MASTER'S THESIS 2020

Eye-tracked gaze gesture implementation in an augmented reality system

Anders Lundqvist Persson

DEPARTMENT OF DESIGN SCIENCES
LTH | LUND UNIVERSITY

EXAMENSARBETE
Designvetenskap

Eye-tracked gaze gesture implementation in an augmented reality system

Anders Lundqvist Persson

Eye-tracked gaze gesture implementation in an augmented reality system

Anders Lundqvist Persson
anders.lundqvist.persson@gmail.com

June 17, 2020

Master's thesis work carried out at the Department of Design Sciences, Lund University.

Supervisors: Günter Alce, gunter.alce@design.lth.se
Diederick C Niehorster, diederick_c.niehorster@humlab.lu.se

Examiner: Johanna Persson, johanna.persson@design.lth.se

Abstract

Gesturing has long been a way of communicating, and is a large part of human nature. It was only natural that this was brought into our technological development when technology advanced. This thesis aims to expand the known human 'library' of gesturing into that of gestural patterns invoked by visual guidance and eye tracking. With the use of a HoloLens to visualize gestural patterns, in combination with an eye tracker from Tobii to invoke and interact with objects, we try to create a modular gesture pattern system. Testing implicates that the developed application without eye tracking is intuitive, easy to figure out and responsive. There are, however, several improvements that can be made in terms of ergonomic and long term use.

Eye tracking implementations to the gaze gesture system is looked into and discussed throughout the thesis, and with further development could be made into a fully viable, modular and intuitive tool for integration of AR with the physical world.

Keywords: Augmented Reality, Eye tracking, Tobii, HoloLens, Gaze Gesture

Acknowledgements

This thesis could not have been completed with the help of my former thesis-partner, Andreas Englesson. As we were working very similar projects in the end, his help and comments were very helpful and helped me push through when it seemed otherwise insurmountable.

I would also like to thank my supervisors Günter Alce and Diederick C. Niehorster for their assistance and understanding with the process and my situation, without which I probably wouldn't have finished this work.

I would also like to thank my good friends Linus Hammarlund, Per Ahlbom and Joakim Hembrink, without them I would have gotten stuck on programming issues far longer than I would like to admit.

And finally I would like to thank anyone who helped me during this process and its completion. This includes, but is not limited to; The staff at IKDC, the staff at Tobii who helped answer my technical issues, my girlfriend, and my family and friends who helped me push through to the end.

Contents

Chapter 1
Introduction

In the first chapter a background to the problem is given, as well as a proper problem formulation with goals and limitations.

1.1 Motivation

The idea of controlling a computer, untethered, and only using hand gestures has been a concept explored by sci-fi for a long time. Ranging from the movie *The Minority Report*, where a detective is controlling a set of monitors using gloves, to newer movies like in the *Marvel Cinematic Universe*, where Tony Stark (Iron Man) controls holograms using both audible and bodily gestures. Augmented Reality, i.e. the ability to overlay the real world with virtual objects and environments, and holograms can be seen in early episodes of Star Trek: The Next Generation, Star Wars, and the likes. All of these systems, except partially for Tony Stark's, have one thing in common, bodily gestures for control. That common thread birthed the idea for this thesis, i.e. controlling holograms and/or technologies **without** these bodily gestures.

Systems such as these offer alternative solutions to interacting with a computer, instead of the classical keyboard and mouse setup. The term for this, and all other information technology, interaction is called human-computer interaction, or HCI, a term popularized by Card, Newell and Moran in 1983 [1]. The systems are created to help the user by offering a simplified or alternative method for interaction. Gesture interactive systems had their issues in the early stages of gesture development, but are now ingrained in the culture and almost natural according to some researchers [2]. There is still some debate whether gestures are 'natural', however, or if they can ever be natural, as they inherently limit some of the trained and known behaviours of humans [3].

You might want a system for gestures without the need for hands, for instance in a sterilized environment, or a dirty one, where you also cannot use the hand gestures associated with a command. Imagine a surgical environment, contact and help for the

elderly during a pandemic, or with functionally impaired patients, such that they are physically incapable of gestures, and sometimes even speech. Techniques used to perform simpler interactions using eye-tracking are available today, and are mostly used in eye-to-speech synthesizers. More technology is being interconnected through the Internet of things, or IoT, i.e. a technical objects ability to communicate by itself, or gather and relay information [4]. And more objects are communicating than ever, a number that is steadily increasing each year [1]. The method of control for these technologies differ. And therefore it is interesting to explore whether or not this no-contact gesture technique, in combination with Augmented Reality, can interact with them.

1.2 Related Work

Using eye gaze systems to control, or interact, with your environment is a large field of study in medical technology. For instance in Shi et. al. [5] where an eye gaze system was implemented to control Information and Communication Technology, or ICT, objects in the users home. Another example is the DOGEye system [6], developed by Bonino et. al., an eye tracking based home control application designed with the *Communication by Gaze Interaction*, or COGAIN, guidelines [7] in mind.

Kullberg and Lindkvist [8] wrote a bachelor's thesis with regard to gaze guidance in augmented reality systems, using the same set-up of a Hololens with Tobii Pro Glasses as will be used in this thesis.

In a broader sense, work with smart home environments has been consistently growing for the last couple of years. Controlling these with augmented reality has been proposed by Marques et. al. [9]. Other suggestions for controlling a smart home include motion matching and smart watches [10] or ZigBee [11] for users with special needs.

Addditionally, Majaranta et. al. [12] researched a gesture-based gaze interface by hinting gaze patterns to a user, that in turn prompted a command. Moreover, Köpsel et. al. researched gaze versus gesture interaction combined with different feedback modalities [13].

1.3 Thesis Vision

Head-worn devices (HWD) for augmented reality have been used for some time, by and large for research purposes. But it is slowly making its way into the everyday user's home and offices. By implementing binocular eye-tracking technology to a HWD one could control an entire system using only the user's eyes. Imagine a modern day living room. The TV is most likely smart, there is probably some type of smart device able to play music, there might be an Alexa standing in the corner. If we consider a really top of the line modern day home, even the lights are smart. With implemented eye-tracking technology in your HWD, it has now become a multi-remote for every single smart device in your vicinity, and perhaps even distant ones. What's to say you cannot control your computer in the other room by virtual means using a different set of gestures? With HWD, eye-tracking and IoT, the only limiting factor of your control is the hardware.

[1] https://www.statista.com/statistics/802706/world-wlan-connected-device/

1.4 Problem Statement

This section will state the goals, problems and proposed solution, as well as the project scope

1.4.1 Goals

The goal of this master thesis is to construct and implement communication between a Microsoft Hololens and a pair of Tobii Pro Glasses 2 for eye tracked gaze gesture interaction. The system should

- Be intuitive to the user
- Respond with as low latency as possible to avoid user irritation
- Avoid issues such as the Midas touch problem [14].

These goals can be summarized into the following goal statement:

> The system should be intuitive and easy to start up and be usable by a diverse group of users. The solution should be ergonomic, both for short, and long term, use. The design should be responsive and well designed with regards to user experience and system design.

1.4.2 Proposed Solution

This thesis proposes a novel solution for targeting and interacting with objects through eye tracking techniques in AR by connecting the AR-capabilities of a Microsoft Hololens with the eye tracking of a pair of Tobii Pro Glasses 2.
By live-streaming the eye trackers gaze coordinates to the Hololens network, the goal is to create a gaze-guided cursor able to interact with objects through gaze guided gesturing. When a user puts on the Hololens, they will be greeted by a quick calibration, then the AR environment will load any virtual objects into view. If the user then passes over the objects it will indicate to the user that it is interactable, to try and avoid, or at the very least minimize, the Midas Touch problem[2] they will not start interaction without prompting.
Interaction will be based on a visual progress-meter of some kind in the users view. By holding their attention to the progress, a user will start interacting with the object, finally giving them a list of options. Interaction with the options will have a shorter initiation sequence, as the user has already showed intent to use the object there is less need to wait and make sure they want interaction.

[2] See section 2.2

1.4.3 Scope of the project and its Limitations

Implementing a solution such as the proposed one involves a high amount of work, and is unfortunately outside the scope of this master's thesis. Things that fall outside the scope are as follows:

- Interaction with any object that is inherently interactable would be favorable, and a large goal of this work. This thesis is, however, more of a proof-of-concept of interactability with eye-tracking and how it should be obtained. And as such, all objects that are interacted with are manually created and not 'read' by the AR-system. Creating a sort of virtual Wizard of Oz environment for interaction where all activities are predetermined.
- Other gesture interaction such as zooming, moving or removing objects are all part of the 'gesture scope' that is known from cell-phones, VR and AR. It would require its own background research for just this part as far as known gestures and interaction should be used for advanced features, and as such has been left out.

1.5 Report Outline

The report is divided into the following chapters:

1. **Introduction.** A background to the problem is given, as well as a proper problem formulation with goals and limitations.
2. **Theoretical Background.** The common terminology and technology are introduced.
3. **Coding Pipeline.** The design of the coding pipeline and system is described. Arguments are brought forwards as to why choices regarding coding and methodology are what they are.
4. **UX design.** The choices regarding UX design and HCI are defended and presented.
5. **Collected system implementation.** The system design, theory and techniques from previous chapters are put together into a complete system.
6. **Trials and Results.** The system is evaluated through user testing, both by 'actual users' and admin-testing. The following results are presented.
7. **Analysis.** The results analysis and evaluations are presented. The analysis is mapped to the previous chapters to simplify reading.
8. **Conclusions and future work.** The work from this thesis is summarized. The main results, evaluations and analysis are presented. Thoughts on future work is discussed.

Chapter 2
Theoretical Background

This chapter serves to introduce common terminology and technologies that are the basis for this thesis.

2.1 Augmented Reality

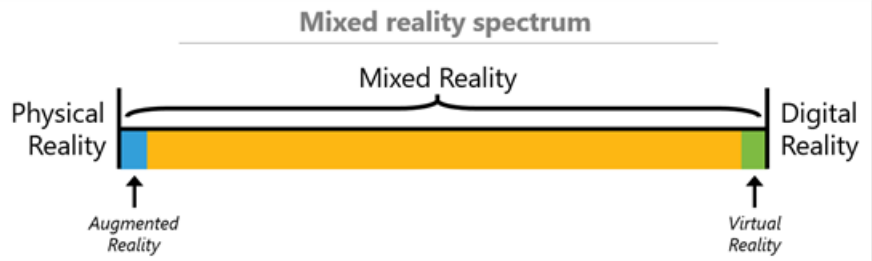

Figure 2.1: The mixed reality spectrum with AR and VR marked at either end. *Source: Microsoft*

Augmented Reality, commonly abbreviated as AR, is a part of the Mixed reality spectrum, as seen in figure 2.1. It combines the physical world that humans live in, with the ability to add digital objects through holographic lenses or cameras. There are differing definitions of AR going around, with one being from Azuma et. al. [15] stating that to be an AR system it must have the following properties:

- It must combine real and virtual objects in a real environment;
- It must run interactively and in real time; and
- It must register (align) real and virtual objects with each other. [15]

If one chooses to follow this definition it would mean that a system would have to geographically (for larger, outdoor systems) and geometrically align created digital objects with the real world, in real time. Any device capable of creating AR can do so by enhancing the natural world, be it with sound, video or GPS data. One could argue that, by a broader definition, listening to music on your phone is AR, as it augments the sounds you hear from your environment. The generated objects by a system can then be manipulated in real time by a user, making it very interesting for applying new technologies. This reality manipulation can be seen in the popular mobile app Snapchat [1], and even in the Swedish post office app, PostNord [2], that lets you view the size of incoming packages in AR if you have a device with ARKit-support [3].

2.1.1 Microsoft HoloLens

Overview

Figure 2.2: A user with a head-worn display attached to his head.

Source: Microsoft

The Microsoft Hololens is a head-worn display that's also untethered, as seen in figure 2.2. It has a semi-transparent lens, allowing the user to see both the physical world and digitally created objects and environments. The Hololens allows for input using speech, gestures, and "gaze-tracking" by use of a head-direction vector, and output using visual and audio-based objects. The Hololens field of view has been estimated at a field of view of $30^o \times 17.5^o$[16], giving a diagonal of 34.73^o using the Pythagorean theorem and it is 16x9 resolution. A mixture of visual studios and Unity is used to place digital objects inside the Hololens viewport. The full documentation regarding mixed reality in Microsoft and the Hololens can be reviewed at their own documentation website [17].

Raycasting

Raycasting is a physics based method inside the Hololens API. With it one can imagine a point firing a laser in a direction, $\vec{D}$, from any point of origin. If that ray were to connect with either the Hololens' mesh overview of the world, or a synthetically created object (a hologram), it would register a ray-collision at a point, P, with coordinates $\begin{bmatrix} x_p & y_p & z_p \end{bmatrix}$. The Hololens will automatically raycast straight forward from its origin when viewing its surroundings, registering any ray it sends out and adapting the response accordingly. If it collides with an object or a 3D-world mesh it will adjust the cursor to indicate interactability or simply that there is something there. Any collision detected by raycasting will yield a collision point, P_n, with which one can calculate a direction, $\vec{D}_n$, with the cast-point origin.

[1] Snapchat - https://www.snapchat.com/

[2] App Store - https://apps.apple.com/se/app/id396871673

[3] As of the writing of this thesis, it is still not developed for the ARCore of Android.

2.2 Eye-Tracking

Eye-tracking in its simplest form is the tracking of a subjects eye movements to extrapolate where that person is looking, the time they spend looking in that direction and how their eyes travel over objects. There are a couple of companies today that offer different kinds of products and solutions for this type of tracking, each company using its own special features to try and make their own product more precis than competitors. Metrics that are often used in the products are, to name a few, gaze direction, time to first fixation and number of fixations, blink rate and duration of blinking, and the diameter of the pupil.
Eye-tracking has immense possibilities of use, and there are already plenty of areas where it is being used to great effect, such as helping someone with a handicap control their computer using only their eyes, or spelling out words on an eye-gaze controlled speech pad, for instance with the EyeMobile Plus[4]. Bonino et. al. lists several distinguishing features of eye tracking [6]:

- it is *faster* than other input media, as observed by Ware and Mikaelian [18]; when a user is about to operate any mechanical pointing device, she usually looks at the intended destination;
- it is *easy* to operate, as apart from extraordinary circumstances, no particular coordination or training is required to observe objects;
- it show where the *focus of attention* of the user is located; without actually intending to, a users gaze can be interpreted as the current focus of their attention;
- it suffers from *Midas Touch* problem: expecting to be able to observe objects without having to interact with them. By using different interaction techniques, such as dwell time or blink select, this problem can be mostly overcome;
- it is *always on*; there are currently no, in present society, natural ways of indicating that one wants to interact with an input device, as with a mouse;
- it is *noninvasive*; by only using eye tracker input, a user can relax their arms and hands, causing less physical fatigue;
- it is *less accurate* than other pointing devices, such as a mouse.

[4] EyeMobile Plus - https://www.tobiidynavox.com/en-GB/devices/eye-gaze-devices/eye-mobile-plus/

2.2.1 Tobii Pro Glasses 2

The Tobii Pro Glasses 2, or 'G2', are an eye-tracker designed as to look as a pair of glasses, as opposed to the commonly used variants that are placed on a monitor or surface to observe the user. The head-worn unit, as seen at the front of figure 2.3 with the recording unit at the back, is equipped with infra-red cameras that record the eyes. The recording unit holds most of the hardware and software used for processing the image data taken by the glasses.

Figure 2.3: The Tobii Pro Glasses 2 *Source: Tobii AB*

Eye-tracking techniques

Pupil Centre Corneal Reflection (PCCR)

Pupil Centre Corneal Reflection is the most common method among eye-tracking devices. By illuminating the eye with a light source, a camera can take detailed pictures of the eye. Image processing algorithms are then used to locate the center of the corneal reflection, and the center of the pupil, illustrated in figure 2.4. Tobii uses near-infrared illumination to create the reflections in their glasses.

With the help of advanced image-processing algorithms, Tobii is able to combine these reflections with a 3D model of the eye to estimate the eyes position in space as well as the point of gaze with high accuracy [19].

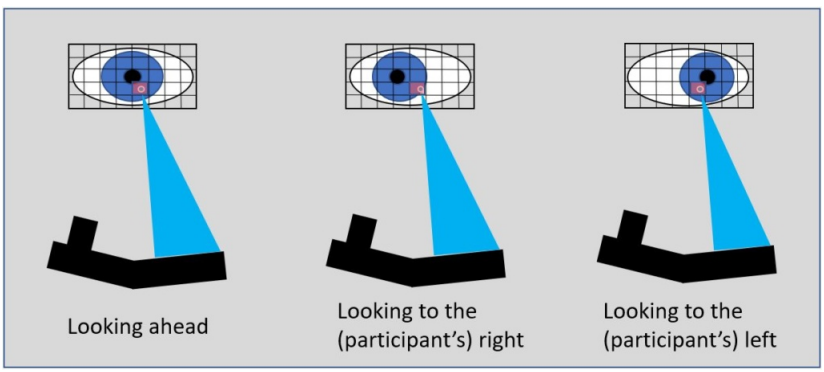

Figure 2.4: Pupil Centre Corneal Reflection *Source: Tobii AB*

Dark/Light Pupil Tracking
There are two different ways of illuminating the pupil when using the PCCR technique in eye tracking. In both methods an illuminator is placed near the eye to cause a reflection from it. The difference between the two methods is the placement of this illuminator. When using the bright pupil tracking, the illuminator is placed as close to the optic axis as possible to make the pupil light up, this is also what causes red-eye in a camera using flash. While using the dark pupil tracking, the illuminator is instead placed away from the optic axis, making the pupil appear darkened instead, as seen in figure 2.5.

Which technique to use depends on many factors such as age, environmental light, and ethnicity. Many of Tobii's eye trackers have both illumination systems built into them, and during calibration the system decides which of them produce the most accurate result. The Tobii pro glasses 2, however, only utilizes dark pupil tracking [20].

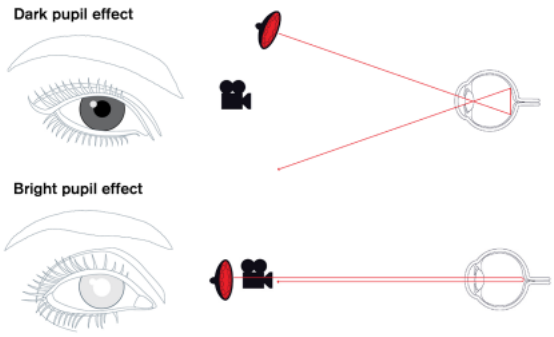

Figure 2.5: Illuminator placement for dark and bright pupil tracking respectively *Source: Tobii AB*

2.3 Human-Computer Interaction

Human-Computer Interaction, is a multidisciplinary field of study within computer technology design, and even more so the interaction between humans, or users, and the computer systems. As technology developed from the massive computers filling up entire buildings to fitting in the palm of a hand, HCI has adapted and evolved as well. As stated by John M. Carroll, one of the founding fathers of HCI [21]:.

> [...] *it no longer makes sense to regard HCI as a specialty of computer science; HCI has grown to be broader, larger and much more diverse than computer science itself. HCI expanded from its initial focus on individual and generic user behaviour to include social and organizational computing, accessibility for the elderly, the cognitively and physically impaired, and for all people, and for the widest possible spectrum of human experiences and activities.*

As humans interact with computers, so can computers interact with humans. Examples of this are when a mobile device tells the user that there are restaurants nearby that would fit

the users previous tastes, this is called *implicit* interaction - i.e. humans interact with non-deliberate input and receive an output from the system. If, on the other hand, the user inputs the question "Are there any restaurants for me nearby?" to the system and receives an output, that's *explicit* interaction.

More than a quarter-century ago, Mark Weiser claimed that technology demands active attention, and that we will never truly experience the potential of technology until it moves into the peripheral [22]. That statement, and the amount of attention required to perform HCI has been looked into since then, for example by Saskia Bakker who researched the use of peripheral attention in technology devices [23].

2.3.1 Gesturing patterns

Gesture patterns are, in its simplest form, a way of interacting with something else without physically touching it. For instance clicking a mouse on a computer screen. The *gesture* is clicking, and the well known gesture pattern for interacting with something on a computer is the double-click. As technology evolved and computer-mice became part of an *interaction library* instead of the sole method of interaction, other gestures and methods arose [2]. Your mobile phone will read gestures such as pinching or scrolling by swiping, the Nintendo Wii uses a pair of white controller-sticks that continuously send their coordinates to the system to form gestures and input. The Hololens uses a set of scene-cameras that pick up gesturing of the hands, and indicates when the hand is or is not readable. One could argue that speech input is a subset of gesture patterns as it is still read by a computer and analyzed to figure out what the instruction actually is.

2.3.2 Visual Ergonomics

> *Visual ergonomics is the multidisciplinary science concerned with understanding human visual processes and the interactions between humans and other elements of a system. Visual ergonomics applies theories, knowledge and methods to the design and assessment of systems, optimizing human well-being and overall system performance. Relevant topics include, among others: the visual environment, such as lighting; visually demanding work and other tasks; visual function and performance; visual comfort and safety; optical corrections and other assistive tools. [24]*

By applying visual ergonomics in ones work, the physical and mental load of the eyes and mind should be put to a minimum degree. It can also reduce physical stress in other parts of the body by not promoting awkward positions or posture when working with an object [25]. When trying to develop a solution for gestures using eye-tracking, the ergonomics of the eye is important to consider at all steps. This includes aspects such as brightness of objects, either virtual or virtually enhanced, the placement of gestures and their overall size etc.

2.3.3 Bottom-up/Top-down Processing & Visual Cues

Bottom-up and top-down processing are terms in attention theory that have to do with humans ability to detect new stimuli. It is said that bottom-up works on raw input, for instance movement in a bush. For this, one doesn't need to actively shift ones focus to the bush to find the source of movement, the body does this itself. Top-down mechanisms use the cognitive systems and strategies that humans have built up over time, basically shifting our direction of focus. A scared individual in the woods, for example, might be even more perceptive to movement, as it could be that of a predator [26]. Because of these inherent abilities of the human attentiveness, it is important to consider when designing a system for interaction. One has to consider if, and how, an object should announce itself as interactable, or how 'flashy' a menu or item should be, lest it steals attention from something else of importance.

Visual cues are ways of actively interacting with a user to shift their attention. They can be explicit, subtle or not there at all, where the range for what classifies as explicit or subtle might vary between users. These cues can be used to tell a user where there are interesting points in a system, as with bottom-up/top-down processing, one can actively shift attention to something by implementing a more explicit cue [27]. One can easily see how this combined with the previously mentioned attention theory introduces methods of interacting with a user from a design perspective.

2.4 Rubber duck debugging

One very efficient way of debugging code without actually using an application-debugging feature is to explain the code, line-by-line, to someone else. This will usually expose any very obvious logical problems or misconceptions of the code. When programming alone, this can be achieved by using so-called 'rubber duck debugging' which replaces the human part with that of an actual rubber duck to which you explain your code. [28] In this thesis work, a pair of sunglasses on a coffee cup has been used instead of said rubber duck. An example of such a 'rubber duck' is shown in figure 2.6, although the cup of choice varied during the course of this thesis.

Figure 2.6: A variant of the 'rubber duck' used for verbal debugging in this thesis work.

Chapter 3
Coding Pipeline

In this chapter, the design of the coding pipeline and system is described. Arguments are brought forwards as to why choices regarding coding and methodology are what they are.

Baseline System
The work in this thesis is based around Kullberg and Lindkvist [8]. In their thesis they managed to connect the Tobii Pro 2 with a Hololens and send gaze data from the glasses to the hololens. This was achieved by connecting the Hololens to the glasses via Wi-Fi, allowing for *http* communication between the two. They examined the conjunction between reaction time and subtle cues, giving them the need of a computer as a server to store and analyse the data. This was achieved by connecting the Hololens to the computers network, and send the test results through *TCP*. The Tobii glasses were connected to the network through a *UDP* protocol. The different protocols are visualised in figure 3.1.

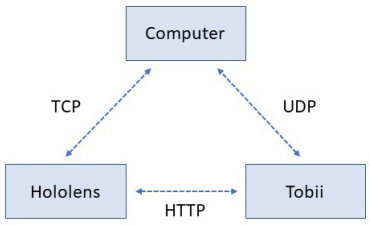

Figure 3.1: The communication protocols *Source: Kullberg & Lindkvist*

3.1 .NET and JSON, C# and Unity

While Unity has its own libraries regarding networking and UDP communication, .NET and JSON has been used instead in this thesis work, as the libraries from the latter are not only stronger and less buggy, but also seem more flexible when it comes to specific cases or

outliers, shown by stackoverflow[1] answers, and general frustration on the Unity developer forums. Firstly, Unity's libraries seemingly will not allow for communication outside of its own creation (see arguments for this in section7.1), i.e. our built environment within the program itself, unless we create several workarounds. The single workaround that is a 'catch all' for this, is to do all communication to the 'outside' without allowing Unity to read it, it is simply handled by the compiler and used inside the application, but is not inherently checked by Unity if it is allowed or not. This is done using the #if preprocessor directive in C# [29], allowing the use of code that will only be read when Unity is not active, as exemplified in this snippet:

```
#if !UNITY_EDITOR
    [Code regarding the Windows socketing
    and networking needed to communicate with
    the Hololens and Tobii G2]
#endif
```

Secondly, and perhaps most importantly, the Tobii G2 API makes use of JSON requests for any communication with the recording device, and as such the library Json.NET by Newtonsoft[2] has been used heavily in this project.

3.2 Calibration

The Tobii G2 can, and if not instructed otherwise **will**, use a baseline eye-model for any and all recordings. This setting can be used for any simple run-of-the-mill testing, such as in Kullberg and Lindkvists thesis [8] where it was only valid to know if a user looked to the right or the left. For any testing with need of accuracy, however, the system needs to be calibrated and the Tobii needs to create a unique eye-model for each user. This is done using HTTP-post requests in the JSON format based on the Tobii developer guide [30], prompting a calibration. The custom-made G2 unit in this project does not have access to the scene-camera that is usually used by the recording unit to calibrate the system by registering a calibration marker held up in front of the camera. To circumvent this, one sends a synthetic marker to the recording unit, along with a set of coordinates, $\begin{bmatrix} X_{mark} & Y_{mark} & Z_{mark} \end{bmatrix}$, in relation to the G2's origin, in millimetres. More on this in section 5.2.

When the system has been calibrated there are two eye-coordinates being sent from the G2 to the system at all times to be read, interpreted and translated. To minimize calculations done in the system when running, only one of these coordinates is handled at any one time by the proposed application. Measuring vergence, i.e. the micro-movements performed by the eyes to focus on objects, is likely not a reliable measurement using binocular pupil-based eye trackers [31]. Therefore it was also deemed unnecessary to perform calculations for both eyes to find the convergence-point when a user focuses on an object. This further reduces the load on the system at any one time, and allows for more complex calculations to take place with lower risk of lag or computational delay.

[1] https://stackoverflow.com/
[2] https://www.newtonsoft.com/json

The right eye has been chosen as the first selection for recording. Since roughly 60-70% of the population is 'right-eyed' [32], it makes sense to set it up this way for general usage and testing. As the selection is controlled by a simple if statement it is also easy enough to change, and could be controlled by a simple question inside the application itself. In this thesis, the assumption is made that users won't know which 'eyedness' they have, and as such the decision to initiate the application focused on the right eye was made.

Chapter 4
User Experience Design

The fourth chapter describes the choices made for gaze interaction, and the UX design as a whole. Arguments are made to defend why certain choices have been made and why other methods or choices have not been selected.

4.1 UX overview

The user experience and interface takes inspiration from a large number of sources. For instance the COGAIN guidelines for gaze based environmental control [7], Majaranta et al. with their work in avoiding the Midas Touch problem via gaze gesturing [12], as well as Delamare et al. [33], Huckauf and Urbina's work in gaze based object selection [34], and Microsoft's own documentation regarding mixed reality [17]. With this in mind a UX system has been developed with three main components, listed as such:

1. **Cursor selection.** The cursors being or non-being in a gaze guided system. It is used to trigger interaction and acts as a visual guide for the user.
2. **Gaze interaction.** The user needs to interact with smart objects in her surroundings while avoiding Midas Touch issues.
3. **User interface.** When interaction has been initiated with an object, the user needs to have clear cut information regarding what can and cannot be done using that object.

*Note: User Interface has been listed as a sub-category of User Experience, as the visuals are counted more as a **part** of the experience, rather than the entire experience itself.*

4.1.1 Cursor selection

When designing a cursor for eye-gaze based interaction, it is very important to consider the visuals of said cursor. While a user may be comfortable using a standardised mouse cursor, the

standard cursor has a direction - being an arrow. Using something with an inherent direction in it might cause what Microsoft declares a 'fleeing cursor' [17, pp. 252] in their documentation regarding eye-gaze interaction. This means that if your cursor has an inherent direction your mind might focus on always looking in that direction, causing the cursor to 'flee' before your eyes. If the input delay for your eye-gaze is ever so slightly offset, or the gaze coordinates in space is off from where you are actually looking (see section 5.2), your eyes might start chasing the cursor, as it consistently grabs your attention and can be interpreted as a visual cue (see section 2.3.3). One step to avoid this is to use a subtle indicator of vision, such as lighting up objects that are being watched, or using a 'non-shape' cursor, such as a small circle or dot with a low alpha-channel. A system based on selecting and interacting with a gesture pattern, such as this, requires that the user can tell when an object is observed, and if the user is not currently focusing correctly they should get an indication of where they are looking. The standardised cursor in a Hololens is a simple dot, and when the Hololens detects an object it can interact with (or a hand, when using the standard gesturing system), it will change into the shape of a torus to indicate that something observed is interactable. The cursor will align itself with the normal of the mesh it is currently colliding with, giving a sense of being able to actually interact with the world around you. To minimise the time for a new user to get initialised into the proposed solution, the same cursor has been selected, albeit with a slightly lower alpha channel. If possible it would be best to have the cursor as a point of light, transforming into a torus of light upon reaching interactable objects as this would further reduce distractions without any visible edges on the cursor. Another benefit of using light instead of objects is to reduce color separation, an artifact created by the nature of the Hololens displays, upon moving the cursor. This will occur on a regular Hololens cursor when a user moves their gaze from one side to the other and is visualised as in figure 4.1.

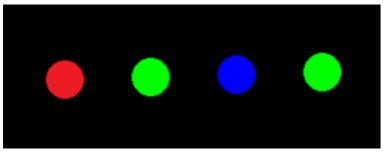

Figure 4.1: Example of color separation *Source: [17, pp. 1638]*

However, due to limitations of the Unity software, the Hololens, and Microsoft Hololens API, light can only be implemented in simulated environments, or in post-processing. As lighting needs immense amounts of calculations to behave correctly.

4.1.2 Gaze interaction

Interacting with the world around us is most commonly done using physical interaction, i.e. touching something with your hand. With gaze interaction it is not as simple, and the major contributing factor for this is the "Midas Touch problem" [14]. Midas Touch in eye tracking is very simply an analogy for gaze interaction with anything you observe, just like King Midas would turn everything to gold. One realises quickly that interaction with everything is not a wanted effect, just as Midas learned that turning everything he turned to gold was, in fact, not a good idea. To circumvent this, researchers have tried a multitude of options. For instance facial muscle movements, as proposed by O. Tuisku et al. [35], that is

both strenuous for a user and requires extra equipment, or simply using voluntary blinking. Blinking, however, as a selection method has the downside of being strenuous for selection for any length of time. Blink detection also has its own built in Midas touch problem revolving around involuntary blinking, something that must be solved with algorithms to determine when a blink is voluntary or not [36], and thus out of this thesis' scope. One of the more commonly used methods is 'dwell time', where the interaction begins after a user has a dwell time, T_d on an object that passes some threshold time T_{thresh}. Dwell time is not only easily implemented, it is also a clear way of showing intent when selecting and has a low learning curve if implemented correctly [12]. When implementing dwell time it is important to take into consideration that a prolonged gaze where the dwell time, T_d, is too long lends to the user having more time to get distracted, or blinking. This would theoretically cancel the dwell, and as such it is important that one designs the system around this if an initial T_d is set for a long time ($> 800ms$ [37]). As such, the dwell time for initial selection in this thesis is set to $500ms$, with an allowance of $100ms$ for any involuntary saccades or blinks that would otherwise cancel the gaze. Simply put:

$$\textbf{if } T_d >= T_{thresh} \longrightarrow \textit{interact with object}$$

When a user has initialised interaction, the gaze pattern nodes will react instantly as long as their conditions are met (see section 5.3) and the node will sequentially fill up a loading-indication, as can be seen in the upper right corner of figure 6.4. If the user subsequently shifts focus from the interacted object, or any of it is interaction-linked children, it will be deselected after a time $T_{ds} = 1000ms$.

4.1.3 User interface

The UI has been designed with ease of access, intuitive use, and usability in mind. Plenty of techniques and theory has been inspired by the 'designer bible', *The Design of Everyday Things* [38]. Meaning that instructions for a user should be heavily implied, or outright spelled out. The mapping of the gesture pattern is designed to, not very subtly, point the user in the right direction and have them realize where to look next by implementing arrows to the next position.

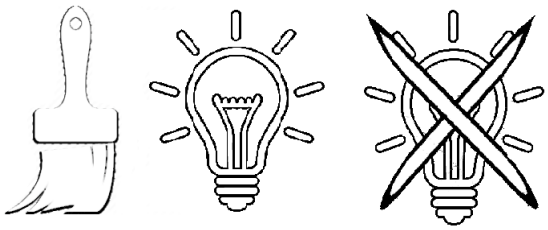

Figure 4.2: Open source ClipArt-icons used in the User Interface, right-most being a modified version of the middle icon.

Different symbols might mean different things in different countries. An easy examples being the 'thumbs up'-gesture or symbol, which is a sign of approval in most western countries, but in western Africa and the middle east it could mean 'up yours' depending on

the country. Or when President H.W. Bush accidentally gave Australians the finger.[1]
Therefore, to design for the general public it is important to consider the symbol usage of ones interface. In an attempt to circumvent any misinterpretations of the symbols used in this UI it is been designed using ClipArt. And within the ClipArt catalogue (that holds more than 13 million open source images), images that are commonly used for the same purpose were selected. For color-selection, a brush, and for the 'light' a lightbulb. The ClipArt selected, as well as a modified one to implicate 'off' can be seen in figure 4.2.

[1]https://www.chicagotribune.com/news/ct-xpm-1992-01-26-9201080471-story.html

Chapter 5
Collected System Implementation

The implemented system is presented in order of 'appearance' in the pipeline. Finalizing in a compiled solution ready for use.

5.1 UDP Communication and Response

The framework used for UDP Communication and Response was a repository on GitHub by user M.A. Baytaş, [1] who developed it based on his work at Qualisys AB, Koç University, as well as Chalmers University of Technology. The scripts set up by mbaytas has been modified to call and respond to the Tobii G2 unit using a modified version of a read-script, called *Models*, by researcher Diederick C. Niehorster at the Humanities Lab at Lund University. This read-script has been written with the intent of grouping response-messages from the G2 unit in their respective classes. It was modified to be able to extract further data that was needed from the G2 in this project.

UDP communication was then setup by translating Tobii's Python-based examples into efficient C# so as to minimize lag and input-delay. Monitoring delay and lag was done with a HoloLens diagnostics tool that kept track of frames per second, network speeds and much more. Selecting certain parameters for lag-reduction and reduce chances of interruptions finalized in a five-second timer for the 'keepAlive'-message that is sent to the G2 to keep it recording and responding. Should the G2 not receive these keepAlives for three of its predetermined intervals, controlled by system-parameters, it stops recording. Setting the keepAlive-rate so high allows for a large amount of errors while still maintaining the connection. As the systems interval time is $\gg 5s$ this allows for several of the keepAlives to fail.

[1] *https://github.com/mbaytas*

5.2 Coordinate system and calibration

As the system is based on direct communication between the Hololens and Tobii G2, we need to be able to read the coordinates live from both systems, and translate them to one system. The Hololens has a world-based coordinate system where its location at startup is the origin, regardless of movement, and the G2 has a fixed origin-position between the eyes that moves with the user as seen in figure 5.1. Because of this it makes sense to translate the G2's coordinates to the Hololens ones rather than the other way around. If we were to translate the Hololens coordinates into the G2's, we wouldn't have any actual location or visual direction in world space without relating it all back to the Hololens origin. Simply placing the G2 inside the Hololens space makes more sense because of this, as we can find the G2's location from the Hololens API using the heads current position and adding the eye offset given by the G2.

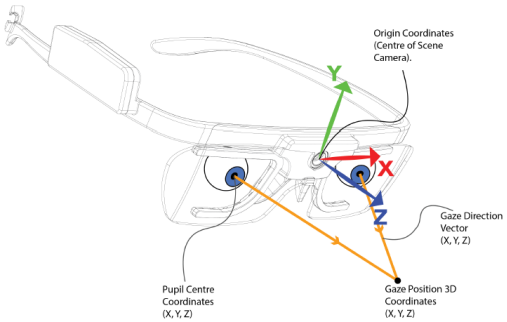

Figure 5.1: The Tobii G2s coordinate system
Source: Tobii Pro Glasses 2 API Developer's Guide v.1.3 - en-US

There are some inherent issues with this translation. Firstly, the G2 glasses are not "aware" of the rotation of the Hololens. They do store a rotation of their own, but we want to minimise the requests sent to the G2 since we want our continuously streamed eye data to be non-interrupted or delayed because of other requests. However, when deploying a Unity project to the Hololens we are working from the basis of the 'main camera', and using the API for this we can simply extract the Hololens' rotation and apply it to the G2 coordinates before translating.
There is also the issue of lacking documentation regarding the Hololens actual origin. We know that its origin is placed at $\begin{bmatrix}0 & 0 & 0\end{bmatrix}$ upon startup of any deployed code, what we don't know is where this origin is with regards to the Hololens' scene camera. As we recall, the G2 has an origin that moves with the user, we can simply translate this origin to always be at the heads current position. The issue with the above statement is that if the Hololens' **actual** origin position is slightly behind, or in front of, its scene camera our offset for where the eyes are positioned will be inherently wrong. Depending on the scale of this location discrepancy the error margin changes. For any object close to us we should experience minor errors, but the further away the object the larger the error, visualized in figure 5.2. Because of this, an assumption is made that the origins of the G2 and the Hololens line up

as figuring out the actual origin position experimentally is outside the scope of this thesis.

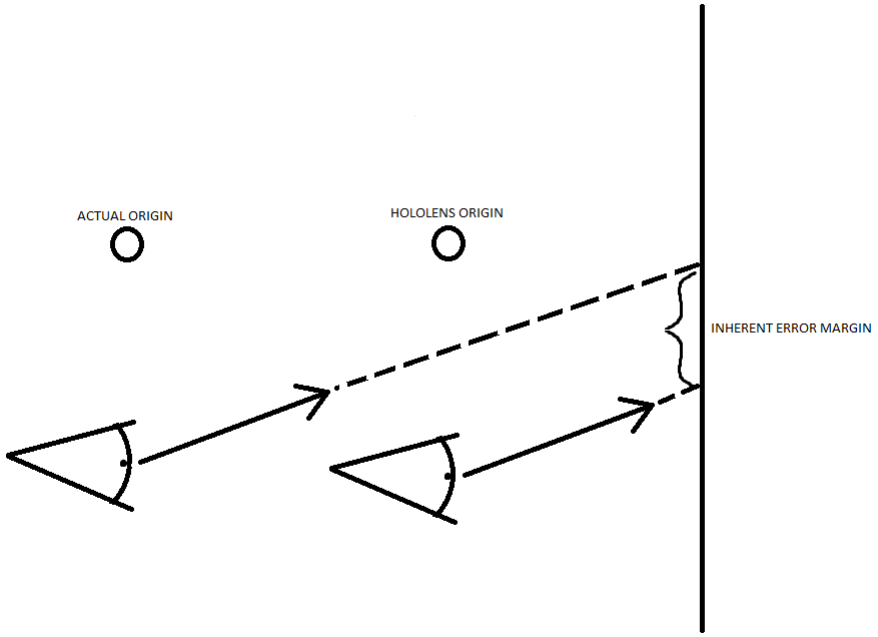

Figure 5.2: Inherent error because of possible shift in Hololens origin

Our current eye position and look-direction (directional vector) is thereby calculated by pivoting the eyes position around the Hololens origin by α degrees, where α is the current rotation of the Hololens stated as a quaternion[2]. This is converted to the Hololens' units of meters and translated onto the current position coordinates of the Hololens, effectively placing our eye in world space. The direction vector is similarly rotated according to α and then geometrically displaced so that it is origin is at the eyes current origin coordinates, O_{eye}.
From this we can raycast, i.e. send an invisible ray from O_{eye} in a direction $\vec{D}_{gaze}$, using Hololens API and query the Hololens for any 'physical' collisions that would occur if this was an actual ray of light, either with a synthetic holographic object or our generated 3D mesh of the surroundings. This, effectively, gives us the real-world coordinates of what we're looking at with our eyes with respect to the Hololens origin.
This coordinate translation should directly enable calibration by providing us with a origin point, and allowing us to place a synthetic marker as mentioned in section 3.2 using these newly translated coordinates. However, for calibration we are not interested in any translated coordinates, we only need to know the distance, Z_{synth}, from our Hololens to the synthetic marker in the G2's coordinate system. To do this, a calibration marker hologram is added in Unity at 1 unit from the main camera and locked in position. By locking it, the marker will always remain at exactly 1 units distance from the camera, and we can put that value in our calibration coordinates mentioned in section 3.2, effectively telling the G2 that the marker is at

$$\begin{bmatrix} 0 & 0 & Z_{synth} \end{bmatrix},$$

where Z_{synth} is the distance from the Hololens' origin, O_{Holo}, to the raycast-collision P_{synth}

[2]https://en.wikipedia.org/wiki/Quaternion#Three-dimensional_and_four-dimensional_rotation_groups

with the marker, or $|O_{Holo} - P_{synth}|$. As far as the G2 is concerned, we have placed the marker with zero vertical or horizontal displacement and it can focus the eye-model around the distance to the marker. When the calibration has completed the translated coordinates become relevant again and we start translating every output of the G2, and using these calibrated coordinates we can now apply a cursor that follows the users gaze properly, and without any relevant input delay as discussed in section 4.1.1.

5.3 Gaze gesture system

The gaze gesturing system is based on a double linked list, this double linked logic means that any node, N, in a specific gesture has read access of nodes N_{-1} and N_{+1}, but has no means of directly reading nodes beyond that reach. The logic of each node is algorithmically overviewed in algorithm 1. This gesture system is based on the gesturing system developed by Majaranta et. al. [12] and further modified into a modular gesture system to allow for extension into any possible combination of patterns.

Algorithm 1 Double linked gaze-element logic

1: N_{prev} ← read access of *previousNode*
2: N ← *currentNode*
3: N_{next} ← read access of *nextNode*
4: **if** N_{prev} exists **then** ***FirstNode*** ← *False*
5: **if** N_{next} exists **then** ***LastNode*** ← *False*
6: *At Each Frame Update*:
7: *condition 1:*
8: **if** *GazedAt*(N) **&** (N_{prev} (IsFull & Exists) **or** *LastNode* = *True*) **then**
9: N ← fillNode.
10: **when** *N* IsFull **then**
11: **set** *N.IsFull* = true
12: *condition 2:*
13: **if not** *GazedAt*(N) **&** N_{next} (IsEmpty & Exists) **or** *FirstNode* = *True*) **then**
14: N ← drainNode.
15: **when** *N* IsEmpty **then**
16: **set** *N.IsEmpty* = true

By use of this logic the nodes will not be loadable in a gesture unless the previous node has been loaded, essentially meaning you cannot start a gesture in the middle. While this might be an annoyance for the user after repeated use of the same gesture, it was deemed better than the alternative of 'Midas touching' gestures where you do not want to interact, as explained in section 4.1.2.

Chapter 6

Trials and Results

The system was evaluated through user testing, both by test-users and admin. The following results are presented.

(Note: Sadly, because of hardware failure, the testing was forced into focusing on user interface, user experience and general feel using only the HoloLens. Because the eye-tracker was being repaired during the time of testing, it is implementation is strictly theoretical. Any and all eye-tracked data is done by admin-testing as no further time for user tests was available.)

6.1 Participants

The tests were performed on a total of 13 people in the age range 23 to 61. 11 of which were 26 or younger, one participant was 41 years old and one was 61 years old. Eight of the participants were men, and five were women. There was no discernible difference between VR or AR experience from younger to older participants. The users subjective view of their background in VR and AR is shown in figures 6.1 and 6.2 respectively. Tests were not timed, but an average time from start to finish was extrapolated from the timestamp of questionnaire entries as 11 minutes and 40 seconds per user test. All participants were either students or employees at the Engineering Faculty at Lund University.

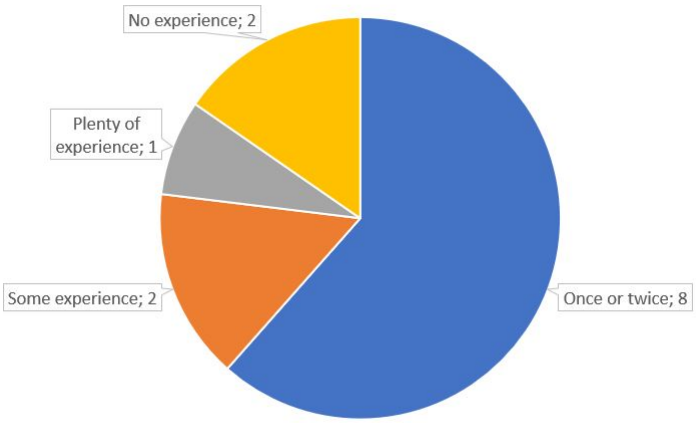

Figure 6.1: Previous VR experience of the users in testing.

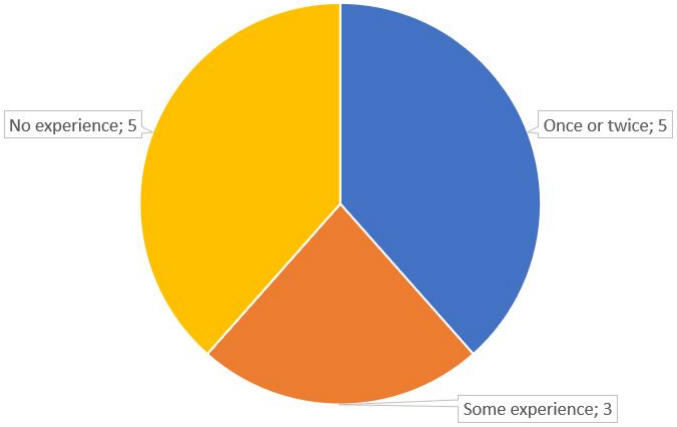

Figure 6.2: Previous AR experience of the users in testing.

A random selection of participants were asked if they would have changed their answers regarding AR experience, knowing that the definition of AR includes things such as Snapchat. Of the five users asked, all would have changed their answer to 'some experience' or 'plenty of experience'.

6.2 Setup

Tests were performed in an empty part of the Virtual Reality lab at Ingvar Kamrads Design Centre, or IKDC, at the Faculty of Engineering at Lund University. The Microsoft HoloLens was used for AR visualization and testing. In an attempt to optimize the HoloLenses spatial detection ability all the lights were on, and the curtains raised to create an ambient light environment.

Without a Lux meter this was estimated as per Microsoft Documentation[1]

> Lighting should be even and sufficiently bright that a human can see without effort, but not so bright that the light is painful to look at.

In figure 6.4 the light in the background is the achieved effect of this.
The participant was seated on a stationary chair, this was to avoid rotation during application start up in an attempt to minimize the initial positions of the holograms. The distances from user to hologram, and the initial gaze point when the application started up can be seen figure 6.3. The distance of the cyan line is set to 1.1m. Task 1, that is shown as a white circle in figure 6.3 is located 50cm to the left of initial gaze, 15cm further away, and 25cm upwards. Task 2, that is shown as a black circle, is located 50cm to the right, 10cm closer to the user, and 25cm down. The locations of these holograms was selected as they were just within range upon startup, but not obvious.

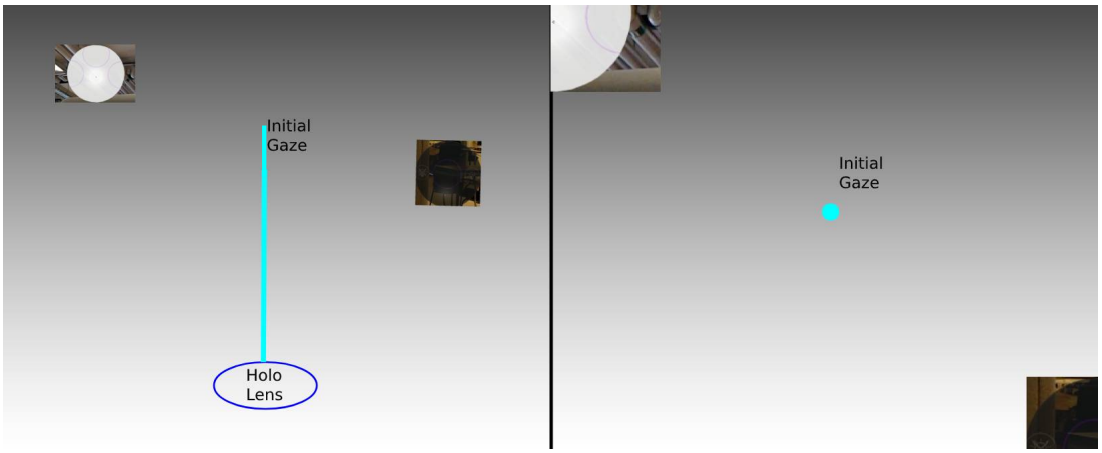

Figure 6.3: Participant setup on application start. Left: The distance between holograms and user. Right: The initial view of a user upon application startup.

6.3 Procedure

Users participating in the tests were asked to sign a consent-form so that their data could be used. If signed, the user was instructed that for the test they were to be using a HoloLens, and that after being fitted with the unit they would have to locate a hologram in the room and paint it first yellow, then blue, and lastly white - in that order. They were instructed that the object was interactable by gaze, but no further instructions as to how interaction was enabled. The user was then to direct their gaze onto the hologram and enable interaction, followed by completing the task at hand. Upon completion of the first task, they were to locate the second hologram, enable interaction and turn the 'light' on and off. Tests were not timed as it was not deemed important for the results, nor a test of how fast a task could be solved.

[1] https://docs.microsoft.com/en-us/hololens/hololens-environment-considerations

Upon startup, a participant was greeted with two visible holograms. To initiate interaction with one of these, they would gaze upon one of the holograms to make a menu pop up (figure 6.4, top left). Further gazing upon a menu item would make it start loading, and tell the user that it was 'focusing' (fig. 6.4, top right). If the user followed the indication arrows for next gaze point, they would finally be able to select from a pre-set menu for that specific task..

The tests were setup with two different methods of gaze interaction, one with a gesture pattern for a more complex multiple choice action, and one with simple activation of an object, as can be seen in figures 6.4 and 6.5 respectively. In the images taken with the Mixed Reality Portal (left part of figure 6.4 and the entirety of figure 6.5), provided by Microsoft for the HoloLens, the 'focusing', and 'losing focus' parts of the pattern could not be captured because of a delay between capturing and image taken and because of this they are shown from the Unity interface.

Lastly, users were asked to fill out a non-standardized questionnaire developed with the help of a UX designer at Mentimeter[2] in an attempt to minimize bias in the questions asked. Participants were not interviewed or further questioned apart from the questionnaire. The entire procedure from introduction to finish can be seen in figure 6.6

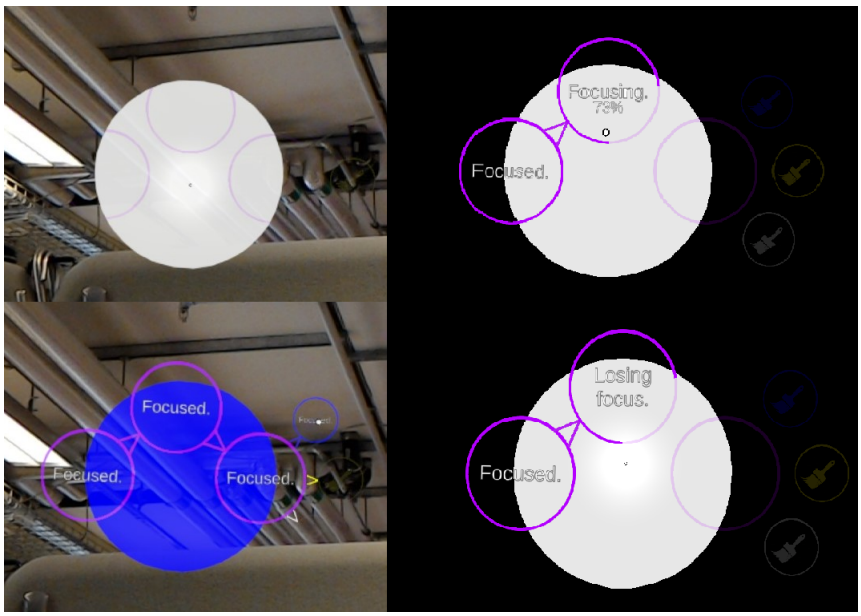

Figure 6.4: Gesture pattern setup. UL: Interaction initialised, UR: Focusing on pattern, LL: Fully focused and colored, LR: Losing focus on the pattern.

[2]https://www.mentimeter.com/

Figure 6.5: Simple activation setup. UL: Interaction initialised, UR: Focused, LL: 'Light' on, LR: 'Light' off.

Figure 6.6: The procedure visualized

6.4 Results

6.4.1 System implementation

Implementation of the G2 data into the HoloLens application was completed successfully. By turning off non-essential Mixed Reality functions, supplied by Microsoft when optimizing a project for Mixed Reality, the load on the HoloLens was lowered drastically. Calculations performed on the inbound data from the G2 affected the application fairly little, by only applying calculations to one eye at a time, fps-rate was increased by an average of 10. Whereas turning off non-essential functions of the Mixed Reality package increased the fps-rate by 20-30. In the end, the application varied between 45 and 60 fps when running. It would have lower fps when updating the environment, and increasing time between update intervals for this update increased the average frame rate while also minimizing noticeable delay for the user. If one didn't know what to look for, the update appeared to not be noticed. It was not asked for in the questionnaire and therefore no conclusion of it can be stated. Worth noting is that the latency issues that could arise are more linked to eye-tracking than head-tracking.

6.4.2 Questionnaire

Most users felt that the difference in selection for task-completion, i.e. a loading selection for the coloring tasks or an instant one for the light, was more or less the same. The ones who did have a preference seems to be fairly evenly divided between load-selection or instant selection, as can be seen in figure 6.7.

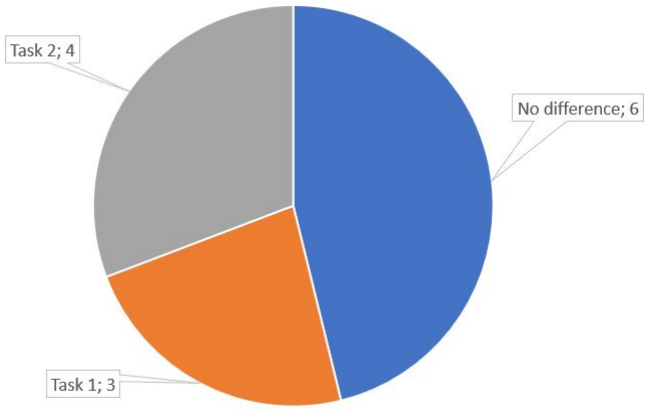

Figure 6.7: Responses regarding if Load-selection (task 1) or Instant-selection (task 2) was the most desired method of selection.

When asked if the users knew what solving a task would do, 9 answered yes. Two users had the opinion that they only knew what solving one of the tasks would do, split evenly

between the two tasks. And two users didn't feel that they understood what solving the tasks would do. The results are shown in figure 6.8.

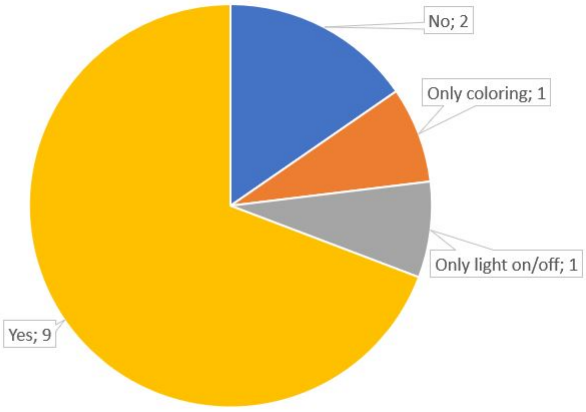

Figure 6.8: Answers to the question "Did you feel that you knew what solving the different tasks would do?"

Users were asked if the visuals of the user interface were clear enough on a scale of 1-10, where 1 was 'Not visible at all' and 10 was 'Very visible'. 12 out of 13 users replied 7 or above, evenly spread from 7 to 9. One user marking visibility at a full 10. Another user reported a general lack of visibility and contrast in the interface as a free answer, and also rated visibility at a 3.

The users were also first given a short and very simplified explanation of the Midas Touch problem, as follows:

> *"The Midas Touch problem" is when a visible object starts initial interaction as soon as we gaze upon it. Did you feel that this system had, or didn't have a midas touch problem?*

They were consequently asked if they perceived the system to suffer from such an issue. The results can be seen in figure 6.9. (*Note: Most users asked for further explanation of the Midas Touch problem and what it would mean for a general system.)*

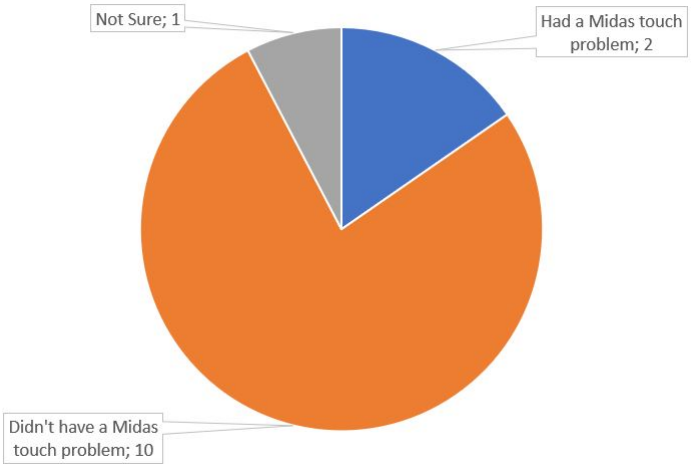

Figure 6.9: Responses regarding The Midas Touch problem.

Responsiveness of the system was rated on a scale of 1-10, where 1 was 'Not responsive at all' and 10 was 'Very responsive'. The ratings given were 7, 9 and 10 with one, two, and ten respondents respectively.

The users were given a chance to comment on the User Interface with anything that springs to mind, as with the visibility comment mentioned earlier. One user would have wanted to know where the holograms were placed in the world with some sort of indication, one user felt that they didn't know when task 1 was 'complete', and one user stated that the only reason they knew what to do with the interface was because they were told what to do.

Following the User Interface questions came a section in the questionnaire regarding ergonomics. The users first rated the ergonomic feel of the headset for the duration of time that they had it on. The rating was on a scale from 1-10, with 1 being 'Not ergonomic at all' and 10 being 'Very ergonomic'. The setup got a fairly high score in terms of ergonomic usage with a 6.7 average, with a lower average score when asked if it would be comfortable for extended use, 4.5 and no users rating a 10 for extended use. The scale was from 1 - 'Very uncomfortable' to 10 - 'Very comfortable'.

The user was then asked about strain in parts of the body during use with a multiple selection question, as well as a text field for optional answers. The selections were: Neck, Back, Forehead, Eyes, Shoulders, Neither, and Other. Six users reported no strain, with one noting that the neck would hurt after repeated or longer use. One noted strain of the eyes, two people responded with forehead as well as neck. Three people wrote the optional answer 'Nose'. One user reported that a broader view-port (inherently limited by the HoloLens) could possibly limit strain over time.

Lastly, the users had to consider whether or not an implemented eye-tracker in the system would make it harder or easier to navigate the UI. Four users responded that it would be easier, one of which noted that it would be faster but more error prone. Seven users thought it would make the system harder to user. One user responded that it would not make a difference, and one user noted that it would be dependent on how it was set up.

6.4.3 Eye-tracking calibration and data

The calibration of the G2 was completed for this project and tested against a virtual wall located 90° from origin, and early calculations rendered the results tabulated in figure 6.1. This calibration was performed using two faulty calculations. One being that the eyes origin-position for raycasting (section 2.1.1) was not properly calculated. It was later learned that its position was never properly rotated along the HoloLens origin-point, and as such would always get slightly skewed results. The second faulty calculation can be seen in the x-axis, and is a result of the first calculation. The G2 'hit' location was calculated using the same method as the HoloLens calculates its 'hit' locations, i.e.:

$$HitPosition = rayOrigin + lastHitDistance * rayDirection$$

The distance between the points has then simply been calculated as:

$$d = \sqrt{(x_H - x_T)^2 + (y_H - y_T)^2 + (z_H - z_T)^2}$$

Where subscript T and H stand for 'Tobii G2' and 'HoloLens' respectively.

Table 6.1: Early calibration results.

130 degrees from origin	**X**	**Y**	**Z**
HoloLens 'hit' location avg.	1,1	-0,228	-1,289
G2 calibrated 'hit' location avg.	1,379	-0,103	-1,53
Difference per axis	-0,279	-0,125	0,241
Distance between 'hit' locations	0,389		
90 degrees from Origin	**X**	**Y**	**Z**
HoloLens 'hit' location avg.	1,1	0,134	0,186
G2 calibrated 'hit' location avg.	1,14	0,213	0,163
Difference per axis [m]	-0,04	-0,079	0,023
Distance between 'hit' locations	0,0915		
40 degrees from Origin	**X**	**Y**	**Z**
HoloLens 'hit' location avg.	1,1	0,161	1,55
G2 calibrated 'hit' location avg.	1,186	-0,103	1,65
Difference per axis	-0,086	0,264	-0,1
Distance between 'hit' locations	0,144		

The calibration of the G2 performed better than expected, even with faulty calculations. Further testing after correcting for the faults provided far better results, they are however not saved and can therefore not be analysed.

Observing the data tabulated in figure 6.1, we can see that the error in the calculated hit positions increases with distance. It is however, because of the faulty calculations, hard to say if this is caused by the error in origin in ray casting for the G2, or because of the G2s difficulty to properly assess long-distance eye tracking.

Chapter 7
Discussion

Results are discussed and analyzed.

7.1 Process

The process for this thesis has been a bit jumbled. As it began as a duo-project, with a wider scope. At about halfway through the thesis work was split into two separate works, with a reduced scope. Nearing the end of it all, vital parts of the hardware broke, further reducing the scope and goals of the thesis. To counter the project setbacks a lot of the steps within the system design has been done in an iterative matter, repeating steps as necessary as new issues arose.
Main problems that arose during this thesis were:

- **Losing a thesis partner** - Solved by reducing thesis scope. Eventually 'replaced' by a Rubber duck (see section 2.4).
- **Sudden hardware failure** - Solved by reducing thesis scope further.
- **Lack of technical support structure** - Working in an entirely new programming language and software demanded lots of help on short notice, lest the project would grind to a halt. My supervisors showed a large amount of support and helped find relevant people to assist. However, my conclusion from this is that this type of work would be best done in collaboration with a company focused on the subject matter.
- **The current pandemic Covid-19** - It might seem trivial to mention, but having a pandemic roaming the country whilst finishing a thesis was not simple. As my work was based around visiting a "public" place, more thought had to be put in to how users could be found. How safe my work area was, more sick-days caused by symptoms that would otherwise be ignored, etc. All of which impeded the work.

Finding the correct way to code for the application was sometimes harder than expected. As mentioned in section 3.1 there seems to be issues with connecting the built in Unity libraries to anything 'outside' of the environment. The shear amount of work-around and bug reports regarding this was enough to dissuade from trying to use these built-in solutions. This could have been because of the version of Unity used, as Unity is very sensitive for version changes. This was noted by my supervisors that the previous work of Kullberg and Lindkvist, around which this thesis was loosely based, had encountered this as well. I myself encountered this version-sensitivity when I accidentally opened up the project in a newer version of Unity, effectively removing dependencies and files it deemed 'old' and then not starting up, as these were missing. As for the outliers and fringe cases mentioned in the coding pipeline, several threads calling for fixes on the Unity developer forums were stated by Unity's staff as fringe, simply because they didn't exceed a threshold number of reports. Working with communication outside of Unity was seen as marked 'unimportant' in several of these help threads when looking for solutions.

7.2 UX Design

It is hard to draw any conclusion regarding if the Midas Touch issue has been solved or not in this system. Since no user in the testing seemed to understand the question and its meaning as it stood it is hard to know if they properly understood the meaning even after further explanation. It is also fully possibly that the further explanation skewed their opinion in any one way, as it was explained by someone whose intent it was to remove the issue. To properly be able to answer the question of the Midas Touch problem being solved or not, one should probably host a small introduction to the issue before testing, and then do the tests. This should minimize researcher-bias in the users questionnaire response if they understand what they are answering before hand.

Based on the questionnaire the UX design was positively regarded by most users, and managed to be both intuitive and respond well to the users intent, as per the goal statement. When looking at the images taken to show the setups in figures 6.4 and 6.5, it would seem that the interface is not very visible in its environment. Based on responses from users, this was not the case apart from one. This could be explained by the mixed reality environment not capturing the full sense of the system when taking the pictures, be it for compression or any other reason. The low visibility and contrast of the images taken in Unity is almost solely because of compression when reducing the resolution of the images.

A relative majority of users had no preference of either selection-type (figure 6.7). For the users that had a preference for any type, the number of respondents is too low to state which was more or less preferred. Data was however ever so slightly skewed towards 'click-like' selection over 'loading'.

7.3 Trials

During testing it was made clear that it would have been easier to give each participant written instructions, as well as use figures to explain the initial gestures needed to enter the application. It did not interfere with the actual tests, as no gestures were usable inside the

application, but it took unnecessary time from the participants to have to 'learn' the gestures such as *click*, or *bloom*. The initial plan was to observe the users behaviour using the Microsoft Device Portal, which shows a live view of the HWD. The 'live' view had a delay of 5-10 seconds and was prone to stuttering, even when at the lowest quality settings. Therefore it was not used, as it annoyed the user more than it helped to have delayed input. Without the live view working, and the ability to observe if a task was complete or not, there were no means of knowing when a participant was finished. There was also the sudden increase in 'learning curve', as participants not only needed to do the tasks but learn the basics of a HoloLens. In hindsight a short introduction should have been added. Some users seemed so fascinated by the AR-technology that they repeated the tasks multiple times, possibly interfering with the already uncertain test times. An argument can be made that this could easily be dealt with by automatically starting testing on prompt, and ending the application once the tasks were completed.

It is a fairly heavy device, the HoloLens, and during testing it was used without the headbands. This meant that most of the weight was on the nose if incorrectly worn. Because of this, all participants received help in fitting to make sure they were as comfortable as possible. During earlier stages of this project, the HoloLens in combination with the modified G2 was used, and was tested by five people in total for fitting. All of which rated it very, or in some cases extremely, uncomfortable, and it would have been interesting to have the setup available for a proper comparison. (*Note: The author of this thesis received short-term nerve damage to the nose from the modified setup. It has recovered since the hardware failure interrupted any further use.*) If the modified setup is to be used in any fashion, long or short term, it has to be modified even further. The main change would be the nose-guard that is currently in a hard plastic material. This works fine with the G2, with a total weight of 45 grams, but when the 579 grams of the HoloLens gets added on top, even with head straps, it is simply too heavy to rest on the nose. As such, the goal of an ergonomic product is somewhat achieved for a simple HoloLens, but not for a modified unit and would require further work. (*Note: There is currently no images of the modified HoloLens setup as the hardware is being repaired. Such images were planned for the last stages. In hindsight maybe they should have been taken earlier on in the process.*)

Chapter 8
Conclusions and Future Work

A novel solution for eye tracking inside a HoloLens using two different devices, the G2 and HoloLens, was successfully implemented but not entirely tested. Initial testing showed promising results but were halted because of hardware failure. Following this a simpler solution was implemented without eye tracking, i.e. a head-gaze tracked version. The user interface was shown to be intuitive and responsive, and should be easy to expand into any needed gesture, or gesture-tree for complicated actions.

The use of head tracking as an alternative method for gesturing when one does not have access to hands seems fully viable, and the work in this thesis should be able to be used as a framework for any future work in the subject without major modifications as the gesturing solution is built to be modular.

Future work would include fully testing the modified solution of HoloLens with G2 tracker. By finely tuning the calibrations, one should be able to get to a accurate estimation of eye position in world space, and as such be able to focus on interactable objects in ones surroundings. This could mean reduced strain on the neck over time, but possibly increase the strain on the eyes.

To summarize; gaze gesture patterns seem to be a fully viable method of interaction in Augmented Reality, but further work should be done in the subject to find out its true potential.

References

[1] S. K. Card, *The psychology of human-computer interaction*. Crc Press, 2018.

[2] M. Bhuiyan and R. Picking, "Gesture-controlled user interfaces, what have we done and what's next," in *Proceedings of the fifth collaborative research symposium on security, e-learning, internet and networking (SEIN 2009), Darmstadt, Germany*, 2009, pp. 26–27.

[3] D. A. Norman, "Natural user interfaces are not natural," *interactions*, vol. 17, no. 3, pp. 6–10, 2010.

[4] K. Ashton *et al.*, "That 'internet of things' thing," *RFID journal*, vol. 22, no. 7, pp. 97–114, 2009.

[5] F. Shi, A. Gale, and K. Purdy, "Helping people with ict device control by eye gaze," in *International Conference on Computers for Handicapped Persons*. Springer, 2006, pp. 480–487.

[6] D. Bonino, E. Castellina, F. Corno, and L. De Russis, "Dogeye: Controlling your home with eye interaction," *Interacting with Computers*, vol. 23, no. 5, pp. 484–498, 2011.

[7] F. Corno, E. Castellina, R. Bates, P. Majaranta, H. Istance, and M. Donegan, "D2. 5 draft standards for gaze based environmental control," *Communication by gaze interaction (COGAIN)*, 2007.

[8] V. Kullberg and E. Lindqvist, "Gaze guidance through head-mounted augmented reality display," 2019.

[9] B. Marques, P. Dias, J. Alves, and B. S. Santos, "Adaptive augmented reality user interfaces using face recognition for smart home control," in *International Conference on Human Systems Engineering and Design: Future Trends and Applications*. Springer, 2019, pp. 15–19.

[10] D. Verweij, A. Esteves, V.-J. Khan, and S. Bakker, "Smart home control using motion matching and smart watches," in *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces*. ACM, 2017, pp. 466–468.

[11] R. Aburukba, A. Al-Ali, N. Kandil, and D. AbuDamis, "Configurable zigbee-based control system for people with multiple disabilities in smart homes," in *2016 International Conference on Industrial Informatics and Computer Systems (CIICS)*. IEEE, 2016, pp. 1–5.

[12] P. Majaranta, J. Laitinen, J. Kangas, and P. Isokoski, "Inducing gaze gestures by static illustrations," in *11th ACM Symposium on Eye Tracking Research and Applications, ETRA 2019*. ACM, 2019.

[13] A. Köpsel, P. Majaranta, P. Isokoski, and A. Huckauf, "Effects of auditory, haptic and visual feedback on performing gestures by gaze or by hand," *Behaviour & Information Technology*, vol. 35, no. 12, pp. 1044–1062, 2016.

[14] R. J. K. Jacob, *Eye Tracking in Advanced Interface Design*. USA: Oxford University Press, Inc., 1995, p. 258–288.

[15] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. Macintyre, "Recent advances in augmented reality," *Computer Graphics and Applications, IEEE*, vol. 21, pp. 34 – 47, 12 2001.

[16] Okreylos, "On the road for VR: Microsoft HoloLens at Build 2015, San Francisco," http://doc-ok.org/?p=1223, added: 2015-05-01, accessed: 2019-10-11.

[17] Microsoft, "Microsoft Mixed Reality Toolkit documentation," https://docs.microsoft.com/en-us/windows/mixed-reality/, accessed: 2019-10-31 as pdf download.

[18] C. Ware and H. H. Mikaelian, "An evaluation of an eye tracker as a device for computer input2," *Acm sigchi bulletin*, vol. 18, no. 4, pp. 183–188, 1987.

[19] Tobii AB, "How do Tobii Eye Trackers work?" https://www.tobiipro.com/learn-and-support/learn/eye-tracking-essentials/how-do-tobii-eye-trackers-work/, accessed: 2019-10-11.

[20] ——, "Dark and bright pupil tracking," https://www.tobiipro.com/learn-and-support/learn/eye-tracking-essentials/what-is-dark-and-bright-pupil-tracking/, accessed: 2019-10-17.

[21] J. M. Carroll and J. Kjeldskov, "The encyclopedia of human-computer interaction," 2013.

[22] M. Weiser, "The computer for the 21st century," *IEEE pervasive computing*, vol. 1, no. 1, pp. 19–25, 2002.

[23] S. Bakker, "Design for peripheral interaction," Ph.D. dissertation, Department of Industrial Design, 2013.

[24] "About visual ergonomics," https://www.iea.cc/about/technical.php?id=51df9aa27ebf9, accessed 2019-10-16.

[25] J. Long and H. Richter, "Visual ergonomics at work and leisure," *Work*, vol. 47, no. 3, pp. 419–420, 2014.

[26] C. E. Connor, H. E. Egeth, and S. Yantis, "Visual attention: bottom-up versus top-down," *Current biology*, vol. 14, no. 19, pp. R850–R852, 2004.

[27] W. Lu, D. Feng, S. Feiner, Q. Zhao, and H. B.-L. Duh, "Subtle cueing for visual search in head-tracked head worn displays," in *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 2013, pp. 271–272.

[28] A. Hunt and D. Thomas, *The Pragmatic Programmer: From Journeyman to Master*. Addison-Wesley Professional, 1999.

[29] Microsoft, "#if (c# reference)," https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/preprocessor-directives/preprocessor-if, accessed: 2019-09-31.

[30] Tobii AB, "Tobii Pro Glasses 2 API - Developer's Guide, v1.3," https://www.tobiipro.com/product-listing/tobii-pro-glasses-2-sdk/, accessed: 2019-10-17.

[31] I. T. Hooge, R. S. Hessels, and M. Nyström, "Do pupil-based binocular video eye trackers reliably measure vergence?" *Vision research*, vol. 156, pp. 1–9, 2019.

[32] W. H. Ehrenstein, B. E. Arnold-Schulz-Gahmen, and W. Jaschinski, "Eye preference within the context of binocular functions," *Graefe's Archive for Clinical and Experimental Ophthalmology*, vol. 243, no. 9, pp. 926–932, 2005.

[33] W. Delamare, T. Han, and P. Irani, "Designing a gaze gesture guiding system," in *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services*. ACM, 2017, p. 26.

[34] A. Huckauf and M. H. Urbina, "On object selection in gaze controlled environments," 2008.

[35] O. Tuisku, V. Surakka, T. Vanhala, V. Rantanen, and J. Lekkala, "Wireless face interface: Using voluntary gaze direction and facial muscle activations for human-computer interaction," *Interacting with Computers*, vol. 24, pp. 1–9, 01 2012.

[36] A. Pasarica, R. Bozomitu, V. Cehan, and C. Rotariu, "Eye blinking detection to perform selection for an eye tracking system used in assistive technology," 10 2016, pp. 213–216.

[37] P. Majaranta and K.-J. Räihä, "9. text entry by gaze: Utilizing eye-tracking," *Text Entry Systems: Mobility, Accessibility, Universality*, 01 2007.

[38] D. A. Norman, *The Design of Everyday Things*. USA: Basic Books, Inc., 2002.