University of Bremen

Faculty of Mathematics and Computer Science

Computer Graphics and Virtual Reality Research Group

Bachelor thesis

Redirected Walking in Virtual Reality during eye blinking

Edited by: Victoria Ivleva (Matr. Nr: 2770973)

Supervisors: Prof. Dr. Gabriel Zachmann

Prof. Dr. Mehul Bhatt

Thesis submitted in partial fulfillment of the requirements for the Digital Media Bachelor of Science degree at the University of Bremen

2016

STATUTORY DECLARATION

I declare that I have authored this bachelor thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

_____ _____

Date Signature

Abstract

This bachelor thesis aimed to explore a technique which helps to achieve more immersion in Virtual Reality (VR) by creating the illusion of free movement in VR by using features of the human visual perception. The main goal is to ascertain if users are able to notice manipulations by changing their position when these manipulations occur during eye blinking. The main technique to generate a free movement illusion in virtual environments is called Redirected Walking (RW). This technique is based on manipulations, such as rotation and translation of virtual objects or of the user's point of view. There are different options to apply Redirected Walking.

In this study, one of the main processes of the human eye is used, namely reflexive blinking. In particular, at the time when a person blinks, the position of his/her point of view in virtual space changes. The main goal of this work is to find out if users are able to notice manipulations by changing their position when these manipulations occur during eye blinking. This experiment uses an HTC Vive Head Mounted Display (HMD), Unity3D Game Engine and a self-made eye blink sensor. The blink sensor is based on an Arduino Uno microcontroller, a LED and a photo sensor.

The experiment was implemented in Unity3D, and participants were standing in a virtual environment that replicated the real room. During the experiment, participants were wearing an HTC Vive HMD and had a wireless StreamVR controller in their hands to choose the answers in VR. The main manipulations during the experiment were rotation and translation along 3 axes and by 7 values.

The results showed that rotations in the negative direction, in particular around the Y-axis, were perceived correctly up to 86-97%, by the Z-axis up to 80-90%, and by the X-axis up to 78-82%. Translations were also perceived correctly, but users tended to choose the positive direction when they were translated along the Z-axis. In addition, participants were evaluated by the Kennedy SSQ [1] to check for physical side effects, and these effects increased after the experiment.

The results of this experiment demonstrate that Redirected Walking in Virtual Reality during eye blinking is possible, but there are some additional factors which need to be considered:

- precise eye blink detection with an eye tracker specially designed to work with the HMD should be ensured;
- exact timing of running the reorientation algorithm is important for perception of this manipulation;
- to avoid the discrepancy between real and virtual movement and to reduce the motion sickness effect, the reorientation should be added to the actual movements;
- it is also necessary to involve most of the human perception channels to reduce cognitive abilities and attention to details. This can be achieved with the use of specific virtual environments, e.g. outdoor nature environments with sounds.

In the context of real applications, it is important to note that people blink infrequently, once per 4-6 seconds.

Keywords: Redirected Walking, HMD, Immersion, Eye Blinking, Saccadic Eye Movement, Natural Walking, Unity 3D, Tuscany, Virtual Reality, Arduino, eye blink sensor.

Structure

Figure list

Table index

Abbreviatons

RW – Redirected Walking

VR – Virtual Reality

fps – Frames per second

ms – Millisecond

HMD – Head Mounted Display

LDR – Light Dependent Resistor

LED – Light-Emitting Diode

COM – Communication Port

FOV – Field of View

1 Introduction

The topic of my bachelor thesis is “Redirected Walking in Virtual Reality during eye blinking”. A Virtual Reality is a computer-generated environment which replicates existing landscape and indoor rooms, or represents imaginary space, where users can move and interact with it.

The most important problem in Virtual Reality is that the physical space is much smaller than the virtual one. The size of the real space limits the ability to move in the boundless virtual space. This, in turn, violates the immersion into the virtual world. Nowadays, scientists are trying to solve the problem with maximal immersion, without any intrusive hardware [2].

The researchers have developed a technology called “Redirected Walking (RW)” which is based on using a Head Mounted Display (HMD) and algorithms based on the use of features of human visual perception. The Redirected Walking technique is based on many different methods of manipulating the user’s orientation or position during natural locomotion in virtual reality, without their noticing. In general, there are several basic options to manipulate the user’s position: translation, rotation, resetting. These manipulations can happen automatically, manually or dynamically [2].

Some of the Redirected Walking experiments are based on the use of saccadic eye movements. At the same time as the user is looking in one direction, some virtual objects or the part of the virtual environment in the opposite direction, from the user’s point of view, are manipulated to correct the user’s walking direction. In some studies scientists directly manipulate the position of the user, like Razzaque et al. [2], or interactively rotate the virtual scene around the participant [3]. There is also a lot of interest in the perceptual aspects of different redirection techniques. Steinecke [4], Neth [5] et al., and Bruder [6], for example, researched how Redirected Walking influences a user’s cognitive abilities.

In this work, the Redirected Walking experiment is based on physiological characteristics of human visual perception. Specifically, the algorithm to redirect the user's walking direction will be launched at the time the user blinks. The details are described in Section “Redirected Walking during eye blink”.

1.1 Motivation

Over the last few years, Virtual Reality has been developing at a rapid rate. A lot of applications and hardware are designed to make some imaginary place possible even without leaving your own room.

There are a lot of problems in the development of Virtual Reality and questions that need unconventional solutions. For example, VR is mostly used indoors, which leads to a smaller physical space than the virtual world where you want to move or play. Since increasing the physical space is not always possible, developers and scientists are trying to solve this problem by different scientific methods. One of them is the Redirected Walking (RW) technique, which can stop the users from running into their own table, by manipulating the position of virtual objects or the users themselves.

This technique is based on physical and physiological characteristics of human visual perception. The main intention of this work is to find out the ability of users to observe manipulations by changing their position if these manipulations occur during their eye blinking.

1.2 Requirement

The aim of this work is therefore to design a system that would allow RW during eye blinking using an HMD and to build an eye blink detection device prototype to make an initial evaluation and ensure the feasibility of the designed system. For this purpose the following hardware and software is needed: Windows x64 Bit PC, 4 GB RAM, Intel Core i5 processor, Nvidia Geforce GTX 970, HDMI 1.4, HTC Vive Head Mounted Display, SteamVR Motion Controller, Arduino Uno Microcontroller, Excelitas LDR04 Photosensor, Unity 3D Game Engine Version 5.4, SteamVR, Arduino Code-Editor, test project scene for Unity 3D with high amount of light or an Adafruit NeoPixel LED strip RGBW.

2 Previous Work

The first successful application in Redirected Walking appeared in 2001 by Razzaque [2] and colleagues, which was based on a pre-programmed amount of rotation. The virtual environment was larger than the real space, and users were able to move naturally, but in a zig-zag pattern. Every time the users turned through 90 degrees in the virtual environment, they actually turned 180 degrees in the real space. This manipulation induced the movement of users in a zig-zag pattern in the large virtual environment, without collisions with real obstacles in physical space. The participants did not pay attention to the difference between their real and virtual movements. However, this Redirected Walking study of Razzaque et al. [2] has a disadvantage - a lack of freedom: the users cannot choose where they want to go, and should follow the prescribed route.

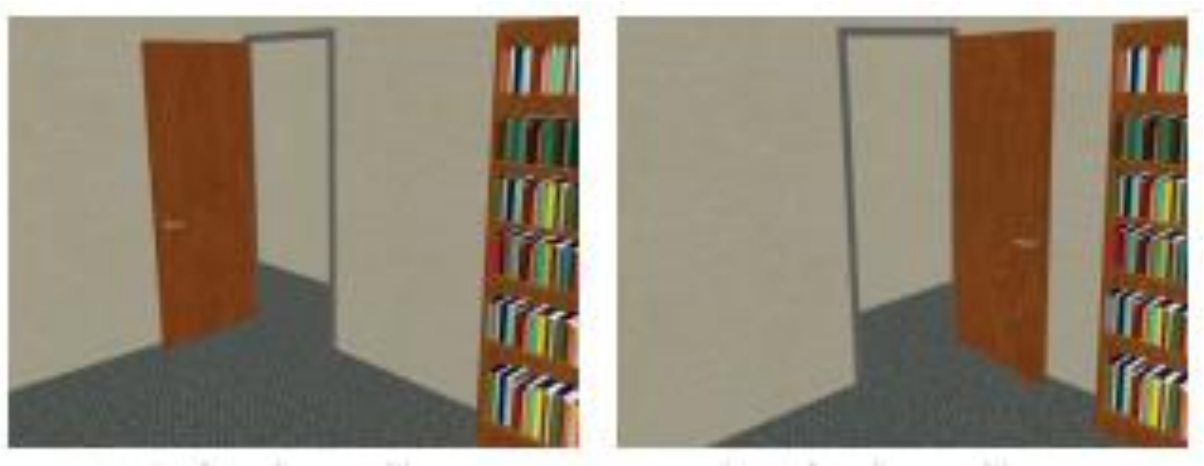


Figure 1 Change blindness redirection [6]

In 2009, Steinecke [4] and his research group analysed human sensitivity to the redirection of walking and looked for some perceptual thresholds that can be detected by users. The important feature of this work was that the Redirected Walking could be implemented with some perceptual manipulations to which the users are less sensitive. They tested the following manipulations: rotation, translation and curvature gains along a circle. The results for translation and rotation manipulations showed that users were less sensitive to an increase in the rate of change, up to 40 percent for rotation and 26 percent for translation. Also, the users did not notice the curvature gain along the circle with radius over 22 meters. The perceptual sensitivity of the users was also researched by the group of scientists [5] in 2012. They explored whether the users sensitivity for the rotation manipulations is dependent on the

users walking speed. The results have showed that the faster users were walking, the more sensitive they were to the manipulation within rotation.

In 2011, Suma et al. [7] and colleagues explored a technique related to Redirected Walking, named Change Blindness Redirection. Change Blindness is a phenomenon, which appears if some part of the environment is out of the users' field of view. If some changes happen at the moment when the part of the virtual space is out of the users view, they would not notice these changes. In this study, users explored the virtual environment, which was four times bigger than the real room. The virtual place that researchers have developed was a building with a lot of rooms and as soon as users entered any of the rooms, the entered door moved to the other side, but the users could not see the door because it was out of their field of view (Figure 1). Only 1 out of 77 participants noticed this change. But this technique is applicable only for indoor virtual environments.

Peck et al. [8] in 2011 conducted research and experiments in Redirected Walking using distractors. Distractors, like the humming-bird that flew back and forth in front of participants, are used to stop users when they are near to the boundary, to prevent collision, or when the users are away from obstacles, to prevent participants from leaving the tracked space.

In 2015, Bolte et al. [9] explored redirection during saccades, short and fast eye movements. They used electrooculography (EOG) to track the saccade, and changed the scene or user position during saccadic movement. The users' view direction was translated and rotated during saccadic movement and fixation. The study showed that users were less sensitive to manipulations during saccadic movement than during fixations. When participants were stable, they could not notice the reorientation (rotation) of ±5 degrees and translation along the line of vision of ±0.5 meters during saccades [9].

This appearance of less sensitivity to manipulation during the eye movement can be explained by the fact that the human visual system uses a built-in prior assumption that the world is stable during the eye movements. This theory and Bolte's study [9] became the basis of the work on redirection during eye blink.

3 Redirected Walking during eye blinking

Redirected Walking is a manipulation technique to re-orient users in virtual environments during natural locomotion. Natural locomotion in a virtual environment is very important to achieve maximum immersion and to eliminate all moments when users might disbelieve. Also, natural locomotion can help avoid the "Motion Sickness", an unpleasant phenomenon that often disrupts users in virtual reality and which appears because of the complex human perception system [10].

The main goal of this study is to find out whether users are able to notice manipulations by changing their position when these manipulations occur during eye blinking.

Blinking is an instinctive closure of the eyelids. The eyelid is used to moisten the cornea with tear fluid and to protect the eye against foreign bodies [11]. A person blinks about 10 to 15 times per minute, so every 4 to 6 seconds; this happens over an average time of 100 to 400 milliseconds [12].

The dark phase during closure of the eyelids is not consciously perceived as the visual perception in relevant brain areas is suppressed shortly before and during blinking. This is a neural mechanism underlying the lack of awareness of the changes in visual input. Blink interrupts visual input, but usually is not noticed because of blink suppression [12] [13]. Additionally, according to current theories, the human brain believes that the world is stable during the eye movements [9].

The blinking happens usually synchronously: on both sides and at the same time. However, it is possible for almost all people to close only one eye. This movement often leads to a specific strain of the eye and face muscles. According to a small study carried out in 2008, women blink faster and more often than men do. In fact, women blink 19 times per minute whereas men blink only 11 times during the same period. Also, older women blink more frequently than young women [14].

These special features of human visual perception are the main reason why the decision was made to apply the redirection during eye blinking.

3.1 Eye blink detection

For the implementation of the experiment, it was necessary to capture the moment when participants were blinking. For this purpose, eye tracking can be used. Eye tracking is a process that measures some features of the eye in order to determine the viewing direction and the visual focus of the eye. The eye trackers locate the iris of the eye in captured video pictures, and specific algorithms, for example the Circular Hough Transformation, recognize rounded patterns in the image [15] and determine the direction of view, i.e. point of gaze. Eye tracking is used in several applications to control and interact with digital devices or for research purposes in design and advertising analyses as well as academic and medical research.

The most common technology used for this type of tasks is real-time eye tracking. An eye tracker called The Eye Tribe (Figure 2) developed by The Eye Tribe Apps company, based in Denmark, was tested for suitability for the specific purposes of this study. This device is based on infrared video and mathematical algorithms which calculate the point of gaze [16].

Figure 2 The Eye Tribe eye tracker [10]

The Eye Tribe is easy to use, but too bulky (20×1.9×1.6cm) to be used in an HMD; moreover, it is not suitable for use at close range due to the fact that a distance of about 30cm to 45cm must be maintained, and such a distance is not required in an HMD [16]. In addition, a video-based real-time eye tracker is not fast enough, mostly 30fps, because the eye cannot move faster than 30 motions per second [15]. Nonetheless, the Eye Tribe has delay of about 20ms. Consequently, no precise information regarding the eye blinking can be provided as a minimum frame rate of about 100fps is required. Measurements should be carried out approximately every 10ms to check whether the blinking process has started, as the minimum duration of the whole process is 100ms.

Another device that can be used for eye tracking is an EOG Electrooculography amplifier (Figure 3). This device is based on tiny electrodes that measure the electric field created by the corneoretinal resting potential. These electrodes are attached to the skin at the inner and outer corners of the eye. When the eye moves, the potential of these electrodes changes and the difference between these individual potentials provides information as to where the eye moved [9].

Figure 3 An EOG [17] device used to measure saccadic movements

The EOG is a precise measuring system, but as the noise level of the EOG signal differs for each participant and may also vary over time, it is challenging to develop and setup a general eye movement classification algorithm. Additionally, the EOG with its electrodes contradicts with the concept of maximum non-intrusiveness of the VR hardware, which supports maximum immersion into VR.

For that reason, the idea was to find some other possibilities to capture the blinking movement in a faster and easier way. For this purpose, the physical characteristics of the eye and the changes that occur during the eyelid closure have been studied.

3.2 Sensor description

During a blink, the eyelid covers nearly the entire eyeball; this feature can be used for eye blink detection. The question is, though, what else changes in addition to the eyelid closure. One of the changes is the light reflection. The eyeball does not reflect most of the light frequencies; instead, it absorbs them, as shown in Figure 5. Furthermore, most of these light frequencies can cause eye damage; for example, the normal light can cause photochemical

damage of the retina, the ultraviolet light causes corneal irritation and cataract, and the infrared frequencies lead to thermal damage of the retina [17]. Unlike the eyeball, the eyelid skin reflects the majority of light frequencies. Therefore, these reflection characteristics are used for eye blink detection.

The first prototype of an eye blink detection sensor was an infrared optocoupler, a device containing an infrared light-emitting diode (LED) and a light-sensitive component, e.g. a Light

Figure 4 LDR4 [29]

Dependent Resistor (LDR). The idea was that the LED should beam into the eye, and the reflected light should be captured by the photocell. Unfortunately, the experiment showed that the optocoupler needed a very close distance to receive the signal and the infrared LED got hot while operating. As a result, under the given conditions, usage of the infrared frequencies emitted by the optocoupler can harm the eye.

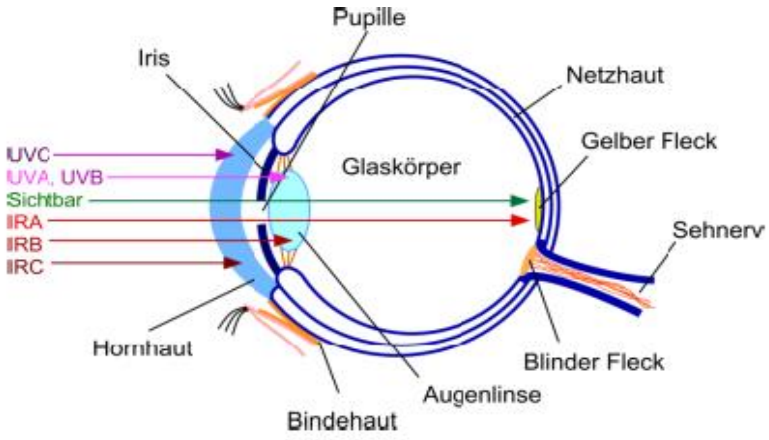


Figure 5 Biological effects of optical radiation on the eye [11] show that the eye absorbs most light frequencies

In the next step, usage of a single light-sensitive component and the light emitted by the HMD screens was considered. The emitted light should be reflected from the closed eyelid and received by a simple photocell which is integrated in the HMD. This method is non-intrusive, but it is necessary that the shown scene be well lit; otherwise, the photocell does not receive enough light. Therefore, later it was decided that an external light source should be used inside of the HMD to make the setup work with every single virtual scene.

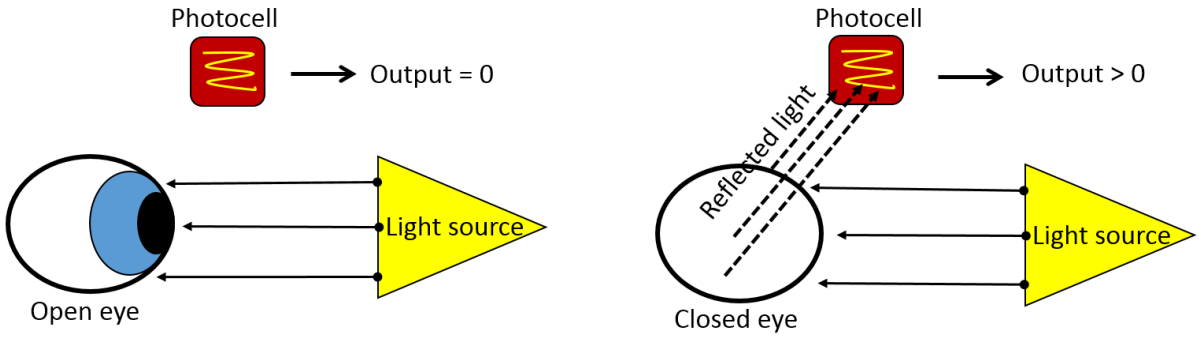


Figure 6 Working principle of the self-made optical sensor: the light emitted by the HMD display is reflected by the eyelid if the eye is closed and the photo sensor produces output bigger than zero

As an external light source, an Adafruit NeoPixel Digital RGBW LED Strip with three small RGBW LEDs was used. This LED strip was placed in the HMD above the eye and was shining downwards. The colour of the LED was green because the used Light Dependent Resistor LDR4 (Figure 4) is sensitive to spectral wave length of 520 to 600nm, which is equivalent to green and yellow coloured light from the visible spectrum. The LDR was placed above and to the side of the eye (Figure 6). The light from the LED was reflected when the eyelid was closed, and this reflected light was detected by the LDR and an output was generated. This output was an integer value from 0 to 256, depending on the amount of light falling on the LDR. The final eye blink device is shown in Figure 8.

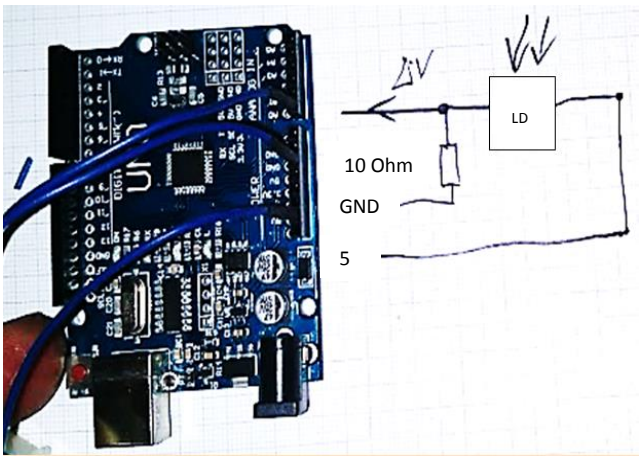


Figure 7 Left: Arduino LDR circuit

The LDR4 had two pins and was interfaced with and controlled by an Arduino Uno microcontroller; the circuit of the device is shown in Figure 7. The communication ran via a Communication Port (COM Port). Over the same port, the Unity Application received sensor data with baud rate of up to 57.600 values per second. The response time of the LDR was 1 to 3ms [18], and the force of data delivery as well as the LED were controlled by the code for Arduino (see Arduino source code in attachment).

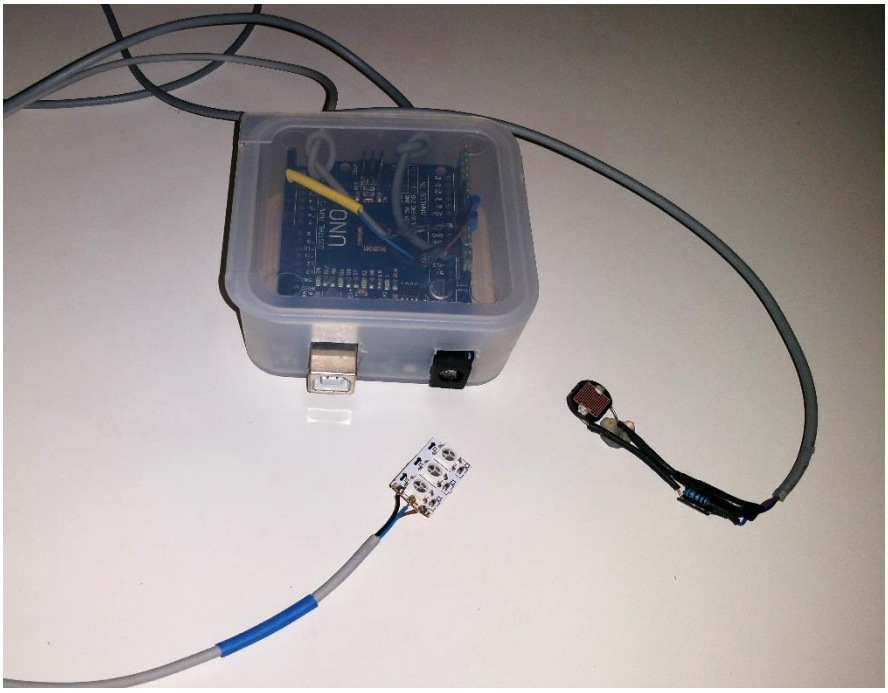


Figure 8 Final eye blink device with an Arduino Uno microcontroller, an LED and an LDR

The sensor evaluation was performed without the HMD in a completely dark room, simulating the conditions in the HMD; the LDR was placed above and to the side of the eye. A smartphone lamp served as the light source. The user's eye was also recorded with a smartphone camcorder, and the abrupt lighting change from dark to light served as the timestamp. The values received from the sensor were stored in a .txt file and then imported and processed in Excel. In Figure 9 LDR evaluation graph, the blue line represents sensor output distribution: depending on the amount of light received by the LDR, the values were between 0 and 3 when the eye was open; with the eye closed, as marked by the red dots, the maximum achieved value was 5. The graph shows fluctuations between 2 and 3 which may be explained by the strong light source and the reflection from the skin around the eye. A more efficient way would be to use a spot light source at a short distance to the eyelid to avoid diffuse reflection from other skin parts. The red dots in the graph represent the moments of eye blinking; it took the LDR about 20ms to notice the change from closed to open eye. This force was controlled

by the baude rate in Arduino Script and was equal to 9.600. By using a higher baud rate value, the reaction time of the sensor can be improved up to 2ms. The LDR response force and accuracy of reaction upon eye closure is highly dependent on the position of the LDR relative to the eye. Also, the open angle of the LDR (±60 degree [18]) must be considered. To make the difference between the reflection from the skin near the eye and the eyelid more noticeable, a reflection booster should be used. It can be a small piece of a highly reflective material, like aluminium foil, that can be placed on the eyelid. Such an auxiliary component would also help make the received reflection values more homogeneous because every person's skin on the eyelid reflects the light differently, depending on the general skin characteristics.

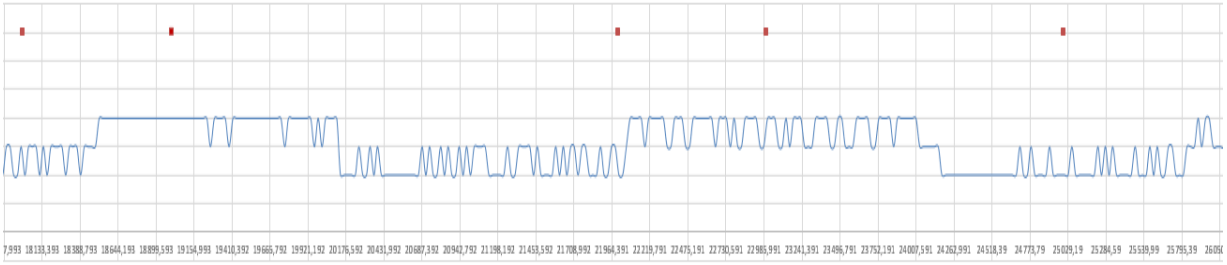


Figure 9 LDR evaluation graph: blue line – sensor output from 0 to 4, red dots – eye closing moments, below – time in ms

As discussed above, the blinking usually happens synchronously: on both sides and at the same time; therefore, one sensor should be enough to track the blinking. The sensor needs to be positioned very precisely to provide correct data; ideally, a positioning system with 3 degrees of freedom should be implemented. Generally, the sensor should be placed above the eyelid at a distance of about 15–20mm in order not to block the eye movements and to capture the light reflected from the eyelid. It was found that the best position was above and at the outside corner of the eye, but this position, distance and angle in particular, should be calibrated for each user due to different head size and facial morphology. In addition, because of using the LED, the LDR should be protected from its direct light.

Figure 10 Left: adjustable sensor positioning design for Oculus Rift DK2; right: sensor and LED positioning in HTC Vive

In this study, various sensor positioning and mounting methods were developed and tested. At the start of this project, Oculus Rift DK2 was used as the HMD. Later on, it was decided to use the HTC Vive HMD. For both HMDs, it was necessary to solve the sensor positioning and mounting problem. The Oculus Rift DK2 HMD differs in its design from the HTC Vive as it has thicker round lenses with a smaller radius than the latter. For each of them, an individual mounting device had to be used (Figure 10), a round-shaped foam material placed around the lens, which can be rotated to change the position of the sensor. Furthermore, the material, soft as it is, allows to push the sensor deeper into the foam and, by doing so, adjust the distance to the eyelid.

Figure 11 shows one of the prototypes for the sensor positioning device. The shown device is based on eyeglasses properly sized to be used in the HMD (HTC Vive). This design ensures accurate and reliable sensor positioning for different users; the distance and the angle to the closed eyelid were almost equal for each user. Unfortunately, when the HMD was placed on the head, the glasses were pressed up and the sensor changed its position relative to the eye making the distance between the LDR and the eyelid too short for correct functioning of the LDR. For this reason, it was impossible to use this device in the HMD.

Figure 11 Universal sensor positioning device design based on eyeglasses to use in HMD

Eventually, the sensor was mounted in the HTC Vive HMD on a flexible material made of 100% silicone (Figure 10) so that the LDR position could be changed if necessary. It was also protected from the LED light with a piece of stable black material. However, this positioning method does not allow for precise calibration of the sensor for each user. To get more

accurate data from the sensor, a high-precision positioning system with 3 degrees of freedom is required.

3.3 Virtual Environment Application

For the Virtual Environment application, Unity 3D Version 5.4 was used. The Unity 3D game engine communicates with the sensor via a Communication Port and receives integer values depending on the light amount captured by the LDR. The main algorithm for redirection should start at the moment when the sensor value achieves a predefined threshold, which means that the user is closing or has already closed the eyes. The threshold first needs to be calibrated for each user via Unity 3D Inspector.

In general, the script for communication between Unity 3D and the sensor contains following commands written in pseudocode (Figure 12): (see the full Unity3D source code for communication with Arduino in attachment)

```
void Start()
    {
        SerialPort.Open();
    }

    void Update()
      {
        ArduinoLight = sp.ReadByte();
      }

    public void processSensorValue(int value)
    {
        if (ArduinoLight == thresholdBlink)
            {
                Reorientations = true;
            }
        }
    }

    public void Close()
    {
        sp.Close();
    }
```

Figure 12 Pseudocode for connection between Unity 3D and Arduino via a COM Port

This script also activates the redirection scripts. Translations along the X-, Y-, Z-axes by 0, ±0.5, ±0.10, ±0.15m and rotations around the X-, Y-, Z-axes by 0, ±5, ±10, ±15 degrees were used as the redirection. The user's Field of View (FOV) (Figure 15) was rotated and translated.

4 User study

4.1 Pre-study

Before the final user study, a pre-study was carried out to check the eye blink sensor connection with Unity 3D, to define the correct position for the sensor and to check the overall performance of the setup and the scene. The pre-study was implemented with Oculus Rift DK2, Unity 3D Version 4.3, an eye blink sensor based on Arduino Uno and an LDR. As a light source only the light from the virtual scene and as a test scene the Tuscany Demo from Microsoft for Unity 3D and Oculus Rift DK2 [19] (Figure 13) were used.

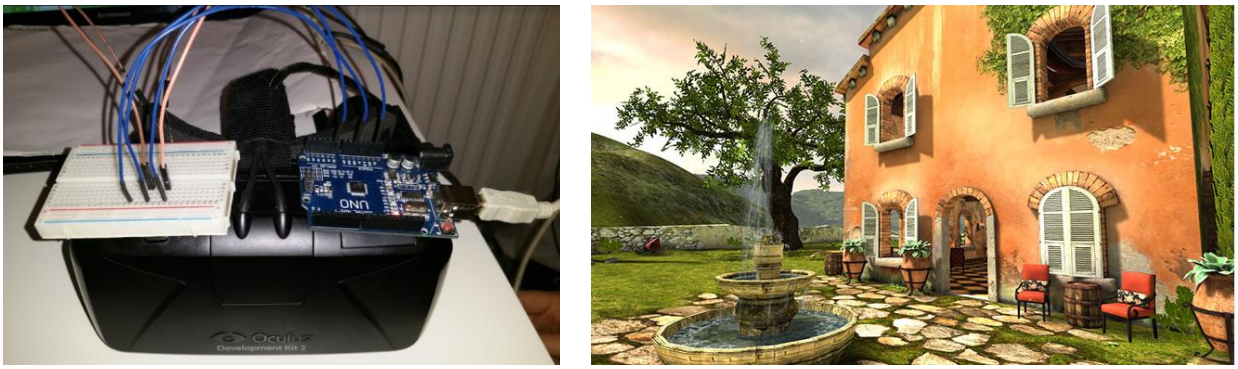


Figure 13 Left: pre-study equipment: Oculus Rift DK2 with Arduino and eye blink sensor setup; right: a screenshot from the Tuscany Microsoft Demo

This pre-study involved three participants: one female and two males aged 14 to 30, all without any prior experience with the HMD studies or 3D games. The participants were sitting during the experiment wearing an Oculus Rift DK2 HMD. They were moving in the scene by using a computer keyboard and were able to move their head. The reorientation included only rotation around the Y-axis by +5 degrees. After the detected blinking, the participants were rotated and then were asked whether they had seen some rotation or not, and if yes, then in which direction. This experiment showed that the sensor had to be very precisely positioned and securely fixed. When the correct position was found and the threshold equal to the light reflected from the closed eyelid was exactly defined, the reorientation was not noticed in 80% of the cases.

The problem in this experiment was that Oculus Rift DK2 did not have full body motion tracking, which was required for the RW experiment, and the Tuscany scene contained many distractors, like floating particles in the air, butterflies, birds, sea waves and sounds. All these

scene details have loaded the user's attention, and the small reorientation manipulations went unnoticed.

4.2 Final experiment

The final user study was performed in the Human Computer Interaction Laboratory (HCI Lab) of the Computer Science Department at the University of Hamburg from 18th of July to 30th of July 2016 with support from PhD student E. Langbehn and Dr. G. Bruder. The main objective of the project was to evaluate the potential of the eye blinks for Redirected Walking in an immersive virtual environment.

The participants were required to stand in the laboratory wearing an HTC Vive HMD and to follow the instructions presented at the display. The participants had to be healthy and free of any disorder of equilibrium or vision.

The experiment took on average approximately 80 minutes. As part of the experiment, participants signed a participant information and consent form (see Participant information and consent form in attachment) which includes information about the experiment, participants' task, benefits and risks related to the participation, e.g. motion sickness, confidentiality, sharing and publication agreement and consent to photo and video recording. Before and after the experiment, participants completed the Kennedy Simulator Sickness Questionnaire [1] to measure the VR induced physical side effects. After the experiment, participants also answered questions about the experiment itself.

4.3 Hypothesis

The experiment's aim was to prove or disprove the following hypothesis: based on the neural mechanism of blink suppression [12] [13] and the information that the human brain believes that the world is stable during the eye movements [9], participants were expected not to notice manipulations by changing their position if these manipulations occurred during eye blinking.

4.4 Experiment design

At first, participants were asked to fill in two questionnaires: a demographic questionnaire with general questions about age, gender and VR experience, and the Kennedy Simulator Sickness Questionnaire (SSQ), which is a standard in VR studies (see Simulator Sickness Questionnaire in attachment). After that, the task was explained to the participants, and the exact procedure was described. Then they were given a controller, which they held in the right or left hand, and an HTC Vive HMD was placed on their head. Inside the HMD, a green LED was shining, and the virtual scene replicated the real room where the experiment was performed (Figure 14).

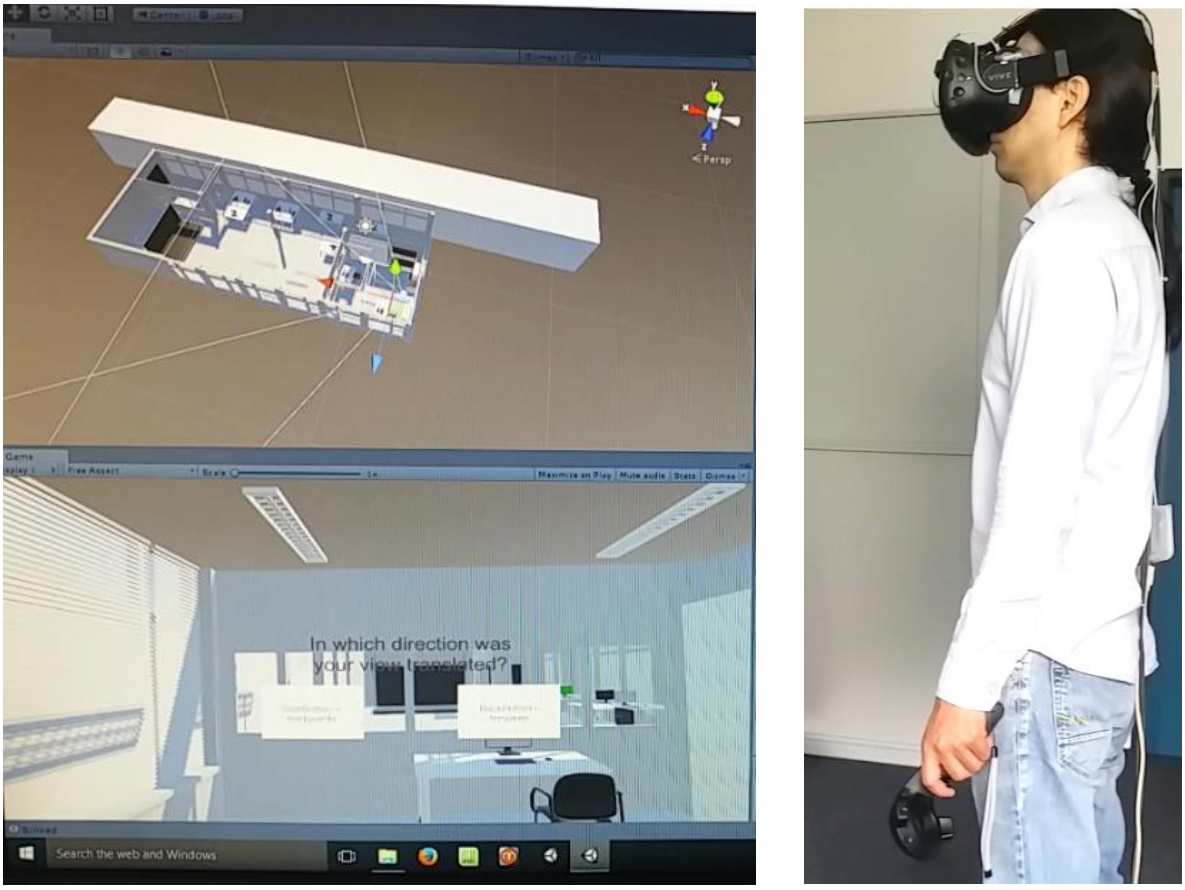


Figure 14 Left: Unity 3D scene seen by participants; right: a participant during the experiment wearing an HMD and holding a controller

The participants were supposed to stand in one place during the whole experiment. The reason for this is that this stable body position ensures more accurate measurement of the user's sensitivity to manipulations. When a person is moving, his/her attention to small changes is lower, while in a static position attention is focused on one particular task. The participants should be more sensitive to smaller changes, especially in an indoor virtual scene with regularities in the seen image. This difference might influence the experiment and users

might notice more manipulations. It is expected that if some manipulations are not or only rarely perceived under the given conditions, they would not be noticed during the natural walking in VR because users would have to pay attention to more things, their cognitive abilities would decrease [6] and they would not be able to discern small reorientations.

After each detected blinking and reorientation, the following question appeared in front of the user: “In which direction were you rotated?“ after rotations and “In which direction were you moved?” after translations, both questions regarding the user’s view. In addition to each question, two buttons with possible answers appeared under the question text. Only one of them contained correct information about the direction where the user was moved, except rotation and translation by 0 degrees and meters, because in this case both answers were false; nevertheless, participants had to answer the question.

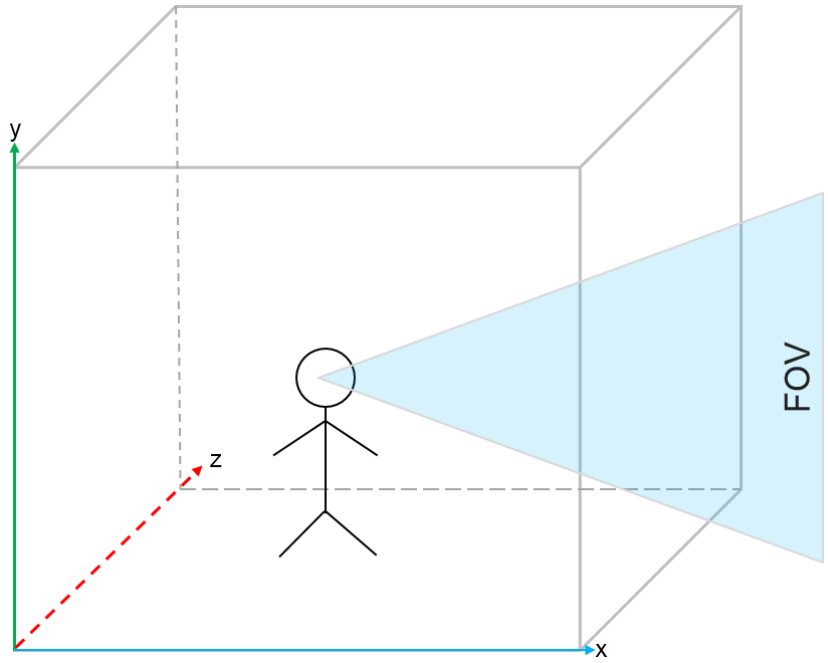


Figure 15 Redirection of user's FOV along 3 axes

At the start of the experiment, participants were given the following task: after the question appeared, they had to select an answer, which they thought was correct, and confirm it by pressing back button or side button on the controller. The answers for rotation were: right, left, up, down; for translation: right, left, forward, backward, up, down. All answers for each user were recorded in a .txt file for evaluation.

Each participant’s FOV was randomly rotated around all three axes X, Y and Z (Figure 15) by 0°, ±5°, ±10°, ±15°. Additionally, each user was also randomly translated along all three axes X, Y and Z by 0m, ±0.5m, ±0.10m, 0.15m.

At the beginning of the experiment for each participant, a trial was carried out, one for translation and one for rotation. During this trial, the sensor was calibrated, participants were asked to blink several times and then to keep their eyes open for some seconds to find the threshold between two eye conditions, open and closed. Additionally, the position of the sensor in the HMD was adjusted, if needed. During the trial, participants practiced choosing the answers with the controller and pushing the buttons.

Each participant was rotated and translated eight times by each angle and each distance around/along each axis (Figure 16). The experiment contained two general runs: rotation and translation. A rotation by +5, +10, +15 degrees around the X-axis corresponded to a rotation upward. A rotation by +5, +10, +15 degrees around the Y-axis corresponded to a rotation to the right. A rotation by +5, +10, +15 degrees around the Z-axis corresponded to a rotation to the right. A rotation by -5, -10, -15 degrees around the X- axis corresponded to a rotation onward. A rotation by -5, -10, -15 degrees around the Y- axis corresponded to a rotation to the left. A rotation by -5, -10, -15 degrees around the Z- axis corresponded to a rotation to the left. A rotation of 0 degree around the X-, Y- and Z-axis changed nothing.

```
        void Start () {
              possibleAxises = new List<Vector3>();
              possibleAxises.Add (Vector3.up);
              possibleAxises.Add (Vector3.right);
              possibleAxises.Add (Vector3.forward);

              possibleDegrees = new List<float>();
              possibleDegrees.Add (-15f);
              possibleDegrees.Add (-10f);
              possibleDegrees.Add (-5f);
              possibleDegrees.Add (0f);
              possibleDegrees.Add (5f);
              possibleDegrees.Add (10f);
              possibleDegrees.Add (15f);

              repetitions = 8;


          for (int i = 0; i < 10; i++)
          {     int chosenTransform = UnityEngine.Random.Range(0,
possibleTransforms.Count - 1);
              int chosenAxis = UnityEngine.Random.Range(0,
       possibleAxises.Count - 1);
             int chosenDegree = UnityEngine.Random.Range(0,
possibleDegrees.Count - 1);
```

Figure 16 Part of the source code for rotation

A translation by +0.5m, +0.10m, +0.15m along the X-axis corresponded to a translation to the right. A translation by +0.5m, +0.10m, +0.15m along the Y-axis corresponded to a translation upward. A translation by +0.5m, +0.10m, +0.15m along the Z-axis corresponded to a

translation forward. A translation by -0.5m, -0.10m, -0.15m along the X-axis corresponded to a translation to the left. A translation by -0.5m, -0.10m, -0.15m along the Y-axis corresponded to a translation downward. A translation by -0.5m, -0.10m, -0.15m along the Z-axis corresponded to a translation backward. A translation by 0m along the X-, Y- and Z-axis changed nothing.

Additionally to these two general parts: rotation and translation, one trial was conducted for each part with also ten test manipulations. After the first part, the participants were offered a break as one part takes 20 to 30 minutes and the HTC Vive HMD has no cooling inside. Besides, there are additional heat sources, such as the LED. Such conditions were uncomfortable for extended periods. All in all, there are 356 translations and rotations in the experiment:

$$Rotation\ per\ user = 3\ axes \times 7\ values \times 8\ times\ for\ each\ combination$$
$$= 168\ times + 10\ test\ runs = 178\ times$$

$$Translation\ per\ user = 3\ axes\ \times 7\ values\ \times 8\ times\ for\ each\ combination$$
$$= 168\ times + 10\ test\ runs = 178\ times$$

Not each blinking movement was recognised by the sensor, so participants sometimes had to blink more than once. The question with the answers appeared 1.5 seconds after the manipulation. It took the participants a few seconds to answer the question; therefore, the experiment took on average 80 minutes per user, including the break between the parts and the time for filling in the questionnaires.

4.5 Equipment

The experiment was performed with an HTC Vive HMD and Steam Valve Version 2.10.91.91 [20] running on Windows 8.1 x64 Bit operating system. The experiment laboratory room scene and the experiment were developed in Unity 3D Game Engine Version 5.4 [21].

Figure 17 Left: HTC Vive [18]; right: Oculus Rift DK2 [18]

For eye blink detection, the following equipment was used: an Arduino Uno microcontroller, an Excelitas LDR04 photo sensor built in the HTC Vive HMD, Arduino Code Editor [22], an LED strip emitting green light built in the HTC Vive HMD, a USB 2.0 cable, and a Windows 8.1 PC (Figure 18).

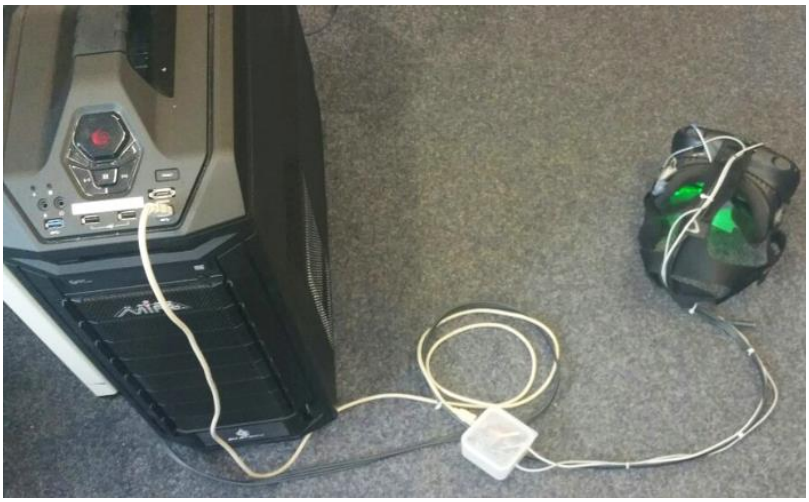


Figure 18 Experiment setup: PC, HTC Vive with an eye blink sensor and an LED turned on

At first, the project was implemented for the Oculus Rift DK2 Head Mounted Display, and the first pre-study was carried out with an Oculus HMD as well. But later it was decided that the HTC Vive should be used instead of the Oculus Rift DK2 (Figure 17) because of better position tracking in space with laser Valve's Lighthouse Position Sensor which can track the user's position in the space of 4.6 x 4.6 meters. The setup includes two wireless handheld SteamVR

Motion Controllers, which are used to answer the questions in the VR scene. The HTC Vive uses two small screens, one for each eye with a resolution of 1.080x1.200mp, and has a wider 110 degree field of view than Oculus Rift DK2 with 100 degrees. The HTC Vive also has lower system requirements: Intel i5 processor, 4GB RAM, while using the Oculus Rift DK2 requires 8GB RAM; moreover, the HTC Vive needs one USB 2.0 port instead of three USB 3.0 and one USB 2.0 ports for the Oculus. Also, the HTC Vive provides better performance and less latency than the Oculus Rift DK2 [23].

4.6 Participants

The user study involved 13 persons. All of them volunteered to be part of this experiment. Since the sensor calibration failed for two participants, the experiment was actually attended by only 11 participants, including PhD, Master and Bachelor students from the Computer Science Department at the University of Hamburg. Some of the Bachelor students got credits for taking part in this experiment. Two of 11 participants were female, 9 male, aged between 21 and 37; the average age was 28. Four of them were wearing glasses, two were wearing contact lenses, and one of the participants had an eye disorder, strabismus, which did not interfere with the experiment. The other 10 users did not have any known vision impairment, such as colour blindness, night blindness, red-green colour weakness or strong eye dominance. None of the participants had a displacement of equilibrium or similar disorders. Nine of 11 participants had already taken part in a study with an HMD, like the Oculus Rift and the HTC Vive, before. All participants have experience with 3D computer games, six of them have much experience, three participants have secondary experience, and two participants have less experience. The participants spend on average 5 hours per week playing computer games. Participant 2 spends the maximum amount of time, 20 hours per week, playing computer games, followed by Participant 1 with 10 hours per week. Participant 8 does not spend any time playing computer games, and the other participants are playing 1 to 5 hours per week. All 11 participants already have experience with 3D stereoscopic displays. The average height of the participants was 1.74m; the tallest participant was 1.85m, and the shortest was 1.61m. All participants were feeling good physically, except Participant 5, who felt tired, and Participant 10, who was not feeling absolutely well physically. See full data in attachment (Personal estimation data).

4.7 Data evaluation method

The data from the experiment with the answers of each user for each redirection technique were written to the .txt files, separately for rotation and translation (Figure 19), in the following form: s0_1_6 = 0.875, where s0 stands for user ID number from 0 to 10; the second

```
Debug.Log ("Left");
                BlinkRotationExperimentTrial
trial = ec.CurrentTrials[ec.CurrentTrialIndex] as
BlinkRotationExperimentTrial;
                trial.result = 0;
                results      +=      "s"      +
ec.participantID + "_" + getAxis(trial.Axis) + "_"
+ getAngle(trial.Degree) + " = " + trial.result +
";\r\n";
```

Figure 19 Part of the Unity3D source code for rotation trial

value 0 to 2 stands for the axes: 0 for the X-axis, 1 for the Y-axis, and 2 for the Z-axis. The third value from 0 to 6 stands for the rotation angle: 0 stands for -15°, 1 for -10°, 2 for -5°, 3 for 0°, 4 for 5°, 5 for 10°, 6 for 15°; or for translation values given in meters: 0 stands for -0.15m, 1 for -0.10m, 2 for -0.5m, 3 for 0m, 4 for 0.5m, 5 for 0.10m, 6 for 0.15m. The last value 0 or 1 stands for the user's answers: 0 for all negative directions (left, down, backward); 1 for all positive directions (right, up, forward). However, the last value can also be a fraction, for example 0.87. As each manipulation was repeated 8 times, an average value from the user's answers was directly calculated.

All this data was imported in Excel (Microsoft Office 2013), sorted with VBA Script (see VBA Excel Macros for data evaluation in attachment) by each axis and by each rotation angle/translation value in meters and compared for each user and generally for all participants.

The data from the Kennedy Simulator Sickness Questionnaire (SSQ) [1] was imported in Excel, and both parts before and after the experiment were at first evaluated separately and then compared with each other for each user.

The data from the questionnaire part, which included personal estimation and additional comments, was evaluated on a question-by-question basis. See complete data in attachment (Personal estimation data).

4.8 Results

This section provides information about the evaluated data received during the user study from the experiment and from the questionnaires which were completed by the participants before and after the experiment. The first two parts describe the results received directly during the experiment and provide participants' answers about the direction of manipulations. The following parts provide evaluation of the questionnaires: personal estimation and the Kennedy Simulator Sickness Questionnaire (SSQ) [1].

4.8.1 Rotation

This part provides an evaluation of the data received from the part of the experiment where participants were rotated around 3 axes. Participants gave the answers about the direction of the current rotation using a SteamVR Controller. This data was sorted by each axis and each angle of rotation, compared for each user and in general for all axes, angles and participants.

The data shown in Table 1 includes the following information: the first column contains participants' ID numbers, values in the second column stand for the X-axis, values in the third column stand for -15 degrees, and values in the fourth column are an average of 8 answers for 8 runs of rotation around the same axis. When a participant chose the positive direction, the value was 1, when the negative direction, the value was 0; all 8 answers for the same axis and the same direction were then added up and divided by 8.

Table 1 Data received for rotation around the X-axis by -15°

User ID	Axis	Value	Answer
s0	0	0	1
s1	0	0	1
s2	0	0	1
s3	0	0	0,75
s4	0	0	0,625
s5	0	0	1
s6	0	0	0
s7	0	0	0,875
s8	0	0	0,5
s9	0	0	1
s10	0	0	1
		Average:	0.795

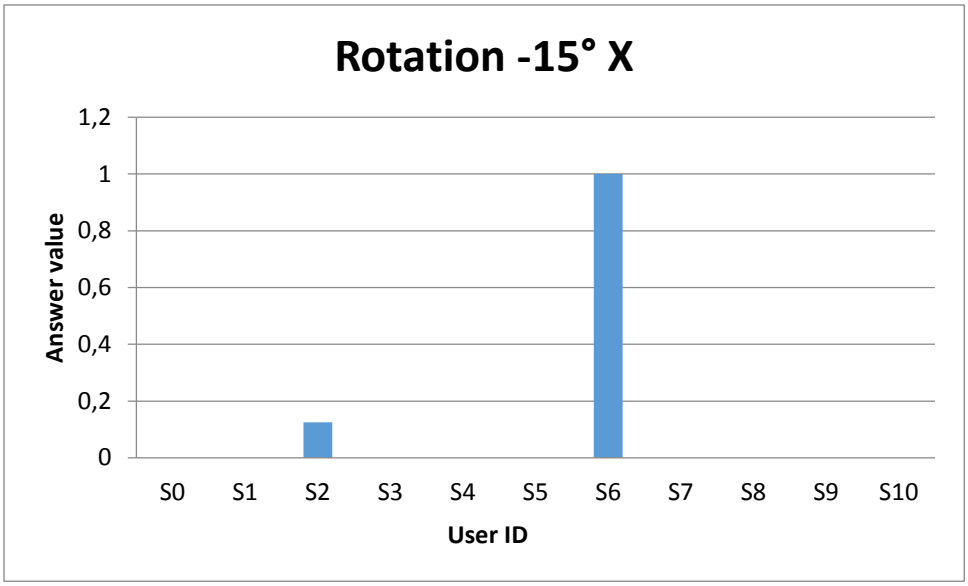


Figure 21 Rotation results -15° X

As we can see from the diagram (Figure 21), most participants (81%) selected the negative direction when they were rotated by -15 degrees around the X-axis. Only 9% of participants selected the wrong direction in 100% of cases. The total average of all answers for the rotation around the X-axis by -15 degrees is 0.102, which means that participants were 90% likely to estimate the manipulation correctly.

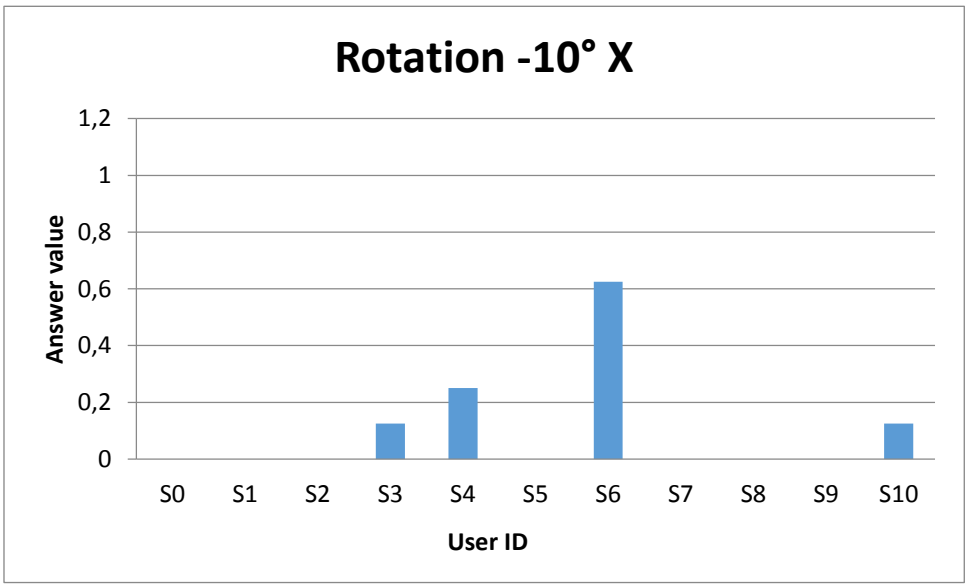


Figure 20 Rotation results -10° X

The next rotation around the X-axis was by -10 degrees. As we can see from the diagram (Figure 20), 63% of participants selected the right direction, 27% chose the correct answer in most cases, and 9% of participants were wrong in 50% of cases. The total average of the

answers for the rotation around the X-axis by -10 degrees is 0.10, which means that participants were with 90% probability able to estimate the manipulations correctly.

The next rotation around the X-axis was by -5 degrees. As we can see from the diagram (Figure 22), 27% of participants chose the correct answer in 100% of cases. 9% of participants chose the right answer in 50% of cases, and 9% of participants believed in most cases that they were rotated in the positive direction. 54% of participants chose the correct answer in almost all cases.

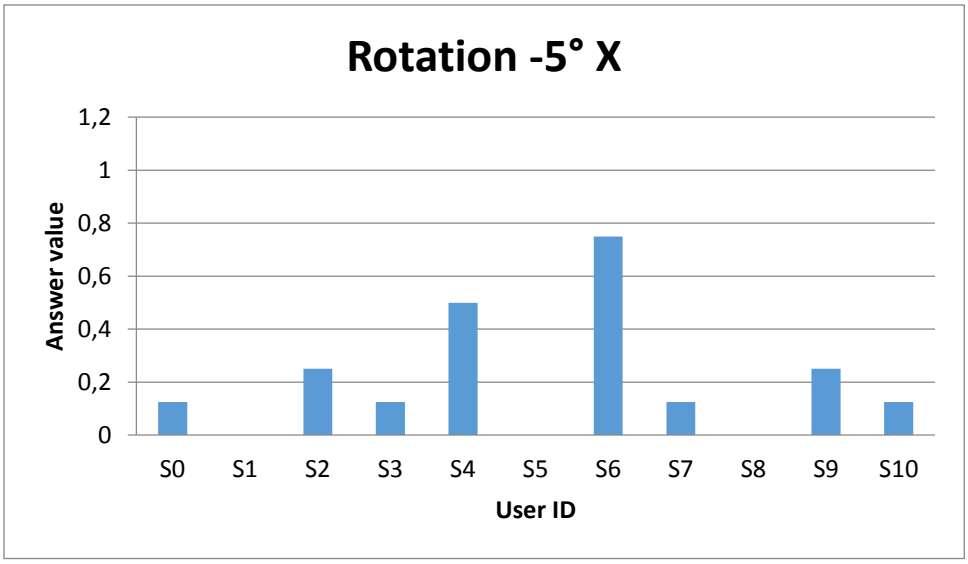


Figure 22 Rotation results -5° X

The total average of the answers for the rotation around the X-axis by -5 degrees is 0.2, which means that participants were able to estimate the manipulations correctly up to 80% in almost all cases.

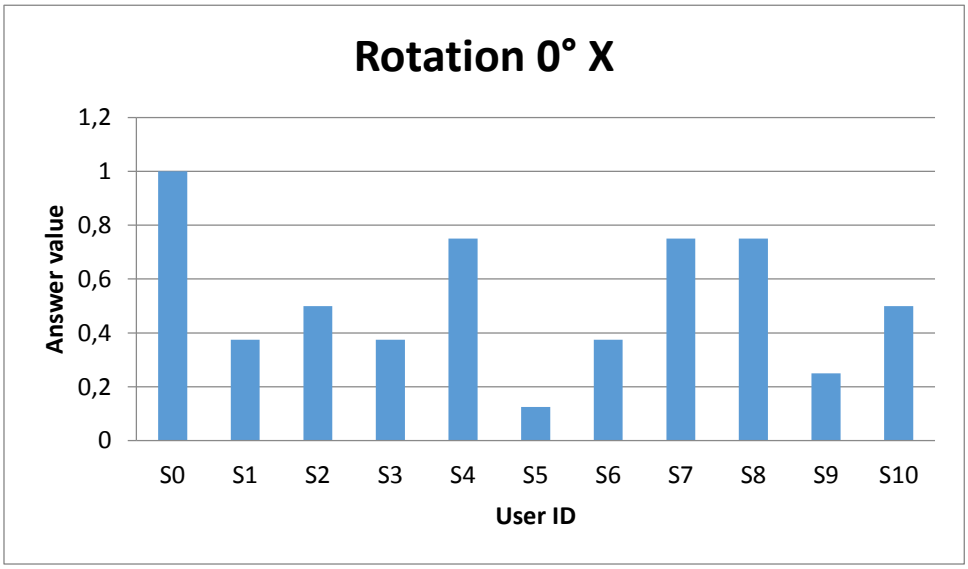


Figure 23 Rotation results 0° X

The next rotation around the X-axis was by 0 degree. As we can see from the diagram (Figure 23), 36% of participants believed that they were rotated in the positive direction, 45% believed that they were rotated in the negative direction, 18% chose the negative direction in 50% of cases and the positive direction in 50% of cases. The total average of the answers for the rotation around the X-axis by 0 degree is 0.52.

The following rotations were in the positive direction, and the correct answer was 1. During rotations by +5 degrees (Figure 24), 90% of participants selected the right answer in almost all cases, and 9% of participants chose the wrong answer in most cases and believed that they were rotated in the negative direction. On average, the answer was equal to 0.8.

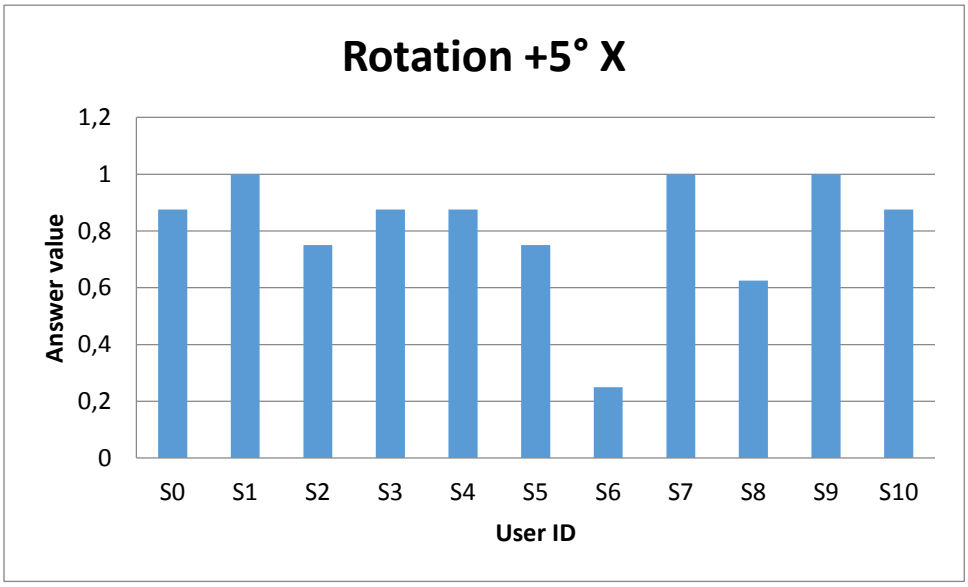


Figure 24 Rotation results +5° X

In the case where the participants were rotated around the X-axis by +10 degree, 90% of participants gave the right answer, and 9% gave the wrong answer in most cases. The average answer was equal to 0.83 which means the correct positive direction. This tendency was further reinforced by the rotations around the X-axis by +15 degrees where only Participant 6 selected the wrong answer in 100% of cases, which is 9% of participants. Another 9% of participants gave the right answer in 50% of cases, and 81% of participants gave the right answer in most cases. The average of all answers is equal to 0.79 which means the positive direction.

Further, the rotations around the Y-axis are described, and the first rotation to be discussed shall be the rotation by -15 degrees (Figure 25), i.e. to the left in the negative direction. In this case, the majority of participants (81%) recognized in which direction they were rotated and gave correct answers in 100% of cases, the average answer was 0.03.

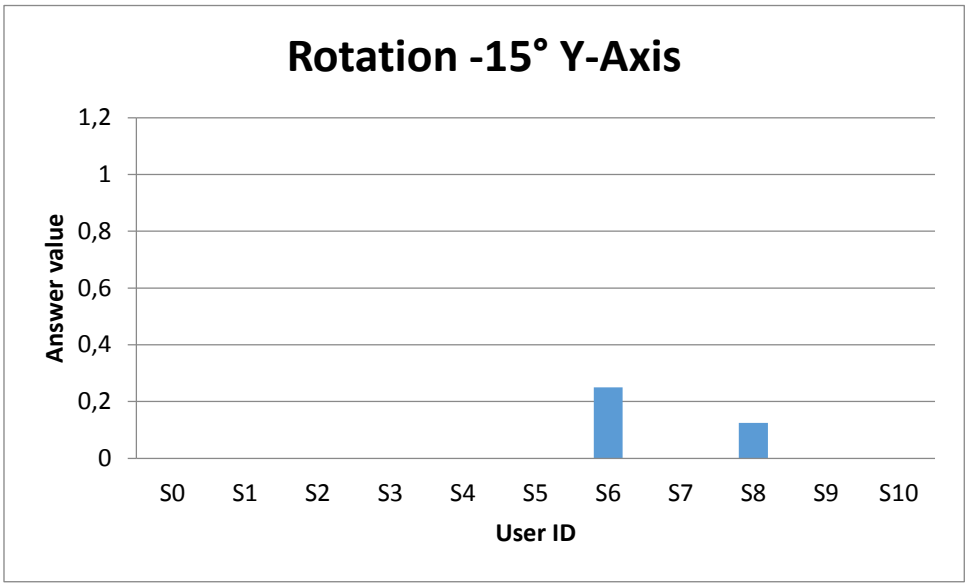


Figure 25 Rotation results -15° Y

Similar results were received from the rotations around the Y-axis by -10 degrees: 63% of participants always chose the right direction, and 36% chose the right answer in most cases. The average answer was 0.08.

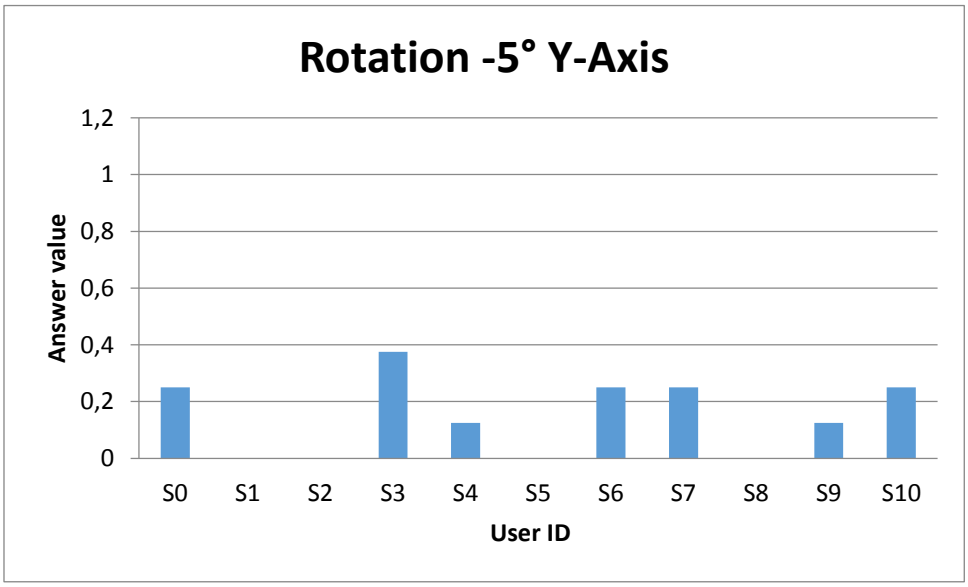


Figure 26 Rotation results -5° Y

During the rotations around the Y-axis by -5 degrees (Figure 26), 63% of participants chose the right answer in almost all, but not all cases, while 36% of participants gave the right answer in 100% of cases. The average answer for this rotation was 0.14.

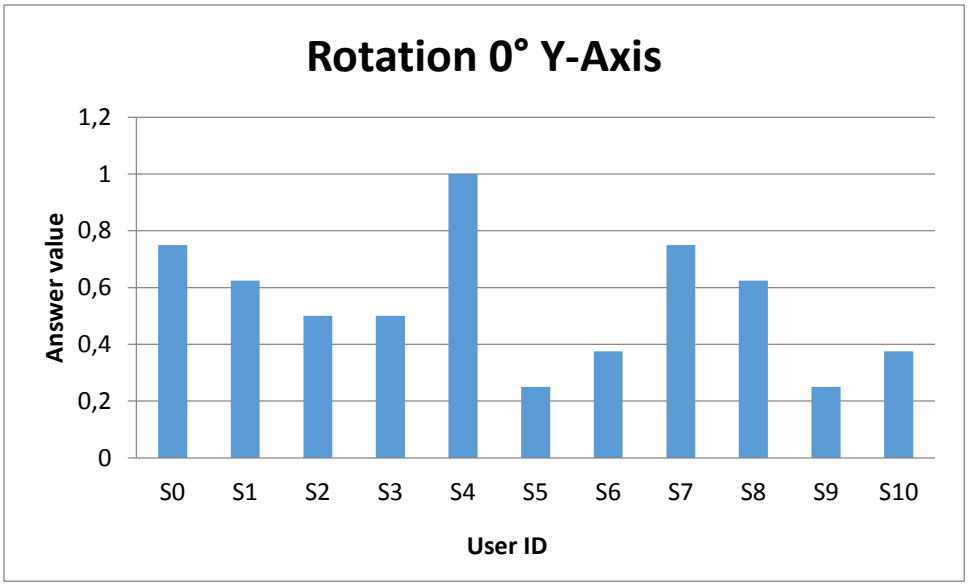


Figure 27 Rotation results 0° Y

During the rotations around the Y-axis by 0 degrees (Figure 27), which means no rotation at all, 45% of participants chose the positive direction, whereas the other 36% chose the negative direction and 27% chose the negative direction in 50% of cases and the positive direction in 50% of cases. On average, the answer was 0.54.

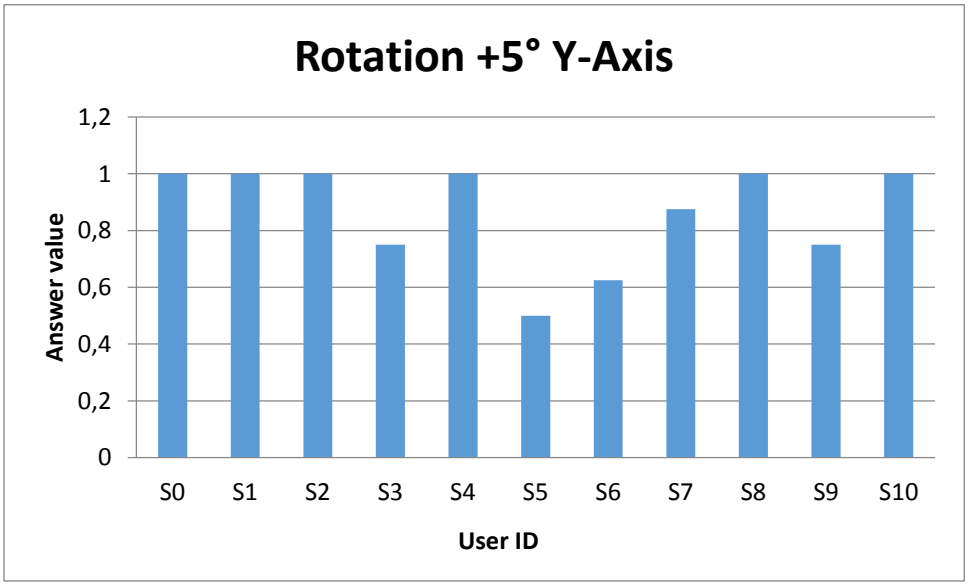


Figure 28 Rotation results +5° Y

The rotations in the positive direction around the Y-axis provided the following data. During the rotations by +5 degrees (Figure 28), 90.9% of participants chose the correct direction in most cases, and the average answer was 0.86. After the rotations by +10 degrees (Figure 29),

also 90.9% of participants chose the correct direction, and the average answer was 0.87. As for the rotation by +15 degrees, 90% of participants chose the right answer in most cases, and the average answer was 0.94. These results show that participants were very sensitive to the rotations around the Y-axis, particularly in the negative direction.

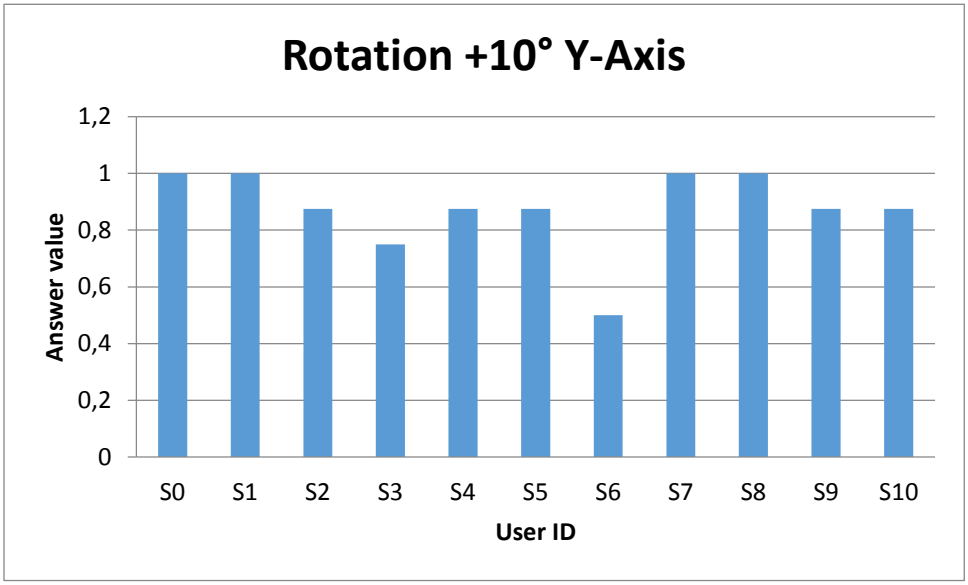


Figure 29 Rotation results +10° Y

The third axis around which participants were rotated was the Z-axis, and the rotations were made to the left or to the right. The results of the rotations around the Z-axis by -15 degrees showed that 81% of participants chose the right direction in most cases, and the average answer was 0.18. When rotated by -10 degrees, 54.5% of participants chose the correct direction in all cases, while 18% chose the right answer in most cases and 27% chose the wrong answer in most cases. The average answer was 0.22. After the rotations by -5 degrees (Figure 31), 90% of participants chose the right answer in most cases, and the average answer was 0.21.

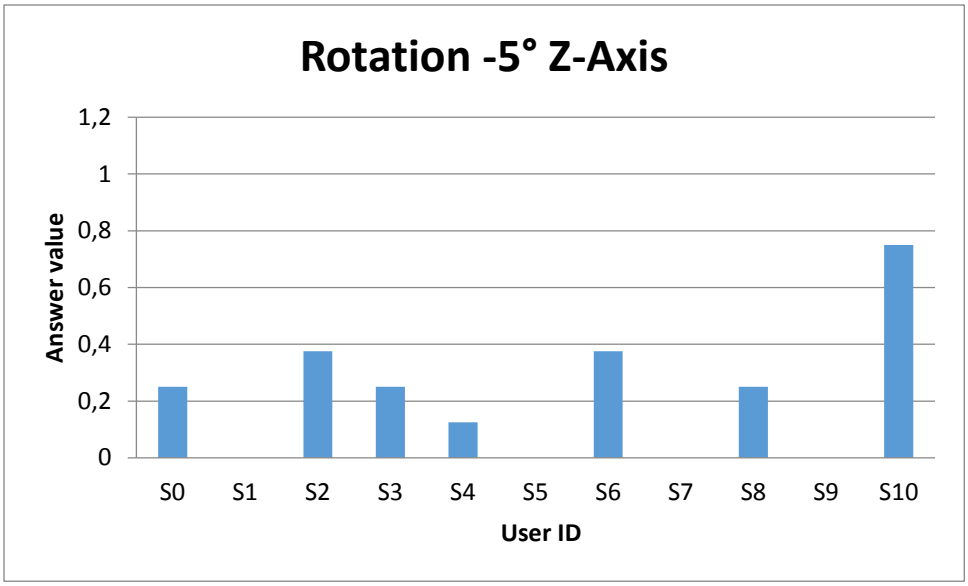


Figure 30 Rotation results -5° Z

As to the rotations around the Z-axis by 0 degrees, the majority of participants (54%) chose the negative direction to the left in most cases, and the average answer was 0.45. When rotated in the positive direction by +5 degrees, 81% of participants gave the correct answer, and the average answer was 0.75. The rotations by +10 degrees led to the result that 81% of participants chose the direction correctly, and the average answer was 0.77. After the rotations by +15 degrees, 100% of participants gave the correct answer in most cases, while the average answer was 0.85. These results showed that participants were sensitive to the rotations around the Z-axis as well, but slightly less than around the Y-axis in both positive and negative directions.

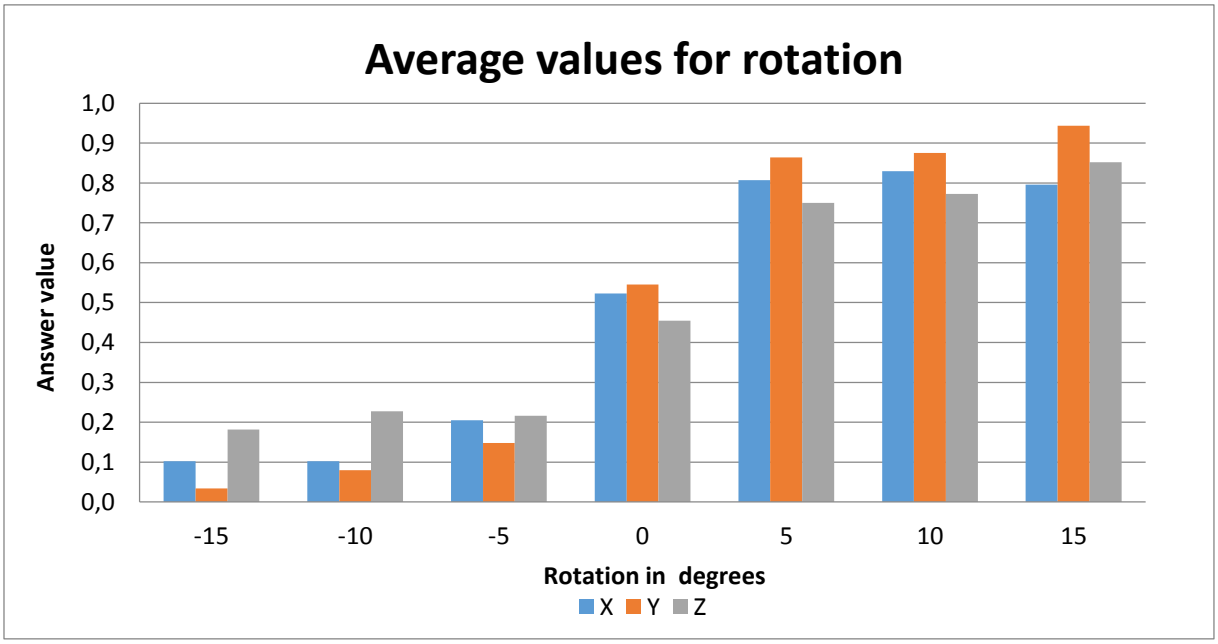


Figure 31 Average values for rotations

The general average for all rotations by all angles around all axes is shown in Figure 31. The bar chart in Figure 32 shows the percentage of the participants' sensitivity to the rotations in general. It can be seen from the chart that participants were more sensitive to the rotations around the Y-axis and less sensitive to the rotations around the Z-axis, while also being more sensitive to the rotations in the negative direction.

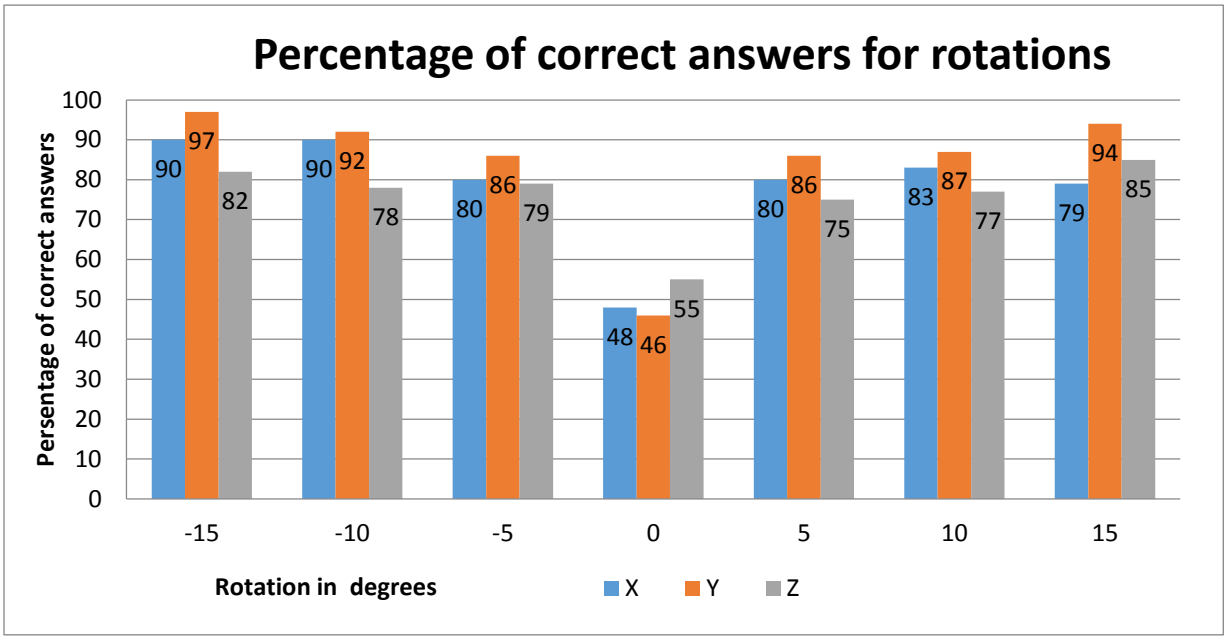


Figure 32 Percentage of correct answers for rotations

Further data from this part of the experiment, which is not covered here, is included in the attachment (see Rotation results).

4.8.2 Translation

This part provides data received from the part of the experiment where participants were translated along three axes. Participants gave the answers about the direction of the current translation using a SteamVR Controller. This data was sorted by each axis and each value of translation, similar to the data from the rotation part.

The answers of the participants were recorded as 0 for the negative direction and 1 for the positive direction. Each translation by each value along each axis was repeated eight times for each participant, and then all these answers were added up and divided by 8 to calculate the average for each user, value and axis.

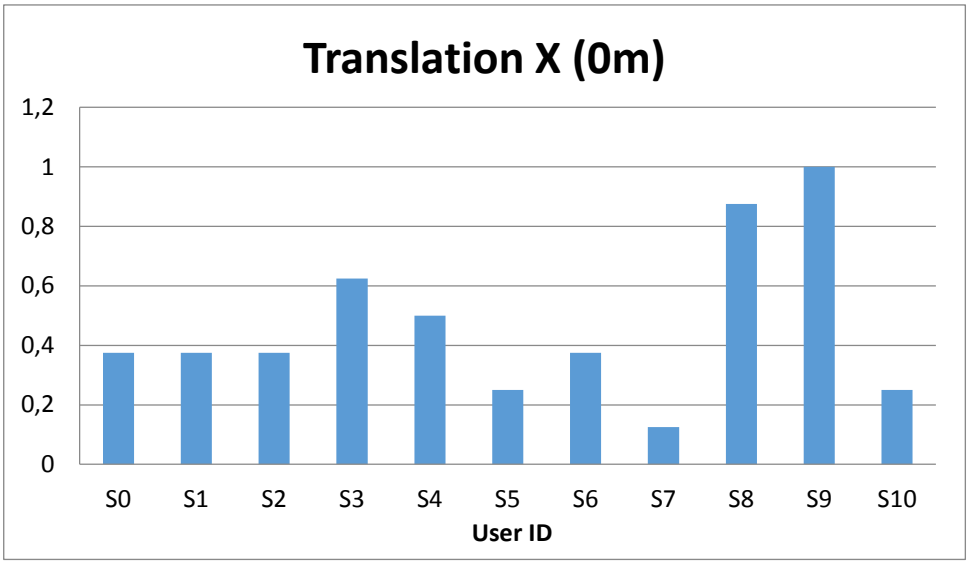


Figure 34 Translation along the X-axis by 0 meters

Unfortunately, a failure occurred during the experiment, and the received data contained only two values (0 and 3) for distance reference instead of 7 values from 0 to 6, where 0 is equal to -0.15m, 1 is equal to -0.10m, 2 is equal to -0.5m, 3 is equal to 0m, 4 is equal to +0.5m, 5 is equal to +0.10m, and 6 is +0.15m. After analysing the data, it was found that the value 3 was equal to 0m as expected, but 0 was equal to other distance references: -0.15m, -0.10m, -0.5m, +0.5m, +0.10m, +0.15m. As a result, the data received from the translation experiment part was analysed for three axes and all users, but only for two distances: for zero and for other six values. Therefore, it makes sense to compare the data by axes.

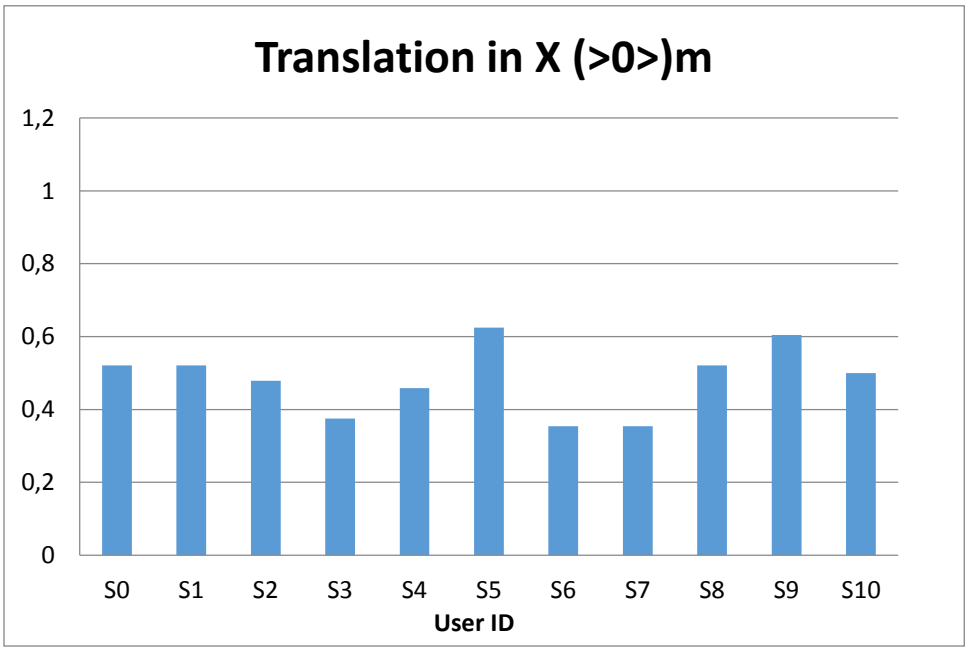


Figure 33 Translation along the X-axis over (>0>) meters

The translations along the X-axis to the right or to the left by 0m (Figure 34) showed that participants (63%) tended to choose the negative direction to the left, and the average answer was 0.46. During the translations over bigger distances along the X-axis (Figure 33) to the right or to the left, the average answer value was 0.48, which means approximately equal distribution of answers in favour of both directions.

The translations up and down along the Y-axis by 0m showed that 72% of participants were more likely to choose the negative direction, i.e. down, and the average answer was 0.39. The answers for the translations over all other distances were divided almost equally, while the average answer was 0.48.

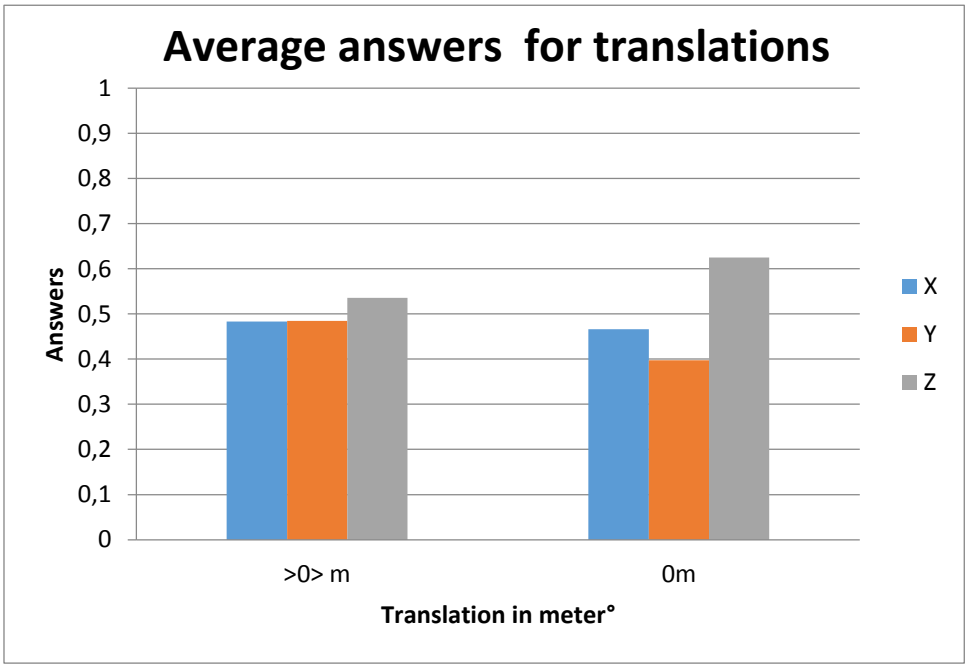


Figure 35 General average for translations

The translations forward and backward along the Z-axis by 0m showed that 72% of participants tended to choose the positive direction, i.e. forward, and the average answer was 0.62. As for the translations over all other distances, the answers were divided almost equally with an average value of 0.53; only 18% frequently selected the positive direction.

The general average for all translations along all axes is shown in Figure 35. This bar chart shows that participants in general were more likely to choose the positive direction, i.e. forward, when they were moved 0m along the Z-axis, and the negative direction, i.e. down, when they were moved 0m along the Y-axis. For all other directions and values, the

answers were divided equally with a slight difference in the Z-axis, where the tendency was towards the positive (forward) direction, as in the case with 0m.

Further data from this part of the experiment, which is not covered here, is included in the attachment (see Translation results).

4.8.3 Personal estimation

After the experiment, participants were asked to answer 10 questions about their personal estimation of the experiment, related to the immersion into the virtual scene and directly aimed at the main task of the experiment. A number of statements had to be evaluated on a scale from 1 to 5, where 1 was the minimum point and 5 was the maximum point. For other questions, detailed answers were required. Full data from this part is provided in the attachment (see Personal estimation data).

The first question in this section was: “How sure are you that you always chose the correct answer?” The participants had to evaluate, on a 5-point scale, the correctness of their own answers about the rotation and translation direction that they gave during the experiment. No one rated the correctness of their answers at 5 or 1, and all values were between 2 and 3. To be more specific, 27% of participants gave a rating of 4, while 54.5% of participants gave a rating of 3 and another 18% of participants rated the correctness of their answers at 2. Based on this data, it can be concluded that most participants were 62% sure that they had given right answers, except Participant 5 and Participant 8.

The next question was directly aimed at the main task of the experiment and was about the cognitive strategy used by the participants to detect rotations and translations. This question required a detailed answer. All participants pointed out that they saw some evidences of their reorientation. The majority of participants (63%) wrote that they were concentrated on some specific points, lines and edges in the scene or the initial position of the objects, which helped them to detect in which direction they were rotated or translated. Participant 6 “tried to build a new impression for each trial”, while Participant 2 used the question text that appeared after the manipulation to detect the reorientation direction. Participant 10 was focusing on the general room geometry: “If the room moves in one direction, choose the other one.”

However, this strategy was useful not for all directions of manipulations. Participant 1 wrote: “Looking straight ahead at some lines, if they rotate or move it is easy to detect if I was rotated. Translating/Rotating up and down was harder to detect.”

An interesting and detailed assessment was given by Participant 9 whose main strategy to detect the translations was to compare images before and after manipulations. To detect the translations to the left and to the right, this participant used the windows in the virtual scene, and to detect the up and down movements, the participant tried to focus on the position of the image centre. It was difficult, but still possible for the participant to detect the translation forward and backward if attention was focused on the image perspective. As for the rotations, this participant used the difference between real orientation of the head and the presented view and between the sense of balance and the presented view. The rotation around the Y-axis was also pointed out: “This was very hard to recognize, often I was lost and didn't know which orientation is real.”

Based on the information above, it can be concluded that the main tools to detect the reorientations were the features in the presented virtual indoor environment, like the legs of the table or window edges, and the displayed question text that unfortunately was also rotated/translated with the participant’s field of view. It is interesting to see that some participants were able to evaluate their own feelings, like head position and sense of balance, in order to detect some manipulations.

The next question (No 2) deals with the immersion into the virtual scene: “Please rate your sense of being in the virtual environment.” The participants had to evaluate their own sense of “being there” on a 5-point scale, where 5 represents their normal experience of being in a place, which means the maximum degree of immersion. No one rated the sense of being there at minimum; Participants 1 and 2 rated their immersion at maximum value 5, while 54.5% of participants gave a rating of 4, another 18% of participants gave a rating of 3, and Participant 6 rated it at 2. This data indicates a high level (76%) of participants’ immersion into the virtual scene.

The next question (No 3) referred to the immersion as well. The participants were asked to evaluate on a 5-point scale if there were times during the experiment when the virtual environment felt like reality for them. Only Participant 1 rated this statement at the highest

level of 5 just as the previous question, while 36% of participants gave the same rating of 4 and another 36% rated it at 3. Interestingly, Participant 2 also gave a rating of 3, whereas the previous question regarding the sense of presence he rated at the maximum level; the same refers to Participant 0. In total, the virtual environment was the reality for participants during the experiment for 64%, thus the grade of immersion falls down by 10% compared to the previous question.

The following question (No 4) was also related to the immersion. The participants were asked to evaluate the virtual scene on a scale from 1 to 5. They could choose between the virtual scene appearing as a simple image to them (point 1) or more like a place they physically visited (point 5). 27% of participants rated the virtual scene at 5, while 45% of participants rated it at 4, 18% at 3, and 9% at 2. All in all, the majority of participants evaluated the scene as a place that they visited, followed by the evaluation as an image that they have seen. The immersion level increased up to 78%.

The following two questions were aimed at awareness of staying in the virtual world. In the first question (No 5), participants were asked to rate the sense of being in the virtual environment or elsewhere on a 5-point scale. No one gave the maximum rating, 72% of participants gave a rating of 4, 18% rated it at 2 and 9% at 3, which means that participants had a sense of being elsewhere at 70% and were not aware that they were actually in the virtual environment. In the second question (No 6) concerning the awareness of being in the virtual environment, participants were asked to rate how often they thought during the experiment that they were really standing in the virtual environment. The point scale for this question was 1 to 5, where 1 means strongly disagree with the statement and 5 means strongly agree. Only 18% of participants did not agree with the statement and gave a rating of 1, while 9% of participants totally agreed with the statement, 54.5% of participants gave a rating of 4, 9% rated it at 3 and another 9% at 2. Overall, participants were 65% sure that they were actually in the virtual environment; however, this result contradicts with the previous one because in this case the higher was the number, the less was the immersion.

The last of immersion-related questions (No 7) was about memorizing, imagination and visual memory of the virtual environment. The participants were asked to evaluate if the virtual environment could be remembered and imagined just like any other real places the

participants had been to during the day. The point scale for this question was 1 to 5, where 1 means strongly disagree with the statement and 5 means strongly agree. No one gave a rating of 1 or 2, while 18% of participants rated the statement at the highest level of 5, 54.5% of participants rated it at 4 and 27% at 3. In total, the virtual environment was memorized and remembered 78% as good as the real places where participants had been the same day.

To calculate the total level of immersion, questions No 2, 3, 4, 5 and 7 were considered, and the total score for the immersion reached 73%.

The following question refers to the attention during the experiment. The participants were asked to evaluate their level of attention on a 5-point scale. All participants reported a high level of attention: 27% of participants rated their attention at 5, 63.6% of participants at 4, and 9% of participants at 3. The total score for the attention was evaluated at 83%, which means that participants were very concentrated during the experiment.

Next, participants were asked about their opinion if the experiment took too long. The point scale for this question was 1 to 5, where 1 means strongly disagree with the statement and 5 means strongly agree. No one gave a rating of 1, 18% of participants rated this statement at 5, 18% of participants at 4, 45% of participants at 3, and 18% of participants at 2. The total score for the duration of the experiment was 67%, which indicated that participants felt that the experiment took too much time and was too lengthy.

In the last part of the questionnaire, participants were given an opportunity to write some additional comments about the experiment. This information is also useful for general evaluation of the experiment; unfortunately, not all of the participants provided feedback.

As an additional comment, participant 1 wrote about the personal feeling that is was hard to stand the whole time. Participant 5 wrote about the seen image and the noticeability of reorientations: it was easier to recognize rotations because a new segment of the space could be seen after rotations which was not the case with translations.

Participant 9 summed up the results of the experiment as follows: “the success rate of the sensor is improved when staring at the "right" position. With perfect timing when the changes in the scene occurred exactly during the eyes shut period, was really difficult to recognize the changes in the environment.”

4.8.4 Motion sickness evaluation

Before and after the experiment, participants filled in the Kennedy Simulator Sickness Questionnaire (SSQ) [1] to measure the VR induced side effects and symptoms, which lead to three general factors: nausea, oculomotor distress and disorientation. This information is important for the experiment because even when participants did not notice the reorientation manipulations, their brain detected discrepancies between the actual movements, the sense of balance and the virtual visual input. These discrepancies cause discomfort, such as nausea and vertigo, which are the evidence that the manipulations were actually noticed.

The Kennedy Simulator Sickness Questionnaire contains 16 statements describing possible side effects of visual simulators which have to be rated on a 4-point scale: none, slight, moderate, severe. The questionnaire was presented to the participants in two languages: German and English (see Simulator Sickness Questionnaire in attachment). The data was collected before and after the experiment to compare the initial and final physical condition of the participants.

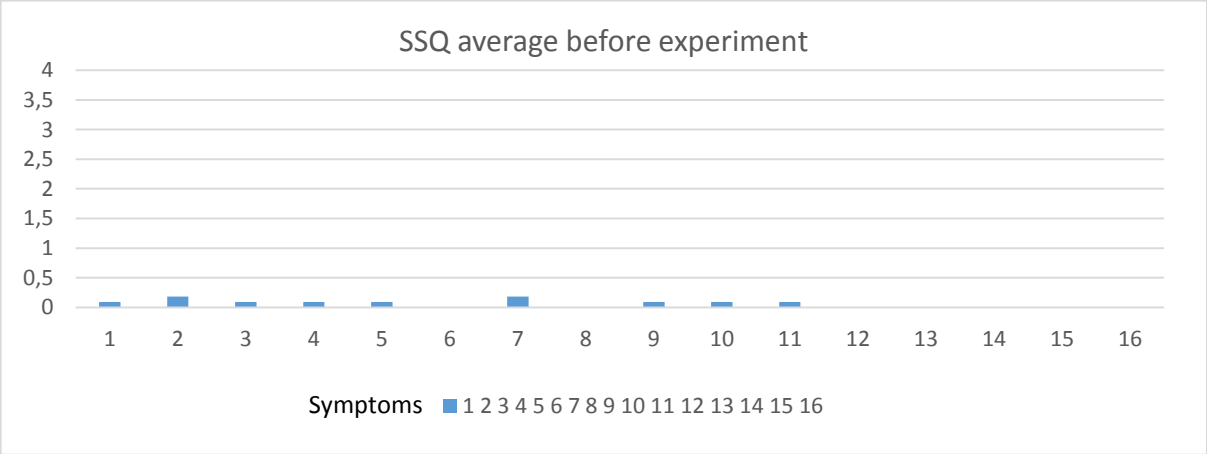


Figure 36 Kennedy SSQ results before the experiment

To evaluate the SSQ results (see Personal estimation data in attachment), the answers and symptoms were coded in numbers: 0 for none, 1 for slight, 2 for moderate and 3 for severe (Table 2), and then converted into a bar chart to visualise the data (Figure 36, Figure 37).

At the beginning of the experiment (Table 2, Figure 36), the majority of participants did not have any symptoms from the SSQ, except for Participant 10 who reported having some symptoms, like headache, general discomfort, sweating and others. Participant 5 was also tired.

Table 2 Kennedy SSQ before the experiment

Kennedy SSQ before											
Symptoms/ User ID	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1 General discomfort	0	0	0	0	0	0	0	0	0	0	1
2 Fatigue	0	0	0	0	0	1	0	0	0	0	1
3 Headache	0	0	0	0	0	0	0	0	0	0	1
4 Eyestrain	0	0	0	0	0	0	0	0	0	0	1
5 Difficulty focusing	0	0	0	0	0	0	0	0	0	0	1
6 Increased salivation	0	0	0	0	0	0	0	0	0	0	0
7 Sweating	0	0	0	0	0	0	0	0	0	0	2
8 Nausea	0	0	0	0	0	0	0	0	0	0	0
9 Difficulty concentrating	0	0	0	0	0	0	0	0	0	0	1
10 Fullness of head	0	0	0	0	0	0	0	0	0	0	1
11 Blurred vision	0	0	0	0	0	0	0	0	0	0	1
12 Dizzy (eyes open)	0	0	0	0	0	0	0	0	0	0	0
13 Dizzy (eyes closed)	0	0	0	0	0	0	0	0	0	0	0
14 Vertigo	0	0	0	0	0	0	0	0	0	0	0
15 Stomach awareness	0	0	0	0	0	0	0	0	0	0	0
16 Burping	0	0	0	0	0	0	0	0	0	0	0

After the experiment (Table 3, Figure 37), the majority of participants reported having some of the symptoms in varying degrees. Only 36% of participants did not show any symptoms. The worst results of all were shown by Participant 7 who had all the symptoms, some of which were strongly expressed. Participants 6 and 8 were also affected by most of the side effects.

Table 3 Kennedy SSQ after the experiment

Kennedy SSQ after											
Symptoms/ User ID	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1 General discomfort	0	1	0	0	0	1	1	2	1	0	1
2 Fatigue	0	2	1	0	0	1	1	3	1	0	2
3 Headache	0	0	0	0	0	1	1	2	0	0	2
4 Eyestrain	0	1	1	0	0	0	2	2	0	0	2
5 Difficulty focusing	0	0	0	0	0	0	1	2	0	0	1
6 Increased salivation	0	0	0	0	0	0	1	2	0	0	0
7 Sweating	0	2	0	0	0	0	1	2	1	1	1
8 Nausea	0	0	1	0	0	0	1	1	1	0	0
9 Difficulty concentrating	0	0	0	0	0	0	1	3	0	0	0
10 Fullness of head	0	0	0	0	0	0	1	3	1	0	0
11 Blurred vision	0	0	0	0	0	0	1	2	0	0	0
12 Dizzy (eyes open)	0	0	0	0	0	0	1	2	1	1	0
13 Dizzy (eyes closed)	0	0	0	0	0	0	1	2	1	1	0
14 Vertigo	0	0	0	0	0	0	1	1	0	1	0
15 Stomach awareness	0	0	0	0	0	0	1	1	0	0	0
16 Burping	0	0	0	0	0	0	1	1	0	0	0

Finally, all users' symptoms were combined into two groups to obtain general information about the side effects of the experiment: nausea and oculomotor distress, according to the Validation of the French-Canadian version of the SSQ developed by the UQO Cyberpsychology Lab [24]. Symptoms 1 + 6 + 7 + 8 + 12 + 13 + 14 + 15 + 16 fall into the category of nausea, and symptoms 2 + 3 + 4 + 5 + 9 + 10 + 11 into the category of oculomotor distress. First, the average value was calculated for each symptom, and then the values for particular symptoms were added up. As a result, the following values were obtained: for nausea 0.364 in the beginning and 3.909 in the end; for oculomotor distress 0.636 before the experiment and 7.454 after the experiment.

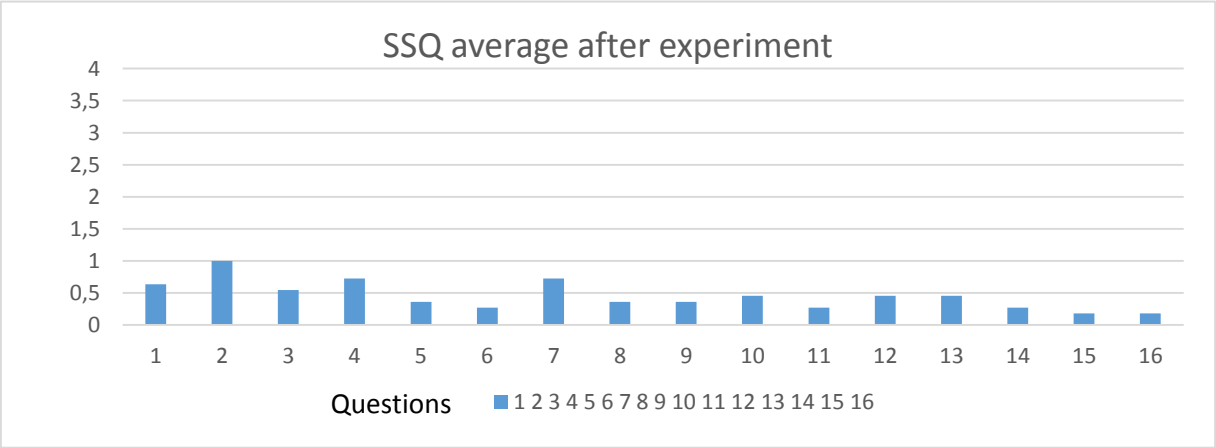


Figure 37 Kennedy SSQ results after the experiment

These results indicate that the reorientation manipulations were detected by the brain areas responsible for the sense of balance because of the discrepancy between the actual movements and the virtual visual input.

5 Discussion

The main purpose of this study was to prove the hypothesis that the Redirected Walking in VR is possible during eye blinking. The findings suggest that participants were able to recognize the manipulations in most cases, but the results were affected by limitations of the experiment.

The first possible explanation for the received results is the static position of the participants as they were standing in one place during the whole experiment. The virtual scene and the objects were static too because a stable body position was expected to ensure more precise measurement of the user's sensitivity to the manipulations than a dynamic one. The reason for this is that attention to small changes is lower during body motion. Due to this static position, the users were able to compare two seen images of the virtual scene before and after the manipulations. They were looking at the same part of the scene all the time, so they focused their attention on some particular points or edges, and this helped them to detect the changes.

As indicated by the participants in the personal estimation part concerning the cognitive strategy used to detect rotations and translations, the main tools to detect the reorientations were the features, such as furniture, windows and other objects in the virtual indoor environment, which was an exact replication of the laboratory where the experiment was conducted. Some features of the experiment design, like the question with the answers, which appeared in front of the participants, also moved during the manipulations and helped the participants to recognize the correct direction of the reorientation. However, this was only possible due to the fact that the participants' position was stable; therefore, they were able to fix the gaze on a single point to define the rotation or to see changes of perspective in the VR scene picture and generally to compare two VR scene pictures after and before the reorientation.

In addition, this static position caused discrepancies between the actual movements and the virtual visual input, which were detected by the brain areas responsible for the sense of balance and resulted in increased nausea and oculomotor distress according to the Kennedy SSQ. One participant directly pointed out the discrepancy between the head position and the

sense of balance. Based on this, it can be assumed that a more appropriate method would have been to add the reorientations to the actual movements, so that the participants were rotated by 30 degrees in the real world and by 40 degrees in the virtual scene.

As an additional comment, Participant 1 wrote about his own feeling that is was hard to stand the whole time. Besides, most participants indicated that the experiment was too long. The experiment included all in all 356 runs of translations and rotations, each followed by a question which appeared 1.5 seconds after the manipulation and had to be answered. It took the participants a few seconds to answer the questions in VR; as a result, one part alone, with rotations or translations, took 20 to 30 minutes. In addition, the participants needed a break between these two parts because the HTC Vive HMD has no cooling and gets warm after a certain period. Again, there was an additional heat source, the LED, so these conditions were unpleasant for extended periods. In addition to that, the user study included another part with questionnaires, which the participants had to fill in, and this task also took up to 30 minutes.

Basically, to decrease the experiment time and improve the experiment, only two axes could be involved into the reorientation; the rotations around the X- and Z-axes, for example, were not necessary in the case of RW, and the translations along the Y-axis could also be omitted.

Another time-consuming and difficult process during the experiment was the sensor calibration. First, the position of the LDR had to be adjusted and the threshold related to the closed eyelid had to be determined for each participant because of different head size and facial morphology. It was difficult to correctly position and secure the LDR inside the HMD due to hindered access and limited space. Consequently, two participants could not continue to participate in the experiment. Also, not each blinking movement was detected by the sensor, so participants sometimes had to blink more than once, and this took plenty of time and was unpleasant because the normal human blinking frequency is approximately once per 4-6 seconds.

Sometimes the blinking was detected not during the closing, but during the opening of the eye, so that the participant was rotated or translated while the eyes were already open and the redirection that occurred at this moment was seen directly. The evidence of this problem is the answer of Participant 1 from the personal estimation part about the cognitive strategy

to recognize the rotations and translations. This participant saw the rotations and translations while opening the eyes, but the particular reason for this disadvantage is not clear. Maybe it was just delay in performance, maybe the blink was detected when the eyes started to open, or the chosen threshold that was supposed to differentiate the two eye states was not completely correct and was reached by opening the eyes instead of closing. Maybe it was influenced by particular features of the eyes of this certain participant. Likewise, Participant 9 noticed that the exact sensor position influenced the success of the experiment.

Additionally, the experiment was influenced by the eye blink sensor function and the algorithm time and force. As some participants pointed out, with the correct sensor position and the perfect timing of the eye shutting and reorientation it was difficult to recognize the changes in the virtual environment.

6 Conclusion

The main objective of the bachelor thesis was to prove the hypothesis that the Redirected Walking in VR is possible during eye blinking. In particular, it was expected that participants would not be able to notice manipulations by changing their position by rotation and translation if these manipulations occurred during eye blinking. This hypothesis was based on the theory about neural mechanism of blink suppression [12] [13] and the theory that the human brain believes that the world is stable during the eye movements [9].

To implement this project, it was necessary to detect the moment when a person is blinking, namely when the eyelid starts to go down or is already closed, to start the algorithm. Several available EOG and video-based eye trackers were investigated with a view to be used for this task. However, due to their complexity or shortcomings: EOG is difficult to procure and to use, and the given Eye Tribe eye tracker [16] was too bulky and unsuitable for use in the HMD, it was decided to solve this problem in a more efficient and cheaper way. Based on the eye characteristics, an eye blink sensor was developed and used for the experiment.

The next step was to design an experiment in the virtual environment during which participants were rotated about or translated along three axes by various angles and over various distances. During the experiment, participants were staying in one place because it was assumed that this position should ensure precise measurement of the user's sensitivity to the manipulations and that attention to small changes is lower during body motion because of human cognitive abilities [6]. The participants were asked directly in the VR environment after each manipulation in which direction they were rotated or translated; this data contains the information about participants' ability to recognize the manipulation. Additionally, the ability to detect the reorientations was proved by the physical side effects, which were evaluated by the Kennedy SSQ questionnaire [1], and the personal estimation questions.

The data received from the answers given directly during the experiment in the VR environment showed that participants in general were sensitive to the rotations, in particular more sensitive to the rotations around the Y-axis, less sensitive to the rotations around the Z-axis, and generally more sensitive to the rotations in the positive direction around all axes. In

the personal estimation part, 62% of participants responded that they were sure that they gave right answers.

When rotated around the X-axis by -15 and -10 degrees, 90% of participants were able to determine the direction of the manipulation correctly. After the rotations by -5, 5, 10 and 15 degrees, 85% of participants, on average, gave correct answers.

When rotated around the Y-axis by ±15 and -10 degrees, over 90% of participants were able to detect the correct direction of the manipulation. After the rotations by ±5 and +10 degrees, the score was over 85%.

When rotated around the Z-axis by ±15 degrees, over 80% of participants were able to determine the correct direction of the manipulation, and during the rotations around the same axis by other angles, the score was over 75%.

During the translation part of the experiment, a failure occurred so that the data was received for only two distances: for translation by 0m and by all others (±0.15, ±0.10, ±0.5). Nevertheless, it was possible to evaluate the data, and it was found that participants in general were more likely to choose the positive direction, i.e. forward, when they were translated along the Z-axis by 0m, and the negative direction, i.e. down, when they were translated along the Y-axis by 0m. For all other directions and values, the answers were divided equally with a slight difference in the Z-axis, where the tendency again was towards the positive direction.

Some of the participants pointed out in the personal estimation part that it was difficult to detect translations up and down (along the Y-axis) and rotations up and down (around the X-axis). In addition, some participants found it difficult to detect translations forward and backward.

The Kennedy SSQ [1] evaluation showed that the physical side effects had increased after the experiment: the nausea value grew from 0.3 to 3.9, and the oculomotor distress value grew from 0.6 to 7.4. This data shows that the brain was able to notice the manipulations because of the discrepancy between the actual movements and the virtual visual input. Additionally, some participants could detect the manipulations based on the difference between the head position and the sense of balance.

The performance of the virtual scene and the HTC Vive HMD was good. According to the personal estimation part, the participants felt immersed in the virtual world. The total score for the immersion calculated from the participants' responses was 73%.

Based on the results listed above, it can be concluded that the initial hypothesis was confirmed only partly, but this can be attributed to the specifics and limitations of the experiment. The data received from the questionnaires and during the experiment clearly showed that not the reorientation itself, but the changes in the seen part of the virtual scene and the discrepancy between the real and virtual movements were recognizable. Unfortunately, sometimes the manipulations were seen directly, as the algorithm started at the wrong moment. However, some participants of the pre-study and the final experiment argued that they did not notice the reorientations when they occurred synchronously with the eye closure.

The above-mentioned facts lead to the conclusion that reorientation during eye blinking is possible with the limitations and findings of this experiment taken into account. First and foremost, perfect synchronization of sensor reaction, eyelid closure and reorientation is needed. Secondly, the changes in the virtual scene have to be introduced exactly during the eyes shut period. Another important factor is the user's body and head motion; one way to hide the discrepancy between the body motion and the sense of balance is to load cognitive abilities and to prevent users from focusing on static scene objects because during the natural locomotion the user's FOV is constantly changing. With all the above factors taken into account, successful redirection is possible.

6.1 Future Work

Based on the findings of this study, it is suggested for future experiments involving eye blink detection that a good eye tracker specially designed to work with the HMD be used, such as the "Pupil" from Pupil Labs [25], which has an extra Monocular Add-on Cup for Oculus Rift DK2. Alternatively, the eye blink sensor developed in this study can be improved by a high-precision positioning system with three degrees of freedom. Yet another way is to use two or more sensors for both eyes to obtain more accurate data; for example, the reorientation algorithm could start, only if both sensors generated the same output.

It is also worth mentioning that people blink infrequently, once per 4–6 seconds, and that is not often enough to perform redirections in small rooms [12].

Another point that should be considered for future experiments is that changes are more easily detected in a small virtual indoor environment than in a virtual outdoor environment, like nature.

It is necessary to involve all possible human perception channels to reduce cognitive abilities and attention to details. Therefore, it can be suggested for future experiments that participants walk and move their body and head freely; this would also make the experiment more interesting and easier for participants. In addition, the scene should contain sounds and some changing details, such as butterflies or sea waves, like in the Tuscany demo scene for Oculus Rift DK2 [19], to capture the users' attention and distract them from small manipulations.

To avoid the discrepancy between real and virtual movement and to decrease the motion sickness effect, the reorientation should be added to the actual movements.

Finally, to save time and improve the efficiency of the RW experiment, the number of rotations and translations could be cut. The experiment conducted in this study showed that it would be enough to perform only the rotations around the Y-axis (left, right) and the translations along the X-axis (left, right) and the Z-axis (forward, backward).

7 References

[1] R. S. Kennedy, N. E. Lane, K. S. Berbaum and M. G. Lilienthal, 1993. [Online]. Available: http://w3.uqo.ca/cyberpsy/docs/qaires/ssq/SSQ_va.pdf. [Accessed July 2016].

[2] S. Razzaque, M. C. Whitton and Z. Kohn, "Redirected walking.," *Proceedings of EUROGRAPHICS.,* pp. 105-106, 09 2001.

[3] S. Razzaque, D. Swapp, M. Slater, M. Whitton and A. Steed, "Eight Eurographics Workshop on Virtual Environments.," *Proceeding EGVE '02.,* pp. 123 - 130, 2002.

[4] F. Steinicke, G. Bruder, J. Jerald, H. Frenz and M. Lappe, "Estimation of detection thresholds for redirected walking techniques. Visualization and Computer Graphics," *IEEE Transactions on,* vol. 16, pp. 17-27, 2010.

[5] C. T. Neth, J. L. Souman, D. Engel, U. Kloos, H. H. Bulthoff and B. J. Mohler, "Velocity-dependent dynamic curvature gain for redirected walking," *Visualization and Computer Graphics, IEEE Transaction,* vol. 18 (7), 2012.

[6] G. Bruder, P. Lubos and S. F., "Cognitive Resource Demands of Redirected Walking.," *In Visualization and Computer Graphics, 2015 IEEE VR,* vol. 21, pp. 539-544, 23 03 2011.

[7] E. A. Suma, S. Clark, D. Krum, S. Finkelstein, M. Bolas and Z. Warte, "Leveraging change blindness for redirection in virtual environments," *In Virtual Reality Conference (VR), 2011 IEEE,* pp. 159-166, 03 2011.

[8] T. Peck, H. Fuchs and M. C. Whitton, “An Evaluation of Navigational Ability Comparing Redirected Free Exploration with Distractors to Walking-in-Place and Joystick Locomotion Interfaces.,” in *Proc. IEEE Virtual Real Conf. 2011 Mar.*, 2011.

[9] B. Bolte and M. Lappe, “Subliminal Reorientation and Repositioning in Immersive Virtual Environments using Saccadic Suppression.,” *Visualization and Computer Graphics.,* 23 03 2015.

[10] “riftinfo.com,” 25 Dezember 2015. [Online]. Available: http://riftinfo.com/oculus-rift-motion-sickness-11-techniques-to-prevent-it. [Accessed 10 Januar 2016].

[11] “www.enzyklo.de,” 2014. [Online]. Available: http://www.enzyklo.de/Begriff/Lidschlag. [Accessed 21 November 2015].

[12] F. H. Adler, “Physiology of the eye, clinical application.,” St. Louis, Mosby, 1953, pp. 8-15.

[13] D. Bristow, J. Haynes, R. Sylvester, C. Frith and G. Rees, “Blinking suppresses the neural response to unchanging retinal stimulation.,” *Current Biology.,* vol. 14, p. 1296–1300, 2005.

[14] C. Sforza, M. Rango, D. Galante, N. Bresolin and V. Ferrario, “Spontaneous blinking in healthy persons: an optoelectronic study of eyelid motion.,” *Ophthalmic Physiol. Opt.,* vol. 28(4), p. 345–353, Jul 2008.

[15] A. Al-Rahayfeh and M. Faezipour, “Enhanced Frame Rate for Real-Time Eye,” *Systems, Applications and Technology Conference (LISAT), IEEE Long Island,* pp. 1 - 6, May 2013.

[16] “http://theeyetribe.com/,” [Online]. Available: www.http://theeyetribe.com/. [Accessed August 2015].

[17] M. Dr. Wittlich, “TROS Laserstrahlung Teil 3: Maßnahmen zum Schutz vor Gefährdungen durch Laserstrahlung.,” in *Technische Regel zur Arbeitsschutzverordnung zu künstlicher optischer Strahlung - TROS Laserstrahlung*, Vols. [Nr. 12-15], GMBl, Ed., 2015, p. 281 .

[18] RS Components , “Data Sheet Light dependent resistors,” RS Components , Corby, Northants, 1997.

[19] “developer3.oculus.com,” [Online]. Available: https://developer3.oculus.com/downloads/. [Accessed February 2016].

[20] “store.steampowered.com,” [Online]. Available: http://store.steampowered.com/about/. [Accessed June 2016].

[21] “unity3d.com,” [Online]. Available: http://unity3d.com/. [Accessed June 2016].

[22] “www.arduino.cc,” [Online]. Available: https://www.arduino.cc/. [Accessed Mai 2016].

[23] “www.virtual-reality-brillen-vergleich.de,” [Online]. Available: http://www.virtual-reality-brillen-vergleich.de/vr-brillen-vergleich/oculus-rift-vs-htc-vive. [Accessed Juli 2016].

[24] S. Bouchard, G. Rouillard and P. Renaud, “Revising the factor structure of the Simulator Sickness Questionnaire,” *Acte de colloque du Annual Review of Cyber Therapy and Telemedicine,* vol. 5, pp. 128-138, 2007.

[25] “https://pupil-labs.com/pupil/,” [Online]. [Accessed 30 Juli 2016].

[26] “learn.adafruit.com,” [Online]. Available: https://learn.adafruit.com/adafruit-neopixel-uberguide/arduino-library. [Accessed July 2016].

[27] “www.arduino.cc,” [Online]. Available: https://www.arduino.cc/en/Guide/HomePage. [Accessed July 2016].

[28] “forum.arduino.cc,” [Online]. Available: http://forum.arduino.cc/index.php?topic=40001.0. [Accessed June 2016].

[29] “http://www.crsltd.com,” [Online]. Available: http://www.crsltd.com/tools-for-vision-science/eye-tracking/bluegain-eog-biosignal-amplifier/. [Accessed August 2015].

[30] “widi-elektronik.de,” [Online]. Available: http://widi-elektronik.de/Widi-Katalog%202012-1012.pdf. [Accessed Juni 2016].

8 Attachment

8.1 Arduino source code

This script contains the code for Arduino Uno to control the Adafruit NeoPixel LED strip RGBW and LDR4 [26] [22] [27]

```
#include <SPI.h>
#include <Adafruit_NeoPixel.h>
#include <avr/power.h>
#include<stdlib.h>

#define PIN    5
#define LEDNUM 3
#define SET_PIXELS_COLOR 0
#define SET_STRIP_COLOR  1
#define UPDATE_STRIP     2

Adafruit_NeoPixel strip = Adafruit_NeoPixel(LEDNUM, PIN, NEO_GRB +
NEO_KHZ800);

unsigned char mode;
unsigned char index;
unsigned char red;
unsigned char green;
unsigned char blue;
int light = 0;
int light_sensitivity = 500;

void setup()
{
  #if defined (__AVR_ATtiny85__)
    if (F_CPU == 16000000) clock_prescale_set(clock_div_1);
  #endif
  strip.begin();

  //strip.setBrightness(255);
  strip.show(); // Initialize all pixels to 'off'

  Serial.begin(57600);


  //Photocell code;
  pinMode(0, INPUT);
  Serial.begin (57600);
}

void loop()
{
  if(Serial.available() >= 5)
  {
    mode = Serial.read();
    index = Serial.read();
    red = Serial.read();
    green = Serial.read();
    blue = Serial.read();

    if(mode == SET_PIXELS_COLOR)
```

```
    {
      setColor(index, red, green, blue);
    }
    else if(mode == SET_STRIP_COLOR)
    {
      for(uint16_t indexLED=0; indexLED<strip.numPixels(); indexLED++)
      {
         setColor(indexLED, red, green, blue);
      }
      updateStrip();
    }
    else if(mode == UPDATE_STRIP)
    {
      updateStrip();
      delay(10);
    }
  }
  //Photocell code;
  light = analogRead(0);
  Serial.write(light);
  //Serial.println(light);
  delay(33);
}

void setColor(unsigned char index, unsigned char red, unsigned char green,
unsigned char blue)
{
      if(index >= strip.numPixels()) return;
    setColor(index, strip.Color(red, green, blue));
}

void setColor(unsigned char index, uint32_t color)
{
      if(index >= strip.numPixels()) return;
    strip.setPixelColor(index, color);
}

void updateStrip()
{
    strip.show();
}
```

8.2 Unity3D source code

8.2.1 Unity3D source code for communication with Arduino

This C# script is part of Unity3D application and is designed to establish connection with Arduino via Communication Port (COM7) to receive the data from the LDR and to control the LED colour and intensity [28].

```
using UnityEngine;
using System;
using System.Collections;
using System.Collections.Generic;
using System.Text;
using System.IO.Ports;
using System.Threading;
```

```
public class LED2 : MonoBehaviour
{
    public static byte SET_PIXELS_COLOR = 0;
    public static byte SET_STRIP_COLOR = 1;
    public static byte UPDATE_STRIP = 2;
    private static byte[] data = new byte[5];
    public float intensityFactor = 0.25f;
    System.Random rnd = new System.Random();

    [SerializeField] //to control the values on GUI;
    TranslationTrainingState translationTraining;
    [SerializeField]
    TranslationTrialState translationTrial;
    [SerializeField]
    RotationTrainingState rotationTraining;
    [SerializeField]
    RotationTrialState rotationTrial;

    [SerializeField]
    String portName = "COM7";
    [SerializeField]
    int baudRate = 57600;
    private SerialPort sp;
    public int ArduinoLight; // output for GUI
    [SerializeField]
    int thresholdBlink = 0;
    [SerializeField]
    int timeout = 1;

    public bool blocked = false;

    void Start()//open connection with COM7 Port;
    {
        sp = new SerialPort(portName, baudRate, Parity.None, 8,
StopBits.One);
        sp.ReadTimeout = 1;
        sp.Open();
    }

    void Update()//read the data from LDR;
      {
        StartCoroutine
        (
            AsynchronousReadFromArduino(processSensorValue, timeout )
        );
    }

    public void processSensorValue(int value)
    {
        try
        {
            //start the reorientation trials;
        if (value == thresholdBlink && blocked == false)
            {
                blocked = true;
                translationTraining.userHasBlinked = true;
                translationTrial.userHasBlinked = true;
                rotationTraining.userHasBlinked = true;
                rotationTrial.userHasBlinked = true;
            }

        }
```

```
        catch (Exception e)
        {
            Debug.Log("Error processing sensor value: " + e.Message);
        }
    }


    public IEnumerator AsynchronousReadFromArduino(Action<int> callback,
float timeout)
    {
        DateTime initialTime = DateTime.Now;
        DateTime nowTime;
        TimeSpan diff = default(TimeSpan);

        do
        {
            // A single read attempt
            try
            {
                ArduinoLight = sp.ReadByte();
            }
            catch (Exception e)
            {
                Debug.Log("Error " + e.Message);
            }

            if (ArduinoLight != 0)
            {
                callback(ArduinoLight);
                yield return null;
            }
            else
                yield return new WaitForSeconds(0.05f);

            nowTime = DateTime.Now;
            diff = nowTime - initialTime;

        } while (diff.Milliseconds < timeout);

        yield return null;
    }

    // Close connection with Arduino;
    public void Close()
    {
        sp.Close();
    }

    public void OnGUI()
    {
        if (Input.GetKeyDown(KeyCode.Space))
        {
            setStripColor
            (
                (byte)(rnd.Next(0, 255) * intensityFactor),
                (byte)(rnd.Next(0, 255) * intensityFactor),
                (byte)(rnd.Next(0, 255) * intensityFactor)
            );
        }
        else if (Input.GetKeyDown(KeyCode.Z))
        {
            setStripColor(0, 0, 0);
```

```
        }

        else if (Input.GetKeyDown(KeyCode.G))
        {
            setStripColor(0, 255, 0);
        }
        else if (Input.GetKeyDown(KeyCode.W))
            {
                  setStripColor(255, 255, 255);
            }

        else if (Input.GetKeyDown(KeyCode.A))
        {
            byte[] colors = new byte[12]
            {
                0, 255, 0, 0,
                1, 0, 255, 0,
                        2, 0, 0, 255
            };
            setPixelsColor(colors);
        }
    }

    private void setPixelsColor(byte[] indexColorValue)
    {
        if (indexColorValue.Length % 4 != 0) return;

        for (int i = 0; i < indexColorValue.Length / 4; i++)
        {
            data[0] = SET_PIXELS_COLOR;
            data[1] = indexColorValue[(4 * i) + 0];
            data[2] = indexColorValue[(4 * i) + 1];
            data[3] = indexColorValue[(4 * i) + 2];
            data[4] = indexColorValue[(4 * i) + 3];

            sp.Write(data, 0, data.Length);

            Debug.Log("PIXELS COLOR >>> " + data[1] + " : " + data[2] + " :
" + data[3] + " : " + data[4]);
        }

        data[0] = UPDATE_STRIP;
        data[1] = data[2] = data[3] = data[4] = 255; // index and RGB values
are discarded
        sp.Write(data, 0, data.Length);
    }

    private void setStripColor(byte red, byte green, byte blue)
    {
        data[0] = SET_STRIP_COLOR;
        data[1] = 255; // index is discarded
        data[2] = red;
        data[3] = green;
        data[4] = blue;

        sp.Write(data, 0, data.Length);
        Debug.Log("STRIP COLOR >>> " + red + " " + green + " " + blue);
    }
}
```

8.2.2 Unity3D source code for rotation

8.2.2.1 Unity3D source code for rotation experiment controller

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
using System;

public class BlinkRotationExperimentController : ExperimentController {

	[SerializeField]
	List<Transform> possibleTransforms;

	private List<Vector3> possibleAxises;
	private List<float> possibleDegrees;
	private int repetitions;

	// Use this for initialization
	void Start () {
		possibleAxises = new List<Vector3>();
		possibleAxises.Add (Vector3.up);
		possibleAxises.Add (Vector3.right);
		possibleAxises.Add (Vector3.forward);

		possibleDegrees = new List<float>();
		possibleDegrees.Add (-15f);
		possibleDegrees.Add (-10f);
		possibleDegrees.Add (-5f);
		possibleDegrees.Add (0f);
		possibleDegrees.Add (5f);
		possibleDegrees.Add (10f);
		possibleDegrees.Add (15f);

		repetitions = 8;

		FillTrials();
	}

	// Update is called once per frame
	protected override void Update()
	{
		base.Update();

	}

	protected override void FillTrials()
	{
		currentTrials = new List<ExperimentTrial>();
	  trainingTrials = new List<ExperimentTrial>();

	  for (int i = 0; i < 10; i++)
	  {
		int chosenTransform = UnityEngine.Random.Range(0,
possibleTransforms.Count - 1);
		int chosenAxis = UnityEngine.Random.Range(0,
possibleAxises.Count - 1);
		int chosenDegree = UnityEngine.Random.Range(0,
possibleDegrees.Count - 1);
		TrainingTrials.Add(new BlinkRotationExperimentTrial(i,
possibleAxises[chosenAxis], possibleDegrees[chosenDegree],
```

```
possibleTransforms[chosenTransform]));
        }

        int trialNumber = 0;

            for(int i = 0; i < possibleAxises.Count; i++)
            {
                  for(int j = 0; j < possibleDegrees.Count; j++)
                  {
                        for(int k = 0; k < repetitions; k++)
                        {
                              int chosenTransform = UnityEngine.Random.Range
(0, possibleTransforms.Count - 1);
                              CurrentTrials.Add(new
BlinkRotationExperimentTrial(trialNumber,                possibleAxises[i],
possibleDegrees[j], possibleTransforms[chosenTransform]));
                              trialNumber++;
                        }
                  }
            }

            ExtensionMethods.Shuffle (CurrentTrials);
      }

}
```

8.2.2.2 Unity3D source code for rotation trial

```
using UnityEngine;
using System.Collections;

public class BlinkRotationExperimentTrial : ExperimentTrial {

      public int result;

      private Vector3 axis;
      private float degree;
      private Transform transform;

      public Vector3 Axis
      {
            get
            {
                  return axis;
            }
      }

      public float Degree
      {
            get
            {
                  return degree;
            }
      }

      public Transform Transform
      {
            get
            {
```

```
                return transform;
            }
        }

        public BlinkRotationExperimentTrial(int trialnum, Vector3 _axis, float
_degree, Transform _transform) : base(trialnum)
        {
            axis = _axis;
            degree = _degree;
            transform = _transform;
        }
}
```

8.2.2.3 Unity3D source code for rotation training state

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;

public class RotationTrainingState : ExperimentState
{
    [SerializeField]
    Transform userPosition;
    [SerializeField]
    Transform userPositionAfterBlink;

    [SerializeField]
    LED2 led;

    [SerializeField]
    Canvas c;
    [SerializeField]
    Text text;
    [SerializeField]
    Text leftButton;
    [SerializeField]
    Text rightButton;

    private bool blinked = false;
    public bool userHasBlinked = false;

    public override ExperimentState HandleInput(ExperimentController ec)
    {
        if    ((Input.GetKeyDown(KeyCode.LeftArrow)    &&    blinked)    ||
(SteamVR_Controller.Input(SteamVR_Controller.GetDeviceIndex(SteamVR_Control
ler.DeviceRelation.FarthestRight)).GetPressDown(SteamVR_Controller.ButtonMa
sk.Grip) && blinked)) // 2AFCT left
        {
            Debug.Log("Left");

            ec.TrainingTrialIndex++;
            blinked = false;
            led.blocked = false;
            // Hide 2AFCT question
            c.gameObject.SetActive(false);
            userPositionAfterBlink.rotation = Quaternion.identity;
        }
        if    ((Input.GetKeyDown(KeyCode.RightArrow)    &&    blinked)    ||
(SteamVR_Controller.Input(SteamVR_Controller.GetDeviceIndex(SteamVR_Control
```

```
ler.DeviceRelation.FarthestRight)).GetPressDown(SteamVR_Controller.ButtonMa
sk.Trigger) && blinked)) // 2AFCT right
        {
            Debug.Log("Right");

            ec.TrainingTrialIndex++;
            blinked = false;
            led.blocked = false;
            // Hide 2AFCT question
            c.gameObject.SetActive(false);
            userPositionAfterBlink.rotation = Quaternion.identity;
        }
        if (ec.TrainingTrialIndex >= ec.TrainingTrials.Count)
        {
            return nextState;
        }
        else
        {
            BlinkRotationExperimentTrial            nextTrial            =
ec.TrainingTrials[ec.TrainingTrialIndex] as BlinkRotationExperimentTrial;
            userPosition.position = nextTrial.Transform.position;
            userPosition.rotation = nextTrial.Transform.rotation;
            return this;
        }
    }

    public override void UpdateState(ExperimentController ec)
    {
        BlinkRotationExperimentTrial                trial                =
ec.TrainingTrials[ec.TrainingTrialIndex] as BlinkRotationExperimentTrial;
        if (trial != null)
        {
            if ((Input.GetKeyDown(KeyCode.Space) && blinked == false) ||
(userHasBlinked && blinked == false)) // Blink
            {
                Debug.Log("Blinked");
                blinked = true;
                userHasBlinked = false;
                userPositionAfterBlink.Rotate(trial.Axis, trial.Degree);
                text.text = "In which direction was your view rotated?";
                if (trial.Axis.Equals(Vector3.right))
                {
                    leftButton.text = "SideButton = left";
                    rightButton.text = "BackButton = right";
                }
                if (trial.Axis.Equals(Vector3.up))
                {
                    leftButton.text = "SideButton = left";
                    rightButton.text = "BackButton = right";
                }
                if (trial.Axis.Equals(Vector3.forward))
                {
                    leftButton.text = "SideButton = down";
                    rightButton.text = "BackButton = up";
                }
                StartCoroutine(ShowGUI());                    // Show 2AFCT
question

            }
        }
        else
        {
```

```
            throw    new    UnityException("couldn't    cast    trial    as
BlinkRotationExperimentTrial");
        }
        c.gameObject.transform.position = Camera.main.transform.position +
Camera.main.transform.forward * 2;
        c.gameObject.transform.rotation = Camera.main.transform.rotation;
    }

    IEnumerator ShowGUI()
    {
        yield return new WaitForSeconds(1);
        c.gameObject.SetActive(true);
    }
}
```

8.2.2.4 Unity3D source code for rotation trial state

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;

public class RotationTrialState : ExperimentState {

    [SerializeField]
    Transform userPosition;
    [SerializeField]
    Transform userPositionAfterBlink;

    [SerializeField]
    LED2 led;

    [SerializeField]
    Canvas c;
    [SerializeField]
    Text text;
    [SerializeField]
    Text leftButton;
    [SerializeField]
    Text rightButton;

    private bool blinked = false;
    public bool userHasBlinked = false;
    private string results = "";

      public override ExperimentState HandleInput(ExperimentController ec)
      {
            if    ((Input.GetKeyDown(KeyCode.LeftArrow)    &&    blinked)    ||
(SteamVR_Controller.Input(SteamVR_Controller.GetDeviceIndex(SteamVR_Control
ler.DeviceRelation.FarthestRight)).GetPressDown(SteamVR_Controller.ButtonMa
sk.Grip) && blinked)) // 2AFCT left
        {
                Debug.Log ("Left");
                BlinkRotationExperimentTrial            trial            =
ec.CurrentTrials[ec.CurrentTrialIndex] as BlinkRotationExperimentTrial;
                trial.result = 0;
                results    +=    "s"    +    ec.participantID    +    "_"    +
getAxis(trial.Axis) + "_" + getAngle(trial.Degree) + " = " + trial.result +
";\r\n";

                ec.CurrentTrialIndex++;
                blinked = false;
            led.blocked = false;
```

```
            // Hide 2AFCT question
            c.gameObject.SetActive(false);
            userPositionAfterBlink.rotation = Quaternion.identity;
        }
            if   ((Input.GetKeyDown(KeyCode.RightArrow)   &&   blinked)   ||
(SteamVR_Controller.Input(SteamVR_Controller.GetDeviceIndex(SteamVR_Control
ler.DeviceRelation.FarthestRight)).GetPressDown(SteamVR_Controller.ButtonMa
sk.Trigger) && blinked)) // 2AFCT right
        {
                Debug.Log ("Right");
                BlinkRotationExperimentTrial            trial            =
ec.CurrentTrials[ec.CurrentTrialIndex] as BlinkRotationExperimentTrial;
                trial.result = 1;
                results    +=    "s"    +    ec.participantID    +    "_"    +
getAxis(trial.Axis) + "_" + getAngle(trial.Degree) + " = " + trial.result +
";\r\n";

                ec.CurrentTrialIndex++;
                blinked = false;
            led.blocked = false;
            // Hide 2AFCT question
            c.gameObject.SetActive(false);
            userPositionAfterBlink.rotation = Quaternion.identity;
        }
            if(ec.CurrentTrialIndex>= ec.CurrentTrials.Count)
            {
                Debug.Log (results);
                string     oldResults     =     System.IO.File.ReadAllText
("D:/langbehn/blink_e1.txt");
                System.IO.File.WriteAllText   ("D:/langbehn/blink_e1.txt",
oldResults + results);
                return nextState;
            }
            else
            {
            BlinkRotationExperimentTrial            nextTrial            =
ec.CurrentTrials[ec.CurrentTrialIndex] as BlinkRotationExperimentTrial;
            userPosition.position = nextTrial.Transform.position;
            userPosition.rotation = nextTrial.Transform.rotation;
            return this;
            }
      }

      public override void UpdateState(ExperimentController ec)
      {
            BlinkRotationExperimentTrial              trial              =
ec.CurrentTrials[ec.CurrentTrialIndex] as BlinkRotationExperimentTrial;
            if(trial != null)
            {
            if  ((Input.GetKeyDown(KeyCode.Space)  &&  blinked  ==  false)  ||
(userHasBlinked && blinked == false)) // Blink
            {
                        Debug.Log ("Blinked");
                        blinked = true;
                userHasBlinked = false;
                userPositionAfterBlink.Rotate(trial.Axis, trial.Degree);
                text.text = "In which direction was your view rotated?";
                if (trial.Axis.Equals(Vector3.right))
                {
                    leftButton.text = "SideButton = left";
                    rightButton.text = "BackButton = right";
                }
```

```
                if (trial.Axis.Equals(Vector3.up))
                {
                    leftButton.text = "SideButton = left";
                    rightButton.text = "BackButton = right";
                }
                if (trial.Axis.Equals(Vector3.forward))
                {
                    leftButton.text = "SideButton = down";
                    rightButton.text = "BackButton = up";
                }
                StartCoroutine(ShowGUI());                    // Show 2AFCT
question

            }
            }
            else
            {
                throw new UnityException("couldn't cast trial as
BlinkRotationExperimentTrial");
            }
        c.gameObject.transform.position = Camera.main.transform.position +
Camera.main.transform.forward * 2;
        c.gameObject.transform.rotation = Camera.main.transform.rotation;
    }

      private int getAxis(Vector3 axis)
      {
            if (axis.Equals(Vector3.right))
                return 0;
            if (axis.Equals(Vector3.up))
                return 1;
            if (axis.Equals(Vector3.forward))
                return 2;
            return 0;
      }

      private int getAngle(float angle)
      {
            if (angle.Equals(-15f))
                return 0;
            if (angle.Equals(-10f))
                return 1;
            if (angle.Equals(-5f))
                return 2;
            if (angle.Equals(0))
                return 3;
            if (angle.Equals(5f))
                return 4;
            if (angle.Equals(10f))
                return 5;
            if (angle.Equals(15f))
                return 6;
            return 0;
      }

    IEnumerator ShowGUI()
    {
        yield return new WaitForSeconds(1);
        c.gameObject.SetActive(true);
    }
}
```

8.2.3 Unity3D source code for translation

8.2.3.1 Unity3D source code for translation experiment controller

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
using System;

public class BlinkTranslationExperimentController : ExperimentController
{

    [SerializeField]
    List<Transform> possibleTransforms;

    private List<Vector3> possibleAxises;
    private List<float> possibleMeters;
    private int repetitions;

    // Use this for initialization
    void Start()
    {
        possibleAxises = new List<Vector3>();
        possibleAxises.Add(Vector3.up);
        possibleAxises.Add(Vector3.right);
        possibleAxises.Add(Vector3.forward);

        possibleMeters = new List<float>();
        possibleMeters.Add(-0.15f);
        possibleMeters.Add(-0.10f);
        possibleMeters.Add(-0.05f);
        possibleMeters.Add(0f);
        possibleMeters.Add(0.05f);
        possibleMeters.Add(0.10f);
        possibleMeters.Add(0.15f);

        repetitions = 8;

        FillTrials();
    }

    // Update is called once per frame
    protected override void Update()
    {
        base.Update();

    }

    protected override void FillTrials()
    {
        currentTrials = new List<ExperimentTrial>();
        trainingTrials = new List<ExperimentTrial>();

        for (int i = 0; i < 10; i++)
        {
            int chosenTransform = UnityEngine.Random.Range(0,
possibleTransforms.Count - 1);
            int chosenAxis = UnityEngine.Random.Range(0,
possibleAxises.Count - 1);
```

```
            int       chosenMeter       =       UnityEngine.Random.Range(0,
possibleMeters.Count - 1);
            TrainingTrials.Add(new       BlinkTranslationExperimentTrial(i,
possibleAxises[chosenAxis],                    possibleMeters[chosenMeter],
possibleTransforms[chosenTransform]));
        }

        int trialNumber = 0;

        for (int i = 0; i < possibleAxises.Count; i++)
        {
            for (int j = 0; j < possibleMeters.Count; j++)
            {
                for (int k = 0; k < repetitions; k++)
                {
                    int   chosenTransform   =   UnityEngine.Random.Range(0,
possibleTransforms.Count - 1);
                    CurrentTrials.Add(new
BlinkTranslationExperimentTrial(trialNumber,             possibleAxises[i],
possibleMeters[j], possibleTransforms[chosenTransform]));
                    trialNumber++;
                }
            }
        }

        ExtensionMethods.Shuffle(CurrentTrials);
    }
}
```

8.2.3.2 Unity3D source code for translation trial

```
using UnityEngine;
using System.Collections;

public class BlinkTranslationExperimentTrial : ExperimentTrial
{

    public int result;

    private Vector3 axis;
    private float meter;
    private Transform transform;

    public Vector3 Axis
    {
        get
        {
            return axis;
        }
    }

    public float Meter
    {
        get
        {
            return meter;
        }
    }
```

```
    public Transform Transform
    {
        get
        {
            return transform;
        }
    }

    public  BlinkTranslationExperimentTrial(int  trialnum,  Vector3  _axis,
float _meter, Transform _transform) : base(trialnum)
      {
        axis = _axis;
        meter = _meter;
        transform = _transform;
    }
}
```

8.2.3.3 Unity3D source code for translation training state

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;

public class TranslationTrainingState : ExperimentState
{
    [SerializeField]
    Transform userPosition;
    [SerializeField]
    Transform userPositionAfterBlink;

    [SerializeField]
    LED2 led;

    [SerializeField]
    Canvas c;
    [SerializeField]
    Text text;
    [SerializeField]
    Text leftButton;
    [SerializeField]
    Text rightButton;

    public bool userHasBlinked = false;
    private bool blinked = false;

    public override ExperimentState HandleInput(ExperimentController ec)
    {
        if    ((Input.GetKeyDown(KeyCode.LeftArrow)    &&    blinked)    ||
(SteamVR_Controller.Input(SteamVR_Controller.GetDeviceIndex(SteamVR_Control
ler.DeviceRelation.FarthestRight)).GetPressDown(SteamVR_Controller.ButtonMa
sk.Grip) && blinked)) // 2AFCT left
        {
            Debug.Log("Left");

            ec.TrainingTrialIndex++;
            blinked = false;
            led.blocked = false;
            // Hide 2AFCT question
            c.gameObject.SetActive(false);
```

```
            userPositionAfterBlink.localPosition = Vector3.zero;
        }
        if    ((Input.GetKeyDown(KeyCode.RightArrow)    &&    blinked)    ||
(SteamVR_Controller.Input(SteamVR_Controller.GetDeviceIndex(SteamVR_Control
ler.DeviceRelation.FarthestRight)).GetPressDown(SteamVR_Controller.ButtonMa
sk.Trigger) && blinked)) // 2AFCT right
        {
            Debug.Log("Right");

            ec.TrainingTrialIndex++;
            blinked = false;
            led.blocked = false;
            // Hide 2AFCT question
            c.gameObject.SetActive(false);
            userPositionAfterBlink.localPosition = Vector3.zero;
        }
        if (ec.TrainingTrialIndex >= ec.TrainingTrials.Count)
        {
            return nextState;
        }
        else
        {
            BlinkTranslationExperimentTrial           nextTrial           =
ec.TrainingTrials[ec.TrainingTrialIndex] as BlinkTranslationExperimentTrial;
            userPosition.position = nextTrial.Transform.position;
            userPosition.rotation = nextTrial.Transform.rotation;
            return this;
        }
    }

    public override void UpdateState(ExperimentController ec)
    {
        BlinkTranslationExperimentTrial               trial               =
ec.TrainingTrials[ec.TrainingTrialIndex] as BlinkTranslationExperimentTrial;
        if (trial != null)
        {
            if  ((Input.GetKeyDown(KeyCode.Space)  &&  blinked  ==  false)  ||
(userHasBlinked && blinked == false)) // Blink
            {
                Debug.Log("Blinked");
                blinked = true;
                userHasBlinked = false;
                userPositionAfterBlink.Translate(trial.Axis * trial.Meter);
                text.text = "In which direction was your view translated?";
                if (trial.Axis.Equals(Vector3.right))
                {
                    leftButton.text = "SideButton = backwards";
                    rightButton.text = "BackButton = forwards";
                }
                if (trial.Axis.Equals(Vector3.up))
                {
                    leftButton.text = "SideButton = down";
                    rightButton.text = "BackButton = up";
                }
                if (trial.Axis.Equals(Vector3.forward))
                {
                    leftButton.text = "SideButton = left";
                    rightButton.text = "BackButton = right";
                }
                StartCoroutine(ShowGUI());                    // Show 2AFCT
question
```

```
            }
        }
        else
        {
            throw new UnityException("couldn't cast trial as
BlinkTranslationExperimentTrial");
        }
        c.gameObject.transform.position = Camera.main.transform.position +
Camera.main.transform.forward * 2;
        c.gameObject.transform.rotation = Camera.main.transform.rotation;
    }

    IEnumerator ShowGUI()
    {
        yield return new WaitForSeconds(1);
        c.gameObject.SetActive(true);
    }
}
```

8.2.3.4 Unity3D source code for translation trial state

```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;

public class TranslationTrialState : ExperimentState
{

    [SerializeField]
    Transform userPosition;
    [SerializeField]
    Transform userPositionAfterBlink;

    [SerializeField]
    LED2 led;

    [SerializeField]
    Canvas c;
    [SerializeField]
    Text text;
    [SerializeField]
    Text leftButton;
    [SerializeField]
    Text rightButton;

    public bool userHasBlinked = false;
    private bool blinked = false;
    private string results = "";

    public override ExperimentState HandleInput(ExperimentController ec)
    {
        if ((Input.GetKeyDown(KeyCode.LeftArrow) && blinked) ||
(SteamVR_Controller.Input(SteamVR_Controller.GetDeviceIndex(SteamVR_Control
ler.DeviceRelation.FarthestRight)).GetPressDown(SteamVR_Controller.ButtonMa
sk.Grip) && blinked)) // 2AFCT left
        {
            Debug.Log("Left");
            BlinkTranslationExperimentTrial trial =
ec.CurrentTrials[ec.CurrentTrialIndex] as BlinkTranslationExperimentTrial;
```

```
            trial.result = 0;
            results += "s" + ec.participantID + "_" + getAxis(trial.Axis) +
"_" + getDistance(trial.Meter) + " = " + trial.result + ";\r\n";

            ec.CurrentTrialIndex++;
            blinked = false;
            led.blocked = false;
            // Hide 2AFCT question
            c.gameObject.SetActive(false);
            userPositionAfterBlink.localPosition = Vector3.zero;
        }
        if    ((Input.GetKeyDown(KeyCode.RightArrow)    &&    blinked)    ||
(SteamVR_Controller.Input(SteamVR_Controller.GetDeviceIndex(SteamVR_Control
ler.DeviceRelation.FarthestRight)).GetPressDown(SteamVR_Controller.ButtonMa
sk.Trigger) && blinked)) // 2AFCT right
        {
            Debug.Log("Right");
            BlinkTranslationExperimentTrial             trial             =
ec.CurrentTrials[ec.CurrentTrialIndex] as BlinkTranslationExperimentTrial;
            trial.result = 1;
            results += "s" + ec.participantID + "_" + getAxis(trial.Axis) +
"_" + getDistance(trial.Meter) + " = " + trial.result + ";\r\n";

            ec.CurrentTrialIndex++;
            blinked = false;
            led.blocked = false;
            // Hide 2AFCT question
            c.gameObject.SetActive(false);
            userPositionAfterBlink.localPosition = Vector3.zero;
        }
        if (ec.CurrentTrialIndex >= ec.CurrentTrials.Count)
        {
            Debug.Log(results);
            string                          oldResults                          =
System.IO.File.ReadAllText("D:/langbehn/blink_e2.txt");
            System.IO.File.WriteAllText("D:/langbehn/blink_e2.txt",
oldResults + results);
            return nextState;
        }
        else
        {
            BlinkTranslationExperimentTrial           nextTrial           =
ec.CurrentTrials[ec.CurrentTrialIndex] as BlinkTranslationExperimentTrial;
            userPosition.position = nextTrial.Transform.position;
            userPosition.rotation = nextTrial.Transform.rotation;
            return this;
        }
    }

    public override void UpdateState(ExperimentController ec)
    {
        BlinkTranslationExperimentTrial                 trial                 =
ec.CurrentTrials[ec.CurrentTrialIndex] as BlinkTranslationExperimentTrial;
        if (trial != null)
        {
            if  ((Input.GetKeyDown(KeyCode.Space)  &&  blinked  ==  false)  ||
(userHasBlinked && blinked == false)) // Blink
            {
                Debug.Log("Blinked");
                blinked = true;
                userHasBlinked = false;
                userPositionAfterBlink.Translate(trial.Axis * trial.Meter);
```

```
                text.text = "In which direction was your view translated?";
                if (trial.Axis.Equals(Vector3.right))
                {
                    leftButton.text = "SideButton = backwards";
                    rightButton.text = "BackButton = forwards";
                }
                if (trial.Axis.Equals(Vector3.up))
                {
                    leftButton.text = "SideButton = down";
                    rightButton.text = "BackButton = up";
                }
                if (trial.Axis.Equals(Vector3.forward))
                {
                    leftButton.text = "SideButton = left";
                    rightButton.text = "BackButton = right";
                }
                StartCoroutine(ShowGUI());                    // Show 2AFCT
question

            }
        }
        else
        {
            throw    new    UnityException("couldn't    cast    trial    as
BlinkTranslationExperimentTrial");
        }
        c.gameObject.transform.position  =  Camera.main.transform.position  +
Camera.main.transform.forward * 2;
        c.gameObject.transform.rotation = Camera.main.transform.rotation;
    }

    private int getAxis(Vector3 axis)
    {
        if (axis.Equals(Vector3.right))
            return 0;
        if (axis.Equals(Vector3.up))
            return 1;
        if (axis.Equals(Vector3.forward))
            return 2;
        return 0;
    }

    private int getDistance(float distance)
    {
        if (distance.Equals(-15f))
            return 0;
        if (distance.Equals(-10f))
            return 1;
        if (distance.Equals(-5f))
            return 2;
        if (distance.Equals(0))
            return 3;
        if (distance.Equals(5f))
            return 4;
        if (distance.Equals(10f))
            return 5;
        if (distance.Equals(15f))
            return 6;
        return 0;
    }

    IEnumerator ShowGUI()
```

```
    {
        yield return new WaitForSeconds(1);
        c.gameObject.SetActive(true);
    }
}
```

8.2.4 Unity3D source code for the feedback on the screen

8.2.4.1 Unity3D source code for experiment launch

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;
using System;

public class IntroState : ExperimentState {

      [SerializeField]
      Canvas c;
      [SerializeField]
      Text text;

      bool next = false;

      public bool Next
      {
            get
            {
                  return next;
            }

            set
            {
                  next = value;
            }
      }

      public override ExperimentState HandleInput(ExperimentController ec)
      {
            if(Next || Input.GetKeyDown(KeyCode.P))
            {
                  c.gameObject.SetActive(false);
            Debug.Log("Training started");
                  return nextState;
            }
            else
            {
                  return this;
            }
      }

      public override void UpdateState(ExperimentController ec)
      {
            c.gameObject.SetActive(true);
            text.text = "Welcome to the experiment. When you click 'next' the
training trials will start.";
        c.gameObject.transform.position = Camera.main.transform.position +
Camera.main.transform.forward * 2;
        c.gameObject.transform.rotation = Camera.main.transform.rotation;
```

```
    }
}
```

8.2.4.2 Unity3D source code between the experiment parts

```
using UnityEngine;
using UnityEngine.UI;
using System.Collections;
using System;

public class BetweenState : ExperimentState
{

    [SerializeField]
    Canvas c;
    [SerializeField]
    Text text;

    bool next = false;

    public bool Next
    {
        get
        {
            return next;
        }

        set
        {
            next = value;
        }
    }

    public override ExperimentState HandleInput(ExperimentController ec)
    {
        if (Next || Input.GetKeyDown(KeyCode.P))
        {
            c.gameObject.SetActive(false);
            Debug.Log("Experiment started");
            return nextState;
        }
        else
        {
            return this;
        }
    }

    public override void UpdateState(ExperimentController ec)
    {
        c.gameObject.SetActive(true);
        text.text = "Training done. Click 'next' to start the experiment";
        c.gameObject.transform.position = Camera.main.transform.position +
Camera.main.transform.forward * 2;
        c.gameObject.transform.rotation = Camera.main.transform.rotation;

    }
}
```

8.2.4.3 Unity3D source code for experiment closing

```
using UnityEngine;
using System.Collections;
using System;
using UnityEngine.UI;

public class OutroState : ExperimentState {

	[SerializeField]
	Canvas c;
	[SerializeField]
	Text text;

	public override ExperimentState HandleInput(ExperimentController ec)
	{
		if(Input.GetKeyDown(KeyCode.Z))
		{
			#if UNITY_EDITOR
			UnityEditor.EditorApplication.isPlaying = false;
			#else
			Application.Quit();
			#endif
		}
		return this;
	}

	public override void UpdateState(ExperimentController ec)
	{
		c.gameObject.SetActive(true);
		text.text = "Thanks for participating!";
	  c.gameObject.transform.position = Camera.main.transform.position +
Camera.main.transform.forward * 2;
		c.gameObject.transform.rotation = Camera.main.transform.rotation;
    }
}
```

8.3 VBA Excel Macros for data evaluation

This is the VBA script to sort the received data in Excel Microsoft Office 2013.

```
Sub sortieren()

Call Modul1.Daten_Sortieren

End Sub

Sub Daten_Sortieren()

Dim i As Integer
Dim B0 As Integer
Dim B1 As Integer
Dim B2 As Integer

B0 = 10
B1 = 10
B2 = 10
```

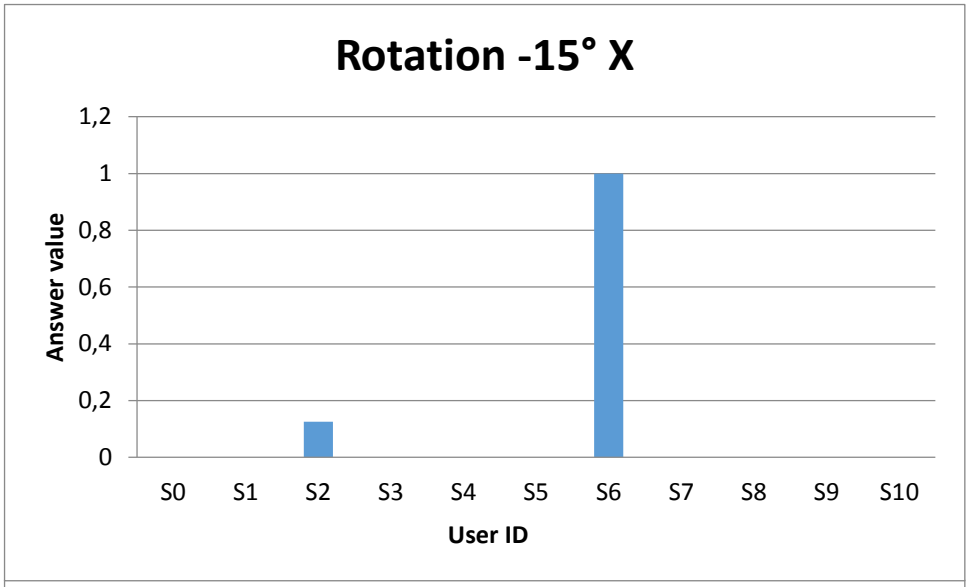
```
For i = 1 To Cells(Rows.Count, 1).End(xlUp).Row

    If Cells(i, 2).Value = 0 Then
        Range(Cells(B0,  10),  Cells(B0,  13)).Value  =  Range(Cells(i,  1),
Cells(i, 4)).Value
        B0 = B0 + 1
    ElseIf Cells(i, 2).Value = 1 Then
        Range(Cells(B1,  16),  Cells(B1,  19)).Value  =  Range(Cells(i,  1),
Cells(i, 4)).Value
        B1 = B1 + 1
    ElseIf Cells(i, 2).Value = 2 Then
        Range(Cells(B2,  22),  Cells(B2,  25)).Value  =  Range(Cells(i,  1),
Cells(i, 4)).Value
        B2 = B2 + 1
    End If
Next
End Sub

Sub Sortieren_Nach_Grad_Z()
    Dim i As Integer
    Dim C_M15 As Integer
    Dim C_M10 As Integer
    Dim C_M5 As Integer
    Dim C_0 As Integer
    Dim C_5 As Integer
    Dim C_10 As Integer
    Dim C_15 As Integer

    C_M15 = 10
    C_M10 = 10
    C_M5 = 10
    C_0 = 10
    C_P5 = 10
    C_P10 = 10
    C_P15 = 10


    For i = 10 To Cells(Rows.Count, 12).End(xlUp).Row

        If Cells(i, 12).Value = 0 Then
            Range(Cells(C_M15, 30), Cells(C_M15, 33)).Value = Range(Cells(i,
10), Cells(i, 13)).Value
            C_M15 = C_M15 + 1
        ElseIf Cells(i, 12).Value = 1 Then
            Range(Cells(C_M10, 36), Cells(C_M10, 39)).Value = Range(Cells(i,
10), Cells(i, 13)).Value
            C_M10 = C_M10 + 1
        ElseIf Cells(i, 12).Value = 2 Then
            Range(Cells(C_M5,  42),  Cells(C_M5,  45)).Value  =  Range(Cells(i,
10), Cells(i, 13)).Value
            C_M5 = C_M5 + 1
        ElseIf Cells(i, 12).Value = 3 Then
            Range(Cells(C_0, 48), Cells(C_0, 51)).Value = Range(Cells(i, 10),
Cells(i, 13)).Value
            C_0 = C_0 + 1
        ElseIf Cells(i, 12).Value = 4 Then
            Range(Cells(C_P5,  54),  Cells(C_P5,  57)).Value  =  Range(Cells(i,
10), Cells(i, 13)).Value
            C_P5 = C_P5 + 1
        ElseIf Cells(i, 12).Value = 5 Then
```

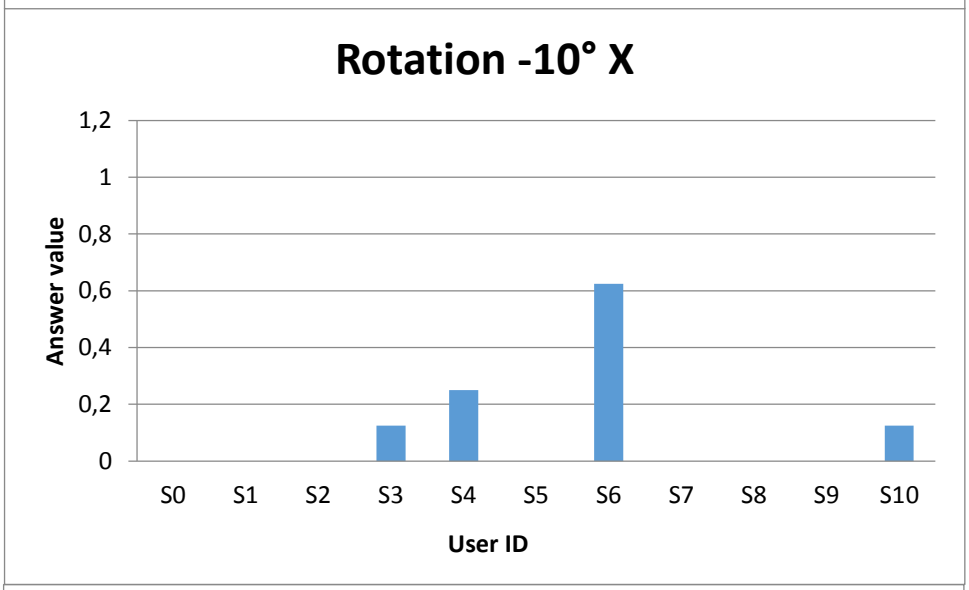
```
            Range(Cells(C_P10, 60), Cells(C_P10, 63)).Value = Range(Cells(i,
10), Cells(i, 13)).Value
            C_P10 = C_P10 + 1
        ElseIf Cells(i, 12).Value = 6 Then
            Range(Cells(C_P15, 66), Cells(C_P15, 69)).Value = Range(Cells(i,
10), Cells(i, 13)).Value
            C_P15 = C_P15 + 1

        End If
    Next
End Sub

Sub Sortieren_Nach_Grad_Y()
    Dim i As Integer
    Dim C_M15 As Integer
    Dim C_M10 As Integer
    Dim C_M5 As Integer
    Dim C_0 As Integer
    Dim C_5 As Integer
    Dim C_10 As Integer
    Dim C_15 As Integer

    C_M15 = 42
    C_M10 = 42
    C_M5 = 42
    C_0 = 42
    C_P5 = 42
    C_P10 = 42
    C_P15 = 42


    For i = 10 To Cells(Rows.Count, 18).End(xlUp).Row

        If Cells(i, 18).Value = 0 Then
            Range(Cells(C_M15, 30), Cells(C_M15, 33)).Value = Range(Cells(i,
16), Cells(i, 19)).Value
            C_M15 = C_M15 + 1
        ElseIf Cells(i, 18).Value = 1 Then
            Range(Cells(C_M10, 36), Cells(C_M10, 39)).Value = Range(Cells(i,
16), Cells(i, 19)).Value
            C_M10 = C_M10 + 1
        ElseIf Cells(i, 18).Value = 2 Then
            Range(Cells(C_M5, 42), Cells(C_M5, 45)).Value = Range(Cells(i,
16), Cells(i, 19)).Value
            C_M5 = C_M5 + 1
        ElseIf Cells(i, 18).Value = 3 Then
            Range(Cells(C_0, 48), Cells(C_0, 51)).Value = Range(Cells(i, 16),
Cells(i, 19)).Value
            C_0 = C_0 + 1
        ElseIf Cells(i, 18).Value = 4 Then
            Range(Cells(C_P5, 54), Cells(C_P5, 57)).Value = Range(Cells(i,
16), Cells(i, 19)).Value
            C_P5 = C_P5 + 1
        ElseIf Cells(i, 18).Value = 5 Then
            Range(Cells(C_P10, 60), Cells(C_P10, 63)).Value = Range(Cells(i,
16), Cells(i, 19)).Value
            C_P10 = C_P10 + 1
        ElseIf Cells(i, 18).Value = 6 Then
            Range(Cells(C_P15, 66), Cells(C_P15, 69)).Value = Range(Cells(i,
16), Cells(i, 19)).Value
            C_P15 = C_P15 + 1
```

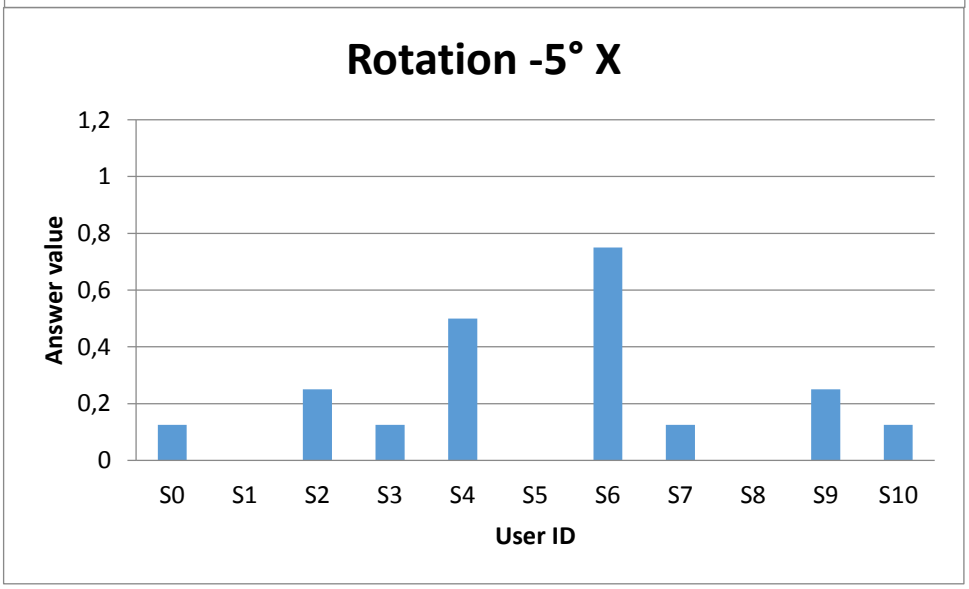
```
        End If
    Next
End Sub

Sub Sortieren_Nach_Grad_X()
    Dim i As Integer
    Dim C_M15 As Integer
    Dim C_M10 As Integer
    Dim C_M5 As Integer
    Dim C_0 As Integer
    Dim C_5 As Integer
    Dim C_10 As Integer
    Dim C_15 As Integer

    C_M15 = 72
    C_M10 = 72
    C_M5 = 72
    C_0 = 72
    C_P5 = 72
    C_P10 = 72
    C_P15 = 72


    For i = 10 To Cells(Rows.Count, 24).End(xlUp).Row

        If Cells(i, 24).Value = 0 Then
            Range(Cells(C_M15, 30), Cells(C_M15, 33)).Value = Range(Cells(i,
22), Cells(i, 25)).Value
            C_M15 = C_M15 + 1
        ElseIf Cells(i, 24).Value = 1 Then
            Range(Cells(C_M10, 36), Cells(C_M10, 39)).Value = Range(Cells(i,
22), Cells(i, 25)).Value
            C_M10 = C_M10 + 1
        ElseIf Cells(i, 24).Value = 2 Then
            Range(Cells(C_M5, 42), Cells(C_M5, 45)).Value = Range(Cells(i,
22), Cells(i, 25)).Value
            C_M5 = C_M5 + 1
        ElseIf Cells(i, 24).Value = 3 Then
            Range(Cells(C_0, 48), Cells(C_0, 51)).Value = Range(Cells(i, 22),
Cells(i, 25)).Value
            C_0 = C_0 + 1
        ElseIf Cells(i, 24).Value = 4 Then
            Range(Cells(C_P5, 54), Cells(C_P5, 57)).Value = Range(Cells(i,
22), Cells(i, 25)).Value
            C_P5 = C_P5 + 1
        ElseIf Cells(i, 24).Value = 5 Then
            Range(Cells(C_P10, 60), Cells(C_P10, 63)).Value = Range(Cells(i,
22), Cells(i, 25)).Value
            C_P10 = C_P10 + 1
        ElseIf Cells(i, 24).Value = 6 Then
            Range(Cells(C_P15, 66), Cells(C_P15, 69)).Value = Range(Cells(i,
22), Cells(i, 25)).Value
            C_P15 = C_P15 + 1

        End If
    Next
End Sub
```

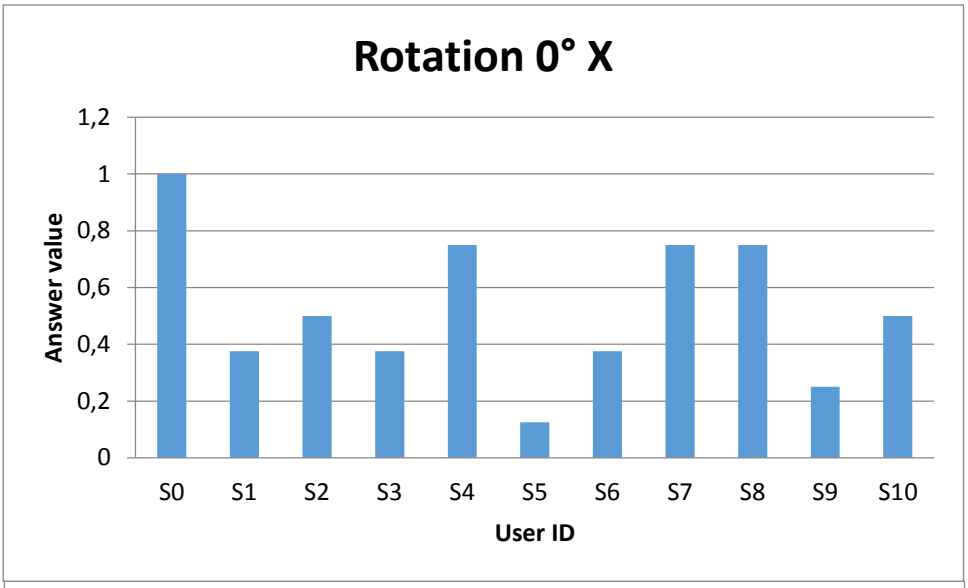
8.4 Experiment results

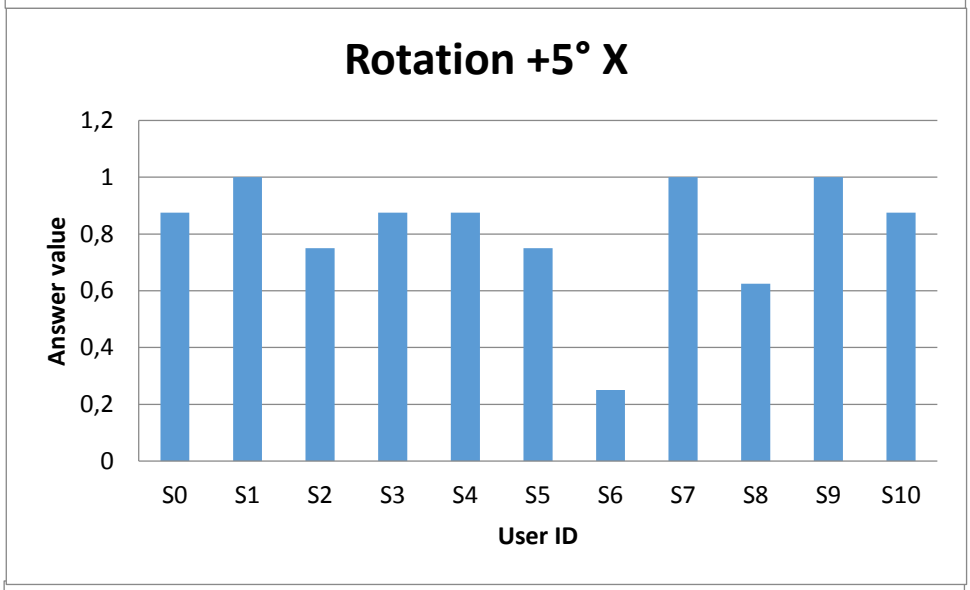
8.4.1 Rotation results

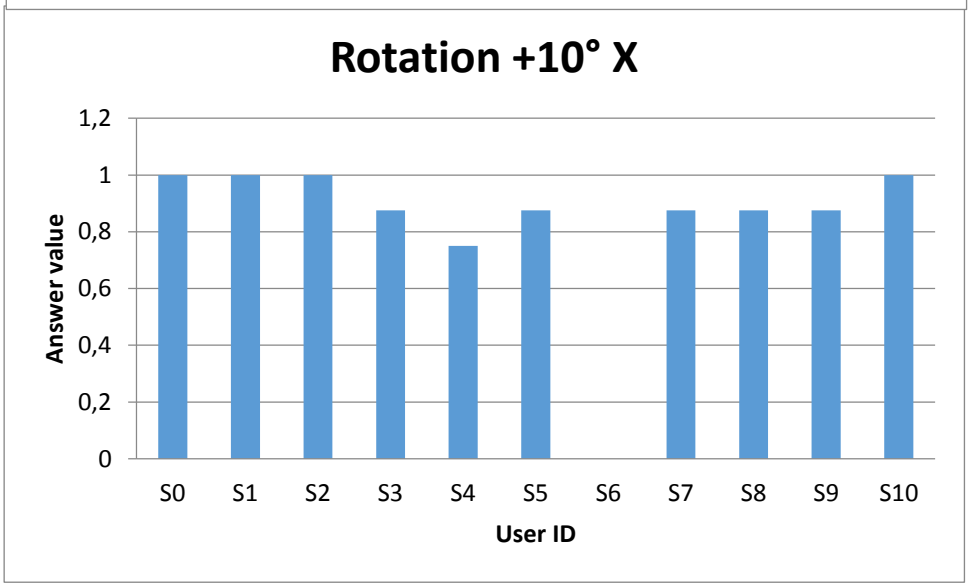


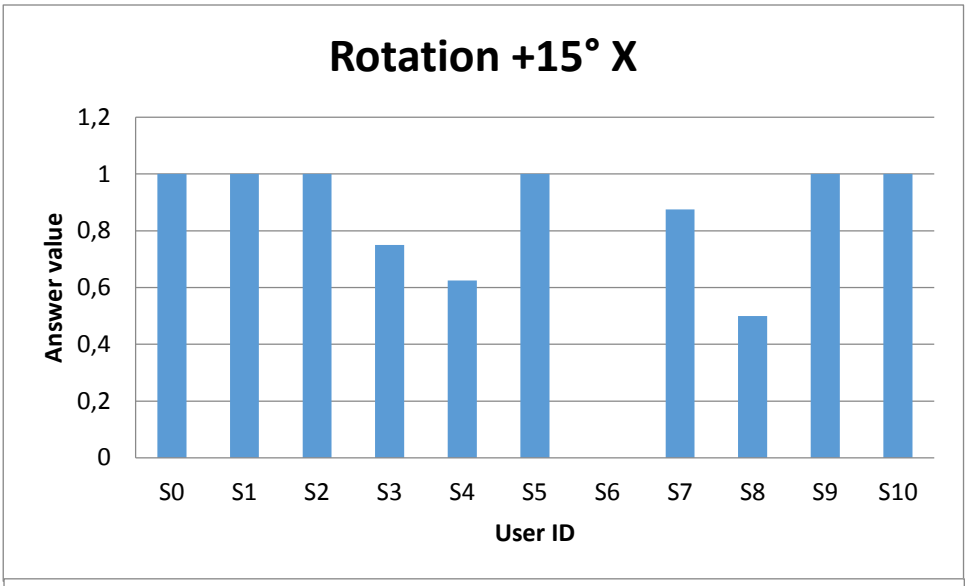


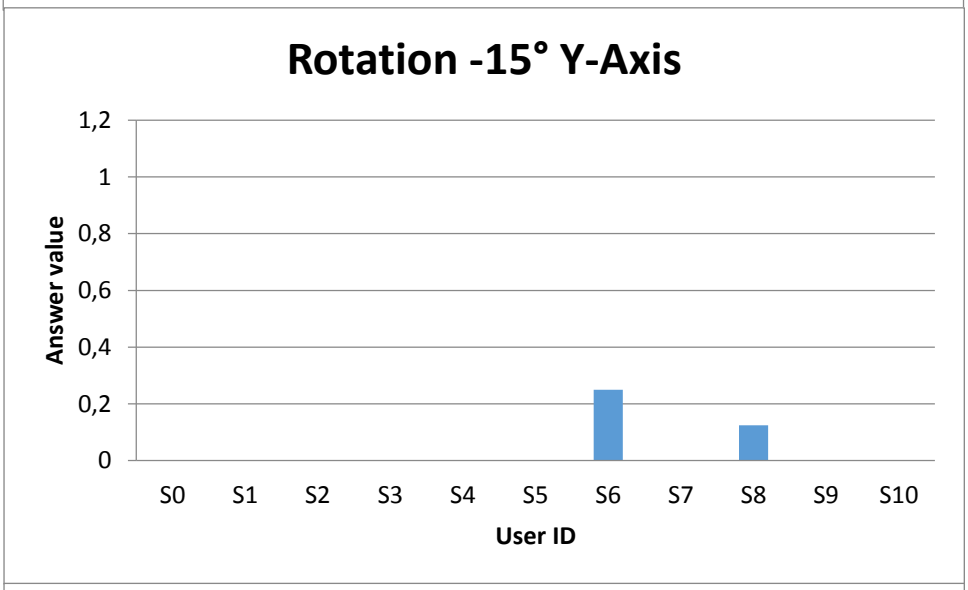


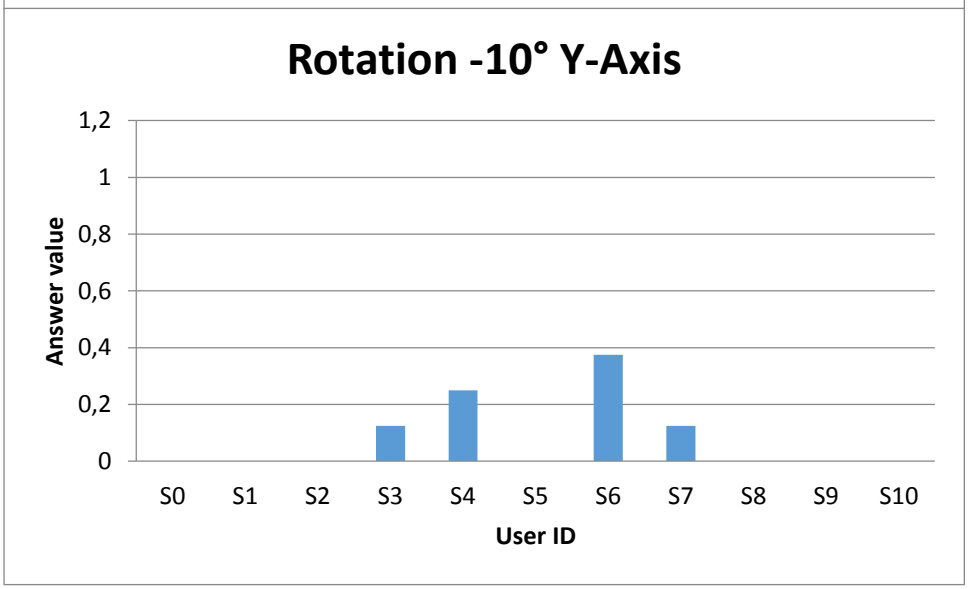

Rotation 0° X
Answer value
1,2
1
0,8
0,6
0,4
0,2
0
S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10
User ID

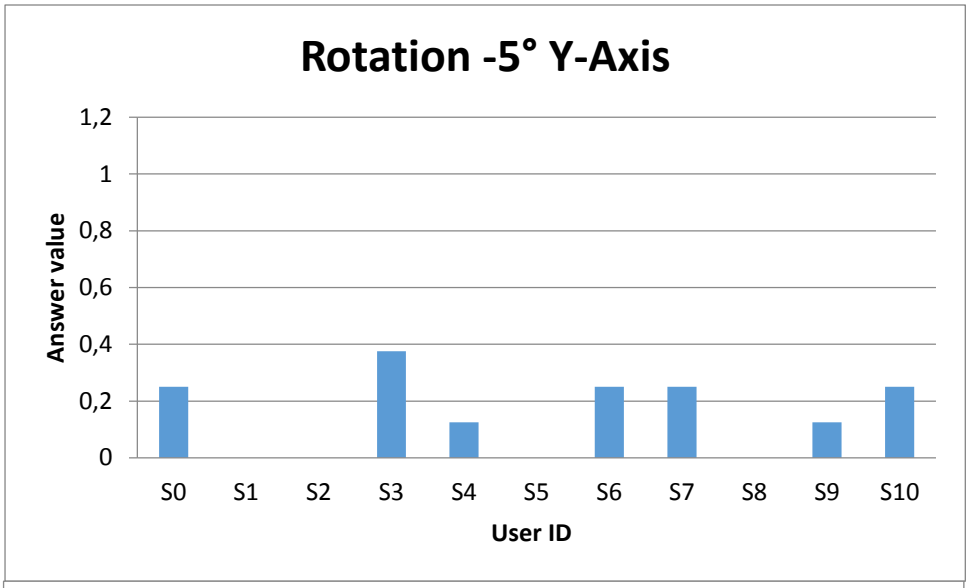

Rotation +5° X
Answer value
1,2
1
0,8
0,6
0,4
0,2
0
S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10
User ID

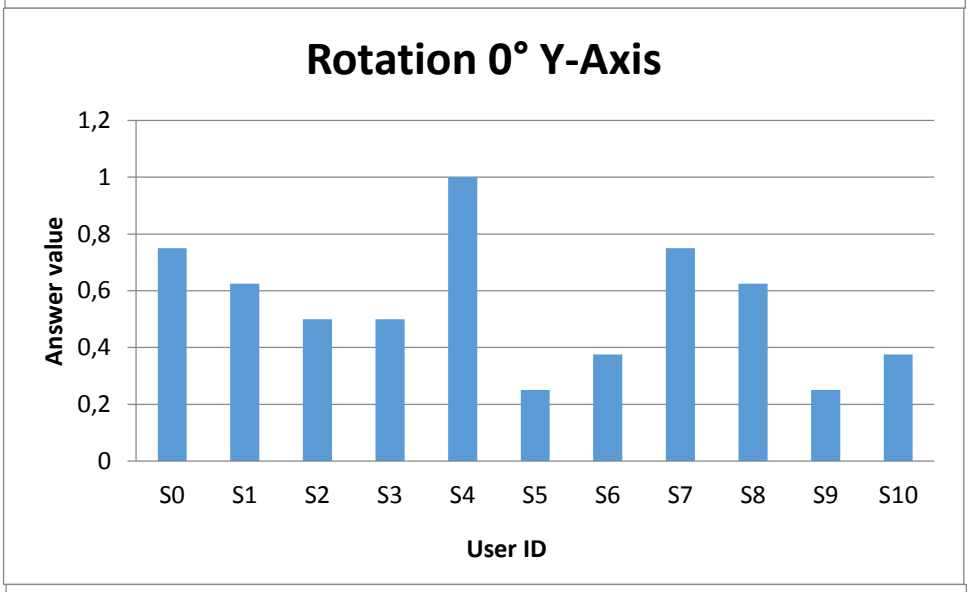

Rotation +10° X
Answer value
1,2
1
0,8
0,6
0,4
0,2
0
S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10
User ID

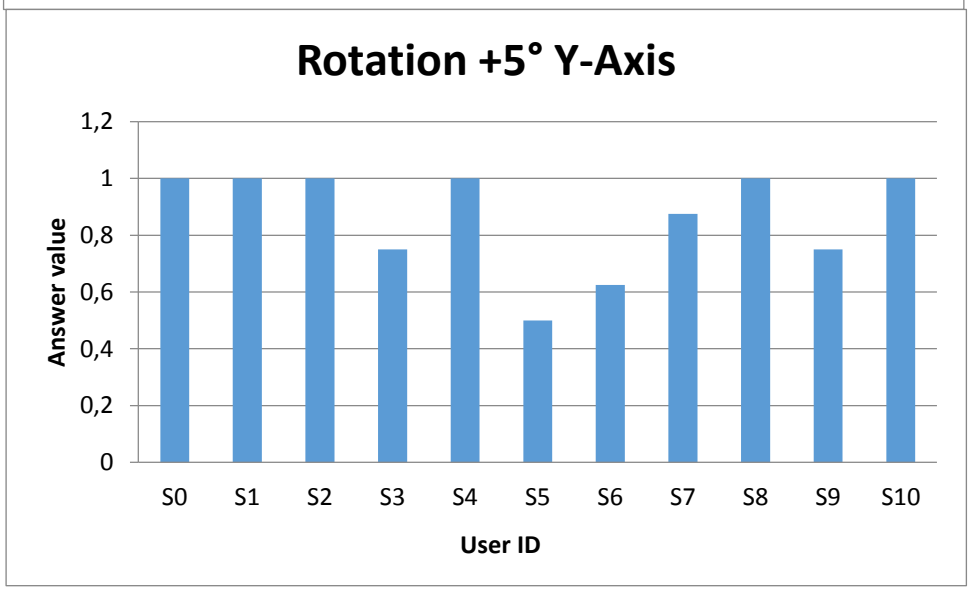

Rotation +15° X
1,2
1
0,8
0,6
0,4
0,2
0
Answer value
S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10
User ID

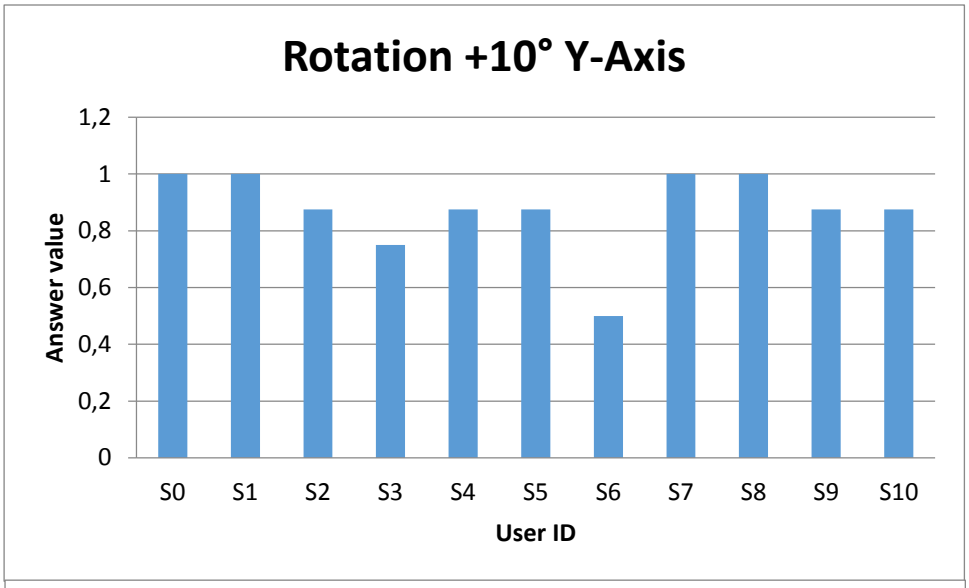

Rotation -15° Y-Axis
1,2
1
0,8
0,6
0,4
0,2
0
Answer value
S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10
User ID

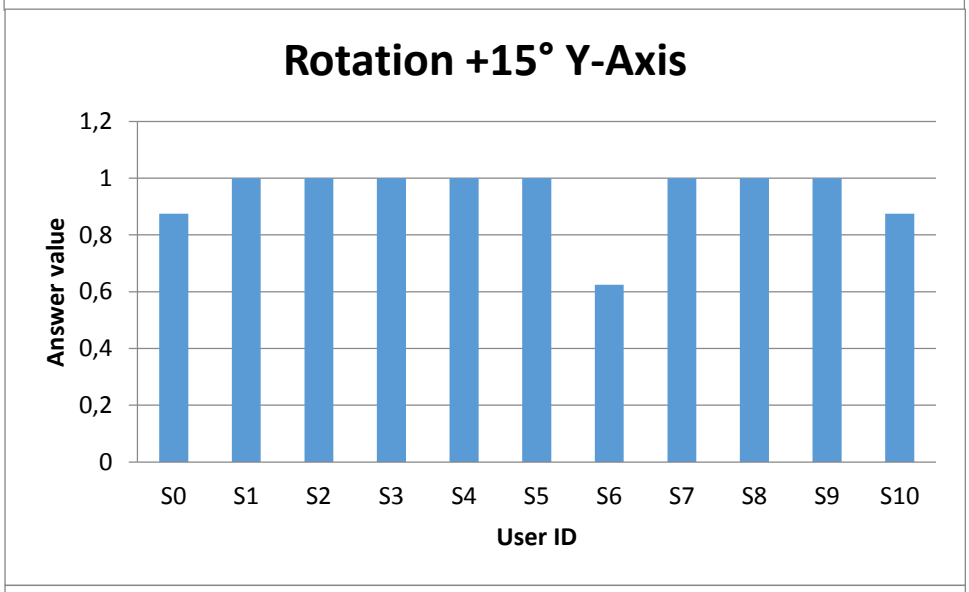

Rotation -10° Y-Axis
1,2
1
0,8
0,6
0,4
0,2
0
Answer value
S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10
User ID

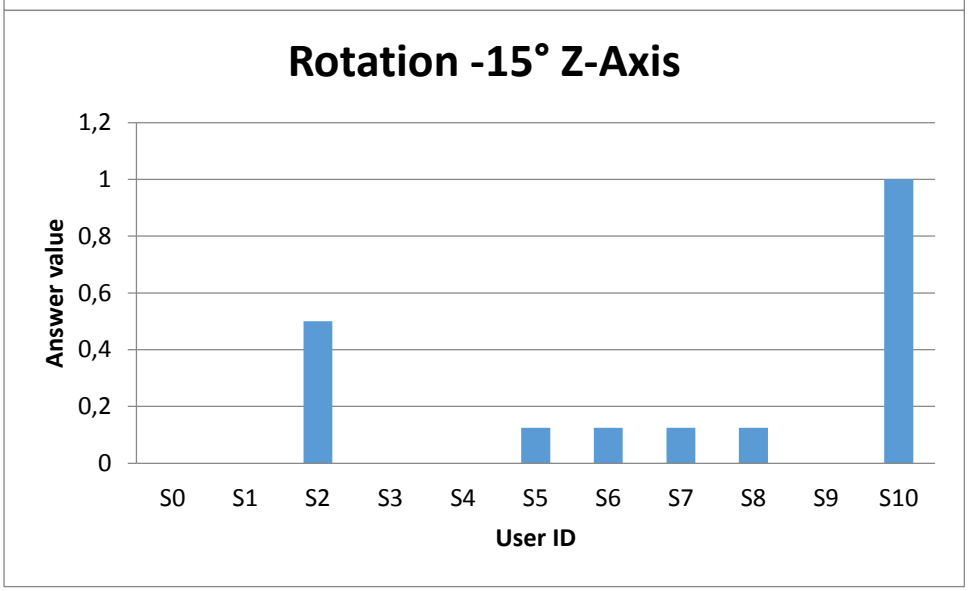

Rotation -5° Y-Axis
Answer value
1,2
1
0,8
0,6
0,4
0,2
0
S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10
User ID

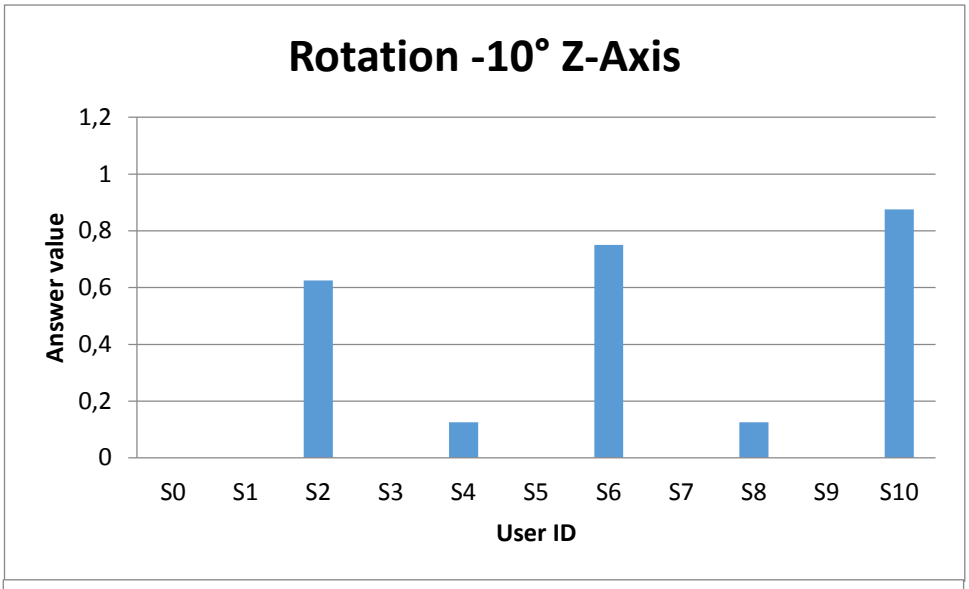

Rotation 0° Y-Axis
Answer value
1,2
1
0,8
0,6
0,4
0,2
0
S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10
User ID

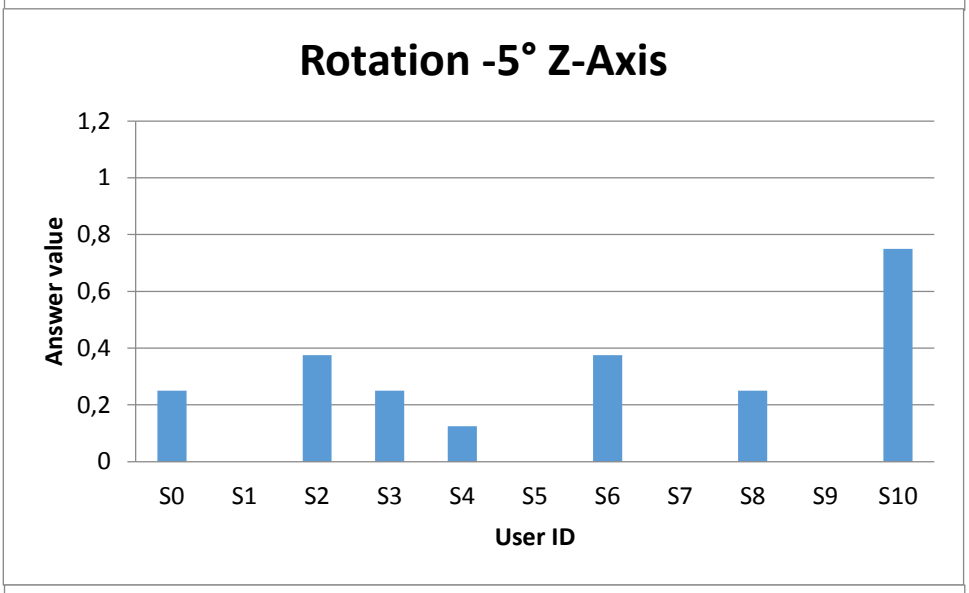

Rotation +5° Y-Axis
Answer value
1,2
1
0,8
0,6
0,4
0,2
0
S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10
User ID

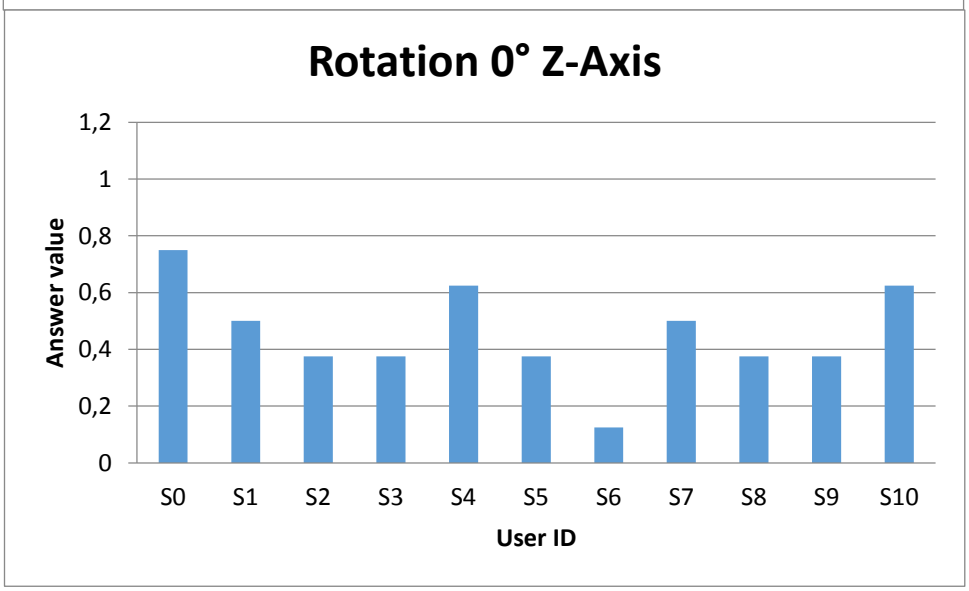

Rotation +10° Y-Axis
Answer value
1,2
1
0,8
0,6
0,4
0,2
0
S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10
User ID

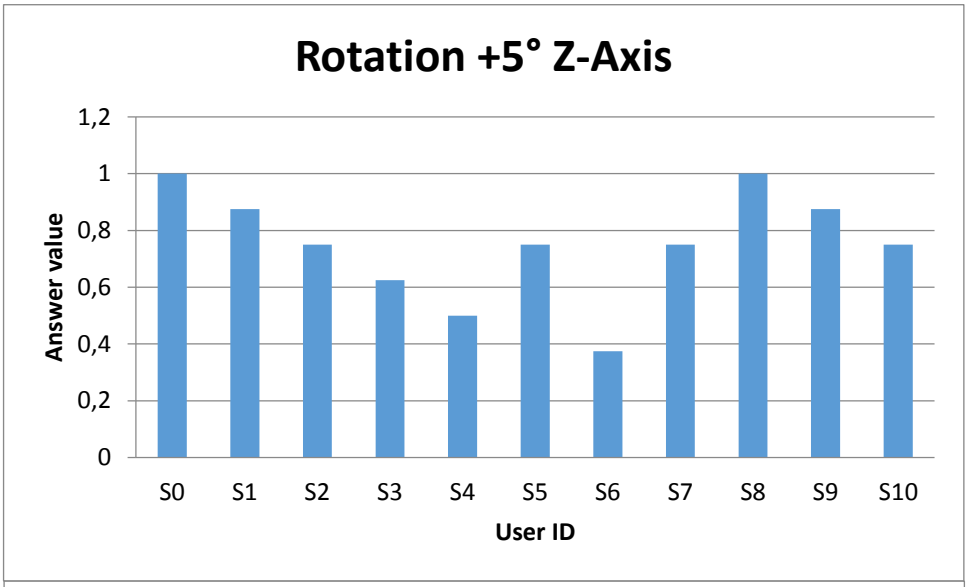

Rotation +15° Y-Axis
Answer value
1,2
1
0,8
0,6
0,4
0,2
0
S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10
User ID

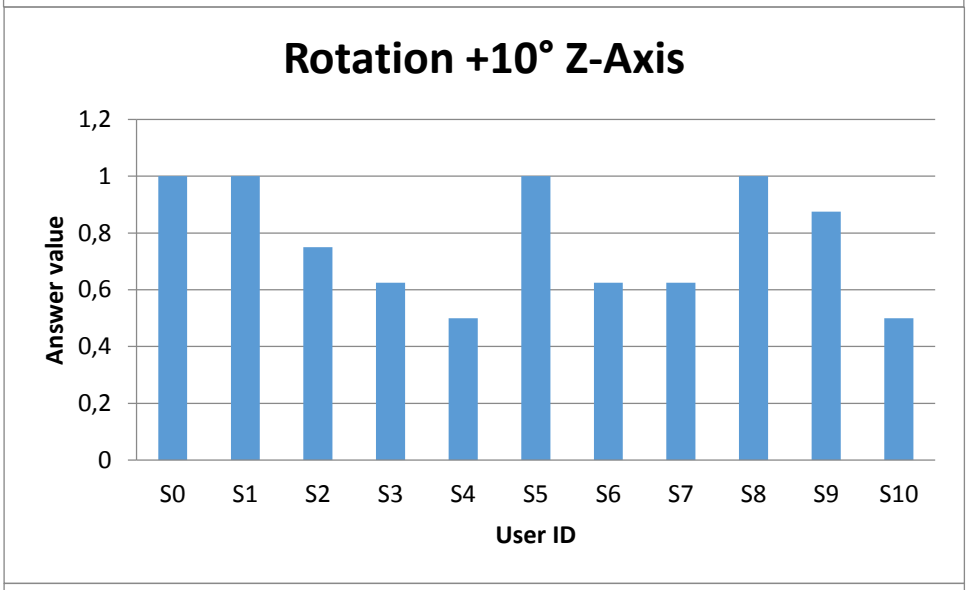

Rotation -15° Z-Axis
Answer value
1,2
1
0,8
0,6
0,4
0,2
0
S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10
User ID

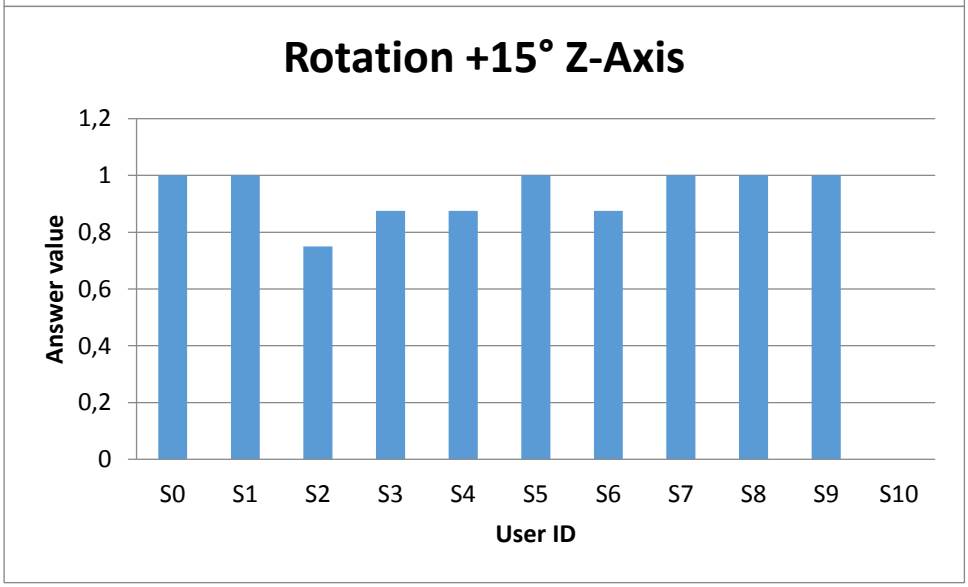

Rotation -10° Z-Axis
Answer value
1,2
1
0,8
0,6
0,4
0,2
0
S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10
User ID

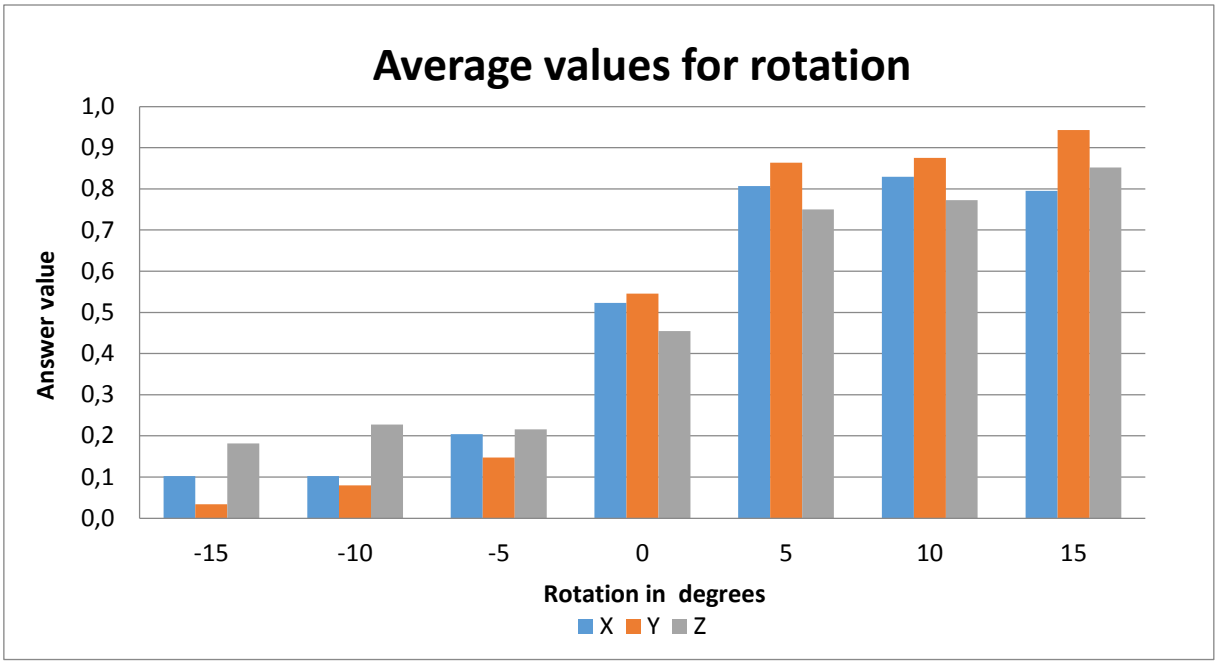

Rotation -5° Z-Axis
Answer value
1,2
1
0,8
0,6
0,4
0,2
0
S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10
User ID

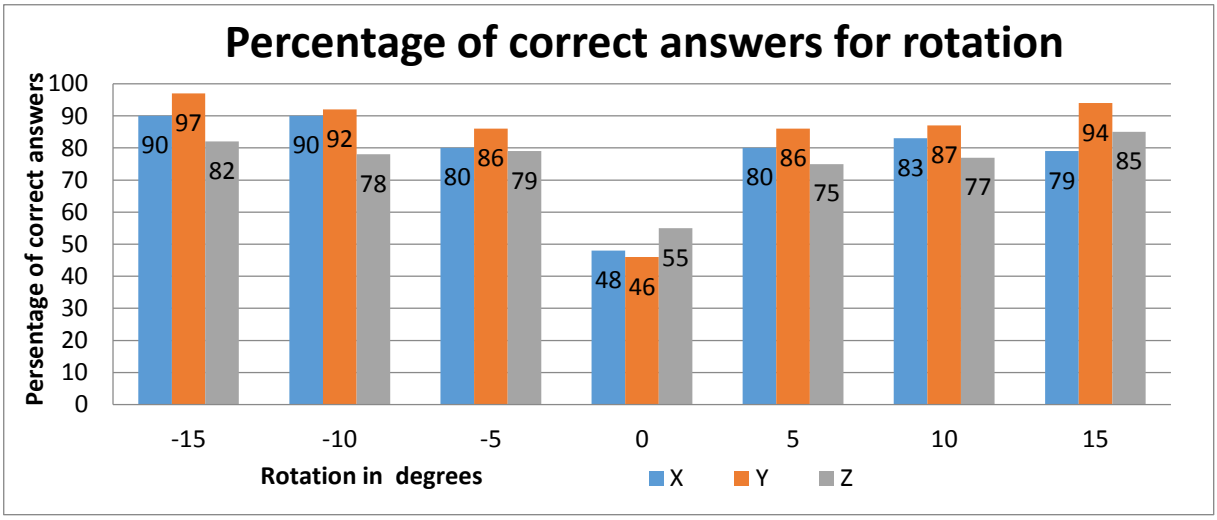

Rotation 0° Z-Axis
Answer value
1,2
1
0,8
0,6
0,4
0,2
0
S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10
User ID


Rotation +5° Z-Axis
1,2
1
0,8
0,6
0,4
0,2
0
Answer value
S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10
User ID


Rotation +10° Z-Axis
1,2
1
0,8
0,6
0,4
0,2
0
Answer value
S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10
User ID

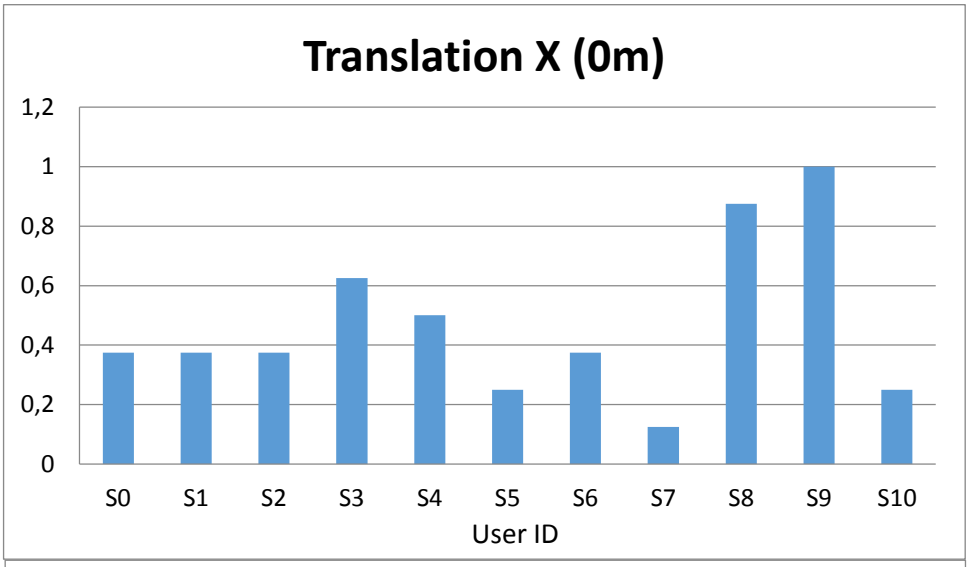

Rotation +15° Z-Axis
1,2
1
0,8
0,6
0,4
0,2
0
Answer value
S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10
User ID

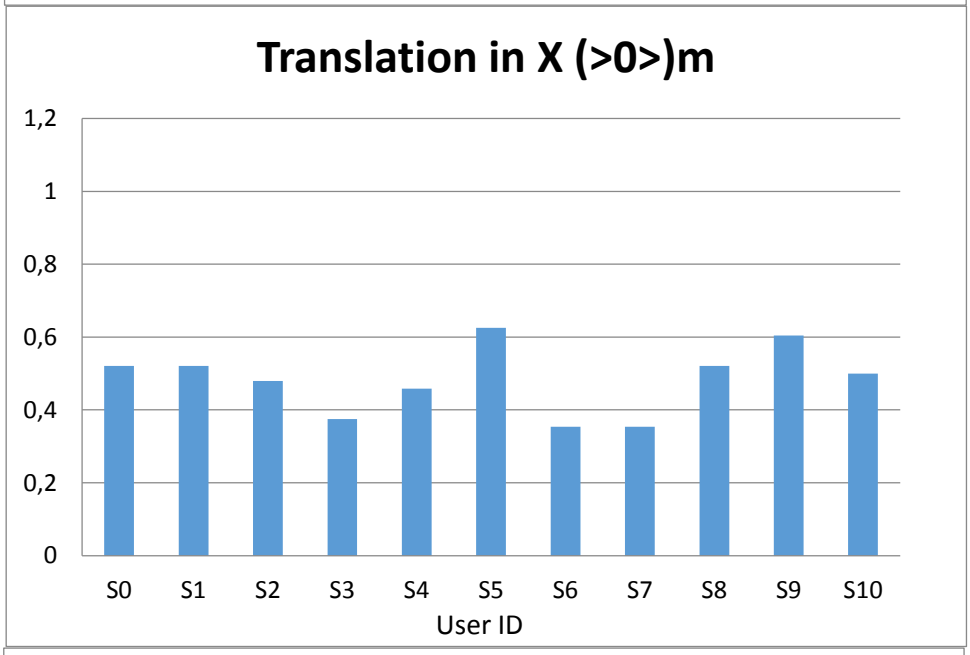


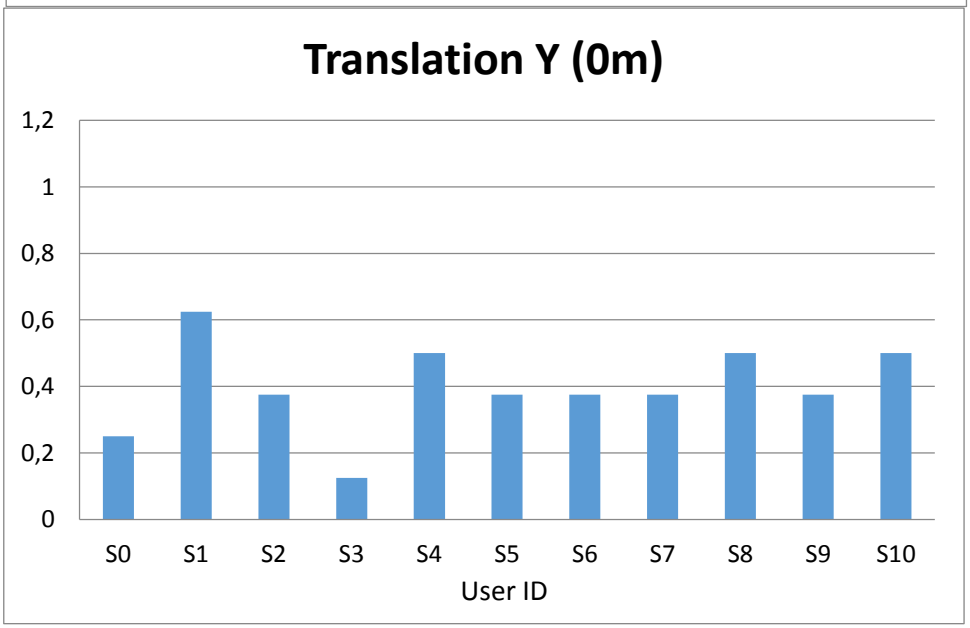


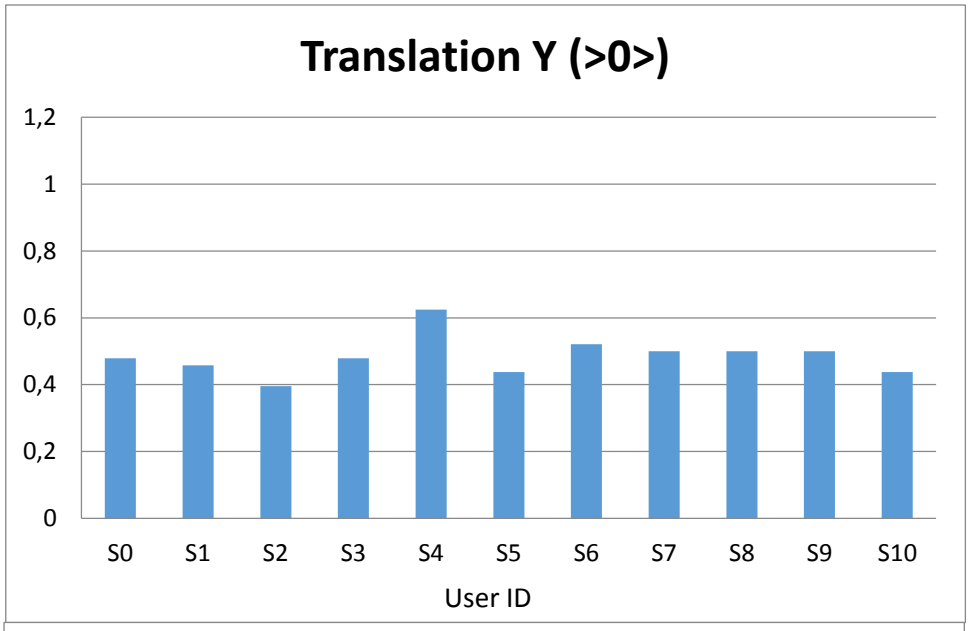
Average value							
	-15	-10	-5	0	5	10	15
X	0,795	0,830	0,807	0,523	0,205	0,102	0,102
Y	0,034	0,080	0,148	0,545	0,864	0,875	0,943
Z	0,182	0,227	0,216	0,455	0,750	0,773	0,852

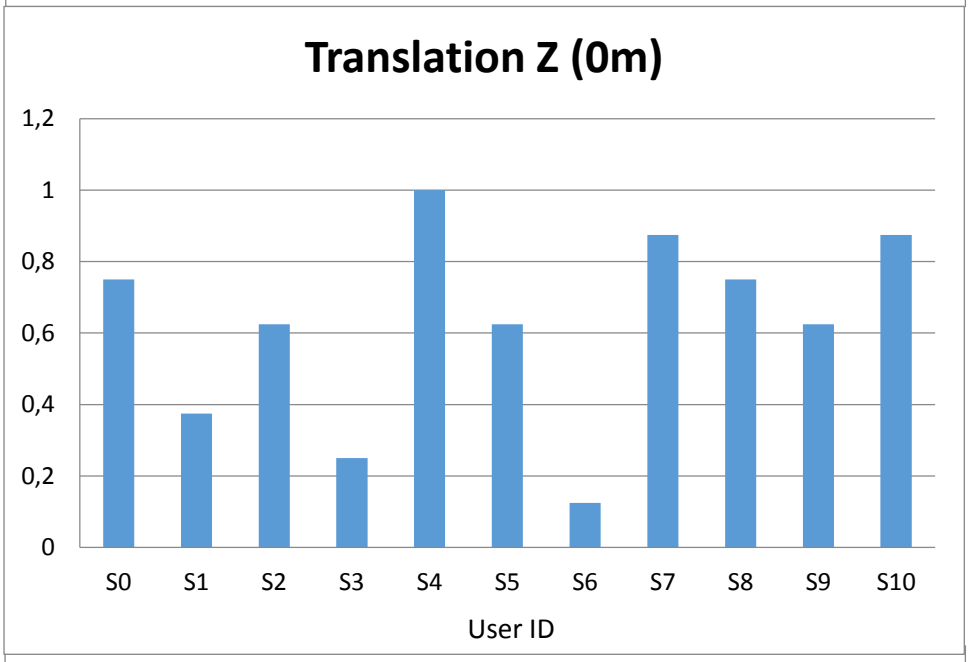
8.4.2 Translation results

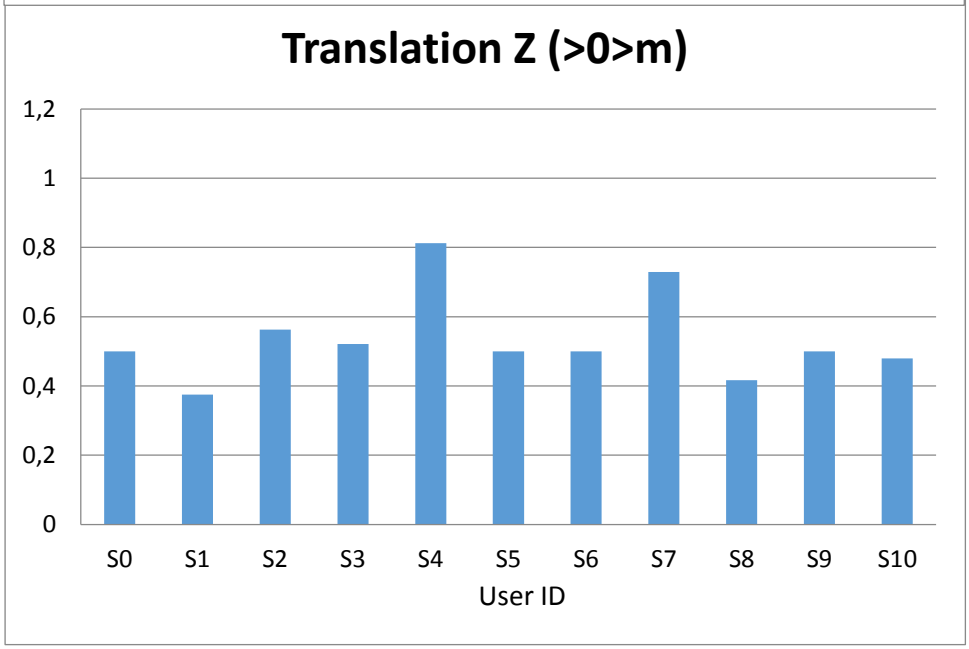


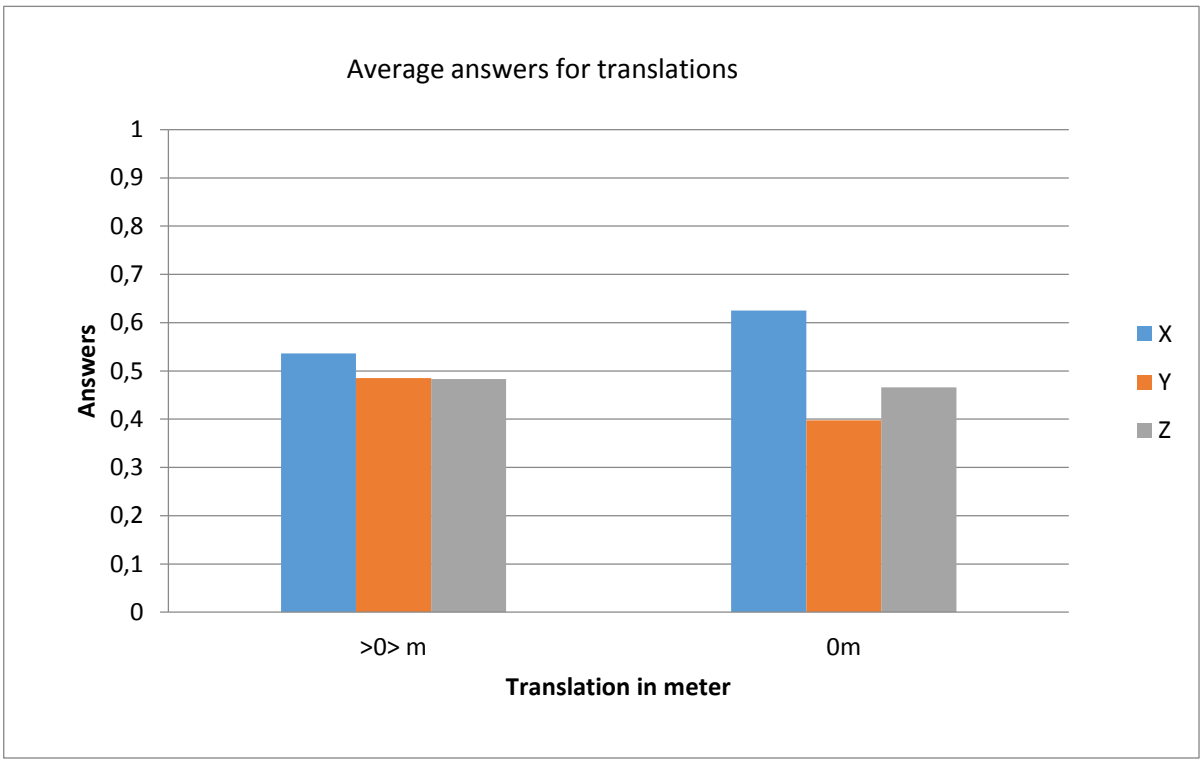





Translation Y (>0>)
1,2
1
0,8
0,6
0,4
0,2
0
S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10
User ID


Translation Z (0m)
1,2
1
0,8
0,6
0,4
0,2
0
S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10
User ID


Translation Z (>0>m)
1,2
1
0,8
0,6
0,4
0,2
0
S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 S10
User ID



8.4.3 Personal estimation data

Start questions										
Start language	en	en	en	en	en	en	en	en	en	en
ID	**0**	**1**	**2**	**3**	**5**	**6**	**7**	**8**	**9**	**10**
Vision correction	None	Glasses	Contact Lenses	Glasses	None	None	None	Glasses	None	Contact Lenses
Do you suffer from a displacement of equilibrium or similar-Gleichgewichtsstörung	No	No	No	No	No	No	No	No	No	No

Do you have a known eye disorder- Color blindness	No	No	No	No	No	No	No	No	No	No
Do you have a known eye disorder- Night blindness	No	No	No	No	No	No	No	No	No	No
Do you have a known eye disorder- Dyschromatopsia red-green color weakness	No	No	No	No	No	No	No	No	No	No
Do you have a known eye disorder- Strong eye Dominance	No	No	No	No	No	No	No	No	No	No
Do you have a known eye disorder- Other								strabismus		
Have you participated in a study with a head-mounted-display like the Oculus Rift before-	No	Yes	No	Yes	Yes	Yes	Yes	No	Yes	Yes
Do you have experience with 3d computer games-	5	5	3	5	5	4	5	2	5	2
How many hours do you play per week-	10	20	1	5	3	2	5	0	4	0
Do you have experience with 3D stereoscopic displays cinema, games etc.-	4	5	3	5	4	4	5	3	5	1
Height // Körpergröße	1,84	1,73	1,61	1,78	1,6	1,83	1,69	1,75	1,85	1,78

Kennedy SSQ before Experiment										
General discomfort // Allg. Unwohlsein	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	1
Fatigue // Erschöpfung	0 None	0 None	0 None	0 None	1	0 None	0 None	0 None	0 None	1
Headache // Kopfschmerzen	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	1
Eyestrain // Überanstrengung der Augen	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	1
Difficulty focusing // Probleme bei der Fokussierung	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	1
Increased salivation // Erhöhte Speichelbildung	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None
Sweating // Schweißbildung	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	2
Nausea // Übelkeit	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None
Difficulty concentrating // Konzentrationsschwierigkeiten	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	1
Fullness of head // Kopf voller Gedanken	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	1

Blurred vision // Unscharfe Sicht	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	1
Dizzy eyes open // Schwindelig o. Duselig bei offenen Augen	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None
Dizzy eyes closed // Schwindelig o. Duselig bei geschlossenen Augen	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None
Vertigo // Gleichgewichtsstörung	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None
Stomach awareness // Den Bauch wahrnehmen	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None
Burping // Aufstoßen	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None	0 None
Kennedy SSQ after experiment										
General discomfort // Allg. Unwohlsein	0 None	1	0 None	0 None	1	1	2	1	0 None	1
Fatigue // Erschöpfung	0 None	2	1	0 None	1	1	3 Severe	1	0 None	2
Headache // Kopfschmerzen	0 None	0 None	0 None	0 None	1	1	2	0 None	0 None	2
Eyestrain // Überanstrengung der Augen	0 None	1	1	0 None	0 None	2	2	0 None	0 None	2

Difficulty focusing // Probleme bei der Fokussierung	0 None	0 None	0 None	0 None	0 None	1	2	0 None	0 None	1
Increased salivation // Erhöhte Speichelbildung	0 None	0 None	0 None	0 None	0 None	1	2	0 None	0 None	0 None
Sweating // Schweißbildung	0 None	2	0 None	0 None	0 None	1	2	1	1	1
Nausea // Übelkeit	0 None	0 None	1	0 None	0 None	1	1	1	0 None	0 None
Difficulty concentrating // Konzentrationsschwierigkeiten	0 None	0 None	0 None	0 None	0 None	1	3 Severe	0 None	0 None	0 None
Fullness of head // Kopf voller Gedanken	0 None	0 None	0 None	0 None	0 None	1	3 Severe	1	0 None	0 None
Blurred vision // Unscharfe Sicht	0 None	0 None	0 None	0 None	0 None	1	2	0 None	0 None	0 None
Dizzy eyes open // Schwindelig o. Duselig bei offenen Augen	0 None	0 None	0 None	0 None	0 None	1	2	1	1	0 None
Dizzy eyes closed // Schwindelig o. Duselig bei geschlossenen Augen	0 None	0 None	0 None	0 None	0 None	1	2	1	1	0 None
Vertigo // Gleichgewichtsstörung	0 None	0 None	0 None	0 None	0 None	1	1	0 None	1	0 None

Stomach awareness // Den Bauch wahrnehmen	0 None	0 None	0 None	0 None	0 None	1	1	0 None	0 None	0 None
Burping // Aufstoßen	0 None	0 None	0 None	0 None	0 None	1	1	0 None	0 None	0 None
Self-estimation after experiment										
How sure are you that you always chose the correct answer	3	3	4	4	2	4	3	2	3	3
Please rate your sense of being in the virtual environment, on a scale of 1 to 5, where 5 represents your normal experience of being in a place. I had a sense of "being there"...	3	5	5	4	3	2	4	4	4	4
To what extent were there times during the experience when the virtual environment was the reality for you- There were times when the virtual environment was the reality for me...	1	5	3	4	3	2	4	3	4	3
When you think back to the experience, do you think of the virtual environment more as images that you saw or more as somewhere that you visited- The virtual environment seems to me to be more like...	4	4	5	4	3	3	4	5	4	2

During the time of the experience, which was the strongest on the whole, your sense of being in the virtual environment or of being elsewhere- I had a stronger sense of...	4	2	4	4	3	4	4	4	2	4
During the time of your experience, did you often think to yourself that you were actually in the virtual environment- During the experiment I often thought that I was really standing in the virtual environment...	1	5	1	4	3	2	4	4	4	4
Consider your memory of being in the virtual environment. How similar in terms of the structure of the memory is this to the structure of the memory of other places you have been today- By 'structure of the memory' consider things like the extent to which you have a visual memory of the virtual environment, whether that memory is in colour, the extent to which the memory seems vivid or realistic, its size, location in your imagination, the extent to which it is panoramic in your imagination, and other such structural elements. I think of the virtual environment as a place in a way similar to	4	5	5	4	3	3	4	4	4	3

other places that I have been today...										
How would you subjectively describe your level of attention during the experiment-	4	4	5	5	4	4	4	4	5	3
Do you think that the experiment took too long-	3	5	2	3	3	3	5	4	2	4
General Information										
Age	25	29	26	29	26	21	37	29	34	23
Profession / field of study Beruf / Studiengang	HCI	phd student	Student	phd student	HCI	Student	Research assistant	Master Studies in Computer Science	Informatik	MCI Student
Gender	Male	Male	Female	Male	Female	Male	Male	Male	Male	Male

Self-estimation after the Experiment

ID	Did you have a cognitive strategy to detect the rotations/translations-	Additional comments
0	Looking straight ahead at some lines. if they rotate or move it was easily to detect if i was rotated. Translating/Rotating up and down was harder to detect	felt very buggy

1	I saw it when i opened my eyes again.	Standing sucks!
2	Yes, looking to the text and some explicit point in the scene.	
3	orientation points	
4	try to look for fixed point and then compare my position to the point whether left, right, up, down and forward, backward. I noticed that in some conditions I am not moving at all for this I just decided randomly.	Interesting experience!
5	ich habe immer die Unterkante des Raums bzw. ein Tischbein bei den Translationen fokussiert. Bei den Rotationen fand ich es einfacher die Tischplatte zu beobachten; ansonsten habe ich bei jedem Trial versucht neu einzuschätzen	die Rotationen fand ich leichter zu bemerken, weil man einen neuen Ausschnitt des Raums gesehen hat (sollte bei Translationen ja eigentlich auch so sein, aber da ist mir das nicht aufgefallen);
6	I tried to build a new impression for each trial.	
7	I tried to guess based on the initial position of the objects.	
8	Not all the time but I was comparing the position of the legs of the table and make conclusions depending whether the front one is left\right from the back one	
9	Translation: -left and right: windows in the scene were unfortunately very helpful -up and down: only by remembering the center position -forth and backward: this was really difficult --> main-strategy was comparing the images Rotation: -pitch: only by checking the mismatch between guessed real orientation of my head and the presented view, moderately difficult -roll: very easy to determine, just by recognizing the mismatch between "Gleichgewichtssinn" and the presented view -yaw: this was very hard to recognize, often I was lost and didn't know which orientation is real	The success rate of the sensor is improved when staring to the "right" position. With perfect timing (changing the scene during eyes shut period) it was really difficult to recognize the change in the environment). Sometimes I saw the active change of the environment during opening phase of my eyes, which made it easy to recognize.

	--> main-strategy was comparing image and recognized feeling for the presented perspective	
10	If the room moves in one direction, choose the other one.	I was not sure if I should tell my own rotation as if i were a camera or of the image i saw, i guess my own

Question/ User ID	0	1	2	3	4	5	6	7	8	9	10	Middl	%
How sure are you that you always chose the correct answer	3	3	4	4	3	2	4	3	2	3	3	3,09090909	61,8181818
Please rate your sense of being in the virtual environment, on a scale of 1 to 5, where 5 represents your normal experience of being in a place. I had a sense of "being there"...	3	5	5	4	4	3	2	4	4	4	4	3,81818182	76,3636364
To what extent were there times during the experience when the virtual environment was the reality for you- There were times when the virtual environment was the reality for me...	1	5	3	4	4	3	2	4	3	4	3	3,27272727	65,4545455
When you think back to the experience, do you think of the virtual environment more as images that you saw or more as somewhere that you visited- The virtual environment seems to me to be more like...	4	4	5	4	5	3	3	4	5	4	2	3,90909091	78,1818182
During the time of the experience, which was the strongest on the whole, your sense of being in the virtual environment or of being elsewhere- I had a stronger sense of...	4	2	4	4	4	3	4	4	4	2	4	3,54545455	70,9090909
During the time of your experience, did you often think to yourself that you were actually in the virtual environment- During the experiment I often thought that I was really standing in the virtual environment...	1	5	1	4	4	3	2	4	4	4	4	3,27272727	65,4545455
Consider your memory of being in the virtual environment. How similar in terms of the structure of the memory is this to the structure of the memory of other places you have been today- By 'structure of the memory' consider things like the extent to which you have a visual memory of the virtual environment, whether that memory is in colour, the extent to which the memory seems vivid or realistic, its size, location in your imagination, the extent to which it is panoramic in your imagination, and other such structural elements. I think of the virtual environment as a place in a way similar to other places that I have been today...	4	5	5	4	4	3	3	4	4	4	3	3,90909091	78,1818182
How would you subjectively describe your level of attention during the experiment-	4	4	5	5	4	4	4	4	4	5	3	4,18181818	83,6363636
Do you think that the experiment took too long-	3	5	2	3	3	3	3	5	4	2	4	3,36363636	67,2727273
											Av		73,8181818

8.5 Simulator Sickness Questionnaire

Kennedy Simulator Sickness Questionnaire (1993) [1]

Instructions: Circle how much each symptom below is affecting you right now.

1. General discomfort	None	Slight	Moderate	Severe
2. Fatigue	None	Slight	Moderate	Severe
3. Headache	None	Slight	Moderate	Severe
4. Eye strain	None	Slight	Moderate	Severe
5. Difficulty focusing	None	Slight	Moderate	Severe
6. Salivation increasing	None	Slight	Moderate	Severe
7. Sweating	None	Slight	Moderate	Severe
8. Nausea	None	Slight	Moderate	Severe
9. Difficulty concentrating	None	Slight	Moderate	Severe
10. « Fullness of the Head »	None	Slight	Moderate	Severe
11. Blurred vision	None	Slight	Moderate	Severe
12. Dizziness with eyes open	None	Slight	Moderate	Severe

13. Dizziness with eyes closed	None	Slight	Moderate	Severe
14. *Vertigo	None	Slight	Moderate	Severe
15. **Stomach awareness	None	Slight	Moderate	Severe
16. Burping	None	Slight	Moderate	Severe

* Vertigo is experienced as loss of orientation with respect to vertical upright.

** Stomach awareness is usually used to indicate a feeling of discomfort which is just short of nausea.

Human-Computer Interaction
University of Hamburg

Participant information and consent form

Please read this document carefully before deciding to participate or not in this study.

Project: Eye blinks and Redirection

People in charge of this project: Eike Langbehn

Objective: The main objective of this project is to evaluate the potential of eye blinks in an immersive virtual environment (IVE).

Nature of your participation: You will be required to wear an HMD and follow the instructions presented at the display.

Participants must be healthy and have no disorder of equilibrium or vision.

Protocol: The experiment will last approximately one hour. The objective is to evaluate the potential of eye blinks in an immersive HMD environment.

You will stand in the laboratory and wear an HMD. You will see a virtual environment. You will be asked to blink and to answer questions.

You can take a short break between trials whenever you want.

Benefits and risks related to your participation:

Your participation in this project will contribute to the advancement of the knowledge about the use of immersive platforms to study human perception.

Your participation in this project should not induce significant difficulties. However, in some cases some people suffer from simulator sickness. The secondary effects match the ones of motion sickness (nausea, vertigo). We remind you that you can ask for a break whenever you want. You acknowledge that it is your responsibility to end the experiment when you feel that your simulator sickness symptoms are approaching your tolerance level.

We hereby inform the participant that immersion can have lasting behavioral influences, and that some of these risks may be presently unknown.

Right to withdraw without prejudice:

Your participation in this project is entirely voluntary. You can withdraw from the study at any time, without consequence, without having to give a reason. You have the option of omitting questions that you do not wish to answer if a questionnaire is used.

If you withdraw from the study, do you wish that all documents related to your participation will be destroyed?

☐ Yes ☐ No

Confidentiality, sharing and publication:

People in charge of the project will hold all your personal information and data gathered during the experiment. Only the data required for the correct execution of the project will be gathered. They can concern the following information: name, gender, demographic information, photo, audio or video recording, results of all the experiments.

All the information gathered during this project will remain confidential within the limits established by the law. Data will be used for research purpose in order to answer to the scientific objectives of the project described in this information and consent form. Data of this project can be published in scientific reviews or shared with other people during scientific discussions.

Video recording and/or photo:

Some photo or video recordings might be taken during the experiments. We would like to be able to rely on these documents for the purpose of training and scientific presentations. If you disagree, all the video recordings and photos concerning yourself will be destroyed at the end of the project with due regard to the confidentiality.

Do you allow us to take photos and record video and to keep them with your research data?

☐ Yes ☐ No

Free and informed consent

I, _____ (*name in capital letters)* declare that I have read and understood this form. I understood the nature and the purpose of my participation in this project. I have been given answers to any question I have about the study.

I freely accept to participate in this project.

Participant's signature: _____

Place and date: _____

Statement of responsibility of the persons in charge of this project

I, Eike Langbehn, declare that the researchers, the collaborators as well as the research team are responsible of the project execution. We are committed to respecting the obligations stated in this document and to keep you informed about any element which could affect the nature of your consent.

Signature: _____

Contact details:

Eike Langbehn, University of Hamburg, Email: langbehn@informatik.uni-hamburg.de

Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Fakultät für Mathematik, Informatik
und Naturwissenschaften

Fachbereich Informatik
Studienbüro

Bescheinigung Versuchspersonenstunden am Fachbereich Informatik

Studiengang : **o B.Sc. MCI**
Nachweis über die Teilnahme an Versuchspersonenstunden zur Anrechnung im Modul InfB-VP/MCI

o andere _____

Semester: **o Sommer 2015** **o Winter 20____/____**

Angaben zur Studentin / zum Studenten	
Name:	Vorname:
Matrikelnummer:	
E-Mail:	

Angaben zur Untersuchung/zum Projekt
Arbeitsbereich: Human-Computer Interaction
Name Projektleiter/in: Eike Langbehn
Titel/Thema: Eye blinks and Redirection
Datum:
Dauer in Minuten:
Anzurechnende VP-Stunden (30 Min. = 0,5 VP-Stunden, 60 Min = 1 VP-Std. etc.): _____ VP-Stunden
_____ _____ **Datum, Unterschrift Projektleitung** **Stempel des Arbeitsbereichs**