

6.3 Evaluation Timeline

Subjective evaluation tests are conducted to assess the quality of the simulated experience of image impairment occurring in a sequence of viewing and subsequent scoring. As each simulated impairment influences various aspects of the VR experience, each simulation is termed a 'View.' The duration during which a user observes any given View is considered the viewing period, and the corresponding scoring process is referred to as the scoring period. This concept is illustrated in 6.2, accompanied by the time blocks denoted as **FV** and **FS**. To visually depict the initiation or conclusion of a viewing period, the system employs a smooth transition, represented by the shortcuts Fade to View and Fade to Score, as indicated in 6.4.5.

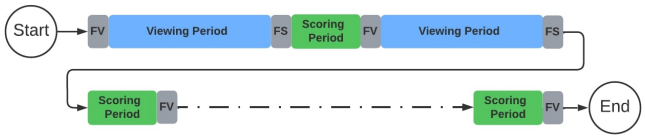


Figure 6.2: The timeline of the subjective test illustrates a sequence of alternating viewing and scoring periods. Between each, there are small blocks representing seamless transitions.

6.4 Main System architecture

Developing a system to simulate various Views involves the coordination of multiple subsystems. The principal subsystems are illustrated in 6.3, along with arrows indicating their intercommunication.

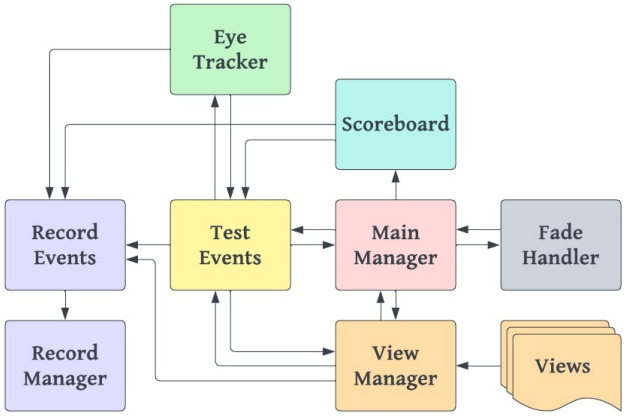


Figure 6.3: A schematic illustrating the primary logic of the application. Intercommunication among components is depicted through arrows. The central components consist of the Main Manager and Test Events, overseeing the majority of processes.

6.4.1 Test Events

As elucidated in the section Internal Communication (6.1.1), events serve as a method of communication between scripts. The Test Events, depicted in figure 6.3, play a pivotal role in establishing the primary communication controller. The rationale behind selecting events as the principal communication mechanism lies in the necessity for numerous subsystems to collaboratively execute functionalities. This approach allows a script to invoke a single statement, prompting all subsystems to respond accordingly. Moreover, the utilization of events facilitates the extensibility of the application; new subsystems can be seamlessly integrated by connecting them to events without requiring an in-depth understanding of other subsystems.

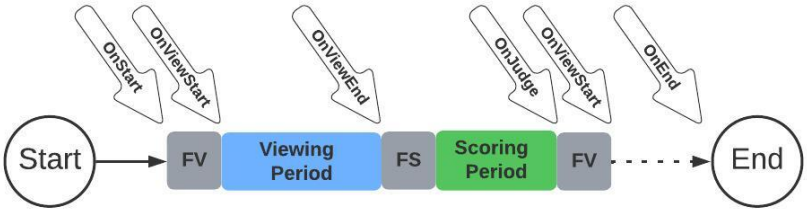


Figure 6.4: Timeline of event calls, encompassing events such as OnStart, OnViewStart, OnViewEnd, OnJudge, and OnEnd.

The set comprises six events, with five being indispensable for the proper execution of tests. Illustrated in figure 6.4, OnGazeDirection is absent as it is called every frame during the viewing period. These events include:

OnStart: This event is triggered when the user opts to commence the test, playing a crucial role in the primary initialization of all subsystems.

OnEnd: In contrast to OnStart, this event signals the conclusion of the test, serving as a reminder for the subject to exit the VR headset.

OnViewStart: This event is invoked at the commencement of the viewing period, necessitating preparation for each View's initialization, given that each View demands a distinct approach.

OnViewEnd: Conversely, OnViewEnd is called at the termination of the viewing period.

OnJudge: Triggered after the subject makes their assessment of the previously viewed View.

OnGazeDirection: An additional event for Views to subscribe to, sending gaze direction.

6.4.2 Main Manager

The Main Manager serves as a fundamental script within the system, responsible for ensuring the accurate sequence of processes. Before each viewing period, the Main Manager initiates a request to the View Manager, prompting the preparation of the subsequent View. Given the Main Manager's lack of knowledge regarding the number of remaining Views in the array, the View Manager relays information about the last View, prompting the Main Manager to conclude the test. Additionally, the Main Manager is tasked with overseeing the proper control of the Fade Handler and the Scoreboard Handler (refer to 6.4.5 and 6.4.6).

Furthermore, the Main Manager houses the timer for the viewing period, and this timer's duration can be adjusted through the Inspector window.

This pivotal role in orchestrating the system's processes underscores the Main Manager's significance in facilitating seamless and controlled execution.

6.4.3 View Manager

The View Manager assumes control over all currently active Views intended for display during the test, overseeing their proper activation, deactivation, and seamless operation.

The storage of Views is organized as follows. Various View types, such as Latency View, Noise View, and others (refer to Section 6.4.4), are individually implemented as components, each encapsulating a unique set of settings. The responsibility of displaying the desired View type with specified properties upon command lies with the View Manager.

At the initiation of the play mode, the View Manager systematically locates all View components. After identification, it requests their active settings and receives a list of indices indicating the active settings. This information is then stored alongside the view type in the *Active View* class, which contains two parameters: a reference to the view and the index of the setting.

When the test is initiated with the **OnStart** event, the View Manager randomizes the order of the Views in the list. Upon request from the Main Manager, the subsystem prepares the next View, sets its configuration, and activates it. Similarly, upon the Main Manager's request for deactivation, the View Manager forwards this command to the current view.

Considering that certain views require dynamic adjustments based on frames, the View Manager also propagates the *Update()* and *LateUpdate()* functions in the form of *Do()* and *LateDo()* methods within the Views. The *LateDo()* method is primarily invoked for views that depend on head position, ensuring the in-game camera has sufficient time to accurately update and copy the real-life position. The View Manager only propagates these methods during the viewing periods, which it initiates by subscribing to the **OnViewStart** and **OnViewEnd** events.

To facilitate recording purposes, the View Manager additionally transmits the name of the current View to Record Events. This comprehensive oversight underscores the View Manager's crucial role in coordinating the display and functionality of active Views throughout the test.

6.4.4 Views

Views refer to simulated image impairments utilized in this study. This section details the means of their simulation and their instruction on their usage. How and which Views were used to compose the test is described in subsequent sections (refer to 7.1).

Backgrounds

The system operates by utilizing 360-degree images as a background. In order to streamline the View settings and eliminate redundant parameters, a deliberate decision was made to provide a curated selection of backgrounds. Specifically, three distinctive background images sourced from Omnidirectional Image Quality Assessment Database [40] are offered (refer to figure 6.5). They are directly inputted into the Material Handler on a background sphere, and Views change this through the Background Handler (see 6.4.9). If a different background were needed, changing the Material Handler to contain an array of different images would be sufficient. These selections were chosen for their unique characteristics: "Hokkaido" offers a darker environment, "Biscayne" displays the highest brightness, and "Flowers" presents a diverse array of colors. Fortunately, Unity materials do not necessitate a predetermined texture, making all images compatible regardless of their resolution.

(a) : Hokkaido **(b) :** Biscayne **(c) :** Flowers

Figure 6.5: 360-degree background images employed in the experiment (images taken form [40]).

Info Class

All designed Views within the scope of this study were conceived with editable parameters to enable the measurement of various degrees of impairment. Employing the same View as a component for multiple types would be impractical due to identical functionality. Consequently, a solution is proposed: an array of *Info Classes* in each type of View, where the settings for each View are configured.

To streamline the core parameters common to all Views, a unified class named *View Info* has been introduced. These fundamental parameters include **View Name** for specifying the name, **Active** to indicate whether the current settings are intended for use in tests, and **Background Texture** for setting the texture on the background sphere. Given that other Views necessitate additional parameters specific to their unique requirements, while still requiring the aforementioned core parameters, distinct *Info Classes* will be derived from the base class *View Info*.

Base View

The Base View functions as a template for other views. As the fundamental logic remains consistent across all views, the Base View provides functions

within its script that are implemented as virtual functions, designed to be customized and overridden by child classes. One function that is intended to remain unaltered is *GetName()*, which retrieves the assigned name of the current setting.

These functions encompass *Activate()* and *Deactivate()*, responsible for preparing the view and cleaning up after the viewing period, respectively. Specifically, *Activate()* and *Deactivate()* handle the application of the **Background Texture** to the background in Base View, and as a result, every View incorporates these base functions. The methods *Do()* and *LateDo()* act as propagation methods for *Update()* and *LateUpdate()* respectively.

Finally, the functionalities of *Set()* and *Get()* are introduced. These functions play a crucial role in facilitating communication with the View Manager. The *Get()* function is employed when the View Manager requests the list of active settings for a given View. Conversely, the *Set()* function is responsible for configuring parameters from the *Info* class for utilization within the View.

Empty View

The Empty View functions as a fundamental demonstration of a 360-degree image without simulating any image impairments. As it does not necessitate the simulation of additional image impairments, it utilizes the default *View Info* class without requiring any supplementary information.

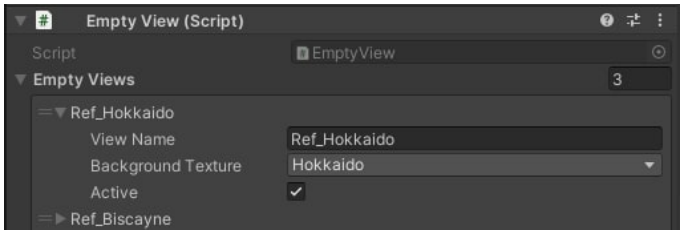


Figure 6.6: Empty View component in Inspector window.

In figure 6.6, all parameters from *ViewInfo* are displayed. It is crucial to emphasize that the parameter **Background Texture** is not a texture parameter but an enumerator. The decision to have a set number of predetermined textures was made to enhance the application's performance. The enumerator comprises three values, which were elaborated in the previous section (refer to Section 6.4.4).

Latency View

To replicate latency, the system employs historical head rotations to simulate display latency. Latency View utilizes the Latency Controller. The *Activate()* method configures the desired latency in frames within the Latency Controller.

Since latency simulation relies on head rotations, and thus needs to be updated every frame, the View also triggers the *LateDo()* method to communicate with the controller.

The Latency View employs *Latency Info* as an *Info Class*. In addition to the inherited parameters, *Latency Info* introduces the **Frame Latency** parameter, specifying the desired latency to be utilized (see figure 6.7).

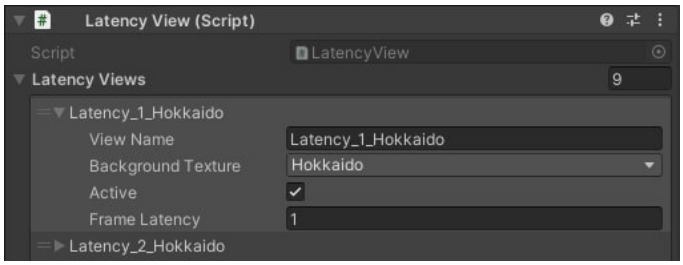


Figure 6.7: Latency View component in Inspector window.

Once the controller is active and regularly updated, it records past head rotations. The input latency, represented by an integer variable, dictates the number of frames to be postponed. For every frame, the controller adjusts the background based on the headset's current rotation and its rotation desired frames in the past. This process creates the perceptual illusion that the headset displays parameters with a frame delay.

Frame Drop View

Similar to latency simulation, the Frame Drop View operates by duplicating frames, thereby presenting the user with an identical display for a specified number of frames, regardless of the user's movements. This effect creates the perceptual illusion of a reduction in frame frequency. The functionality is facilitated through interaction with a controller named Frame Drop Controller.

Unlike the Latency View, the Frame Drop View is not continuously active throughout its duration. Upon activation, the Frame Drop Controller operates in an *on/off* state. During the *on* state, the past head rotation is recorded and applied to the local rotation of the background, while the background pivot updates to match the current head rotation. Subsequently, the controller transitions to the *off* state, resetting the background rotation and ceasing the update of pivot rotation. The *off* state persists for a set duration.

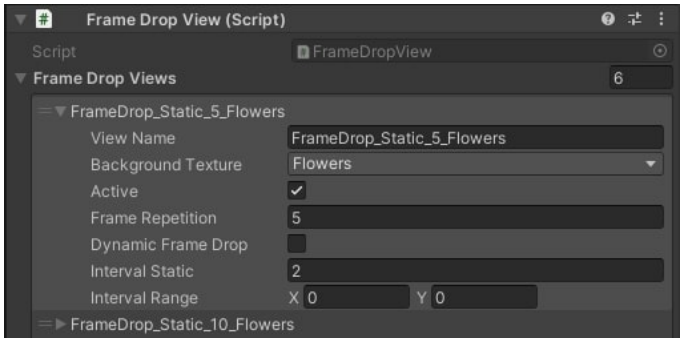


Figure 6.8: Frame Drop View component in Inspector window. The **Dynamic Frame Drop** checkbox isn't active therefore the View will use **Interval Static**.

The *Frame Drop Info* class introduces several parameters. Firstly, the duration of the *on* state is determined by an input integer referred to as **Frame Repetition**. The duration of the *off* state depends on the state of the **Dynamic Frame Drop** checkbox. When unchecked, the view will adopt a static value specified in the **Interval Static** parameter for each *off* duration. If the checkbox is checked, the view will utilize a random interval within the values specified in the **Interval Range** parameter (see figure 6.8).

Noise View

The Noise View was devised to simulate additive noise effectively. This simulation leverages the manipulation of the Albedo color in the Layer sphere material. The Albedo color dictates how the surface color responds to incident light, and it can alternatively incorporate any image as a source of color, thereby introducing texture. When utilizing an image format with an alpha channel, such as PNG, and configuring the material's rendering mode to transparent, the material becomes translucent.

Upon the activation of the view, the designated texture is applied to the material of the Layer sphere. The Noise View provides three options for manipulating the Layer sphere. The first option involves no manipulation, resulting in the Layer sphere maintaining a consistent rotation. The second option entails a random rotation of every frame. Lastly, the Layer sphere can be configured to rotate with the camera GameObject, thereby preserving a constant relative rotation to the headset at all times.

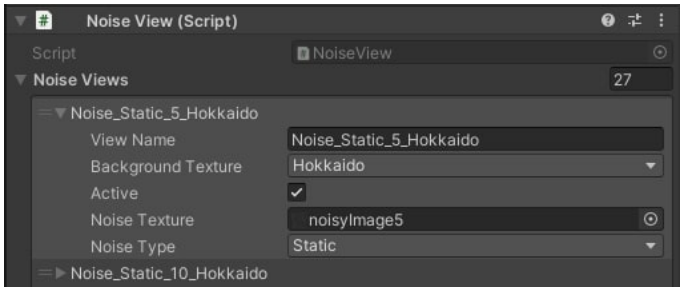


Figure 6.9: Noise View component in Inspector window.

The chosen texture is derived from the **Noise Texture** parameter within the *Noise Info* class. The method of manipulating the Layer sphere is elucidated by the enumerator **Noise type** parameter, which includes the values: **Static**, **Dynamic**, and **Headset Static** (see figure 6.9).

Gaze Follow View

The functioning of the Gaze Follow View closely mirrors that of the Noise View. It employs the Layer sphere as a mask for users to observe, with the key distinction being that, unlike the Noise View, the mask in the Gaze Follow View dynamically follows the user's gaze direction.

Similar to the controller, the Gaze Follow View features a Gaze Follow Layer situated on the Layer sphere. The placement of the controller on the sphere is chosen for its potential to facilitate easier configuration in the event of using multiple masks in future studies. The Layer sphere's pivot replicates both the rotation and position of the user's headset. Furthermore, upon activating the Gaze Follow Layer, the script subscribes to the *OnGazeDirection* event, incorporating a function that aligns the sphere's rotation with the direction of the user's gaze.

While this approach would be effective under the assumption of perfectly accurate gaze direction calculations, a discrepancy was identified during the development phase. While maintaining gaze direction on a fixed point, the desired mask exhibited slight oscillations. This error is likely attributed to the limitations of eye-tracking technology, subtle eye twitching, or a combination of both. To rectify this issue, the mask will now smoothly move towards the correct rotation, implemented through *Slerp()*, instead of abruptly snapping into place. Additionally, an observation was made that the rotation was off by 90 degrees in the y-axis, which was mitigated by incorporating an offset.

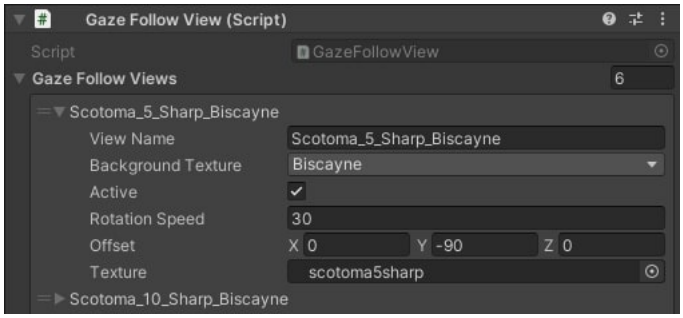


Figure 6.10: Gaze Follow View component in Inspector window.

Within the *Gaze Follow Info* class, the **Texture** parameter is utilized as the mask. It includes a **Rotation Speed** parameter, denoting the arbitrary speed of the *Slerp()* operation. Through testing in the development phase, an optimal rotation speed of 30 was determined. The last parameter is **Offset**, which corrects the rotation (see figure 6.10).

Flicker View

The Flicker View was devised to replicate the effect of flickering lights by rapidly alternating the brightness of a scene between darker and lighter states.

Various methods can be employed to implement flickering, and our proposed solution involves integrating a post-processing unit in Unity (refer to 6.4.9). The tool employed for adjusting brightness in the scene is the Color Grading effect, specifically the Post-exposure component. To facilitate effective control, the Post Process Controller is utilized.

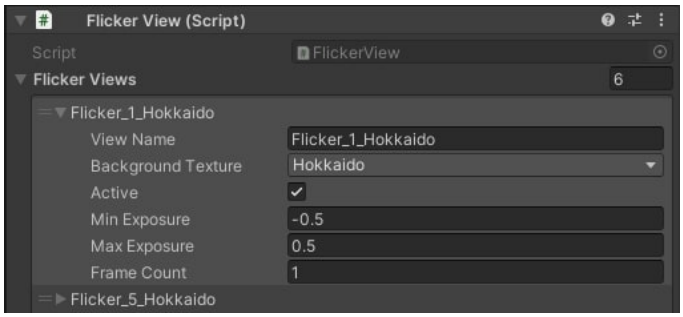


Figure 6.11: Inspector window displaying the Flicker View component.

Within the *Flicker Info*, three additional parameters are present. Two of these parameters, namely **Max Exposure** and **Min Exposure**, function as setters for the darker and lighter values of the scene brightness. The third parameter, **Frame Count**, specifies the duration for which one exposure persists, measured in frames (refer to 6.11).

Color Matrix View

To explore various color-related impairments (4.2.1), the Color Matrix View was developed. This View facilitates the adjustment of each color level according to specific requirements.

It leverages the post-processing unit through the Color Grading effect, specifically employing the Channel Mixer, thus it uses Post Process Controller. After the image is rendered, the Channel Mixer dynamically adjusts the RGB values based on input parameters, essentially forming a 3×3 matrix. Consequently, the RGB color is transformed into a new color space.

Figure 6.12: Default setting of Channel Mixer in Post-processing, for the color red.

Traditionally, the values in such matrices fall within the range of 0 to 1. However, in the Unity post-processing framework, the default values are set to 100 (see figure 6.12). Consequently, the default matrix corresponds to 100 times the identity matrix.

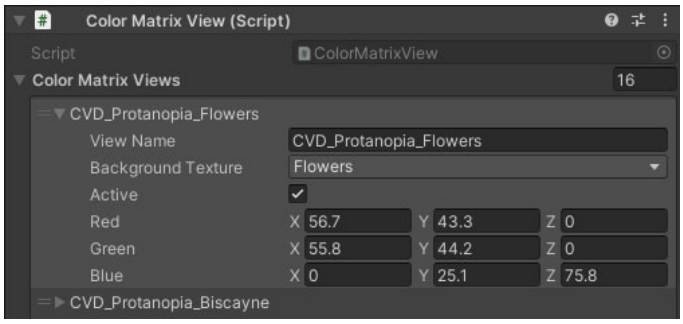


Figure 6.13: Color Matrix View component in Inspector window.

Figure 6.13 illustrates an additional parameter in the *Color Matrix Info* class. These parameters, namely **Red**, **Green**, and **Blue**, are represented as Vector3 values and collectively contribute to the creation of a 3×3 matrix, as visually depicted.

CVD View

This section introduces a method for color editing that necessitates a calculated color matrix. However, it is important to note that the color matrix does not accurately represent how the human eye perceives colors, as discussed in section Color Vision Deficiency 5.4. The human eye is equipped with

three color-sensing cones, as detailed in the context of CVD. The CVD View adjusts color based on alterations in the LMS spectral functions.

In conjunction with the Post Process Controller, this View relies on the *CVD* script. The *CVD* script plays a crucial role in this process, as it is responsible for calculating the color matrix required to modify the rendered image. This calculation involves utilizing premade files that contain the spectral sensitivity of LMS cones and their interaction with RGB. By altering the spectral sensitivity, different matrices can be computed. The CVD View facilitates the adjustment of each LMS spectrum. Upon activation, the View calculates a color matrix based on the shifted spectra and applies it to the scene through the Post Process Controller.

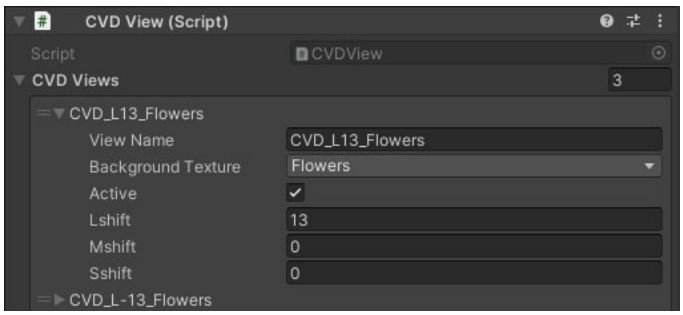


Figure 6.14: CVD View component in Inspector window.

Unique parameters within the *CVD Info* include integers representing the shift of the spectrum, namely **Lshift**, **Mshift**, and **Sshift** (see fig 6.14).

6.4.5 Fade Handler

The Fade Handler was developed to streamline the transition between the viewing and scoring periods. It employs the Fade sphere, manipulating the alpha value of its material. As the viewing period concludes, the entire screen undergoes a gradual fade to gray, by making the sphere visible. Thus signaling the end of the viewing period, commonly referred to as Fade to Score (see 6.2). This gray screen remains until the scoring process concludes. To seamlessly transition back to the viewing period, the gray screen gradually fades into invisibility, revealing the desired content clearly—referred to as Fade to View (see 6.2)

The duration of this transition can be adjusted in the Inspector window. Once the transition is completed, the Fade Handler notifies the Main Manager that the test can resume.

6.4.6 Scoreboard

The Scoreboard, depicted in 6.3, embodies a collection of GameObjects and scripts collaborating to form this subsystem. One integral component is the

Scoreboard GameObject, illustrated in 6.15, comprising six cubes of distinct colors. The bottom row's represents QoE score (refer to 2). Equipped with a handheld controller, the subject triggers the *Score Point* script upon entering the collider of any cube. Upon entry, the selected cube slightly enlarges, resetting to its original size upon controller removal. This precaution prevents accidental triggering of an undesired cube, requiring the user to maintain the controller in their chosen cube to register their response.

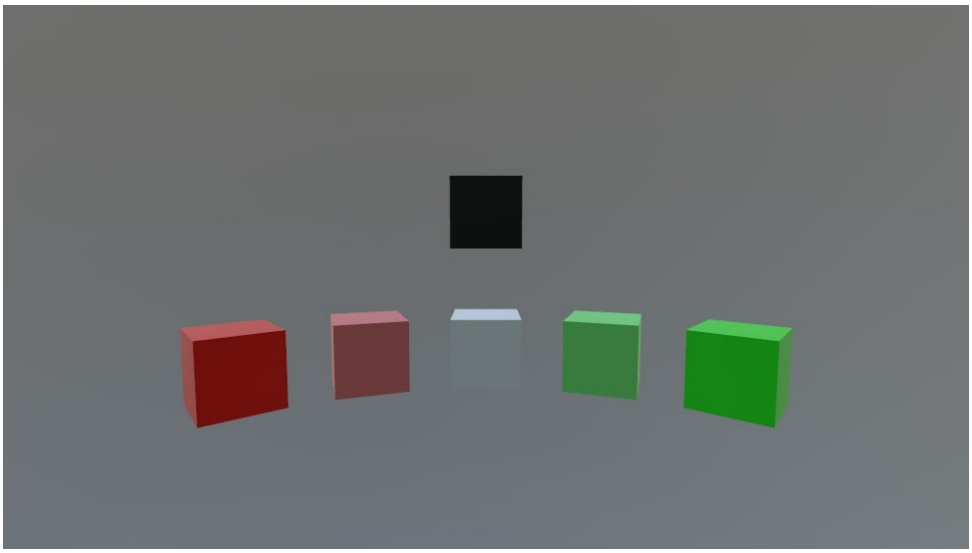


Figure 6.15: In the in-game screenshot of the Scoreboard, the bottom row of cubes represents a scale from one to five. The rightmost cube, highlighted in bright green, corresponds to the value one. The black cube at the top symbolizes the score 'Unwell.'

Upon exceeding a predefined threshold size, the script transmits information about the selected score to Record Events and commands the display of the scoreboard, signaling the continuation of the test. The desired size and rate of change are editable in the Inspector window of *Score Point*. Finally, the Scoreboard handler ensures the correct positioning of the Scoreboard GameObject relative to the user and manages the visibility of the Scoreboard and handheld controller solely during the scoring period.

In the accompanying 6.15 screenshot, the in-game Scoreboard is visualized. For this test, participants are afforded the option to select from five discrete values to articulate their opinions. If the need arises for a different number of options or a movable slider to represent opinions, such adjustments can be made without disrupting other subsystems.

As the scoreboard depended on controllers, there was a chance that, due to unforeseen issues, the controllers might either fail to work or not connect. To address this concern, the scoreboard was additionally linked to a script that functioned similarly, allowing the reporting of scores by pressing number keys. Each key corresponded to a specific score, with the number '6' indicating an 'Unwell' score.

6.4.7 Eye Tracker

The system not only facilitates viewing and gathering subjective scores but also records information concerning the user's gaze direction through a script called Eye Tracker. This is possible due to the VR system VIVE Pro eye (3.3.1). As illustrated in 6.3, the Eye Tracker operates independently of any subsystem, connecting solely to Record Events and Test Events.

The primary function of the Eye Tracker is to monitor the gaze position and relay this data to Record Events. However, the task is more intricate than it might appear. The system employs the public ViveSR plugin for Unity to enable eye tracking. Utilizing the ViveSR library, the system can retrieve gaze direction in the XYZ coordinates. Unfortunately, the provided vector only denotes the eye position within the goggles, not the comprehensive direction of a person's gaze. Furthermore, the Z-axis can be disregarded as it lacks meaningful information, as depicted in 7.2.

(a) : Side view **(b) :** Top view

Figure 6.16: Side and top view of VR Headset with eye tracking vector[1].

To accurately determine the user's eye gaze direction, the system must also track the rotation of the person's head. As the camera automatically follows the user's head position, the system retrieves information about position and rotation from the object with the camera component.

Rather than mathematically calculating gaze direction, the system leverages Unity's behavior with child objects. It employs a GameObject named Head, which autonomously updates itself with the rotation and position of the camera. The Head has a child object called the Eye, inheriting the rotation of the Head and possessing an additional rotation of its own. The Eye's rotation is updated with parameters obtained from ViveSR. Additionally, it utilizes a third object called Gaze, positioned as a child of the Eye with a

[1]Images source: `https://developer.vive.com/resources/openxr/openxr-mobile/tutorials/unity/getting-data-eye-gaze/`

z-direction offset of 10. The Gaze object points in the direction of the user's gaze, allowing the system to compute the Gaze vector from the positions of the Gaze and Head GameObjects. Subsequently, the vector is normalized.

The gaze direction vector is transmitted to the **OnGazeDirection** event in Test Events for Views to utilize. The Eye Tracker periodically records its findings, with the time period adjustable in the Inspector window. Ultimately, the Eye Tracker conveys information regarding head rotation, eye rotation, the overall gaze vector, and the time of measurement. However, eye tracking is not infallible; during rapid eye movements or blinking, the system may fail to register eye position. In such instances, head and time are recorded consistently, while eye and Gaze provide empty information.

The Eye Tracker boasts an additional functionality, namely View skipping. Recognizing that certain Views may be disorienting or unpleasant for the user, closing the eyes for a specified duration will automatically conclude the viewing period. The duration for Skip activation can be modified in the Inspector window.

6.4.8 Record Manager

The Record Manager plays a crucial role in the comprehensive recording of information intended for preservation. Triggered by the **OnStart** event, it initiates the creation of a folder bearing the name of the test subject in the specified directory path. Both the directory path and the folder name can be modified within the Inspector window. Within this subject-specific folder, four additional folders and one file are generated.

The file, designated as a text file named "Score" captures the subject's rating for a given View. It is stored in the format *"ViewName ViewScore"*. The folders, named according to their content (Head rotation, Eye rotation, Overall gaze, and Time), serve as repositories for detailed information. Each of these folders contains files corresponding to the specific Views, labeled as *ViewName*. An example is located in Section Captured Data 7.3.2 This structured approach ensures organized and retrievable storage of pertinent data related to the test subject's responses and interactions during the evaluation process.

6.4.9 Side subsystems

Beyond the core system logic, the system depends on a few subsystems that are not essential for the test to take place but provide minor enhancements for both the individual being tested and the person overseeing the experiment.

User Interface

The User Interface (UI) serves as a means for users to interact with the software application. The implementation of the UI is aligned with this purpose. As discussed in the previous chapter, numerous internal settings can be modified in the Inspector window. However, displaying such parameters in the UI for the test overseer becomes redundant after the edits are completed. Therefore, the UI is designed specifically for making small changes that are necessary or ensuring individual subject test accuracy.

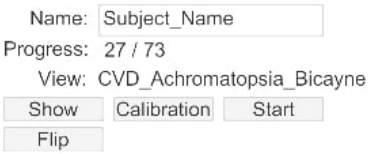


Figure 6.17: User interface for preparing the test.

The UI features an input field labeled *Name*, intended for the user to enter the subject's name. Below this field is a progress indicator displaying the rank of the current View in relation to the maximum number of Views. This provides the overseer with an approximate indication of the remaining length of the test. Next to the View is the name of the current View. In case of unexpected errors with any View, the overseer can identify which View caused the issue. Additionally, if the subject provides remarks or feedback, the overseer can associate them with the corresponding View.

There are four buttons. As discussed in the Basic Mechanics section 6.2, the sphere around the user needs to be inverted. Unfortunately, this inversion is not saved when saving the Unity project, requiring the inversion to occur each time the test is conducted. Since all spheres are disabled before the test, the **Show** button makes the Fade sphere visible, subsequently changing to the **Hide** button, which renders the Fade sphere invisible. The **Flip** button inverts the faces, and the change becomes visible in the Game window.

The **Calibration** button performs eye calibration using the eye calibration method from the *ViveSR.anipal.Eye* library. This calibration ensures that each subject has the correct position of the HMD gear, as well as eye movement calibration for eye tracking purposes. Finally, the **Start** button initiates the test, launching the OnStart event

Audio Signaling

Another smaller subsystem is dedicated to audio, comprising the Audio Manager and Audio Controller. These components work in tandem, enabling the playback of various audio clips. Several audio clips have been integrated, signaling the start and end of a View, the recording of the user's score, and the conclusion of the entire test. These audio cues prove particularly valuable

when subjects choose to skip the current View by closing their eyes. The played audio clips serve as a notification, informing them that the skip has occurred, allowing them to safely open their eyes again.

Controllers

Simulated artifacts often exhibit interrelated mechanics, rendering the implementation of redundant code for executing nearly identical functions unnecessary, especially considering the multitude of Views involved. In order to streamline and consolidate the execution processes, we introduced several controllers. Each of these controllers is equipped with the capability to simulate specific features, thereby facilitating a more cohesive and efficient approach. This methodology not only promotes code simplicity but also enables the simultaneous simulation of multiple artifacts with minimal additional coding effort.

Handlers

To facilitate convenient access to the Layer and Background spheres, the Views leverage the assistance of the Main Handler. By interacting with the Main Handler, Views can obtain references to essential elements, including the camera GameObject, as well as other scripts like Layer Handler and Background Handler associated with the respective spheres. These auxiliary handlers also provide access to references such as scripts, materials, or other components. Special case This streamlined approach empowers the Views to effortlessly modify sphere parameters. For instance, a View utilizing the Layer sphere can employ the Layer Handler to make the Layer sphere visible.

Post Processing

Post-processing is a widely utilized application in computer graphics, serving to refine the final rendered image of a scene. This toolset offers a variety of features aimed at editing the appearance of the rendered image.

In the context of Unity, the incorporation of post-processing begins with the establishment of a post-processing volume. This volume defines the spatial region within which the desired effects are applied. Following this, a post-processing layer is added to the camera GameObject, ensuring that the output of the camera is modified by the specified post-processing effects.

Post-processing provides numerous tools for image enhancement. One example is **Color Grading**, which allows for comprehensive adjustments to various aspects of color within the image. However, it's important to note that certain effects, such as **Motion Blur** and **Depth of Field**, are unavailable for non-stereoscopic cameras and cannot be used in VR applications.

Frame Counter

Given that the number of Views changes dynamically during the testing period in terms of frames, it becomes essential to gauge the average time between frames. To fulfill this purpose, a script named the *Frame Counter* has been employed to calculate the average delay between frames specifically within the testing period. Subsequently, during the **OnEnd** events, the script will display this average delay for the overseer to gather and analyze.

Collecting data from all participants, I determined their mean time between frames. The calculated mean and standard deviation were ($\mu = 22.9$ ms, $\sigma = 0.332$). While the interval between frames is approximately 23 ms, it primarily serves as informative data. Nevertheless, I will continue to refer to View parameters regarding time in frames.

Recorder

To illustrate the influence on the image, it was essential to employ a method for capturing the scene. Unity offers a specialized plugin named Recorder, which provides a range of versatile options for capturing in-game views.

The Recorder plugin includes various recording modes, such as image sequence, movie, GIF animation, animation clip, and audio. To capture screenshots in our image study, we utilized the image sequence feature with the recording mode set to Single Frame. This involved capturing the Game View in PNG format with a resolution of 1920×1080.

6.5 Legacy Application

Before developing the application detailed in this chapter, I had previously created a similar one with a comparable objective: to present various images for participants to assess subjectively. That application was created within another project. At that point, my understanding of Unity was not as comprehensive, resulting in an application lacking proper structural organization. Despite this, it provided a valuable opportunity for enhancement with a clearer vision. This section will elucidate the design of the earlier application, its utilization, and how the ongoing development has refined and built upon it.

The objective of the previous application was to collect subjective opinions on 360-degree images. In contrast to the current thesis, the images in that application did not incorporate simulated impairments. Before the presentation, the images underwent distortion through different compression methods, including JPG, HEIC, AVIF, JXL, and their respective undistorted references. The reference set comprised five distinct 360-degree images depicting different scenes, three of which are utilized in the current application. The

compressions were executed with diverse properties, resulting in a total of 126 images displayed.

As each image was unique, necessitating individualized skyboxes, I initially chose to alter the skybox material between scoring and viewing periods. Regrettably, this led to a slight lag, which participants found noticeable. Consequently, I modified the View activation to occur during the scoring period. Similarly, due to the limitations in dynamically rotating the skybox for views prone to issues like frame drop and latency, the method of using an inverted sphere was adopted.

I supervised a considerable number of participants during the tests. Due to the extensive number of test images, each test session lasted approximately 30 minutes. I observed physical discomfort and signs of exhaustion among participants, likely attributed to the repetitive exposure to practically the same set of five images. Consequently, I endeavored to minimize the test duration while ensuring an adequate number of tested views for the current pilot experiment, as detailed in 7.1.

The scoreboard underwent improvements based on participant feedback. In the current version, participants passed the controller through a single cube, resulting in a brief delay between scoring and recording the score. This lack of delay occasionally led to incorrect scores being recorded, as participants inadvertently passed the controller through the wrong cube. Additionally, the absence of an "Unwell" cube in the original scoreboard, although seemingly unimportant, was reconsidered. It was recognized that this option should always be available, even if no participant would select it.

The legacy application also captured eye, head, and gaze data similar to the current one. However, individual data gathered from different images were consolidated into a single file along with timestamp information. Each new image introduced a disruption in the data flow by inserting a line indicating the viewed image. Additionally, when recording gaze direction, four values were logged, with the first three representing XYZ data and the fourth indicating the time when the sample was taken. Both of these methods had a detrimental impact on subsequent data analysis. This discrepancy was a driving factor in streamlining the current data gathering process into individual files, as elaborated in the Capture Data section (7.3.2).

In the conceptualization of the design, I observed that various tests often cater to specific requirements. Consequently, I aimed to create an application composed of several subsystems with limited interconnections. This modular approach allows for the utilization of the core system while facilitating easy editing, removal, or addition of new subsystems.

It is noteworthy that the legacy application served as a robust foundation for the current one. Additionally, the analysis of image quality assessment was undertaken by my fellow bachelor student. His findings and data analysis are detailed in his bachelor thesis [34].

6.6 Limitations

The system was developed in Unity, providing a familiar environment. However, it has certain disadvantages, such as the inability to display certain post-processing effects (refer to 6.4.9). In the future, it would be advantageous to develop software that offers the utmost freedom in determining what is displayed. An alternative approach, as demonstrated by [26], involves using the Unreal Engine. However, it is important to consider that any development environment might have limitations to avoid causing inconvenience for the user.

While conducting the calibration of the eye-tracking feature, I identified a minor disturbance. This issue arose because the gaze direction was computed based on the relative position of the user's head rather than the center of the display sphere, leading to a slight misalignment. Although this does not impact the recorded data for eye movement and head rotation, it should be taken into account when employing gaze direction for precision analysis. Future studies might explore alternative gaze-tracking methods, such as employing ray tracing.

The participants were tasked with freely perceiving the displayed image and providing ratings afterward. However, this activity diverges from common VR practices, which typically involve interacting within a created 3D environment or engaging in various types of games. This particular aspect was not considered in the proposed solution. There is a possibility that certain impairments may be overlooked when individuals are focused on specific tasks. Therefore, future studies should consider incorporating tasks for users while simulating distortions, as explored in the thesis [24] or [10].

While in the development phase, I observed image morphing, particularly when the rendered image significantly differed from the previous one. Unfortunately, this distortion was not visible in the game view or the Steam VR display, preventing me from capturing this anomaly. The origin of such artifacts was unclear initially; I suspected that the morphing occurred due to the overload in frame calculations or potential overheating of the processor or headset. To address this concern, I designed *Info Classes* to ease the calculation load. This approach appeared to resolve the issue, as none of the participants reported experiencing such disturbances when directly queried. Investigating the root cause of this problem would be valuable. Ultimately, it is crucial to prioritize optimization to minimize the occurrence of unwanted artifacts.

Chapter 7

Pilot Experiment

This thesis incorporates a pilot experiment designed to assess the practical functionality of the developed application. I conducted these experiments with the participation of a diverse group of university students, aiming to obtain a comprehensive understanding of how well the application performs in varied scenarios.

This chapter will provide information about the participants in the experiment, its design, and the results. It also offers statistical evaluation of results. The conducted experiment aims to showcase the capabilities of the developed application.

7.1 Subjective Experiment Design

I designed the pilot experiment incorporating Views and their corresponding info classes. The test comprised a total of 73 Views, resulting in a VR testing duration of approximately 20 minutes. Each of these Views was thoughtfully crafted to highlight specific aspects of the application's capabilities. Most of these Views had simulated impairment with different parameters. This intentional diversity in the content aimed to enhance the experimental experience and evaluate how well the application performs in various situations.

Table 7.1 provides a summary of all Views created for the experiment. Each subsection will highlight the chosen parameters and the rationale behind their selection. The naming convention for each View adheres to a consistent structure: *(Impairment Name)_(Type/Value)_(Background Name)*. Examples for each View will be presented in dedicated sections. They will also elaborate on the anticipated outcomes for each tested impairment.

Background:			Hokkaido	Biscayne	Flowers
Reference:			☑	☑	☑
Latency [frame]:		1	☑	☑	☑
		2	☑	☑	☑
		3	☑	☑	☑
Noise [density %]:	Static	5	☑	☑	☑
		10	☑	☑	☑
		20	☑	☑	☑
	Global Dynamic	5	☑	☑	☑
		10	☑	☑	☑
		20	☑	☑	☑
	Headset Static	5	☑	☑	☑
		10	☑	☑	☑
		20	☑	☑	☑
Flicker [frame]:		1	☑	☑	☐
		5	☑	☑	☐
		10	☑	☑	☐
Frame Drop [frame]:	Static Interval	5	☐	☐	☑
		10	☐	☐	☑
		20	☐	☐	☑
	Dynamic Interval	5	☐	☐	☑
		10	☐	☐	☑
		20	☐	☐	☑
CVD:	Protanomaly		☐	☑	☑
	Protanopia		☐	☑	☑
	Deuteranomaly		☐	☑	☑
	Deuteranopia		☐	☑	☑
	Tritanomaly		☐	☑	☑
	Tritanopia		☐	☑	☑
	Achromatopsia		☐	☑	☑
	Achromatomaly		☐	☑	☑
Scotomas [diameter]:	Sharp edge	5	☐	☑	☐
		10	☐	☑	☐
		20	☐	☑	☐
	Grade edge	5	☐	☑	☐
		10	☐	☑	☐
		20	☐	☑	☐

Table 7.1: A table displays various settings for individual Views and the corresponding backgrounds they were utilized with. Rows highlighted in blue pertain to technical impairments, while those in green are associated with visual factors.

7.1.1 Reference Images

Incorporating a reference is crucial for comparing image impairments to their default states. The Empty View serves as a reference background, devoid of additional artifacts. Three Views were employed for each background type, adopting the naming convention *Ref_(Background Name)*.

7.1.2 Technical Impairments

This section delineates the simulations conducted to assess impairments arising from the technical aspects of the VR experience site. In total, 48 views were simulated, encompassing four distinct impairments across various parameter settings.

The chosen simulated impairments were selected to encompass all potential causes mentioned in Chapters 4 and 5. Initially, simulating various noise types (7.1.2) helps replicate poor image quality, akin to a faulty headset display. Latency (7.1.2) is directly associated with latency, as described in Section 5.1. The flickering effect (7.1.2) is often attributed to a mismatch between the frame rate of the display device and the frame rate of the content being shown (refer to Section 4.1.2). Lastly, frame drop (5.2) occupies the space between latency and frame rate since, due to substantial latency, the frame rate freezes.

Noise

Noise plays a vital role in signal processing. The simulation of noise explores three types, all implemented using the Noise View (6.4.4).

Static Noise Type is employed to simulate the distortion of the background image. This approach aims to replicate a scenario where the noise remains constant in time.

Simulating noise that varies with time is achieved using **Dynamic Noise Type**. By presenting only a small fraction of the entire 360-degree image to the user, the remainder becomes unused in the case of static noise. Employing random rotation in every frame ensures that the user perceives dynamic noise with minimal ability to discern that it originates from the same image. Utilizing this approach allows for circumventing the need for large files for video of noise. This simulates potential errors in communication between the headset and PC, hardware limitations, and related issues.

The study also simulates noise that is static to the headset, referred to as **Noise Type Headset Static**. In this scenario, regardless of the user's headset direction, the same noise pattern remains visible. This setup replicates an error in the headset display.

It was also considered to use the Gaze Follow View to have static noise fixed to the gaze direction. However, this setup cannot be justified as the image displayed in the headset is solely dependent on head rotations.

The research encompassed examinations of three noise textures across all available backgrounds for each type of noise, leading to a total of 27 distinct settings. The noise textures, generated in Matlab, were sized to match the reference image "Flower" using the *CreateSPnoise* function, which incorporates salt and pepper noise to ensure visible noise in both dark and light areas of the image. The resulting images were produced with noise

densities of 5%, 10%, and 20%. The naming convention for Views followed the format *Noise_(Noise Type)_(Noise Density)_(Background Name)*.

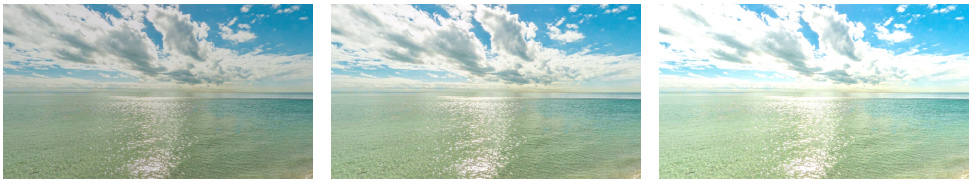
The primary objective was to observe the impact of noise types on QoE. I anticipated that static noise would be less noticeable to participants. Similarly, I expected that as the noise density increased, the QoE would deteriorate.

Flickering

Flickering is a phenomenon that may manifest in various display systems, including VR. The causes of flickering can range from issues related to frame rates to discrepancies in refresh rates (4.1.2), among other factors.

In this study, the background images "Hokkaido" and "Biscayne" were selected for testing due to their extreme brightness and darkness, respectively. The focus of the tests centered on assessing the frequency of flickering rather than the brightness of the images. To achieve this, the parameters **Max Exposure** and **Min Exposure** were set to 0.5 and -0.5, respectively, emphasizing the impact on flickering frequency. The visual effects of these parameters can be observed in figure 7.1.

(a) : Dark Hokkaido **(b) :** Normal Hokkaido **(c) :** Ligth Hokkaido



(d) : Dark Biscayne **(e) :** Normal Biscayne **(f) :** Light Biscayne

Figure 7.1: Six images were captured using the Recorder (see 6.4.9). On the left side, both images were darkened using a post-exposure value of -0.5. The middle image represents the default setting of 0, while the image on the right is brighter with a post-exposure value of 0.5.

Tested frequencies were specifically configured using the parameter **Frame Count**, which could be interpreted as half-period. It was set to values of 1, 5, and 10, providing a comprehensive analysis of the flickering phenomenon in the context of varying frequencies, in a total of 6 tested Views. A standardized naming convention is employed, denoted as *Flicker_(Frame Count)_(Background Name)*.

I hypothesize that higher frequencies (indicated by lower **Frame Count**) will result in a worse QoE score. However, I expect that the effect of flickering

will have the same impact on both backgrounds used.

Latency

The simulation of latency was achieved through the utilization of the Latency View (6.4.4). Three latency scenarios were simulated 1,2 and 3 frames in **Frame Latency** parameter. Given that frame latency would persist throughout the entire viewing period, minor latencies were chosen to mitigate potential discomfort. Latency tests were conducted on all background images, leading to a total of 9 Views utilized. The Views utilizing latency were named: *Latency_(Frame Latency)_(Background Name)*, creating names such: *Latency_2_Flowers*.

Despite setting the latencies to small values, I anticipate that increasing the latency will degrade the QoE score.

Frame Drop

In the context of technical artifacts, the final simulated anomaly is frame drop or simply lag. Lag, in this context, refers to a scenario where computational processes become overwhelmed by complexity, resulting in frames being calculated after their anticipated time with respect to the refresh rate. To emulate this phenomenon, we employed the Frame Drop View (6.4.4).

We conducted tests on two types of lag, each with three different durations, resulting in a total of six Views. The first type represents a system periodically engaging in complex processes every 2 seconds. The second type simulates lag spikes occurring semi-randomly, potentially due to suboptimal system optimization. For the latter, we activated the **Dynamic** checkbox with an **Internal Range** of 0 to 3. In both types, the lag persisted for 5, 10, and 20 frames, set in **Frame Repetition** parameter. The Frame Drop View names were as follows: *FrameDrop_(Frame Drop type)_(Frame Repetition)_(Background Name)*

7.1.3 Visual Impairments

In contrast to technical impairments, visual impairments pertain to defects in the human visual system. The primary motivation for introducing visual impairments into the test was to assess the system's ability to simulate eye defects.

Color Vision Deficiency

In this thesis, an exploration was conducted on the perceptual effects of viewing images as individuals with CVD. We had the chance to employ the "Colorblind Effect" plugin available on the Unity Asset Store, as demonstrated

in [12]. However, given the nature of our system and its limited adaptability, we opted to generate CVD simulations using our methods. To achieve this, the CVD View (6.4.4) was considered, offering the opportunity to examine the correlation between individual shifts in spectral functions and the QoE. However, recognizing the potential for test fatigue and the resultant lengthening of the testing process when using multiple Views, a strategic decision was made to employ premade color matrices with the Color Matrix View (6.4.4).

Protanopia	56.7	43.3	0	Tritanopia	95	5	0
	55.8	44.2	0		0	43.3	56.7
	0	24.2	75.8		0	47.5	52.5
Protanomaly	81.7	18.3	0	Tritanomaly	96.7	3.3	0
	33.3	66.7	0		0	73.3	26.7
	0	12.5	87.5		0	18.3	81.7
Deuteranopia	62.5	37.5	0	Achromatopsia	29.9	58.7	11.4
	70	30	0		29.9	58.7	11.4
	0	30	70		29.9	58.7	11.4
Deuteranomaly	80	20	0	Achromatomaly	61.8	32	6.2
	25.8	74.2	0		16.3	77.5	6.2
	0	14.2	85.8		16.3	32	51.6

Table 7.2: Premade color matrices for Color Matrix View.

It is crucial to acknowledge that anomalous color vision deficiencies (protanomaly, deuteranomaly, tritanomaly) each have their unique color matrices. As they depend on various factors related to function, even slight changes could result in slightly different matrices. This aspect could be intriguing to explore in future studies.

Color matrices were taken from [1]. They were adjusted to seamlessly integrate with post-processing. The specifics of these matrices can be found in table 7.2. The evaluation encompassed testing each type of CVD on two distinct background images, namely "Biscayne" and "Flowers," culminating in a total of 16 views. The naming convention for these follows a structure as *CVD_(Color Blindness type)_(Background Name)*.

Given the subjective nature of color enjoyment, results from CVD Views will be compared against objective metrics.

Scotoma

Precisely replicating the genuine impact of a scotoma requires an understanding of the inherent properties associated with this blind spot. However, this study did not delve into the comprehensive examination of this specific impairment. Consequently, the inclusion of Views simulating scotomas serves the purpose of demonstrating the potential of the developed application.

In alignment with the approach outlined in [48], two types of masks were adopted: one with a sharp edge and the other with a gradually fading edge.

The assessment was conducted for a singular background, namely "Biscayne", with three sizes of scotomas, resulting in 6 total Views. The View used was Gaze Follow View (6.4.4).

(a) : Sharp edge **(b) :** Gradually fading edge

Figure 7.2: Examples of the employed masks. Images are not to scale.

The scotoma masks were generated using a Matlab function called *CreateScotoma*. With diameters calibrated relative to the individual's visual field at 5, 10, and 20 degrees. The diameter of 10 degrees was tested in [48]; therefore, the experiment was extended to include both double and half the original diameter. For the GE masks, the intensity linearly decreased from 80% of the diameter to 120% of the diameter after reaching that threshold. Created masks were set in **Texture** parameter. The names used for Views with blind spots were named *Scotoma_(Scotoma Diameter)_(Scotoma Type)_(Background Name)*.

7.2 Participans

This section of the chapter is dedicated to participants of the pilot subjective experiment. Encompassing a description of the tested group, the pre and post-test procedures for administering the questionnaire, and instructions on how individuals can interact with the application. Additionally, I have included the most common remarks from participants regarding the test and the procedures.

I enlisted 25 participants for my pilot experiment, comprising 4 females and 21 males. All participants were students aged between 19 and 25, with a median age of 21. Half of the participants had unimpaired vision, while the remaining 12 wore glasses or contact lenses. The recruitment took place at the Czech Technical University of Prague, and all participants volunteered willingly. Many expressed eagerness to take part due to their limited or nonexistent experience with VR.

7.2.1 Procedure

To commence the test, the procedure was verbally explained in detail. This explanation covered each simulated impairment and how to provide a subjec-

tive score. Participants were also informed about the option to close their eyes to skip a View if they wished. Following this explanation, participants were reconfirmed for their willingness to proceed, now fully informed about the test details.

After consenting to the testing, participants completed two questionnaires. The first gathered general information, including name, age, VR experience, and the use of glasses or contact lenses. The second questionnaire focused on simulator sickness, addressing general symptoms that might arise during VR experiences. The symptoms can be viewed in table 7.3. They assess each symptom on a scale from 0 to 3, where 0 corresponds to "None," 1 to "Mild," 2 to "Moderate," and 3 to "Strong." The second questionnaire was administered after the testing, and the results were subsequently compared. Both questionnaires can be found in the appendix in the survey folder.

After completing the questionnaires, participants were guided on how to wear the headset. Those with glasses were given the option to keep them on if they fit comfortably inside the headset. However, none chose this option due to discomfort while wearing both glasses and the headset.

Once the headset was correctly worn, calibration took place using the headset's calibration function, adjusting the headset position, lens distance, and eye tracking. Subsequently, the actual test commenced, preceded by a final verbal confirmation from participants expressing their understanding of all instructions.

The test duration was approximately 20 minutes, with a supervisor present throughout. All participants acknowledged the supervisor's presence. No tests were interrupted due to cybersickness or other discomfort. Only one test experienced a minor issue – the controller for participants to input scores had run out of batteries. Nevertheless, this was promptly addressed as the overseer pressed the corresponding number keys as instructed by the participant.

After the test concluded, participants were assisted in removing the headset. The second questionnaire was then filled out, followed by an opportunity for participants to share their thoughts about the application.

7.2.2 Participants comments

Some participants shared their perspectives on the test, and since some opinions were similar, this section highlights the main ideas. The overall discomfort will not be discussed further, as it was evident that simulating image impairments in VR would not provide a pleasant experience.

Primarily, a significant number of participants expressed strong dislike for any impairment in the form of a screen not being synced to the headset, such as Latency and Frame Drop View. Many believed that extended exposure to such issues would induce feelings of sickness, but fortunately, each View lasted only 10 seconds. One participant with extensive VR experience automatically skipped any View with Latency or Frame Drop, knowing the potential effects

and wishing to avoid them. Conversely, another participant mentioned that they didn't mind the lag because they were accustomed to it.

A secondary group of participants explained their scoring rationale. Initially, they scored based on their enjoyment. However, their scoring shifted when they encountered highly distorted Views because they "now knew what was the worst."

One participant expressed dissatisfaction, finding the test dull. Similarly, another participant explained that when attempting to relax, they felt sick, so they tried to stay focused. This sentiment is understandable, considering participants only saw three types of background images and were tasked to perform any type of action.

In the second questionnaire, nearly all participants expressed confusion regarding the "head fullness" symptom. To elaborate briefly, the consensus explanation for this symptom was described as "the sensation of overwhelming information to process in the head." This definition gained agreement from a participant with regular VR experience, who noted experiencing it after prolonged sessions in virtual reality.

Finally, there were opinions about Views with visual impairments. Many said that they enjoyed color-adjusted images since they were new and unique to them. Also, a few said that any background, even with lower quality, still could fit with the overall aesthetics of the environment when aiming, for example, for "pixel art" or "horror" style. Similarly, the blind spot simulation served as an entertaining view, where the participants were surprised that the blind spot followed their gaze.

The application received minimal comments, and notably, there were no specific remarks or concerns raised regarding potential errors in the application.

7.3 Results

In this part of the chapter, the outputs of the pilot experiment are outlined. The term "output" encompasses various types of information obtained from subjective tests.

One form of output involves the questionnaires, which will be compared and analyzed to understand their impact on participants' well-being.

Furthermore, data were gathered through eye tracking, and I will illustrate the format in which the data are recorded. Subsequently, I will explore different approaches for analyzing eye-tracking data.

Lastly, a statistical analysis will be performed on the QoE scores derived from the test. This analysis will entail comparing individual Views and parameters using a range of statistical methods. However, it is important to note that the primary objective was not to achieve statistically significant evaluations of impairments but rather to showcase the capabilities of the

created system. Therefore, the statistical analysis does not extensively explore this aspect.

7.3.1 Simulator sickness

Each participant completed a simulator sickness questionnaire (taken from [16], can be found in digital attachment), and by comparing the pre and post-test results, we can analyze the impact it had on participants in terms of health symptoms.

Table 7.3 illustrates the means of pre and post-test symptoms along with their differences. The anticipation was that symptoms would worsen after the test due to general fatigue and unpleasant images. It is evident that the primary symptoms that worsened include 'General discomfort,' 'Headache,' 'Eye fatigue,' and 'Head fullness.' This observation isn't surprising as these symptoms are commonly experienced after VR usage. Additionally, it can be noted that the largest difference in symptoms is '0.44,' which does not represent half of the quantified step on the scale. Again the scale is from 0 to 3, where 0 corresponds to "None," 1 to "Mild," 2 to "Moderate," and 3 to "Strong."

Symptoms	Pre-test	Post-test	Difference
General discomfort	0.32	0.6	-0.28
Fatigue	0.8	0.96	-0.16
Headache	0.04	0.48	-0.44
Eye fatigue	0.44	0.88	-0.44
Difficulty concentrating	0.44	0.48	-0.04
Increased salivation	0.04	0.12	-0.08
Increased sweating	0.32	0.4	-0.08
Nausea	0.08	0.24	-0.16
Head fullness	0.48	0.76	-0.28
Blurred vision	0.16	0.28	-0.12
Loss of balance (with open eyes)	0	0.08	-0.08
Loss of balance (with closed eyes)	0.04	0.16	-0.12
Dizziness	0.04	0.16	-0.12
Nausea	0.08	0.08	0
Belching	0.04	0.04	0

Table 7.3: Symptoms table: The left column contains the names of each symptom, followed by the means of pre and post-test values for all participants. The right column illustrates the average difference between pre and post-test values.

7.3.2 Captured data

The developed application successfully captured several data. This section will illustrate how the collected data is stored and the format in which it is structured.

At the initiation of each tested subject, the overseer is tasked with entering the names of participants into the system's UI (see Section 6.4.9). The participant's name serves as the folder's name, containing organized data. Inside the folder, the structure follows the pattern shown in 7.4.

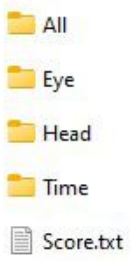


Figure 7.3: Organizational breakdown of each name file: The "All" folder encompasses the comprehensive gaze direction data, "Eye" focuses on exclusive eye movements, "Head" captures head rotation, and "Time" records the timing of the reading session. The "Score.txt" file houses the subjective QoE assessment for the respective participant.

The directories labeled "All," "Eye," "Head," and "Time" encompass individual text files for each displayed view in the test. The filenames correspond to the names assigned to each view, facilitating straightforward data analysis for both individual and specific views.

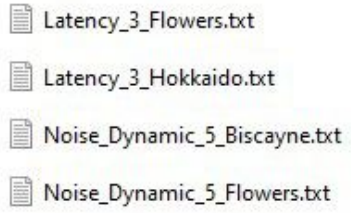


Figure 7.4: A segment within any inner directory consists of files named after the respective Views in which the data was recorded.

Depending on the file's location, it contains one of four types of data, as explained and visualized in figure 7.5. In subfigure 7.5a, empty lines are visible, indicating instances when eye tracking fails to record data (such as when the eyes are closed). This also underscores the importance of time tracking when the system attempts to capture eye movement, as highlighted in figure 7.5d. Because the files are of the same length, it is possible to precisely determine when and for how long the eyes were closed.

```
0.9649855 -0.05048182 0.2573997
0.9637311 -0.05251988 0.2616561



0.9302683 0.1664647 0.3269411
```

(a) : All data

```
3.755839 -8.687564
3.061673 -8.736523
1.861309 -9.176277
14.50754 -10.94754
14.99188 -11.16785
13.79502 -11.65831
```

(b) : Eye data

```
356.9804 96.92049 351.3708
356.9256 97.07448 351.3416
356.8713 97.14777 351.3402
356.8672 97.16081 351.3501
356.8608 97.1682 351.4379
356.8896 97.13916 351.5083
```

(c) : Head data

```
4.464675
4.51387
4.521667
4.530869
4.553332
4.598597
```

(d) : Time data

Figure 7.5: Collecting data involves utilizing the *Record Manager* (see 6.4.8). Subfigure (a) illustrates the format of gathered data for gaze direction, displaying coordinates XYZ of the normalized vector of gaze direction. Subfigure (b) presents purely eye data in the X and Y directions in degrees. Head rotation is captured in XYZ degrees in subfigure (c), while subfigure (d) records the time elapsed from the start of the View.

```
CVD_Achromatomaly_Flowers 5
Noise_Static_10_Flowers 3
Noise_HeadsetStatic_20_Hokkaido 4
Flicker_10_Biscayne 5
CVD_Deuteronomaly_Biscayne 1
Scotoma_10_Sharp_Biscayne 5
Flicker_1_Hokkaido 5
FrameDrop_Static_5_Flowers 5
CVD_Protanomaly_Biscayne 1
CVD_Deuteranopia_Biscayne 1
CVD_Protanopia_Biscayne 2
CVD_Trinopia_Biscayne 5
Scotoma_20_Grade_Biscayne 5
Latency_3_Biscayne 5
Noise_Static_10_Biscayne 4
Noise_Dynamic_20_Biscayne 5
FrameDrop_Dynamic_20_Flowers 5
FrameDrop_Dynamic_10_Flowers 5
```

Figure 7.6: A portion of participant scores, where each line comprises the View name and the score assigned by the participant to that specific View, separated by space.

Figure 7.6 illustrates the format for storing QoE scores. This format also reveals the sequence in which Views are presented to each individual participant. The scores range from 1 to 5, with 1 representing the highest score (refer to 2).

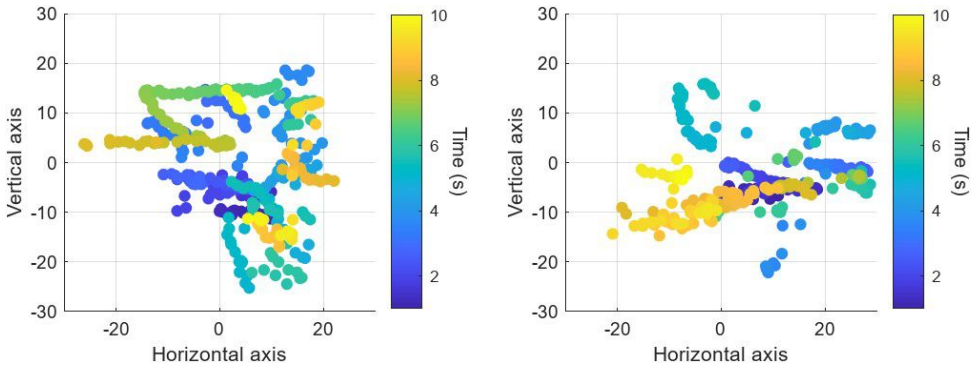
7.3.3 Eye tracking

In addition to the subjective QoE score, the system collected various types of eye-tracking data. The recording, facilitated by the Record Manager, captured extensive information related to eye movements for each View.

Motion Analysis

As the system records the positions of the eye, head, and gaze along with the corresponding timestamps, it enables the visualization of the subject's motion within a specific perspective. This section will demonstrate this capability.

We successfully analyzed eye movements in Matlab, integrating this data with the reported time. The graphical representation of exclusive eye movements is illustrated in figure 7.7. We presented the eye movement patterns of a single individual for two distinct Views. Specifically, we exhibited eye movements derived from a reference image and another with a simulated blind spot, aiming to observe and compare behavioral differences in the presence and absence of a blind spot.



(a) : Normal **(b) :** Blind spot

Figure 7.7: Example of captured eye movement: a) no impairment, b) with blind spot simulation.

The same process can be done with the gaze direction. The collective gaze direction is recorded as an XYZ vector. Through projection, this XYZ vector can be accurately mapped onto a 2D plane. Visualizing such data involves overlaying it onto a background, which itself is a 2D representation of a 360-degree image.

The calculations were performed using Matlab. Initially, utilizing gathered time data, individual gaze positions were mapped to specific points in time. This process allows for the visualization of the progression of gaze movement, as depicted in figure 7.8.

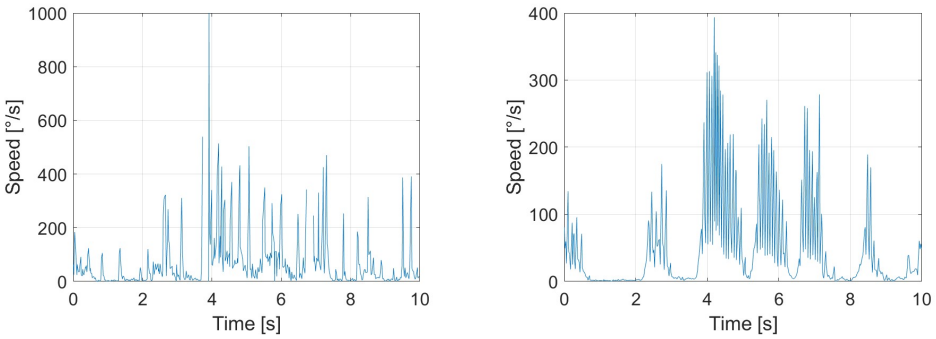
Figure 7.8: Observing an individual's gaze synchronized with a relevant background image. The figure also depicts gaze position based on time by utilizing time data.

Speed Analysis

This section aims to demonstrate the feasibility of illustrating the subject's movement speed over time.

To visualize speed, the approach involves computing the difference between two samples and dividing it by the difference in corresponding time samples. The speed analysis of eye and head movements is conducted in MATLAB. The eye movement is represented in XY degrees, and the head rotation is represented in XYZ separately. Consequently, the speed will be presented as degrees over time.

In the case of head movement speed calculation, I opted to exclude the Z rotation, which represents the head roll, as it does not significantly impact the overall direction. It is acknowledged that mapping the distance between two samples onto a sphere is necessary for accurate position representation. However, between two consecutive samples, the X and Y differences were not substantial enough to create a significant distinction between mapping on a 2D plane or a sphere. Therefore, for both eye and head movements, the overall magnitude was calculated using the Pythagorean theorem.



(a) : Eye speed **(b) :** Head speed

Figure 7.9: Charts illustrating the temporal evolution of the speed (in degrees per second) for the specified component, with (a) representing eye speed and (b) representing head speed.

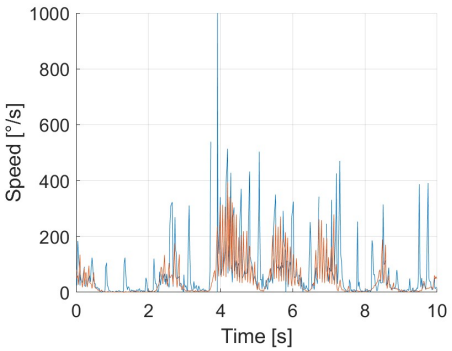


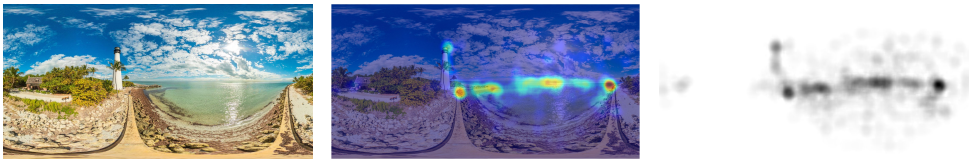
Figure 7.10: Comparison graph depicting the speed variations of the eye and head over time.

I presented the individual speed profiles for both the eye and head of a randomly selected individual, as shown in figure 7.9. Additionally, figure 7.10 offers a comparison of both speeds in a single graph. Upon examining the graphs, it becomes apparent that there are distinct periods of movement and stillness in both eye and head speed. This observation is further emphasized in the combined graph, where the periods of movement for the eye and head overlap.

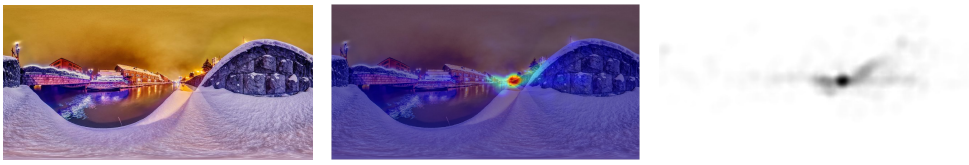
Heatmaps

In this section, we delve into the gaze direction data analysis, unlocking the ability to unveil the prevalence of frequently visited locations or regions of interest. Rather than representing gaze positions as isolated points (as seen in Motion Analysis 7.3.3), we extend our approach to encompass entire areas. Through the strategic overlaying of these areas, we unveil a comprehensive magnitude map that vividly illustrates the significance and patterns of common places, resulting in the creation of a captivating heatmap effect.

Following projection, the gaze vector is translated onto an XY grid, with each point represented by a single pixel. By convolving each pixel with a custom mask that accurately reflects the focus of human vision. I employed a circular mask. The inner circle, with a diameter of 8 degrees, was completely filled, symbolizing paracentral vision. Progressing from the end of the paracentral vision, the mask linearly fades to a diameter of 18 degrees, representing macular vision.



(a) : Biscayne background **(b) :** Biscayne heatmap **(c) :** Biscayne average gaze

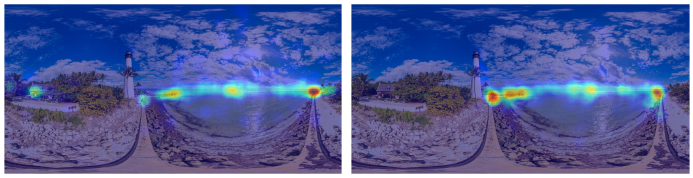


(d) : Hokkaido background **(e) :** Hokkaido heatmap **(f) :** Hokkido average gaze

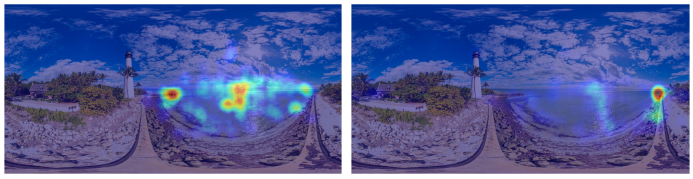
(g) : Flowers background **(h) :** Flowers heatmap **(i) :** Flowers average gaze

Figure 7.11: Display of heatmaps for individual backgrounds is featured here. The left column exhibits reference background images, while the right column presents the averaged gaze positions of reference Views, from all participants. The middle column displays a heatmap illustrating common areas of fixation.

Overlaying individual heatmaps and subsequently averaging the values allowed me to generate a visual representation of common fixation positions. By associating the corresponding background and applying a colormap to the averaged gaze positions, where the color red indicates a higher concentration of fixations. These heatmaps are depicted in figure 7.11. It is evident that common fixation points, such as the lighthouse, ocean horizon, and shore, can be observed for the Biscayne background.



(a) : Participant 8 **(b) :** Participant 17



(c) : Participant 22 **(d) :** Participant 25

Figure 7.12: Heatmap of 4 random participants over every View using Biscayne background.

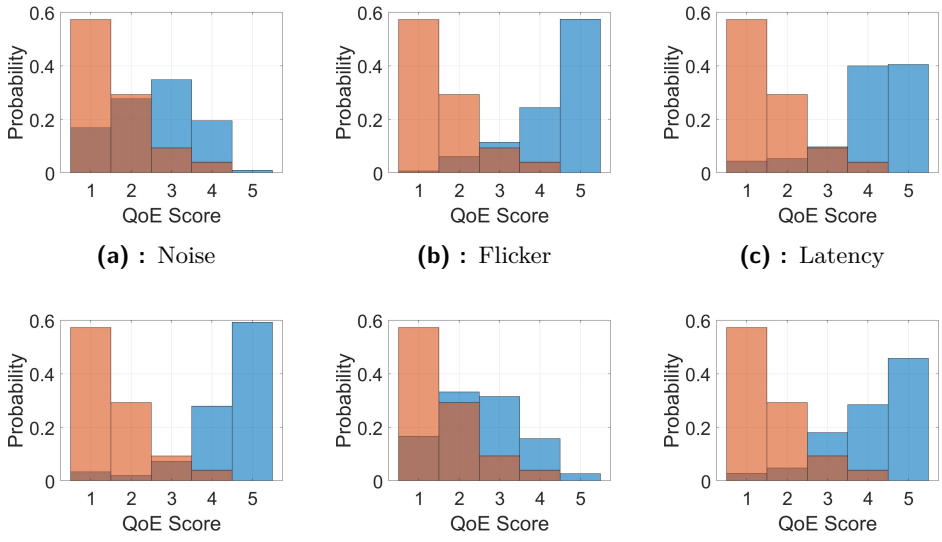
However, we can also generate heatmaps specific to each individual. This approach reveals distinct fixation points for each participant, as illustrated in figure 7.12. It is evident that Participant 22 exhibited a more scattered viewing pattern, whereas Participant 25 appeared to concentrate on a single point.

This feature enables the expansion of the project, such as the analysis of fixation points, for example.

7.4 Subjective Score

In this section, I will analyze the QoE scores gathered. They were the main measure gained from subjective tests. They are illustrated in Figure 7.6, ranging between values 1 to 5 (refer to Section 2).

To visualize distinctions in QoE scores, I have generated histograms that compare the reference to each simulated impairment, as depicted in figure 7.13. The reference QoE score is predominantly centered around 1 (indicating the best score), with some dispersion into 2 and 3. Notably, the impairment histograms exhibit distributions near scores 4 and 5, except for noise (see figure 7.13a) and color vision deficiency (see figure 7.13e).



(a) : Noise **(b) :** Flicker **(c) :** Latency

(d) : Frame Drop **(e) :** CVD **(f) :** Scotoma

Figure 7.13: Normalized histograms contrasting the reference QoE score in red with the simulated impairment QoE score in blue. Each histogram compares the reference to a specific impairment, namely: (a) Noise, (b) Flicker, (c) Latency, (d) Frame Drop, (e) CVD, and (f) Scotoma.

7.4.1 Statistical Analysis

The following section will delve into the statistical analysis of various simulated impairments. Primarily, the analysis will center on the differences in QoE scores between the reference and impairments. In some instances, a more in-depth examination was conducted to highlight the varying effects each impairment has on QoE.

The pilot experiment served the sole purpose of testing the developed system, resulting in a straightforward analysis. However, for a comprehensive study, a more extensive subject pool and a more detailed specific impairment sequence would be essential.

The statistical tests employed included the Kruskal-Wallis test and post hoc analyses. This method was selected due to the non-parametric nature of the gathered data. Since the testing approach for many impairments was similar, numerical values for the test results will not be provided. Instead, the results will be presented in a clear and comprehensible table. The methodology behind the creation of these tables is demonstrated in the Noise section 7.4.1.

Noise

In this section, we will explore the impact of noise on QoE. I will compare different types of noise to the reference group and present the test results, illustrating them in the subsequent table.

In the hypothesis, noise types (see Section 7.1.2) are denoted as "NS" for Noise Static, "ND" for Noise Dynamic, "NH" for Noise Headset Static, and "R" for reference. The example hypotheses for noise are as follows:

$\mathbf{H_0}$ *: The mean ranks of the groups NS, ND, NH, and R are the same*

$\mathbf{H_A}$ *: There is a significant difference in the mean ranks among the groups NS, ND, NH, and R*

Kruskal-Wallis test showed that the groups are different ($\chi^2(3, 749) = 117.5, p < 0.01$), therefore rejecting the null hypothesis H_0 and accepting H_A. Using post hoc analysis showcased that every noise type is significantly different from reference scores ($D(R, NS) = -116, p < 0.01$, $D(R, ND) = -235.8, p < 0.01$, $D(R, NH) = -250.0, p < 0.01$), where the function D(,) denoted the different in the mean ranks. The dynamic and headset static noise didn't significantly differ ($D(ND, NH) = -14.2, p = 1$, however static noise differed from both ($D(ND, NS) = -118.8, p < 0.01$, $D(NH, NS) = 133.0, p < 0.01$)).

Noise	Ref	Static	Dynamic	Headset S.
Ref				
Static				
Dynamic				
Headset S.				

Table 7.4: A table contrasting various noise types (Static, Dynamic, Headset Static) with both the reference and each other. The color signifies the association between the row group and the column group. White cells indicate they are the same group. Red and green denote a significant difference, with green cells indicating a smaller mean of ranks and red cells signifying the opposite. Gray cells represent no significant difference.

The results are presented in a table for clarity, where both rows and columns correspond to the analyzed groups, see table 7.4. Row names indicate the first group compared to the one in the columns. White cells signify that no comparison was performed as they represent the same group. Gray cells indicate non-significantly different groups, such as Dynamic and Headset Static Noise ($D(ND, NH) = -14.2, p = 1$), where the threshold was set to 0.05. Green cells indicate significant differences, with the row group having a better (smaller) mean of ranks ($D(R, NS) = -116, p < 0.01$). Red cells also represent significantly different groups, but the row group has a worse (larger) mean of ranks ($D(NS, R) = 116, p < 0.01$).

Table 7.4 presents the statistical outcomes for various noise types. Similarly, I applied Kruskal-Wallis and post hoc tests to examine noise densities (see 7.1.2) for each specific noise type, and these results are provided in table 7.5. The tables serve illustrative purposes, and numerical values of p-values and other details were excluded for simplicity.

Static	5%	10%	20%
5%			
10%			
20%			

(a) : Static Noise

Dynamic	5%	10%	20%
5%			
10%			
20%			

(b) : Dynamic noise

Headset S.	5%	10%	20%
5%			
10%			
20%			

(c) : Headset Static noise

Table 7.5: Three tables illustrate the statistical relationships among various noise densities (5%, 10%, 20%) for each noise type. Table (a) portrays the relationships in static noise densities, (b) in dynamic noise densities, and (c) in headset static noise densities.

Upon reviewing table 7.5, it becomes evident that the 5% noise density consistently results in better QoE scores compared to other noise densities across all noise types. Interestingly, there is a noteworthy observation that, in all noise types, there was never a significant difference between QoE scores for 10% and 20% noise densities.

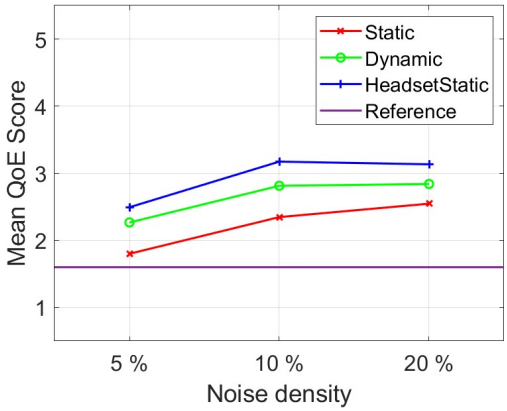


Figure 7.14: A graph illustrating the mean QoE scores based on noise density. It includes all three types of noise, accompanied by a reference line that showcases the mean QoE score of the reference Views.

I have generated a graph displaying the mean score values corresponding to different noise densities for all noise types, as depicted in figure 7.14. The average scores for all groups surpass those of the average reference. This highlights that every simulated noise negatively affects the QoE for

participants. Additionally, there is a gradual increase in QoE scores with higher noise density, a trend expected as more of the screen is covered with noise. Notably, Headset Static noise exhibits the highest score across all noise densities; however, no statistically significant differences were identified, as indicated in table 7.5.

Flickering

In this section, I will perform a statistical analysis on the Views affected by the flickering impairment (7.1.2). Three types of flickering were simulated, differing in the duration of either the light or dark part staying on the screen, referred to as the half-period.

Flicker	Ref	1 frame	5 frames	10 frames
Ref				
1 frame				
5 frames				
10 frames				

Table 7.6: A table contrasting various flickering half-periods (1 frame, 5 frames, 10 frames) with both the reference and each other.

Table 7.6 illustrates that all frequency types significantly deteriorate the QoE compared to the reference. Interestingly, the 5-frame half-period did not exhibit a significant difference from the other half-periods.

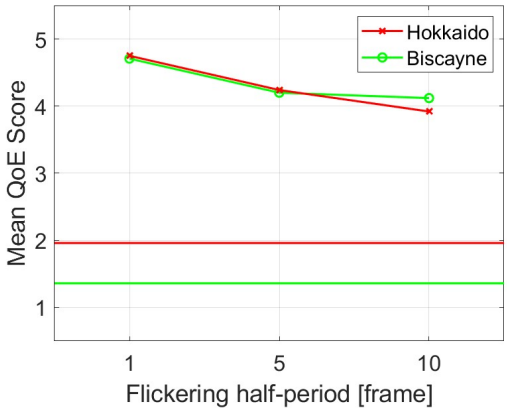


Figure 7.15: A chart displaying the mean QoE scores corresponding to different half-periods for two backgrounds: Hokkaido (in red) and Biscayne (in green). The two lines at the bottom represent the reference scores for each background.

I have created a graph presenting the mean QoE scores based on different half-periods 7.15. The graph indicates a deterioration in scores with smaller half-periods, signifying higher frequencies. Additionally, it showcased that the flickering degrades QoE similarly regardless of background image quality.

Interestingly, for 1 and 5-frame half-periods, Views using both backgrounds demonstrated similar scores. However, with the 10-frame half-period, their scores diverged, but in the opposite direction from their references. This suggests that, although the Hokkaido reference background had an overall worse QoE score than Biscayne, it performed better in the 10-frame half-period. This could be due to Biscayne being a brighter image, therefore the flickering affects it differently. However, that is only speculation.

Latency

This section concentrates on examining latency Views. The test involved simulating three different degrees of latency across various backgrounds (see Section 7.1.2).

Latency	Ref	1 frame	2 frames	3 frames
Ref				
1 frame				
2 frames				
3 frames				

Table 7.7: A table contrasting various simulated latency (1 frame, 2 frames, 3 frames) with both the reference and each other.

The outcomes of the Kruskal-Wallis test and post hoc analysis can be observed in table 7.7. As anticipated, all simulated latency scenarios performed more poorly than the reference. Surprisingly, different types of latency do not show statistically significant distinctions from one another. This unexpected finding is likely due to the similarity in the degrees of latency among them.

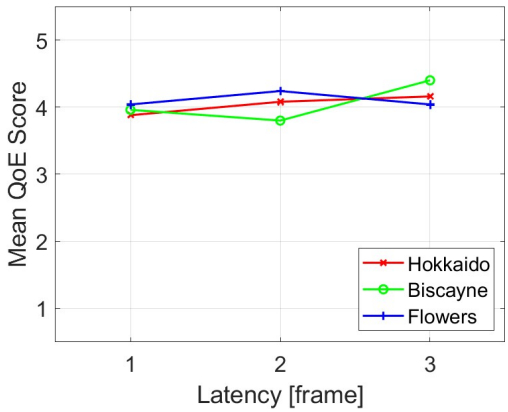


Figure 7.16: A chart displaying the mean QoE scores based on simulated latency for all three backgrounds: Hokkaido (in red), Biscayne (in green), and Flowers (in blue).

We can visualize the impact of latency on scores for each background, as illustrated in figure 7.16. The mean QoE score appears relatively consistent

across all latencies and backgrounds. However, it is noteworthy that even with the smallest simulated latency (1 frame), the score significantly deteriorates.

Frame Drop

This section delves into the impact of frame drops or simulated lag on QoE scores. Two types of frame drops were simulated: one where the lag occurs periodically and the other where it happens semi-randomly. For both types, the lag persistence was configured at 5, 10, and 20 frames. See Section 7.1.2 for more detail.

Frame D.	Ref	Static	Dynamic
Ref			
Static			
Dynamic			

Table 7.8: A table contrasting simulated frame drop types (static, dynamic) with both the reference and each other.

Comparing the two types of lag to the reference, the results are presented in table 7.8. Both types negatively impact the QoE score; however, no significant difference was found between the two lag types.

Static	5 frames	10 frames	20 frames
5 frames			
10 frames			
20 frames			

(a) : Static Frame Drop

Dynamic	5 frames	10 frames	20 frames
5 frames			
10 frames			
20 frames			

(b) : Dynamic Frame Drop

Table 7.9: Two tables illustrate the statistical relationships among the lag persistence (5, 10, 20 frames) for both types of Frame Drop: (a) Static, (b) dynamic.

Similarly, I have generated a table for lag persistence for both types of frame drops, as shown in table 7.9. Here, we can note that for both lag types, the test did not identify significant differences among lag persistence groups. This may be attributed to the limited number of tested views, considering that Frame Drop Views were exclusively tested on the Flowers background.

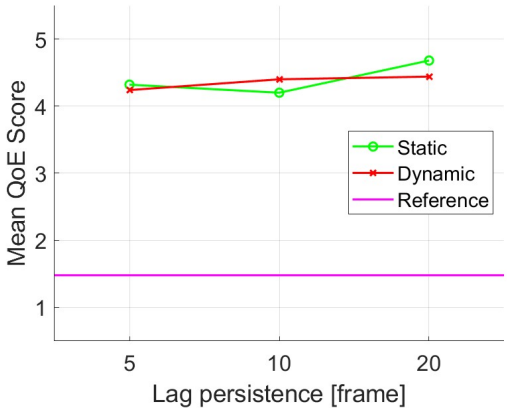


Figure 7.17: A graph depicting the mean QoE scores based on lag persistence. It showcases two types of Frame Drop along with a reference line.

Finally, I have plotted the dependency of QoE scores on lag persistence in figure 7.17. It is noticeable that the scores cluster around the same point for both types of lag. Additionally, a slight overall increase in QoE score with growing lag persistence is observed; however, it's important to note that this is merely an observation, as the statistical tests do not support this trend (refer to table 7.9).

Scotoma

This section is devoted to the comparison of the scotoma's effects (7.1.3). Three scotoma sizes were tested: 5, 10, and 20 degrees in diameter. There were two types, one with a sharp edge (sharp) and one with a gradually fading edge (grade).

Scotoma	Ref	5 degrees	10 degrees	20 degrees
Ref				
5 degrees				
10 degrees				
20 degrees				

Table 7.10: A table contrasting simulated frame drop types (static, dynamic) with both the reference and each other.

Table 7.10 illustrates the relationship between different sizes of scotoma. In the table, we observe that the test did not identify significantly different scotoma size groups, unlike when comparing any size of scotoma to the reference.

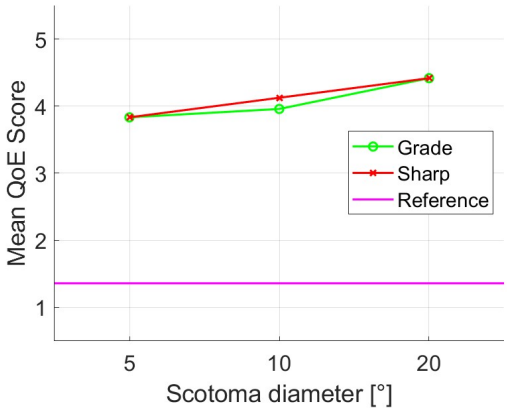


Figure 7.18: A graph illustrating the dependency of mean QoE scores on scotoma size. The green line represents the gradually fading edge (Grade), the red line represents the sharp edge (Sharp), and the reference line showcases the average score of the background used.

Finally, I have plotted the mean QoE score dependency on scotoma size in figure 7.18. I overlaid two dependencies for the grade and sharp edge. We can observe that the score doesn't change drastically in relation to the type of edge. However, there is an ever-so-slight increase in score with the growth of the scotoma.

CVD

In this section, we highlight how various types of CVDs impact the QoE score. We tested eight types of CVDs (see Section 7.1.3), including protanomaly, protanopia, deuteranomaly, deuteranopia, tritanomaly, tritanopia, achromatomaly, and achromatopsia. The evaluation was conducted with two different backgrounds, Flowers and Biscayne.

CVD	Ref	Py	Pa	Dy	Da	Ty	Ta	Ay	Aa
Ref									
Py									
Pa									
Dy									
Da									
Ty									
Ta									
Ay									
Aa									

Table 7.11: An informative table delineating distinctions among simulated types of color vision deficiencies. CVDs consist of Py (protanomaly), Pa (protanopia), Dy (deuteranomaly), Da (deuteranopia), Ty (tritanomaly), Ta (tritanopia), Ay (achromatomaly), and Aa (achromatopsia).

The Kruskal-Wallis test results are presented in table 7.11. The findings indicate a predominantly negative impact on the scores across various types

of CVDs. Notably, conditions involving the absence of cones, such as -opia, demonstrated lower scores compared to other CVDs.

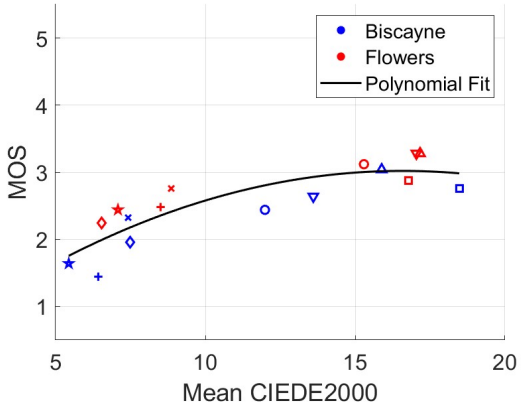


Figure 7.19: A graph illustrating the correlation between the mean opinion score (MOS) and the objective metric CIEDE2000. The color of each dot corresponds to different backgrounds, with Biscayne represented in blue and Flowers in red. The shape of the dots signifies various types of CVDs: Protanopia(**o**), Protanomaly(+), Deuteranopia(∇), Deuteranomaly($\star$), Tritanopia($\square$), Tritanomaly($\diamond$), Achromatopsia($\triangle$), Achromatomaly($\times$). The graph was also subjected to second-degree polynomial regression fitting.

Finally, I plotted the mean opinion score (MOS) against the mean of the objective metric CIEDE2000 (see figure 7.19). The objective metric (2.1) was calculated in Matlab using the function imcolordiff. To obtain the objective metric, I captured 360-degree images from the Unity environment using the recorder described in Section 6.4.9. To mitigate the potential impact of the 3D environment on color perception due to simulated lighting, I took images directly from the scene for both reference and CVD images.

After acquiring reference images and images for all CVDs for both backgrounds, I conducted comparisons in MATLAB. The MOS values were derived from recorded scores. The resulting graph illustrates the correlation between the objective metric and subjective scores, further emphasized by the display of second-degree polynomial regression.

Additionally, different types of CVDs are visually distinguished using various dot shapes, enabling the observation of their differences. Notably, CVDs of lower severity, such as protanomaly, deuteranomaly, tritanomaly, and achromatomaly, exhibited lower scores in both MOS and mean CIEDE2000. This pattern is also evident in table 7.11. When comparing individual CVDs, it is observed that the same CVD for both backgrounds tends to be relatively close to each other.

Chapter 8

Conclusions

This thesis extensively explores image impairments in virtual reality and their impact on Quality of Experience. The primary objective was to design a system capable of simulating various types of impairments for subjective measurement.

As detailed in Section 6.5 on Prior Applications, I previously developed a subjective measurement approach for Image Quality Assessment for a different project. This earlier project provided valuable insights for conceptualizing the architecture of the current system. The resulting system is constructed from multiple subsystems operating in harmony. The key advantage lies in its expandability, allowing for the alteration of different subsystems. This intentional design choice provides a basic structure while acknowledging the potential need for a customized approach for each test.

The application facilitated the sequential simulation of various impairments, regardless of the means and methods employed in the simulation process. These simulated impairments were consistently presented against static 360-degree images serving as backgrounds. The methods involved manipulating the background, incorporating layers to obstruct portions of vision, and introducing effects that influenced the lighting within the scene. These simulated artifacts spanned both technical and visual dimensions, with their origins and previous investigations documented in Chapters 4 and 5. On the technical spectrum, our simulations encompassed latency, noise, flicker, and frame drops. From a biological standpoint, our simulations included replicating color blindness and blind spots.

In the pilot experiment, a deliberately restricted number of variations for each impairment was incorporated to ensure a focused analysis rather than seeking statistically significant results. This deliberate approach was undertaken not to derive significant outcomes but rather to verify the effective functionality of the system.

While the primary objective of the application has been achieved, there remain avenues for improvement in future studies, as delineated in Section 6.6. A key limitation to acknowledge is the exclusive focus on 360-degree images and their respective impairments. Enhancements could be made by extending

the application to encompass 3D environments with associated assets, potentially incorporating movement options to facilitate a more comprehensive exploration of the scenes. Additionally, the current system is tailored for static background images, where the sole task assigned to the subjects is observation. A more nuanced approach involving subjects in specific challenges could provide a richer understanding of how image impairments influence the quality of experience.

The application incorporates an automated subjective scoring mechanism, enabling participants to assess image impairments on a scale ranging from one to five. This user-friendly scoring system is facilitated through the handheld controller. The outcomes of this assessment are detailed in Section 7.4.1, where statistical differences are analyzed. Across all impairments, a discernible decline in the overall experience is observed when compared to reference conditions. Interestingly, minimal distinctions are noted when comparing individual settings within a specific impairment.

In addition to facilitating subjective tests, the system incorporated functionality for capturing eye movement data. These data not only served the purpose of simulating blind spots but were also recorded for future research endeavors. For instance, in Section 7.3.3, we presented an analysis of an individual's eye and gaze movements. Although no statistical measurements were conducted, this feature holds the potential for a more in-depth understanding of individuals with impaired vision and insights into their adaptive strategies.

Similarly, Section 5.4 presents an experiment on color vision deficiency, analyzed in Section 7.4.1, involving the testing of only 8 CVDs. However, by utilizing the CVD View (6.4.4), there is potential for testing the Quality of Experience on individual shifts in LMS spectral sensitivity functions. Through minor adjustments or by introducing a new View, the opportunity arises to further refine color adjustments, thereby offering a broader range of color-distorted samples. This aspect could be enhanced in future studies.

Overall this thesis establishes the fundamental framework for subjectively assessing the quality of experience in simulated image impairments. The system is designed with a high degree of customization, allowing for tailored adjustments to suit specific purposes. Furthermore, it demonstrates its capabilities through a pilot experiment, accompanied by a concise analysis of the gathered data. This analysis not only illuminates insights into the quality of experience but also unveils additional measurable aspects of human behavior beyond the primary focus on experience quality.

Bibliography

[1] Coblis — color blindness simulator (jan 2024): https://www.color-blindness.com/Coblis-color-blindness-simulator/ , their github page (jan 2024): https://gist.github.com/Lokno/df7c3bfdc9ad32558bb7.

[2] Hardware recommendations for virtual reality: https://www.pugetsystems.com/solutions/more-workstations/virtual-reality/hardware-recommendations/.

[3] Understanding scotoma: Symptoms, causes, different types: https://www.visioncenter.org/conditions/scotoma/.

[4] S. Agrawal, A. Simon, S. Bech, K. Bærentsen, and S. Forchhammer. Defining immersion: Literature review and implications for research on immersive audiovisual experiences. 10 2019.

[5] R. Albert, A. Patney, D. Luebke, and J. Kim. Latency requirements for foveated rendering in virtual reality. *ACM Trans. Appl. Percept.*, 14(4), sep 2017.

[6] C. Anthes, R. J. García-Hernández, M. Wiedemann, and D. Kranzlmüller. State of the art of virtual reality technology. In *2016 IEEE Aerospace Conference*, pages 1–19, 2016.

[7] D. C. Beidel, B. C. Frueh, S. M. Neer, C. A. Bowers, B. Trachik, T. W. Uhde, and A. Grubaugh. Trauma management therapy with virtual-reality augmented exposure therapy for combat-related ptsd: A randomized controlled trial. *Journal of Anxiety Disorders*, 61:64–74, 2019. Virtual reality applications for the anxiety disorders.

[8] Y. Boas. Overview of virtual reality technologies. In *Interactive Multimedia Conference*, volume 2013, 2013.

[9] K. Brunnström, S. Beker, K. D. Moor, A. Dooms, S. Egger, M.-N. Garcia, T. Hossfeld, S. Jumisko-Pyykkö, C. Keimel, M.-C. Larabi, B. Lawlor, P. L. Callet, S. Möller, F. Pereira, M. Pereira, A. Perkis, J. Pibernik,

A. M. G. Pinheiro, A. Raake, P. Reichl, U. Reiter, R. Schatz, P. Schelkens, L. Skorin-Kapov, D. Strohmeier, C. Timmerer, M. Varela, I. Wechsung, J. You, and A. Zgank. Qualinet white paper on definitions of quality of experience. 2013.

[10] K. Brunnström, E. Dima, T. Qureshi, M. Johanson, M. Andersson, and M. Sjöström. Latency impact on quality of experience in a virtual reality simulator for remote control of machines. *Signal Processing: Image Communication*, 89:116005, 2020.

[11] V. Clay, P. König, and S. König. Eye tracking in virtual reality. *J Eye Mov Res*, 12(1), Apr. 2019.

[12] G. Craddock et al. Emulating perceptual experience of color vision deficiency with virtual reality. In *Transforming our World Through Design, Diversity and Education: Proceedings of Universal Design and Higher Education in Transformation Congress*, volume 256, pages 378–389, 2018.

[13] S. Davis, K. Nesbitt, and E. Nalivaiko. A systematic review of cybersickness. In *Proceedings of the 2014 Conference on Interactive Entertainment*, IE2014, page 1–9, New York, NY, USA, 2014. Association for Computing Machinery.

[14] K. Fliegel, J. Kolmašová, and J. Švihlík. Performance comparison of perceived image color difference measures. In A. G. Tescher, editor, *Applications of Digital Image Processing XLI*, volume 10752, page 107522U. International Society for Optics and Photonics, SPIE, 2018.

[15] G. Gonçalves, P. Monteiro, M. Melo, J. Vasconcelos-Raposo, and M. Bessa. A comparative study between wired and wireless virtual reality setups. *IEEE Access*, 8:29249–29258, 2020.

[16] J. Gutierrez, P. Perez, M. Orduna, A. Singla, C. Cortes, P. Mazumdar, I. Viola, K. Brunnström, F. Battisti, N. Cieplińska, et al. Subjective evaluation of visual quality and simulator sickness of short 360 videos: Itu-t rec. p. 919. *IEEE transactions on multimedia*, 24:3087–3100, 2021.

[17] A. Hamad and B. Jia. How virtual reality technology has changed our lives: An overview of the current and potential applications and limitations. *International Journal of Environmental Research and Public Health*, 19(18), 2022.

[18] M. Huang, Q. Shen, Z. Ma, A. C. Bovik, P. Gupta, R. Zhou, and X. Cao. Modeling the perceptual quality of immersive images rendered on head mounted displays: Resolution and compression. *IEEE Transactions on Image Processing*, 27(12):6039–6050, 2018.

[19] X. Huang, K. Xie, S. Leng, T. Yuan, and M. Ma. Improving quality of experience in multimedia internet of things leveraging machine learning on big data. *Future Generation Computer Systems*, 86:1413–1423, 2018.

[20] Y. Huang, K. Yao, J. Li, D. Li, H. Jia, Y. Liu, C. K. Yiu, W. Park, and X. Yu. Recent advances in multi-mode haptic feedback technologies towards wearable interfaces. *Materials Today Physics*, 22:100602, 2022.

[21] M. Javaid and A. Haleem. Virtual reality applications toward medical field. *Clinical Epidemiology and Global Health*, 8(2):600–605, 2020.

[22] J. A. Jones, E. Luckett, T. Key, and N. Newsome. Latency measurement in head-mounted virtual environments. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 1000–1001. IEEE, 2019.

[23] P. Juluri, V. Tamarapalli, and D. Medhi. Measurement of quality of experience of video-on-demand services: A survey. *IEEE Communications Surveys Tutorials*, 18(1):401–418, 2016.

[24] K. Krösl. *Simulating vision impairments in virtual and augmented reality*. PhD thesis, Wien, 2020.

[25] R. Lavoie, K. Main, C. King, and D. King. Virtual experience, real consequences: the potential negative emotional consequences of virtual reality gameplay. *Virtual Reality*, 25:69–81, 2021.

[26] J. Lewis, D. Brown, W. Cranton, and R. Mason. Simulating visual impairments using the unreal engine 3 game engine. In *2011 IEEE 1st International Conference on Serious Games and Applications for Health (SeGAH)*, pages 1–8, 2011.

[27] G. M. Machado, M. M. Oliveira, and L. A. F. Fernandes. A physiologically-based model for simulation of color vision deficiency. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1291–1298, 2009.

[28] Z. Makhataeva and H. A. Varol. Augmented reality for robotics: A review. *Robotics*, 9(2), 2020.

[29] S. Mann, T. Furness, Y. Yuan, J. Iorio, and Z. Wang. All reality: Virtual, augmented, mixed (x), mediated (x, y), and multimediated reality. *ArXiv*, abs/1804.08386, 2018.

[30] R. Q. Mao, L. Lan, J. Kay, R. Lohre, O. R. Ayeni, D. P. Goel, and D. de SA. Immersive virtual reality for surgical training: A systematic review. *Journal of Surgical Research*, 268:40–58, 2021.

[31] D. C. Niehorster, L. Li, and M. Lappe. The accuracy and precision of position and orientation tracking in the htc vive virtual reality system for scientific research. *i-Perception*, 8(3):2041669517708205, 2017.

[32] Y. Niu, H. f. Zhang, W. Guo, and R. Ji. Image quality assessment for color correction based on color contrast similarity and color value difference. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(4):849–862, 2018.

[33] I.-T. R. P.10. Vocabulary for performance and quality of service, amendment 2: New definitions for inclusion in recommendation itu-t p.10/g.100. 2008.

[34] A. Pinsker. Image quality assessment in immersive image display systems. 2023.

[35] K. Raaen and I. Kjellmo. Measuring latency in virtual reality systems. In K. Chorianopoulos, M. Divitini, J. Baalsrud Hauge, L. Jaccheri, and R. Malaka, editors, *Entertainment Computing - ICEC 2015*, pages 457–462, Cham, 2015. Springer International Publishing.

[36] L. Rebenitsch and C. Owen. Estimating cybersickness from virtual reality applications. *Virtual Reality*, 25(1):165–174, 2021.

[37] S. Rokhsaritalemi, A. Sadeghi-Niaraki, and S.-M. Choi. A review on mixed reality: Current trends, challenges and prospects. *Applied Sciences*, 10(2), 2020.

[38] P. Sachse, U. Beermann, M. Martini, T. Maran, M. Domeier, and M. R. Furtner. "the world is upside down" – the innsbruck goggle experiments of theodor erismann (1883–1961) and ivo kohler (1915–1985). *Cortex*, 92:222–232, 2017.

[39] M. Seufert. Fundamental advantages of considering quality of experience distributions over mean opinion scores. In *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–6, 2019.

[40] M. Simka, L. Polak, J. Kufa, M. Novotny, A. Zizien, and K. Fliegel. Omnidirectional image quality assessment database (omniqad): Description and examples. In *2023 33rd International Conference Radioelektronika (RADIOELEKTRONIKA)*, pages 1–5, 2023.

[41] F. Škola, S. Tinková, and F. Liarokapis. Progressive training for motor imagery brain-computer interfaces using gamification and virtual reality embodiment. *Frontiers in human neuroscience*, 13:329, 2019.

[42] J.-P. Stauffert, F. Niebling, and M. Latoschik. Latency and cybersickness: Impact, causes, and measures. a review. *Frontiers in Virtual Reality*, 1, 11 2020.

[43] N. Stewart Rosenfield, K. Lamkin, J. Re, K. Day, L. Boyd, and E. Linstead. A virtual reality system for practicing conversation skills for children with autism. *Multimodal Technologies and Interaction*, 3(2), 2019.

[44] V. C. Tashjian, S. Mosadeghi, A. R. Howard, M. Lopez, T. Dupuy, M. Reid, B. Martinez, S. Ahmed, F. Dailey, K. Robbins, B. Rosen, G. Fuller, I. Danovitch, W. IsHak, and B. Spiegel. Virtual reality for

management of pain in hospitalized patients: Results of a controlled trial. *JMIR Ment Health*, 4(1):e9, Mar 2017.

[45] G. Tsaramirsis, S. M. Buhari, K. O. Al-Shammari, S. Ghazi, M. S. Nazmudeen, and K. Tsaramirsis. Towards simulation of the classroom learning experience: Virtual reality approach. In *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 1343–1346, 2016.

[46] M. Varela, L. Skorin-Kapov, and T. Ebrahimi. *Quality of Service vs. Quality of Experience*, pages 85–96. 03 2014.

[47] S. Vlahovic, M. Suznjevic, and L. Skorin-Kapov. A survey of challenges and methods for quality of experience assessment of interactive vr applications. *Journal on Multimodal User Interfaces*, 16(3):257–291, 2022.

[48] D. V. Walsh and L. Liu. Adaptation to a simulated central scotoma during visual search training. *Vision Research*, 96:75–86, 2014.

[49] M. Warburton, M. Mon-Williams, F. Mushtaq, and J. R. Morehead. Measuring motion-to-photon latency for sensorimotor experiments with virtual reality systems. *Behavior Research Methods*, pages 1–21, 2022.

[50] I. Wechsung, K.-P. Engelbrecht, C. Kühnel, S. Möller, and B. Weiss. Measuring the quality of service and quality of experience of multimodal human–machine interaction. *Journal on Multimodal User Interfaces*, 6, 07 2012.

[51] S. Weech, S. Kenny, and M. Barnett-Cowan. Presence and cybersickness in virtual reality are negatively related: A review. *Frontiers in Psychology*, 10, 2019.

[52] D. Willis, W. Powell, V. Powell, and B. Stevens. Visual stimulus disrupts the spatial localization of a tactile sensation in virtual reality. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 484–491, 2019.

[53] M. Xu, C. Li, S. Zhang, and P. Le Callet. State-of-the-art in 360 video/image processing: Perception, assessment and compression. *IEEE Journal of Selected Topics in Signal Processing*, 14(1):5–26, 2020.

[54] A. Zheleva, W. Durnez, K. Bombeke, G. Van Wallendael, and L. De Marez. Seeing is believing: The effect of video quality on quality of experience in virtual reality. In *2020 Twelfth International Conference on Quality of Multimedia Experience (QoMEX)*, pages 1–4, 2020.

[55] D. J. Zielinski, H. M. Rao, M. A. Sommer, and R. Kopper. Exploring the effects of image persistence in low frame rate virtual environments. In *2015 IEEE Virtual Reality (VR)*, pages 19–26, 2015.

Appendix A

List of attachments

The electronic attachments include the following:

- Result/ - folder containing data created by the system
- Survey/ - folder containing simulator sickness and general info questionnaire
- Matlab/
  - CreateScotoma.m - function for creating scotoma image
  - CreateSPnoise.m - function for creating transparent noise image
  - GetGazeMovementDataset.m - processes data from specified subdirectories and organizes data into maps for All, Eye, Head, and Time data per subject.
  - GetHeatmap.m - function processes eye movement coordinates, maps them onto a 360-degree image, and generates a normalized heatmap, incorporating a circular mask
  - AnalyseHeatmap.m - averages heatmaps (created by GetHeatmap.m) and displays it over the background
  - Ciedetest.mlx - computes and compares the CIEDE2000 color differences between reference and tested images for different types of CVDs
  - EyeMovement.mlx - reads eye movement data and plots them with corresponding timestamps
  - GazeMovement.mlx - reads gaze movement data and plots them with corresponding timestamps with background image
  - SpeedAnalysis.mlx - computes rotational speeds of head and eye movements over time
  - CreateScoreMap.mlx - reads and processes data from a subdirectory and organizes score data into a map
  - GetDataset.m - returns the score dataset of a given factor in the form of a table

- DataAnalysis.mlx - does statistical tests of subjective scores
- CreateHistrograms.mlx - creates histograms comparing reference scores and impairments

Appendix B

Download Instructions

B.1 Introduction

This appendix provides detailed instructions on how to download the Artefacts Simulation project. This project is a crucial component of the diploma thesis titled "Simulation and Evaluation of Image Impairment Impact on Quality of Experience in Virtual Reality Environments." The purpose of this guide is to facilitate the acquisition of the project resources necessary for further exploration.

B.2 Download

To initiate the project, you need to download or clone several essential pieces of software from public domains:

- **Unity Hub:**
  `https://unity.com/download`
- **SteamVR:**
  `https://store.steampowered.com/app/250820/SteamVR/`
- **Artefacts_Simulation:**
  `https://gitlab.fel.cvut.cz/spanajak/artefacts_simulation`

To clone the repository, open Git in the desired folder and execute the following command:

```
git clone https://gitlab.fel.cvut.cz/spanajak/artefacts_simulation.git
```

B.3 Setup

After downloading Unity and the project, launch Unity Hub. If necessary, complete the registration process (registration is free).

1. Click on the **Open** button in Unity Hub.
2. Navigate to your project folder, where you should find two folders: `.git` and `ArtefactsSimulation`.
3. Select the `ArtefactsSimulation` folder and click open.
4. Unity Hub will automatically download the correct version of the project and configure the necessary settings. The project will be opened.
5. Locate the `Assets/Scenes` folder in the project window.
6. Open the `SampleScene` by double-clicking.

The project requires two additional packages from Unity Package Manager: the `SteamVR Plugin` and `XR Plugin Management`. These packages should automatically clone alongside the project.

Within the scene, all essential objects are arranged. Steam VR will handle the camera tracking. The key scripts vital to the functionality of the project are situated within the MainManager and its associated children.

It's essential to acknowledge that this project was specifically crafted to utilize the eye-tracking capabilities of the HTC Vive Pro Eye. Consequently, alternative headsets will not support this feature, leading to an impact on the GazeFollowView functionality.

The project can be accessed in *Play Mode* (by pressing the play button) without the VR headset or SteamVR (an error will appear). While this version can still demonstrate the core system, its capabilities will be significantly diminished.