Enabling people with motor impairments to get around in 360° video experiences: Concept and prototype of an adaptable control system

Julia Deneke
Abstract

Enabling people with motor impairments to get around in 360° video experiences: Concept and prototype of an adaptable control system

Julia Denée

This thesis is about an alternative control system for interacting with 360° videos and focuses on people with physical impairments (PI). People with PI face restrictions that keep them from exploring the world like able-bodied people. For example, a walk in the forest can be impossible because of lacking accessibility or harmful weather conditions. My contribution is located in a bigger project, the Experience Library 360°, that aims to provide virtual, interactive and immersive experiences of real-world locations. As VR applications are often controlled by head movements or mouse clicks, people with certain PI are excluded. My goal was to find alternative control units to make the experience accessible, and to validate the idea in practice.

360° video technology is still young and there has been no previous research in the context of assistive technologies. I used a literature study to gain understanding of the field and to generate conceptual ideas, a target user interview to evaluate those ideas, and a prototyping phase to build a proof-of-concept.

The result is an operable prototype that features an intermediate box to which a control unit can be connected via USB. It includes an API that re-maps controller input to video commands. The proof-of-concept validates a part of the system, but has not been user tested. My research shows that it is possible to use alternative control units to exclude potentially less people from use. Those ideas can be applied in other human-computer interaction contexts relevant to people with and without physical impairments.
# Table of Contents

1 INTRODUCTION  .................................................. 1
   1.1 Motivation .................................................. 1
   1.2 Objectives and Research Questions ...................... 2
   1.3 Delimitations and Scope ................................... 4

2 RELATED WORK ................................................. 6
   2.1 360° Videos and Virtual Reality .......................... 6
   2.2 Physical Impairments ....................................... 7
   2.3 Control Units and Input Tools ............................ 8

3 RELEVANT FRAMEWORKS AND CONCEPTS .................. 16
   3.1 Non-Excluding Design ..................................... 16
   3.2 Universal Design .......................................... 17
   3.3 Ability-based Design ...................................... 19

4 METHODOLOGY ................................................ 21
   4.1 Overall Methodology ....................................... 21
   4.2 Literature Study ........................................... 22
   4.3 Target User Interview ..................................... 23
   4.4 Prototyping .................................................. 23

5 CONCEPTUALISATION .......................................... 25
   5.1 Functional Requirements ................................... 25
   5.2 Expressive Power of Control Units ....................... 27
   5.3 Mapping ..................................................... 30
   5.4 Preliminary Evaluation ..................................... 32
   5.5 Selecting Control Units .................................... 34

6 PROTOTYPING THE CONTROL SYSTEM .................... 37
   6.1 Sketches and Design Considerations ..................... 37
   6.2 Building ..................................................... 38
   6.3 Final Prototype ............................................. 41

7 DISCUSSION .................................................... 46
   7.1 Conclusion .................................................. 46
   7.2 Future Work ................................................ 49

REFERENCES ....................................................... 51
Tables and Figures

Table 1: Functional requirements for controlling a 360-degree video. 26
Table 2: Assessment of Dimensions of expression and minimum number of expressions of different control inputs. 30
Table 3: Distribution of commands on the example control units. 43

Figure 1: Example of movements and devices used to control a VR headset. 2
Figure 1: Examples of mechanical switches. 9
Figure 2: Example of a wheelchair joystick. 10
Figure 3: Example of a touchpad. 10
Figure 4: Interface of gesture-controlled Camera Mouse. 11
Figure 5: Interface of Click Control system. 11
Figure 6: Sounds used by the Vocal Joystick engine. 12
Figure 7: Example of eye wink-based control system. 13
Figure 8: Gaze-based system EyeDraw. 14
Figure 10: Overview of the adapted design process and methods used. 21
Figure 11: 360-degree video interface with arrows. 31
Figure 12: 360-degree video interface controlled by drag-and-drop, and swiping on the smartphone. 31
Figure 13: Simplified spectrum of the user's freedom of movement. 35
Figure 14: Combined spectra of the Dimensions of expression of different devices and the user's freedom of movement. 35
Figure 15: First sketch of the prototype setup. 37
Figure 16: Screenshot of Arduino serial output. 39
Figure 17: First iteration of the application. 40
Figure 16: Video globe viewed from the outside. 40
Figure 19: Final prototype setup with computer, Arduino “box” with USB Host Shield, and two example control units. 41
Figure 20: File structure of the main program. 42
Figure 21: User interface with pointers, arrows and time counter. 44
Appendix

Appendix A: Overview of Control Alternatives
Appendix B: Application Workflow
Appendix C: Arduino Code Documentation
Appendix D: Processing Code Documentation
1 Introduction

This thesis describes the development of a prototypical control API that will be used as part of the project Experience Library 360°. The project aims to provide people with (primarily) physical impairments with the ability to experience environments in a way that resembles the physical presence as much as possible. This includes the possibility to select different paths through the environment. My thesis work comprises a pilot literature study that lead to taxonomy of control units and input technologies, as well as to a proof-of-concept for further development. In this chapter I will describe the motivation for dedicating my thesis work to the Experience Library project further. I will introduce the project and its aims and explain where my own contribution is located. Then, I will discuss the objectives and research questions that I will try to answer with my work. Finally, I will briefly explain the limitations of this thesis.

1.1 Motivation

As humans are curious by nature, exploring the world is part of our quality of life. It should be available to every person in the best way possible. While certain impairments do affect a person’s life greatly, this does not generally mean that they are not as interested in exploring as everyone else. People with spinal cord injuries often face many restrictions when it comes to making experiences in outdoor environments. They are limited in their freedom of movement, ranging from difficulties walking, moving their arms and hands, the absence of fine-motor control, tremor, or even the inability to turn their heads. They might not get a chance to visit cultural and historical sites because some of those places cannot be accessed by wheelchair. This type of exclusion through “clear, physical barriers” is also referred to as hard exclusion (Oestreicher, n.d.). Furthermore, environmental factors can make it difficult to be outside for longer periods of time, such as cold weather or snow. A longer exposure to cold can result in frostbite due to low blood circulation or damage to the technical equipment. Skin conditions associated with the impairment can force a person to avoid the sun or even daylight completely.

With the means of technology currently available, people with motor impairments can be enabled to move around in otherwise inaccessible places. Even so, there are many places that are not accessible even by very sophisticated technology. One way of increasing the accessibility of difficult environments could be to use 360-degree video technology, where we can create immersive experiences that comes as close to real-life experiences as possible and convey a feeling of “being there”. While this is not to be regarded as a replacement for the world outside, it can be used to offer a more equal accessibility to special environments and even provide a visitor’s experience, independent of physical impairments. Lastly, applications that are created for people with impairments as the primary target group can in most cases be used by people without impairments as well. Considering certain limitations in the first place is crucial to reach a design solution that excludes as few people as possible. Even though the current focus of the project lies on people with motor impairments, it can be applied to a broader context of use.

The Experience Library project is intended to enable the viewer to experience real-life situations in an immersive and realistic fashion they could not access otherwise. This pos-
sibility already exists in the form of videos with a first-person perspective. With the development of affordable 360-degree cameras and applications, there are many activity movies available on YouTube. The only issue with those videos is that they are pre-recorded, and the viewer has to follow a person as they choose the path. This is where the Experience Library 360° intends to make a difference, by providing a means for the viewer to choose among several paths in the physical environment – similar to in animated games. The goal is to create a method for building a collection of 360-degree videos with an interactive component. That is, each experience consists of multiple video snippets joined by “decision points” where the viewer is allowed to choose from two or more different paths. The possibility to choose “where to go” in the filmed environment enables them to explore the displayed scenery to its full extent. Thanks to the 360-degree aspect, the user can also simultaneously look around in the environment throughout the whole video. All videos will be filmed with an omnidirectional camera. The viewing setup will consist of a pair of VR goggles, e.g. Oculus Rift or Samsung VR, or a large screen that allows for some level of immersive feel.

While most 360-degree viewing applications are controlled by movements of the head (or drag-and-drop operations in some cases), not all of the required movements are possible for people with certain motor impairments. As mentioned above, some people cannot move their heads fully, if at all. In order to perform input with the hands, a certain degree of fine-motor control is required which not everyone possesses. Therefore, in order to achieve maximum inclusion a set of alternative controllers needs to be available. Since motor impairments exist in different forms and combinations, the available control facilities should also be adaptable to individual characteristics.

This thesis attempts to lay the foundations for such a control system, beginning with the identification of movement requirements for navigating 360-degree videos, investigating the suitability of possible input tools, and resulting in a suggestion for an API that facilitates the connection of those tools. We will for this purpose explore the notion of “dimensions of expression”. In the following two sections, I will go into detail about the research questions to be answered as well as delimit the scope the thesis will cover.

### 1.2 Objectives and Research Questions

In my thesis work, I will discuss the initial development of an adaptable and reusable control system to navigate 360-degree video experiences despite major motor impairments. The research presented is located in the field of Universal or Non-Excluding Design. The video experiences (made available through a collection of videos, developed in the Experience Library 360° project) can be a powerful tool to enable people with physical impairments to experience parts of the world from the same perspective as able-bodied people can. As different assistive tools and digital applications give us the possibility to work around bodily limitations of physical nature, the focus can be shifted to the strengths and abilities that are actually there.
360-degree video technology is being used in multiple disciplines and for different purposes. However, as far as we have found, it has not been studied in the context of Non-Excluding Design. Research has been done to explore alternative ways of controlling devices and applications, often motivated by the aim to provide aid to people with impairments. Which is to say that many, if not most of the technologies needed are already available, but need to be connected to each other for the purpose of the project. Even though a variety of alternative control units has been studied, it turns out not all are equally suitable for this work. Therefore, one of the main objectives of this thesis was to gather information about different alternative control units, evaluate their control properties, and to select those that are qualified for use in the Experience Library 360° project. This was done to learn about devices we can work with, in order to find as many different ways as possible to include people with different physical properties. Another goal was to create a simple interface (API) to facilitate the connection of different types of assistive devices to the Experience Library software. The resulting navigational interface should pose as little intrusion as possible and feel comfortable to the user. This led me to the following research questions:

1. What alternative, suitable input devices and assistive tools are available for controlling 360-degree videos?

2. What are appropriate criteria to determine the suitability and applicability of such input devices?

3. How can different input devices be connected in practice to make the control system flexible and adaptable?

In order to answer the first research question, an overview of available technologies and control units had to be created. Information about the existing body of research was acquired through an extensive literature search, partly built on a systematic review of access technologies (Tai, Blain, & Chau, 2008). The findings were categorised by the type of input they afford. The different technologies are discussed in Chapter 2.3 as a part of the thematic background, and were used to create the initial taxonomy described in Chapter 5 (Conceptualisation).

To answer the second research question, it was necessary to determine what aspects the control units in question need to be suitable for. Since this research project is the first of its kind, all conceptual ideas had to be built from scratch. As a starting point, an analysis of the functional requirements of a 360-degree video was carried out, including the full range of possible commands. Different limitations were considered to define a set of minimum requirements – starting from the assumption that some people have very little or no ability to move. Together with the leader of the Experience Library 360° project and reviewer of my thesis, Lars Oestreicher, the Dimensions of expression of a control unit and its Maximum number of expressions were discussed as criteria for judging suitability.\(^1\)

The third research question draws a line from the technical aspects of the solution to the philosophy of Non-Excluding design: making the video experiences accessible for people with different physical properties, allowing for as small a range of mobility as possible.

---

\(^1\) Note that this concerns the overall suitability of the tool for the type of application discussed in this thesis. For usability and accessibility aspects, an individual classification will be needed for each user, depending on the impairment.
As a proof-of-concept, a first prototype of an API was created to enable the connection of the control units of choice using an Arduino board as a communication interface between controller and software. It was developed with the intention to prove that the necessary controls can be modified and re-mapped according to the needs of the individual user. The successful implementation of the prototype works as a validation of the concept as well as a foundation for future development.

1.3 Delimitations and Scope

The focus of this thesis lies on finding suitable hardware alternatives for people with restricted mobility to control the viewing experience of 360-degree videos. Yet, there is not enough space to ask and answer all upcoming questions. Other aspects of the 360-video experience, such as immersion and transitions between video clips, are addressed by the work of my classmates (Abou Torab & Tapia, 2018). While this thesis covers the conceptual aspects of the control system, it will not produce an elaborate technical solution. The Experience Library 360° project is still in a rather early stage of idea formation and development, and the outcome should be a prototype as a proof-of-concept that builds a basis for further iterations. Due to time and resource constraints, only one iteration of the design cycle (Norman, 2013) could be performed. This area of research is somewhat sensitive, and the target group is rather specific. Therefore, the evaluation phase had to be kept concise, using qualitative feedback from one target user that has given her explicit consent to the interview. That round of feedback was used to confirm the concept, but not the final prototype.

The main target group are people with motor impairments, like those resulting from spinal cord injury. While the idea of an experience library offers potential for use in a variety of contexts, this thesis concentrates on the aspects that are relevant to people with motor impairments. Yet, since the research problem is situated within the field of Non-Excluding Design, I would like to point out that it is nearly impossible to enable everyone, despite their physical properties, to use the solution. The efforts of this thesis are aimed at laying the foundation for an alternative hardware solution that excludes as few people as possible from use. From the perspective of Non-Excluding Design, the most important measure by which the system should be judged by is not efficiency, but the operability of a device for an individual user. Not everyone can use the solution that is considered “most efficient”. What matters instead is that an option is found that works for the person in practice. This includes identifying possibilities to reuse existing tools as described in my work, as well as the development of new tools from scratch if necessary.

Apart from the functionalities and expressive power of different control units, there are other aspects that can influence the choice of technology, such as economy factors. The cost of a solution is important once it shall be used in a real-life context. Most people cannot afford to spend large amounts of money on assistive devices. Still, the cost factor was not considered as a criterion in this work. This is mainly for two reasons: the solution does not consist of a specific device; instead, the criteria for the devices should be universal in a sense that the user can choose the control units they want to connect.

Furthermore, some researchers of assistive technologies mention that input devices should be “comfortable”, “aesthetically pleasing” and, if possible, also wireless (Evans, Drew, & Blenkhorn, 2000; Kim & Cho, 2002). These criteria, again, become especially
important in a real-life context. The aesthetics of a solution are closely related to the emergence of social stigma, and therefore influence strongly how (un)comfortable a person is with using it. As the work of this thesis is happening at an early stage, it was decided to focus the main attention and efforts on making the control system work. While it exceeds the scope of this thesis, making the solution pleasant to use should in no way be disregarded in following iterations.

It is also the case that aspects such as the ones mentioned above are to a high degree individual and as such not possible to address at this level of research. Regardless of which technology we use, the final physical implementation still needs to be adapted on an individual level. A solution that works for one user cannot be generalised to work the same way for others. Accordingly, statements about the usability of an adaptation are only valid for the person it was made for. As this stage of development has not been reached, user tests were not a suitable method to answer my research questions. The discussion revolves around what the prototype can generally express, but not around interaction quality. Therefore, this thesis neither contains an evaluation of quantitative data.
2 Related Work

In this chapter, I will give a background on different aspects of my thesis project. An introduction is given on the use of 360-degree video technology and relevant existing research projects. I will also give a short overview over common conditions that lead to physical impairments and how they affect the individual’s ability to move. The largest part of the background section is devoted to the discussion of different control units that were developed to assist people with physical impairments.

2.1 360° Videos and Virtual Reality

Over the last few years, 360-degree videos as well as virtual reality viewers have become increasingly popular. This offers the possibility to create immersive environments either completely virtual, or based on recordings from real life. With the commercial availability of specialised cameras, viewing devices (e.g. Oculus Rift, Google Cardboard) and this type of video being featured on the platform YouTube, it has become easy to experience 360-degree videos.

As they add another dimension to the viewing experience through conveying the feeling of being present in the scene, 360-degree videos have been of interest in different fields of research. Studies have been done on mobile devices, addressing social behaviour during 360-degree video experiences, as well as in a medical context (M. Van den Broeck, Kawsar, & Schöning, 2017; Chambel, Chhaganlal, & Neng, 2011; Izard et al., 2017; Neng & Chambel, 2010; Ramalho & Chambel, 2013; Tang & Fakourfar, 2017; W. Van den Broeck, Pamplona, & Fernandez Langa, 2017). While research exists on using 360-degree videos in the context of impairments, the purpose of those applications is different from what we aim to provide (Gelsomini, Garzotto, Matarazzo, Messina, & Occhiuto, 2017).

When preparing the literature research for my thesis, I only found a few studies that touch on similar aspects of 360-degree video as the project I am part of, namely exploring real-life locations through videos. Research has been done on the exploration of cultural heritage sites in an interactive manner – places that are not easily accessible to every user around the world (Kwiatek, 2012; Kwiatek & Woolner, 2009, 2010). Another exploration-focused approach was described by Noronha et al. who developed Sight Surfers, an interactive web application that resembles a video version of Google’s Street View (Noronha, Álvaro, & Chambel, 2012). The researchers used georeferenced videos that are synchronised with a map application so they can be tied to the place where they were shot. On the map, different video trajectories are linked, giving the user the possibility to choose the next clip. Their goal was to enable users to explore places “in other users’ shoes” – an aim similar to that of the Experience Library 360°. A related, commercial product is Google’s Expeditions, an application that provides virtual school trips for children². Yet, none of those approaches include support for people who cannot use conventional devices to navigate 360-degree videos or virtual environments in general. Therefore, they do not cover the aspect most relevant to my own contribution to the project.

---

² Further information: https://edu.google.com/products/vr-ar/expeditions/
Besides the positive aspect about “being there”, virtual reality also comes with certain drawbacks that cause discomfort for the viewer and might even be aggravated in people with certain impairments. Generally, those symptoms are grouped under the term *cybersickness* and include fatigue, eye strain, headache, dizziness, nausea, among other things. Factors that contribute to cybersickness are individual, but also depend on device and task (Davis, Nesbitt, & Nalivaiko, 2014).

### 2.2 Physical Impairments

There is a wide range of physical impairments that affect the individual’s mobility and movement ability in different ways. They can be both congenital (inborn) and acquired, and there are different health conditions that lead to the development of certain impairments. In this chapter, I want to give an introduction into common causes of physical impairments and their characteristics. This section is intended to provide insight into different impairments to create a better understanding of abilities and limitations they can entail, before taking a closer look at what types of devices are used as alternative solutions.

The list below is based on an analysis with focus on physical impairments that can hinder computer and technology use in general (Sears, Young, & Feng, 2009). The authors define the term *impairment* as “a loss or abnormality of body structure or function” that is caused either by a health condition or the person’s context. They distinguish between four categories of physical impairments that directly affect human-computer interaction:

- **Structural deviations.** Body parts that deviate from the norm or are missing completely, due to congenital absence or amputation.
- **Mobility functions.** This includes limited range of bone and joint motion or stiffness. Conditions associated with this type of impairment are osteoarthritis, rheumatoid arthritis, repetitive strain injuries, muscular dystrophy (making it difficult to fully extend joints), and Parkinson’s disease (difficulties bending the elbow).
- **Muscle power functions.** Those are affected by health conditions that cause weakness or paralysis such as cerebral palsy, multiple sclerosis, muscular dystrophy (paralysis does not occur), ALS, locked-in syndrome, spinal cord injuries (paralysis can or will occur)
- **Movement functions.** This includes uncontrolled movement, shaking, muscle spasms or stiffness. Conditions associated with those properties are ALS, brain injury, cerebral palsy, locked-in syndrome, multiple sclerosis, Parkinson’s disease, stroke, and tremors.

It should be noted that different health conditions can affect a person’s mobility, muscle power and movement functions to varying degrees. A more detailed description can be found in the publication (Sears et al., 2009). Some of the conditions mentioned become progressively severe over time (e.g. ALS, arthritis, multiple sclerosis) while others are static (e.g. spinal cord injuries, locked-in syndrome). Some of the conditions can also affect a person’s ability for speech, excluding voice-based interaction from the list of alternatives. There can be big differences in residual abilities despite the “same” health condition, which makes the above classification too inaccurate to be used in the conceptual part of my thesis work. While it gives an initial overview and can be helpful to delimit the useful-
ness of particular control units given a person’s diagnosis, I will continue with a classification that is oriented more towards the actual availability of movements for input. My decision is based on the frameworks described in Chapter 3 and particularly Chapter 3.3 (Ability-based Design) as well as practical reasons. Besides the importance of understanding limitations among target users it is equally important to take on a constructive perspective and work with the existing properties.

As follows from the conditions mentioned above, people with physical impairments face different challenges when interacting with technological devices. Decreased precision and accuracy lead to performance errors, which in turn increases time that is required to make corrections (Trewin & Pain, 1999). In a 2013 study on touchscreen use, Anthony et al. observed e.g. a need for longer dwell times, difficulties with sliding or dragging among people with limited muscle control or tremor, an increased need for precision, and interaction through various body parts (nose, toe, palm). The authors found adaptations in form of device positioning, screen protectors, and (self-made) physical barriers to facilitate interaction. On the other hand, some participants were not able to use the device at all (Anthony, Kim, & Findlater, 2013). This illustrates the diversity among physical impairments of different kinds and the need for individual adaptability.

## 2.3 Control Units and Input Tools

In this section, I will present and discuss different control units that have been developed to support and assist people with motor impairments in using computer applications. The intention is to give an overview of available input tools and their strengths and weaknesses. It constitutes the base for any assessments and concepts described in later chapters. Appendix A contains a list of the controllers that I have examined in more detail, as their characteristics and interaction concepts are of special interest to the project.

The different controller types are loosely arranged by the amount of physical control a person needs to use them, ranging from high to very low. Since accessibility can vary greatly depending on the individual user, that order is not definite. While some devices require certain freedom of movement and precision, others do not require any physical movement at all. On the other hand, movement-free control units come with their own set of challenges. The studies presented in this section were found through performing a free search for controllers and navigation units for people with motor impairments, and by following up on papers included in a systematic review of assistive tools (Tai et al., 2008). For a detailed list of databases, search criteria and keywords see Chapter 4 (Methodology). I would like to point out that my literature study does not satisfy the criteria for a full systematic review (Kitchenham, 2004). Also, the publications discussed do not represent the complete body of research.

Generally, there are at least two different approaches towards accessibility. One way is to change parts of a standard control, e.g. by exchanging a smaller button for a bigger one so it requires less fine-motor skill to use. The other way is to design for accessibility and adaptability right from the beginning (theoretical frameworks covering this are discussed in Chapter 3). The first approach is referred to as retrofitting accessibility. It inevitably involves additional cost and often comes with significant problems (Wentz, Jaeger, & Lazar, 2011). Even though the second approach is preferable, it is more realistic to consider both. While the tools and technologies discussed here were developed as a dedicated
alternative to standard control units, they are in most cases used to make a standard application (e.g. text entry, video games) accessible for users with physical impairments.

Regardless of which approach that is used, the current prototype can be used in order to provide for a greater flexibility, both for retrofitting and for predevelopment accessibility. Through this package any controller can be exchanged for another, as long as they provide at least a similar number of dimensions of expression.

MECHANICAL SWITCHES
Basic mechanical switches are based on an on/off functionality and allow for binary input (Figure 1). There are different kinds of microswitches available, such as pull, push, squeeze, blink, sip or puff (Beukelman & Mirenda, 2013). One important advantage is low cost (Folmer, Liu, & Ellis, 2011). Microswitches can be attached to any body part and modified to make use of a person’s existing motor abilities (Folmer et al., 2011; Lancioni et al., 2004). Multiple switches can be used in combination to actuate different kinds of responses through simple interactions (Lancioni, O’Reilly, Oliva, & Coppa, 2001). Due to this simplicity, they can be operated by people with very limited motor behaviour. They are, for example, being used to facilitate communication for people with multiple and severe impairments of both physical and intellectual nature (Lancioni et al., 2004, 2001).

On the downside, holding down a switch for longer is not possible for everyone, for example due to pain or weak muscles, and some types of switches do not afford for holding at all (Folmer et al., 2011). If the functionality of a switch is limited to on/off only, input becomes too restricted for more complex applications and interfaces. However, switches do gain additional quality when they are used together with switch access scanning technology. This selection technique uses a scanner that iterates through a set of options until the user selects the desired option with the switch (Blackstien-Adler, Shein, Quintal, Birch, & Weiss, 2004; Demasco & McCoy, 1992; Simpson & Koester, 1999). Folmer et al. investigated the usefulness of a hold-and-release mechanism connected to the scanner that allows for mixed non-linear input and overall faster navigation in a video game environment (Folmer et al., 2011).

JOYSTICKS
The joystick is a popular input device used to control video games, based on an airplane’s side-stick. It consists of a stick on a base that can move in different directions, and one or more optional buttons (Christensson, 2007). Examples are Sony’s PlayStation Analog Joystick3 and the DualShock Analog Controller4: both provide the user with two analogue sticks to control a part of the gaming interface. The left stick is used to control the field of view, and the right stick is used to control directional movement5.

---
3 Further information: https://en.wikipedia.org/wiki/PlayStation_Analog_Joystick
However, joysticks are not only relevant in the context of games. They are also used as assistive devices for people with motor impairments, e.g. in combination with a virtual keyboard (Demasco & McCoy, 1992; Song, 2010). Furthermore, a joystick is also an inbuilt part of most power wheelchairs (Figure 2), used to control directional movement, speed and brakes, among other things (Stewart, n.d.). Ahn et al. developed the Octopus Launcher, an interaction framework that reuses the directional controls of a power wheelchair to operate a smartphone (Ahn, Kim, & Suh, 2014). It replaces touch interactions as well as hardware buttons, as the absence of fine-motor skills is a common consequence of different physical impairments. Advantages of this approach are familiarity with the controller, and low additional cost, since it is already part of the equipment (Ahn et al., 2014). Similarly, Wobbrock et al. investigated the reuse of a power wheelchair joystick as a device for text entry (Wobbrock, Myers, Aung, & LoPresti, 2003). As a commercial example, the wheelchair manufacturer Permobil offers a Bluetooth feature for controlling computers and mobile devices with the joystick.

**TOUCHPADS**

A touchpad is a “flat control surface” that is controlled with one or more fingers (Christensson, 2016), see Figure 3. Touchpads are commonly built into laptops, and are also available in the form of touch screens and buttons, offering multi-touch interaction on smart devices, such as swiping and tapping (“Glass gestures,” n.d.). Malu and Findlater did a preliminary study on the accessibility of Google Glass, concluding that the inbuilt touchpad alone is not suitable for most people with motor impairments (Malu & Findlater, 2014). This is mainly due to limited range of motion. A follow-up study showed that switch-based wearable touchpads are a feasible control alternative (Malu & Findlater, 2015). One strength of these touchpads is that they can be attached in different locations, either on the wheelchair or on the user’s body. Another general advantage of touchpads is that they require less impact than, for example, switch-based devices and can also be used by people with weaker muscles. On the downside, some of the participants expressed concerns regarding stigmatisation, as they felt that touchpads would be awkward to use in public (Malu & Findlater, 2015).

**GESTURAL INTERFACES**

Gestural interfaces are powerful control units for people with motor impairments, as they only require gross movements from the user (see Figure 4 for an example interface). They do not rely on accurate finger movements and effective input does not suffer from a lack of fine-motor skills. Other benefits are that there is no need for cables or sensors, a rather small learning effort (only a few gestures need to be memorized), applicability in noisy environments, and easy integration into existing systems.

---

environments, and the fact that the user does not need to wear any device (Jiang, Duerstock, & Wachs, 2014). Since the user does not stand out by wearing additional gear, there is less risk of social stigmatisation. On the downside, motion sensing is complex, and the interface might be very restricted in its possibilities for interaction (Reilly & O’Malley, 1999).

Reilly & O’Malley developed an infrared-based gestural interface that detects movement of different body parts (head, face, feet, hands) (Reilly & O’Malley, 1999). Therefore, it is adaptable to users’ individual physical properties. The system features click actions and decision strategies that allow for more complex input, hence the system can be used for any software application. A similar interface is the Camera Mouse, a system that can be adapted to track different features of the user’s body (e.g. nose or thumb) with a video camera and triggers mouse clicks depending on dwell time (Betke, Gips, & Fleming, 2002). Jiang et al. propose a gestural interface that uses machine vision with a Kinect sensor to recognize user features (Jiang et al., 2014). It tracks movement of the hands and allows for a “lexicon” of ten different hand gestures. A commercial example is the KinesicMouse, an application that recognises a large variety of facial signals through a Kinect sensor to control mouse, keyboard and game controllers.\footnote{Further information: \url{http://kinesicmouse.xcessity.at}}

**HEAD MOVEMENT**

While people with spinal cord injuries tend to have a very limited or no capacity for movement in their limbs, they might have residual mobility in the neck. Therefore, several interfaces and control units were developed that require the user to only move their head to produce input. Strictly speaking, some interfaces controlled by head movement mentioned here also fall into the category of gestural interfaces.

Chen et al. and Evans et al. developed infrared-based systems to replace the keyboard or mouse (Chen et al., 1999; Evans et al., 2000). Those systems require the user to wear an infrared-emitting device on their head and point at the desired key or directional command. The system by Chen et al. includes a tongue touch panel for turning the infrared transmission on or off. Research has been done on using sensors to detect the position and movement of the user’s head in order to control cursor movement or directional commands (Chen, 2001; Kim & Cho, 2002; Pereira, Expedito, De Faria, & Vivacqua, 2011). The system proposed by Chen includes a touch switch that enables the user to click by puffing

---

7 Further information: \url{http://kinesicmouse.xcessity.at}
their cheeks, whereas Kim & Cho use eye blinks to trigger clicks (Chen, 2001; Kim & Cho, 2002).

Another possibility to detect head movement is computer vision. Perini et al. developed the *Face Mouse* that uses a webcam for input (Perini, Soria, Prati, & Cucchiara, 2006). It tracks facial features of the user to move the cursor and relies on dwell time for triggering mouse clicks. Similarly, the systems proposed by Reilly & O’Malley and Betke et al. can also be set up to use movements of the head as an input (Betke et al., 2002; Reilly & O’Malley, 1999). *Click Control* (see Figure 5), a camera-based mouse replacement system was developed to reduce accidental clicking (Kwan, Paquette, Magee, Lee, & Betke, 2011). The user performs gestures with their head to confirm and choose a type of click (left, right, double). Although not targeted directly at users with physical impairments, Pai et al. propose a hands-free navigation system for 360-degree video that relies on a combination of head movement and eye tracking (Pai et al., 2017).

**VOICE-BASED INTERFACES**

As people with motor impairments are very restricted in their range of motion, there is only little physical potential to be exploited for assistive tools. At the same time, many of them can use their vocal system fully (Bilmes et al., 2005). The human voice holds a range of characteristics that allow for faster and more complex input than other alternatives. Bilmes et al. developed the *Vocal Joystick*, a voice-based control engine (Bilmes et al., 2005). It utilises several dimensions of the human voice like loudness, vowel quality, and discrete sounds to control movement speed, direction and button presses of a mouse cursor (Figure 6). Similar approaches using different vocal qualities can be found (Igarashi & Hughes, 2001; Olwal & Feiner, 2005). Based on the *Vocal Joystick* engine, a drawing application controlled by voice commands was developed (Harada, Wobbrock, & Landay, 2007). A video game controller developed by Pereira et al. uses single syllable commands in combination with head movements and provides the same functionality as a joypad (Pereira et al., 2011).

![Figure 7: Sounds used by the Vocal Joystick engine. (Source: Harada et al., 2007)](source:haradaetal2007)

A more minimalistic type of voice input is humming. As it does not rely on vowels, recognition is not biased by the user’s native language (Bilmes et al., 2005; Poláček & Mikovec, 2010; Won, Lee, & Smith, 2007). Poláček and Mikovec developed a method to control different clicking actions through humming, based on pitch and length of a sound (Poláček & Mikovec, 2010). In a study by Sporka et al., humming was found to be faster and more accurate than speech input, but required more training (Sporka, Kurniawan, Mahmud, & Slavík, 2006). An even less obtrusive kind of control input is *subvocal* or silent humming, as investigated by Won et al. (Won et al., 2007). It does not require any conceivable sound at all, instead bone-conducted sound is received through an insertion microphone in the ear.

**EYE WINKS**

Interfaces that only require the user to wink their eyes for input can be used by people with severely limited or no mobility. Benefits of such systems are that they are nonintrusive, comfortable for the user, and that they work in noisy environments (Grauman, Betke,
Lombardi, Gips, & Bradski, 2003; Miluzzo, Wang, & Campbell, 2010; Shaw, Crisman, Loomis, & Laszewski, 1990). They need no or little additional equipment and are therefore usually low in cost. Most eye wink interfaces require the user to be able to execute voluntary winks, some even of a single eye, as detection mechanisms rely on wink duration to determine the intentionality of the input action.

Shaw et al. investigated an infrared-transmitting module that can be attached to the user’s eyeglasses (Shaw et al., 1990). It detects reflections from the sclera or eyelid and allows for a small set of commands. BlinkLink automatically detects winks and measures their duration to trigger mouse clicks (Grauman et al., 2003). The system combines the binary input from winks and allows for a small vocabulary. It does not require cursor motion, but instead relies on a scanning mechanism. A similar system proposed by Mamatha & Ramachandran (2009) allows for a larger number of commands (see Figure 7). Miluzzo et al. developed the EyePhone, a hands-free system that relies on eye movements and winks to interact with a smartphone (Miluzzo et al., 2010). It utilises the inbuilt front camera of the phone to track eye movements, and enables the user to click a button by winking at it. Matthies et al. used data recorded from a commercial neuro headset to control iPad functionalities, such as playing and pausing songs through eye winks (Matthies, Antons, Heidmann, Wettach, & Schleicher, 2012). Furthermore, eye gestures have been researched in the context of video games and their influence on game mechanisms, e.g. (Velloso & Carter, 2016).

EYE GAZE
Like eye winks, eye gaze requires very little physical movement from the user, and works even without having to control blinking. Other advantages include naturalness, speed, low effort, comparably low cost, and the fact that eye movement provides context for other types of interaction (Smith & Graham, 2006). It is counted as a promising input method for people with severe motor impairments and has been studied in different contexts, for instance the control of power wheelchairs (Lin, Ho, Chen, Chiu, & Yeh, 2006; Nguyen & Jo, 2012). Regarding computer use, an early example is EagleEyes, a system that enables the user to control different applications through eye gaze (Gips & Olivieri, 1996). Also, the use of eye gaze for games has gathered attention among HCI researchers over the past decade (Bulling, Roggen, & Tröster, 2008; Sundstedt, 2012; Velloso & Carter, 2016). Overall, it appears that gameplay performance strongly depends on the type of game and the acts eye gaze is used for (Vickers, Istance, & Smalley, 2010), a finding that can be transferred to other types of applications like the interactive video player of the Experience Library 360°.
An important issue with eye gaze input is the *Midas Touch* problem: if a command is issued just by looking at something, the user cannot look at the interface without involuntarily making an input (Jacob, 1991). It is often approached by implementing dwell time as a parameter, so the user is required to fixate a place on the screen long enough for the action to be triggered. This is, for example, used in gaze-based drawing software (Hornof & Cavender, 2005) to switch between looking and drawing mode or to click buttons.

**OTHER INPUT OPTIONS**

Besides the input technologies described in the previous paragraphs there are methods available that require no physical movement from the user. This includes measurement of muscle contractions, changes in skin conductivity as well as recordings of brain activity. I do not consider the methods described in the following directly relevant to the Experience Library project. This is due to reasons such as limited availability, high learning curve, discomfort, and some of them being very intrusive. Still, I decided to include them for the sake of completeness.

One way of generating input without having to move a body part is *surface electromyography* (SEMG). Myoelectric signals are produced through the contraction of skeletal muscle and can be detected with small electrodes that are placed on the surface of the skin, right above the contracting muscle. This type of signal is commonly used to control prosthetic and orthotic limbs, but it can also be applied to facilitate communication for people in a locked-in state (e.g. due to ALS) (Gryfe, Kurtz, Gutmann, & Laiken, 1996). Different systems have since been developed, such as a SEMG-controlled telephone (Chen, Lai, Luh, & Kuo, 2002) and discrete mobile interfaces (Costanza, Inverso, & Allen, 2005). The technology has also been used in the development of hands-free control systems for electric wheelchairs (Felzer & Freisleben, 2002; Oonishi, Sehoon Oh, & Hori, 2010). A disadvantage of the use of electrodes is that an electrolytic gel needs to be applied to the skin to enable conductivity. This results in drying out the skin which, in turn, reduces signal quality. Also, the technology depends on the strength of the muscle contraction (Tai et al., 2008).

Another method that has been researched to facilitate communication for locked-in individuals is *electrodermal activity* (EDA). A pair of electrodes is used to measure changes in skin conductivity which are elicited by the autonomic nervous system. The autonomic nervous system regulates unconscious actions (heart rate, sweating, pupil dilation) as a response to different types of stimuli (Tai et al., 2008). Studies have shown that changes in skin conductivity can potentially be used to produce binary output (Moore & Dua, 2003; Tsukahara & Aoki, 2002), but training is necessary to consciously raise and lower EDA signals (Moore & Dua, 2003).

Furthermore, there are different methods available that generate input from brain wave activity. One technology is *electroencephalography* (EEG) which uses surface electrodes...
to record brain activity from the scalp. EEG-based brain-computer interfaces (BCI) work without the activation of any muscle. Tai et al. describe two categories: potentials evoked through external stimuli (e.g. to detect the point of gaze using elements flashing at different frequencies), and potentials that the user can learn to control with adequate training (Tai et al., 2008). Thus, imagined movement can be used to generate the same response as actual physical movement, enabling the user to control interfaces with their brain only. EEG-based BCIs are used to control home environments (Goo, Suh, Cho, Park, & Hahn, 2005) as well as video games (Kapeller, Hintermüller, & Guger, 2012; Lotte, 2011). Drawbacks of EEG include, among other things, a steep learning curve, signals that are easily disturbed by environmental factors, currently very limited possibilities for interaction, and long-term discomfort caused by gel-based electrodes (Lotte, 2011; Tai et al., 2008).

Lastly, Tai et al. mention two types of brain-computer interfaces that are much more invasive as they require surgery. Electrocorticography (ECoG) requires electrodes to be placed on the brain’s surface. Intracortical recordings rely on electrodes placed in the brain’s cortex to directly access neuron activity (Tai et al., 2008). While both technologies have produced faster more accurate signals than EEG, they are out of question for my thesis work due to their availability and invasive nature.
3 Relevant Frameworks and Concepts

In this chapter, I will discuss the theoretical frameworks and concepts that build the ideological base for the Experience Library 360°. Three different frameworks are presented that are concerned with designing for people with impairments of all kinds. I will briefly discuss the origin and intention of each framework, explain the fundamental principles and guidelines for design. Further, I will go into points of criticism and how my own research project fits into the context.

3.1 Non-Excluding Design

The term Non-Excluding Design was coined by my teacher and supervisor Lars Oestreicher (Oestreicher, n.d.), and it is also the name of the same-titled course at Uppsala University. With the common goal of designing for inclusion and equality, Non-Excluding Design and Universal Design share the same basic principles. Universal Design claims by right of its name that it is feasible to design one product to be used equally by everyone, but this is not always the case. Non-Excluding Design, however, targets the design process in order to exclude as few people as possible from performing a task. Designing to enable the user is central to the approach, as well as anticipating and reducing the negative effects of a design. While it is almost impossible not to exclude anyone at all, the term Non-Excluding Design was chosen to raise awareness for impairments and societally inflicted disabilities. It can be considered a continuous work in progress. In the following, I will outline the different parts of the Non-Excluding design process.

- **Awareness of disabling factors.** Disability is most often rooted in society, in that it fails to accommodate for certain impairments. It is important to make people in diverse contexts aware of impairments, in order to help them act appropriately.

- **Observation.** The designer needs to understand individual needs and difficulties to design a helpful product. The best way to do this is by observing the use of technology in real-life contexts, similar to the user-centered design process.

- **Testing.** The designer should try out their own product under different conditions, such as testing the incline of a wheelchair ramp. Even though this only produces small insights, it can be enough to identify flaws that would have otherwise not been obvious to the designer.

- **Acceptance.** In order to successfully work with people with impairments, the designer needs to accept and respect their properties as normal, and take their feedback seriously.

- **Understanding.** As mentioned above, designers only get a limited insight into the perspective of a person with one or multiple impairments. It is important to understand the scope of the situation that is being designed for, even though full understanding is hard to develop from the perspective of an able-bodied person.

In certain aspects, the Non-Excluding Design framework differs from common practice in the HCI field: Generally, there is less of an emphasis on seeing problems in the design of a certain interface, since this can be considered a rather negative perspective. Instead, the focus lies on finding new possibilities and alternatives that make use of what is there, often
by providing alternative ways of interacting. In this regard, Non-Excluding Design overlaps with the mindset proposed in the Ability-based Design framework (Wobbrock, Kane, Gajos, Harada, & Froehlich, 2011) in that it emphasises ability over disability. An important aspect of Non-Excluding Design is the focus on finding excluding factors in a certain design, and providing the means to eliminate them.

### 3.2 Universal Design

The term *Universal Design* is used synonymously with *Design for All*. In its origins, Universal Design was a movement initiated in response to a change in demographics, namely a growing number of people with impairments in the society. Also, the legal situation changed: From the 1970s onward, laws were passed to prohibit the structural discrimination of people with impairments (Center of Universal Design, 2008). To provide guidance to different design disciplines, a group of architects, engineers and design researchers at North Carolina State University developed the *Principles of Universal Design* (Connell et al., 1997). The seven principles are outlined in the following paragraphs.

- **Equitable Use.** A product should be equally usable for every person, despite their abilities. If a design cannot be used in the same way by everyone, alternative ways of use should be available. Also, safety, security and privacy should be ensured independent of user context. This design principle aims at reducing stigmatisation and separation based on ability. It also includes making a product appealing to all possible users.

- **Flexibility in Use.** A product should accommodate for different users’ preferences and abilities. This does not only include impairments, but also properties like left-handedness. The design should support the user in executing precise and accurate actions, as well as it should be adaptable to individual pace.

- **Simple and Intuitive Use.** A product should be intuitive to use. It should be easy to understand, despite of previous knowledge, literacy and language skills, and the user’s level of concentration. The design should not be more complex than necessary and match the user’s expectations. This also includes appropriate feedback. Generally, information should be presented in a consistent manner.

- **Perceptible Information.** It should always be possible to perceive information through different senses, e.g. not reducing them to the visual or auditory channel only. This ensures that users with different sensory abilities have equal chances to take in information. Information should be legible and stand out from the background. Also, it should be possible to process with different assistive devices, such as screen readers.

- **Tolerance for Error.** A product should be designed in a way that hazards and undesired consequences are minimized, e.g. by making certain functionalities less accessible or eliminating them overall. Considering that humans always make mistakes, it should not be possible to execute a dangerous action. If this cannot be avoided through design, warnings need to be provided.
- **Low Physical Effort.** The use of a product should put minimal stress on the user. It should be designed in an ergonomic fashion and require as little movement as possible. Depending on the physical properties of different users, this entails a need for adaptability.

- **Size and Space for Approach and Use.** The design of a product should allow for different body sizes and postures. Regardless of their physical properties, the user should be able to reach all elements that can be manipulated. Also, there should be enough space for personal assistance, or to fit in assistive tools.

As stated in the name, this design framework focuses on the creation of a universal solution; one that fits all users equally well. While most of the principles are closely linked to those of user-centered design, the goal of universality adds to the complexity of the design process. As individual abilities vary from person to person, it is difficult to anticipate all the possible properties that a design needs to consider. Wobbrock et al. criticise the framework for producing “one size fits all” solutions, which may not be appropriate in the context of computer systems (Wobbrock et al., 2011). Instead, design can make use of the adaptable qualities of today’s technology. Another point of criticism is that the Universal Design framework only provides a list of principles, but no guidance as to how they can be implemented in practice (Oestreicher, n.d.).

Even though the Universal Design framework is not tailored specifically to the design of computer systems or interactive interfaces, it still provided important input for the work of this thesis. It builds the base for the other frameworks mentioned in this section. Looking at it through the eyes of a student educated in Human-Computer Interaction, most of the above principles can be considered common sense. Yet, they form a basic checklist for accessibility concerns that informed each phase of the design cycle.

**UNIVERSAL DESIGN IN ICT**

Following up on the criticism outlined above, Stephanidis et al. state that “accessibility should be a design concern, as opposed to an afterthought” (Stephanidis, Benyon, Crerar, Wilkinson, & Marcus, 2001). This so-called reactive approach often happens unguided and does not follow a standard procedure. It is problematic as technology advances quickly and causes extra effort every iteration. The authors present *User Interfaces for All* as an approach to transfer the principles of the Universal Design framework to the context of human-computer interaction. They promote a user-centered approach to HCI that includes the user in the whole design process. This is to ensure that products “can automatically address the possible range of human abilities, skills, requirements and preferences” (Stephanidis et al., 2001).

Consequently, the process of designing user interfaces for all requires the designer to quit studying the “average user”, which is common among HCI practitioners. In today’s information society, there is not only one single user group constituted of computer-literate, performance-oriented, able-bodied workers. Additionally, there are differences in interests, personal preferences, and information needs (Stephanidis et al., 2001). Hence, designers need to consider a much more heterogeneous target group, leading to a need to revise established methods and tools. This does not only apply to changing the development process to factor in the need for adaptation, but also to the evaluation phase since the adaptive qualities of a prototype need to be tested in context. As in my own research project, the issue arises that any final solution needs to be tailored to each user individually.
and that the usefulness of a system cannot be fully generalised. While it is possible to test general characteristics for adaptation, it cannot be predicted how well it works for the individual in practice.

While the UI for All framework is very relevant as it discusses the challenges of Universal Design in the context of HCI, the focus is on accessibility and usability, and not yet on providing an enjoyable full user experience. Even though it emphasises human diversity, the term “disability” is used (Stephanidis, 2013), indicating a focus on weaknesses, not strengths. A more recent approach to this issue is presented in the following section.

### 3.3 Ability-based Design

The Ability-based Design framework was initially presented in a paper by Wobbrock et al (2011). It builds on different approaches that target inclusion and accessibility in design, such as Universal Design and Design for All. According to the authors, previous frameworks tend to focus on disability, whereas Ability-based Design emphasises the user’s abilities and how they can be used to allow for the best performance possible – what can a person do? Thus, it can be regarded as a further development of existing ideas.

Among more traditional approaches, it is common to treat software as a given element, and to develop specialized hardware to accommodate for different abilities and properties. In the Ability-based Design framework, this process is reversed: software should be designed to enable the use of “off-the-shelf” hardware. While this may not be possible in the context of every impairment, the use of common assistive input tools is facilitated. Adaptive interfaces play an important role in the framework, even though they are not considered mandatory. The initiators of Ability-based Design developed seven guiding principles that are outlined in the following paragraphs.

- **Ability.** The designer focuses on ability, not on dis-ability. The focus should lie on what potential users are capable of doing.
- **Accountability.** Poor performance is not considered the fault of the user. In order to counteract it, the system will be changed, instead of pressuring the user to adapt.
- **Adaptation.** An interface should either be self-adaptive or adaptable by the user in order to provide the best possible match.
- **Transparency.** At all time, the control over the interface lies with the user. This includes functions such as reverting, overriding or restoring data or settings.
- **Performance.** The system can observe and monitor user performance in order to predict it.
- **Context.** The system has the ability to sense context, and therefore can anticipate its’ effects on the user’s abilities.
- **Commodity.** The system is low in cost and makes use of readily available hard- and software.

It is to be noted that the first three principles are always required in the ability-based design process, whereas principles four to seven are optional. Principles five and six speak for the design of intelligent and adaptable systems (as well as principle three, to an extent). Overall, the design principles target the designer’s way of thinking in the context of inclusion. The
Ability-based Design framework expresses a positive outlook on the topic of impairments, in that it focuses on strengths instead of weaknesses. The common attitude of regarding an impairment as a deficiency makes it impossible to create an equal and non-discriminative environment. Often, the need for adaptation falls back onto the user who might not have the means to make it possible by themselves.

While the framework promotes a revised approach towards designing for equality, I would argue that is most useful as a guide for the conceptualisation and development of new products. Implementing something after release takes work, which is why it might be difficult to apply the framework in its entirety to existing designs. Sometimes, it is not about developing a completely different product, but about enabling people to take part in activities that use already popular devices. Examples of this are video games and smartphone applications (Ahn et al., 2014; Pereira et al., 2011).

In total, the Experience Library project fits well into the context of the Ability-based Design framework. That is, the ideas behind the project are reflected by principles one to three. As the video library does not cover the scope of a whole computer system, it does not include functionalities to monitor performance or to sense context. The main focus lies on creating a user-adaptable interface, in order to make it work with “off-the-shelf” devices. While economy factors were not considered in the current version (see Chapter 1.3), I would argue that there is potential to also satisfy the Commodity principle in future iterations. Also, the Transparency principle will necessarily be relevant in future versions, as soon as a more sophisticated user interface is designed.
4 Methodology

In this chapter, I will discuss the methods that were used in this thesis. To fit the task, methods associated with both user-centered design as well as technical human-computer interaction were used. First, I will give an overview of how the methods were applied over the course of the different project phases and then critically examine them in more detail in the following subchapters.

4.1 Overall Methodology

The Universal Design framework builds on the basic principles of user-centered design (Stephanidis, 2013). This means that it is roughly structured into the phases of observation, idea generation, prototyping and testing (Norman, 2013). The structure of this research project, however, deviates from that process in certain aspects. This is due to the overall goal being different – proof-of-concept rather than a finished solution – and, as a result, the main tasks relying less on user input at the current stage of development. Since there is a certain innovative component to the project, the process had to be tailored to meet that challenge. Proof-of-concept is a method of technical HCI that is considered the most fundamental approach for validation in innovation-driven research (Hudson & Mankoff, 2014). Because of the added technical component of my objectives, it made sense to build parts of the process around that method. An overview over the project phases is given below.

![Diagram showing the adapted design process and methods used](image)

**Figure 10: Overview of the adapted design process and methods used.**

1. **Understanding.** The first phase was similar to the observation phase in the user-centered design process. I chose the term “understanding” instead, as the goal was to get an overview over research on different technologies and artefacts relevant to the project. A literature study was chosen for this phase. Unlike described by Norman, no research in the style of “applied ethnography” needed to be performed. The target group and its limitations were already well-known by the beginning of the project, and certain requirements followed as a logical consequence. Also, the design problem approached here is for the most part one of physical accessibility and less a question of conflicting mental models or the like.

2. **Conceptualisation.** This phase resembled the idea generation phase of the user-centered design process. Based on previously gained insights, criteria for interaction and the qualification of control units were collected, grouped and formed into concepts. Again, as the final goal of the design was defined beforehand, idea generation was rather constrained. The aim of this phase was to identify minimum requirements and to construct plausible interaction models for the system to build on.
3. **Preliminary evaluation.** In this phase, a small and informal test of the previously sketched concepts was performed. An interview with a person of the target audience was chosen for the evaluation. This was done to discuss my findings from a more personal and real-life perspective, as opposed to the theoretical analysis done beforehand. Also, it served to verify the ideas that would build the base for the development of the control system.

4. **Proof-of-concept.** After forming theoretical concepts and validating them on paper, the question for the last phase was “does this work in practice?”. As even small details can break the success of practical implementation, the next reasonable step was to build the part of the project that my research focuses on. A prototype of the control system was developed to explore the re-mapping of commands according to the models formed in phase two. As the overall goal of this thesis is met with the successful implementation of said system, no testing phase as found in the user-centered design process was conducted.

### 4.2 Literature Study

The literature study was conducted in the first phase of the thesis work. In order to get an overview over the current stage of research regarding assistive devices and control units for different interactive media, the choice of databases was kept general. Google Scholar was searched to access a broader range of results that are not restricted to subjects related to information technology. As a complement to this, the ACM Digital Library was used to access more up-to-date research, especially in regard to the topic of different control units for people with physical impairments.

Keywords used in the search include the following: “motor impairments navigation controls”, “motor impairments controls”, “impairments navigation controls”, “impairments controls”, “motor impairments games”, “impairments navigation controls”, “gestural interface impairments”, “joystick motor impairments”, “eye gaze navigation”, “eye winks game”, “eye wink game controller”, “eye tracking game controller”, “speech control video games”, “humming control”, “humming control video games”.

Both older and newer studies were used in the literature search to facilitate a good understanding of the research area. Journal articles as well as conference papers were included. As there are many kinds of assistive tools and alternative control units for people with different impairments, studies covering impairments and restrictions relevant to the project were selected. This means that only solutions for people with motor impairments were included. Research on devices targeting other physical properties like blindness or deafness was excluded from the search, unless there was an overlap with the project focus. The results of the literature search were categorised by the kind of impairment investigated, as well as the type of technology used. Chapter 2.3 reflects this categorisation.

It is to be noted that the literature study is not a complete systematic review. It focuses on single examples of, sometimes very specific, solutions. This is due to the nature of the research area, characterised by highly individualistic properties of target users and great variety in their needs and preferences. The literature study should neither be regarded as a representative collection of all currently available control units, as research results were chosen in terms of their relevance to the problem at hand. Secondly, most of the control
units investigated were developed in a research context, hence they are not available off-the-shelf. While the results of the literature study show existing possibilities, not all of the alternatives might be obtainable in a real-life context. Also, as mentioned in 1.3, cost was not considered as a limiting factor. Similarly, not all existing solutions are available for use in this project.

On the upside, a diverse range of studies was examined, and different forms of physical impairments were covered, including varying extents of freedom of movement. Many of the approaches were developed in collaboration with people with impairments, thus addressing different needs, properties, and limitations. In sum, the studies collected contain a greater variety of user input than I could have likely gathered from conducting preliminary interviews myself. Still, I would argue that it is necessary to validate my assessment of the findings with at least one person from the target group. As a designer, I have limited insight and understanding, despite the effort made to comprehend the challenges that target users are facing.

4.3 Target User Interview

An interview with a potential target user was conducted after assessing the findings from the literature study. The interviewee has different motor impairments and uses an electric wheelchair controlled with a joystick. She has very little mobility in her left arm, and limited mobility in her hands. For actions like looking around, the wheelchair is turned instead. Besides being part of the target population, the interviewee works for an organisation that belongs to the International Independent Living Movement. Due to this, her expertise is not limited to personal experience only. She was asked about experiences with different devices and tools and her opinion about the concepts described in Chapter 5.

The interview was held in a semi-structured way, following loosely a set of questions prepared beforehand.

The interview was intended to falsify or validate the findings from the literature study and the conceptual base of the control system. It should help the designer better understand the requirements for the subsequent work. The results are discussed in Chapter 5.4. As only one person was interviewed at this stage, they cannot be viewed as representative of the target group. A full representation of the target population is difficult to reach either way, as people with motor impairments are a very heterogeneous group. It is debatable whether it would have been necessary to interview a larger group of people at this stage of development, since there will never be a “one fits all” type solution. Furthermore, recruiting participants from this specific target group can be challenging.

4.4 Prototyping

Beaudouin-Lafon and Mackay define the term prototype as a “concrete representation of part or all of an interactive system” (Beaudouin-Lafon & Mackay, 2012). For this thesis, the goal was to build a prototype as a proof-of-concept to show that the use of alternative control units to navigate 360-degree videos is feasible. The prototype was intended to validate the interaction concepts described in Chapter 5 and give an idea about how the control API could be structured. In a basic manner, it also visualises the consequences of user interaction. For a detailed description of the building process see Chapter 6.
Prototypes and prototyping techniques can be described using four dimensions: **Representation, Precision, Interactivity, Evolution** (Beaudouin-Lafon & Mackay, 2012). The representative dimension of a prototype can range from simple paper sketches to sophisticated computer simulations. The authors also distinguish between **offline** and **online** prototypes; online prototypes being everything that requires a computer to run on (also called **software prototypes**). As a consequence, they are necessarily high-fidelity and require more time and effort than offline prototypes. The prototype created as part of this thesis is an online prototype, since it includes a computer program and several input devices. While this kind of prototype is uncommon in early-stage design, I would argue that the complexity and fidelity of a prototype depends on the context of the project. In this case, the goal was to validate the feasibility of the technical implementation which required actual development work. Testing the concepts with a mock-up or paper prototype would not have been sufficient.

A point of criticism brought up by Beaudouin-Lafon and Mackay is that online prototypes come with constraints caused by the environment they are developed in, limiting ideation and exploration of the design space. In the case of this thesis, design space and goal were already defined by the beginning of the work, as well as functional requirements were set before beginning the prototyping process.

The dimension of **Precision** denotes the level of detail at which the prototype is to be evaluated. It replaces the terms low-fidelity and high-fidelity, as those include how the prototype compares to the final system. At the same time, Beaudouin-Lafon and Mackay argue that a prototype can be detailed but imprecise. This means that only those details relevant for the upcoming evaluation cycle are implemented in full, whereas others are left open for future iterations. In the prototype at hand, this is the case for the user interface. It was only sketched out roughly, while the technical aspect was implemented with more detail and care. The visual side of the prototype exists to show the effects of interaction, but it will not be subject to evaluation in the next cycle.

The interactive dimension of a prototype describes the extent to which the user can interact, from “watch only” to a fully interactive representation. Additionally, a prototype might possess limited interactivity while providing a life-like feel of the experience (fixed-path), or it could demonstrate many interactions at the cost of system performance (open prototype). Eventually, the extent of interactivity depends on the context of use. This also includes accessibility to different users. The prototype built for this thesis could count as (almost) fully interactive on a fixed path when operated by a person with average mobility. As the connection of more specialised input devices is still pending, its interactivity is reduced to “watch only” for users with certain motor impairments.

Finally, the evolutionary dimension of a prototype describes its life span, ranging from throwaway or rapid prototypes to iterative and evolutionary prototypes. While rapid prototypes are not meant to be used as part of the final system, they can be online too, and therefore requiring a certain effort to build. This may sound counterintuitive at first, but Beaudouin-Lafon and Mackay argue that sometimes a precise software prototype is needed to discover interaction problems. The prototype built within the scope of this thesis could therefore be classified as a rapid one. It is not aimed at becoming a part of the finished system, but was intended to show possibilities for an actual implementation in a simplified way. This is also mirrored in the tools and technologies that were used to build it.
5 Conceptualisation

In this chapter, I will discuss conceptual considerations on how different types of control units can be utilised in the Experience Library 360°. I will begin with an overview of functionalities that are required to successfully control 360-degree videos, and discuss the findings of that examination. Then, I will give an assessment of the control units introduced in Chapter 2.3. In order to build up a taxonomy, I developed two concepts for classification, the Dimensions of expression and Maximum number of expressions. Further, I will give an account of an interview with a target user that was intended to verify my ideas. Lastly, I will discuss an approach for selecting control units based on the findings from the conceptualisation phase.

5.1 Functional Requirements

360-degree videos include more functionalities than regular videos. The most important aspect is that they allow the viewer to look around in every direction. Instead of a level surface, the screen can be imagined as the inside of a sphere on which the video is displayed. One goal of the Experience Library project is to provide the user with decision points within a video experience. In order to have full control over the possibilities offered, a certain set of commands is needed.

The following sections can be read as a requirements analysis for interaction with 360-degree videos. I will give an overview of the functionalities of 360-degree videos and classify them by their importance. This serves as a preparation for determining the necessary inputs and expressive power a control unit has to provide in order to be sufficient. Subsequently, I will examine the different control alternatives with regard to the expressive power they provide, and present arguments for the selection of those that are relevant to the control system for the Experience Library.

MOVEMENTS AND ACTIONS

There are different movements and actions that can be performed to control a 360-degree video. They were split up into looking, movement, navigation and meta acts and are listed in Table 1. Looking acts describe how the user orients themselves in all directions of the visual space. This category comprises the functionalities that are special to 360-degree videos. Movement acts describe how the user moves along the timeline of the video. This includes going forward and back, as well as standing still, which equals pausing the video altogether. In the paused state, the user retains the possibility to look around the visual space while all movement in the environment is frozen, turning the experience into an interactive photo. Navigation acts are the choices that the user can make about the path they want to follow. They occur at the intersection between different video clips. Lastly, meta acts describe “classic” video controls, such as jumping to a specific point in the timeline. Those are separated from the immersive aspect of the video and it should be considered that they might take away from the degree of immersion of the whole experience.
<table>
<thead>
<tr>
<th>Movement or activity</th>
<th>Type of act</th>
<th>Description</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rotate field of view to the right</td>
<td>Looking</td>
<td>Enables the user to swing their field of view to the left.</td>
<td>Primary</td>
</tr>
<tr>
<td>Rotate field of view to the left</td>
<td>Looking</td>
<td>Enables the user to swing their field of view to the right.</td>
<td>Primary</td>
</tr>
<tr>
<td>Rotate field of view up</td>
<td>Looking</td>
<td>Enables the user to swing their field of view up and down.</td>
<td>Secondary</td>
</tr>
<tr>
<td>Rotate field of view down</td>
<td>Looking</td>
<td>Enables the user to swing their field of view down.</td>
<td>Secondary</td>
</tr>
<tr>
<td>Play from beginning</td>
<td>Meta</td>
<td>Plays the video from the start from any time marker.</td>
<td>Secondary</td>
</tr>
<tr>
<td>Select a specific point on the timeline</td>
<td>Meta</td>
<td>Jumps to a chosen marker in the video’s timeline. This feature could be useful when the video is interrupted and continued at a later point in time, for example when a user fatigues easily or cannot focus for the duration of the whole video.</td>
<td>Secondary</td>
</tr>
<tr>
<td>Adjust speed</td>
<td>Meta</td>
<td>Enables the user to adjust walking speed in the video, for example if they wish to take a closer look at something in passing. It could be interesting to implement an option for time lapse/slow motion, but it could also make the experience less immersive.</td>
<td>Tertiary</td>
</tr>
<tr>
<td>Fast forward/backward</td>
<td>Meta</td>
<td>Winds the video forward or backward to the desired time marker. This action might not be needed, if the user can select a specific time marker to jump to. More user feedback needed to decide if this action is necessary.</td>
<td>Tertiary</td>
</tr>
<tr>
<td>Move forward (Play)</td>
<td>Movement</td>
<td>Starts playing the video; equals with moving forward through walking, running, driving or any other type of movement.</td>
<td>Primary</td>
</tr>
<tr>
<td>Stop moving (Pause)</td>
<td>Movement</td>
<td>Pauses the video at any point. This is not exactly the same as if the user would just stop moving, as the whole visual environment is paused and not only the viewer’s movement.</td>
<td>Primary</td>
</tr>
<tr>
<td>Move backward (Reverse)</td>
<td>Movement</td>
<td>Enables the user to move in the opposite direction on the same path. This would require backtracking the path in a separate video.</td>
<td>Secondary</td>
</tr>
<tr>
<td>Choose path to the left</td>
<td>Navigation</td>
<td>Enables the user to choose a path to the left when they arrive at a crossing, which means that another video sequence is played in the following.</td>
<td>Primary</td>
</tr>
<tr>
<td>Choose path to the right</td>
<td>Navigation</td>
<td>Enables the user to follow a path to the right.</td>
<td>Primary</td>
</tr>
<tr>
<td>Choose path straight ahead</td>
<td>Navigation</td>
<td>Enables the user to follow a path that leads straight ahead.</td>
<td>Primary</td>
</tr>
</tbody>
</table>

Table 1: Functional requirements for controlling a 360-degree video.
Prioritising Controls

It needs to be considered that a user might be severely restricted in the movements they can do, or a control unit might not have enough input options for all of the commands listed in Table 1. Therefore, it is necessary to determine which acts are indispensable to experience 360-degree video as intended, and which ones could be omitted. For example, shifting the field of view horizontally is a vital aspect of 360-degree video and needs to be included. If left out, the user would see a regular video from a fixed point of view. Also, the user needs to be able to choose the path on which they would like to continue once they arrive at a decision point. Playing and pausing the video are counted as acts crucial to the viewing experience.

While the act of going back on the same path can be considered valuable and could add to the viewing experience, it is not indispensable. Acts like “fast forward” or “select a specific point on the timeline” could be left out, as they are not directly part of the experience. Omitting them could contribute to the degree of immersion of the video, since they could be considered “unnatural” or to interrupt the flow. Further, there is no crucial need for the user to be able to select playback speed. Yet, this could be useful, as small decreases or increases in speed can be helpful while not being very noticeable. The ability to jump to a point on the timeline could be useful if the viewer has to stop the video and wants to continue watching from where they left off at a later occasion.

To make the distinction clear, each act was assigned a tag that specifies its importance. Acts tagged as primary are considered essential for controlling the 360-degree video experience. In other words, they make up the minimum requirements that any control unit needs to fulfil in order to be a standalone option. There is a total of seven acts that need to be included, “play” and “pause” being possible to implement as a toggle switch. The minimum requirements are made up of looking acts, movement acts and navigation acts. Acts tagged as secondary include further looking acts, movement acts and meta acts. They are not essential to controlling the viewing experience but could be seen as “nice to have”. Lastly, there are meta acts tagged as tertiary, as they can be left out without taking much freedom from the user. As mentioned above, they could even be detrimental to the degree of immersion, hence their usefulness could be worth evaluating through user testing.

Wording of the Input Options

The labels for the different acts presented in Table 1 were chosen to be neutral so they do not indicate a certain type of locomotion. Other than walking, videos could include bicycle rides, swimming or skiing. Overall, comprehensible and meaningful labels should be available for all input options so that they can be easily memorised. They should clearly express the type of act that is being controlled, to help the forming of correct mental models. As the control system is still in its initial stage, wording as a design decision is not of high relevance yet. Further annotations on the wording of labels are discussed in Chapter 7.2.

5.2 Expressive Power of Control Units

Different controllers allow for different numbers of input options. Some of the devices and tools presented in Chapter 2.3 lend much freedom to the user to control hardware and software applications, while others only allow for binary input. As this does have to do with the user’s physical properties in some cases, it also greatly depends on the controller itself.
Therefore, I decided to introduce the term *Dimensions of expression* as a description of interactive quality. It points at the flexibility of each controller in regard to the room it leaves for the designer to implement a variety of commands – it includes the different input options of a controller and the resulting possibilities for interaction. In Chapter 5.1, a minimal set of seven commands was identified to be essential for the user to be in control of their viewing experience. Yet, there are another seven commands that can be considered useful or beneficial to the experience. Since different control units fit different physical properties, it is likely that a trade-off still needs to be made. A controller that hosts all 14 input options is optimal in terms of simplicity, but not necessary.

In order to select a set of control alternatives that work well with the intended concept, it is important to determine how many dimensions of expression each alternative allows for. While it may be difficult to decide upon an exact number as there are variations between individual devices of one kind, a general trend could be spotted. I examined different input tools presented in Chapter 2.3 and identified the *Maximum number of expressions (MNE)* as a means to count the number of input options of the given controller examples. This number was extracted from each study according to the input options that were presented, though it seems possible to expand the range of input options in most cases. It should be read as an approximation, and not as a fixed score. An assessment of the *Dimensions of expression* of each controller type is listed in Table 2 below. Only the control units feasible for use within the scope of this project are being discussed.

<table>
<thead>
<tr>
<th>Type of controller</th>
<th>Dimensions of expression of the controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mechanical Switches</td>
<td>A single, simple mechanical switch only has the options 0 and 1 (on/off) and does not offer much freedom to express commands (Lancioni et al., 2004, 2001). It is a very minimalistic alternative that can, however, be expanded by adding more switches to the set, to include a greater set of functionalities. A single switch can also be coupled with a scanner (Blackstien-Adler et al., 2004; Demasco &amp; McCoy, 1992; Simpson &amp; Koester, 1999) and utilise a hold-and-release mechanism to include a bigger variety of input commands (Folmer et al., 2011). Yet, the dimensions of expression are somewhat limited as the iterative nature of the scanning mechanism consumes a considerable amount of time and attention from the user.</td>
</tr>
<tr>
<td>MNE = 8&lt;br&gt;hold-and-release switch in combination with scanner described in (Folmer et al., 2011)</td>
<td></td>
</tr>
<tr>
<td>Joysticks</td>
<td>Regarding the mechanical aspect, a joystick can be designed in different ways. It can have a set number of stages that transmit commands, like the wheelchair joystick utilised for the Octopus Launcher that offers a total of 8 positions (Ahn et al., 2014). The advantage of this is that it requires less fine steering and the clicking into place provides haptic feedback to the user. On the other hand, a joystick that allows for fluent motion could be preferable for the acts of looking around in the visual space. This resembles the way modern game controllers work; usually the left control stick of a joypad exists for the player to shift their field of view. Accordingly, familiarity with existing options is a factor worth considering.</td>
</tr>
<tr>
<td>MNE = 8</td>
<td></td>
</tr>
<tr>
<td>Touchpads</td>
<td>Depending on the size and number of the touchpads, they can offer a certain variety of commands. They allow for different input gestures such as tapping or swiping. The type used by Malu and Findlater can be attached to whatever part of the body or wheelchair the user wishes to, making it easy to place them in accessible locations (Malu &amp; Findlater, 2014, 2015). Due to this, it is possible to distribute the pads depending on the kind of</td>
</tr>
<tr>
<td>MNE = (&gt;4)&lt;br&gt;(expansion should be possible)</td>
<td></td>
</tr>
<tr>
<td>Type of controller</td>
<td>Dimensions of expression of the controller</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>input (e.g. looking or movement acts). Including the user’s spatial sense might help to make the input options intuitive and distinct from each other.</td>
<td></td>
</tr>
<tr>
<td>Gestural Interfaces</td>
<td>Some gestural interfaces detect movement of different body parts (Betke et al., 2002; Reilly &amp; O’Malley, 1999), while the others use rough gestures of the hands as an input (Jiang et al., 2014). The systems can be adapted to the motions the individual user can perform, which makes it difficult to assign an exact number of possible expressions. Another challenge lies in the setup that is required to detect and correctly interpret the gestures. The gesture lexicon described by Jiang et al. consists of relatively universal gestures that fulfil the minimum requirements for control units relevant to this project (Jiang et al., 2014). The systems described by Betke et al. as well as Reilly and O’Malley are intended to replace a computer mouse, hence the cursor can be moved to every place on the screen with clicks being performed after a certain dwell time (Betke et al., 2002; Reilly &amp; O’Malley, 1999). This allows for a very high number of expressions.</td>
</tr>
<tr>
<td>Head movement</td>
<td>Gestures performed with the head might not offer as many possibilities as arm or hand gestures. On the other hand, they require less freedom of movement from the user. In combination with clicking mechanisms or dwell time, there is potential to use head movements for directional commands (Chen, 2001; Kim &amp; Cho, 2002; Pereira et al., 2011). Yet, if the neck is flexible, it would generally allow the user to perform the looking acts characteristic for 360-degree videos as intended, although adjustments to the sensitivity of the gyro sensor might be necessary. It needs to be considered that those “natural” acts should not be overwritten, in order to not confuse the user.</td>
</tr>
<tr>
<td>Speech</td>
<td>Speech processing is a powerful and expressive tool for controlling an interface, if the user is able to produce distinct sounds. Sounds and syllables are easier to detect than full words, and they can be produced fast. Making use of different vocal characteristics, a sufficient range of expressive dimensions can be produced (Bilmes et al., 2005; Pereira et al., 2011). Possible drawbacks could be processing time and a lack of accuracy, as well as native speaker and pronunciation bias that make it difficult to learn using the tool correctly. Also, noisy environments make this input method almost impossible to use. However, next to eye movement- and humming-based solutions, interfaces based on speech are among the tools that are accessible to most users.</td>
</tr>
<tr>
<td>Humming</td>
<td>Humming is related to speech control, but it does not offer as many distinct command options. Some of the expressive power is lost because the mouth stays shut, providing one information channel less to the detecting device. By itself, humming appears to be rather limited and might not provide sufficient input options for controlling a 360-degree video. Still, it can be considered a valuable tool in combination with others.</td>
</tr>
<tr>
<td>Eye winks</td>
<td>Eye winks only require minimal movement from the user, while their expressive power can be rated as rather high, as described in (Mamatha &amp; Ramachandran, 2009). This great variety of command options relies on very subtle signals that use binary input (open/closed eye) as a base to form combinations. An obvious downside is that the user already needs their...</td>
</tr>
</tbody>
</table>
Type of controller  | Dimensions of expression of the controller
--- | ---
eyes to watch the video, so constant voluntary winking could be very irritating. Therefore, the method could also result in increased eye strain or headache.

Eye gaze  | Similar to eye winks, eye gaze can be a valuable alternative, especially for individuals with locked-in syndrome. Interfaces such as EyeDraw provide a wide range of inputs to the user who selects a key or a button by directing their gaze towards them (Hornof & Cavender, 2005). While eye gaze can be utilised to build an elaborate control system, it suffers from the same issues as eye wink control: the visual channel is already taken up by watching the video. This was also found to be a problem in gaze-based video game control (Isokoski & Martin, 2006).

MNE = >14*  | *Some alternatives exceed the functional requirements specified in Table 1 by far, so this number was chosen as reasonable limit.

As the assessment reveals, some alternatives are actually too constrained to be used as the only input tool to control a 360-degree video; for example, single switches or humming. Taking into account the physical limitations of the target group, a way to work around this can be to combine tools of different kinds. With such a multimodal approach, the range of input options can be broadened without requiring the “perfect” tool or even developing highly individualised devices, eventually increasing expressive power (Oviatt & Cohen, 2000). Enabling the combination of different units makes the control system more flexible and adaptable to the needs of single users. This approach builds on the existing body of research presented in Chapter 2.3 in that none of the devices need to be considered “insufficient” per se and therefore do not have to be excluded from the pool of possible control units.

Another critical issue is the suitability of eye-based input technologies. They do appear promising with respect to their high number of possible expressions but using them would mean to occupy the user’s visual channel for two different actions. That is, looking and interacting. While this has been implemented successfully in the EyeDraw software (Hornof & Cavender, 2005), the overall context of application is not the same. Instead of moving a cursor with the eyes, a viewer of 360-degree video would need to move the whole contents of their field of view. This leads to a conflict in senses and can be expected to be problematic. Accordingly, difficulties were observed in controlling looking and movement acts in first-person perspective games only with eye gaze (Isokoski & Martin, 2006; Smith & Graham, 2006). Even though there is potential to use eye-based input to control other types of acts than looking, the feasibility of this input technology should be judged carefully and needs further evaluation.

### 5.3 Mapping

Making a 360-degree video application accessible for different input tools means to remap the original functionalities to the alternative control system. This includes the acts listed in Table 1, so that all gestures and button presses can be replaced through alternative control units. In modern VR headsets, looking acts are detected through a gyro sensor in the device.
and translated into a shift in the field of view. On-screen choices like *meta acts* can, for example, be performed with an external controller that hosts a set of buttons.\(^8\) In other video players, it is possible to perform *looking acts* via clicking on arrows (Figure 11) or by using drag-and-drop (Figure 12). As those actions cannot be performed by people with certain motor impairments, they need to be accessible through another controller of choice.

---

\(^8\) A popular example is the Samsung Gear VR, consisting of a pair of goggles and a remote control. [https://www.samsung.com/global/galaxy/gear-vr/]
As stated in 5.2, not all of the available input devices offer enough options to satisfy the basic (or extended) requirements for controlling 360-degree videos. In many cases, a combination of devices allows for a more flexible choice. This implies that it will be necessary to split some of the functionalities and map them to the different devices. In doing so, consistency should be kept to support the user in forming a correct mental model of the control system. Acts of one type should stay together, that is, they will be mapped to the same input device. For example, all looking acts could be mapped to one device, while another device hosts the remaining movement and navigation acts. Alternatively, the latter two could be distributed between two controllers. As there are three, respectively four acts of one type, all devices generally offer enough input options for this distribution pattern. This might however not fully translate to every individual implementation.

An important concept related to those ideas is natural mapping. Norman discusses it extensively in “The Design of Everyday Things” (Norman, 2013). In his words, natural mapping means “taking advantage of spatial analogies”. The same layout, shape or direction of movement are used for an input option to create an unambiguous relationship with the resulting action. In the context of 360-degree video, for example, moving a joystick to the left should result in the field of view shifting to the left. The same principle applies to a cluster of switches or a number of touchpads attached in different locations on the user’s body or wheelchair. However, it needs to be considered that natural mapping underlies cultural aspects, e.g. in regard to directions or the perception of time.

Under the aspect of mapping, navigation actions take up a special position. As they comprise the commands relevant to choosing paths in different directions in one video experience, they are linked to the mechanism for selection and transition explored by my classmates (Abou Torab & Tapia, 2018). They investigated different kinds of transitions between video clips to find out what feels most natural to the user. Also, they tested the acceptance of elements such as arrows and doors to indicate possibilities for navigation. Our efforts were intended to complement each other by providing the visual side of the interaction on one hand, and the technical implementation of the navigation process with alternative devices on the other. The commands discussed in this chapter will be used at those graphical decision points. As my work also includes a basic navigational interface that uses similar elements, the next question will be how to tie the two mechanisms together.

5.4 Preliminary Evaluation

The concepts described in previous sections were verified with the help from a person of the target group. I visited her to discuss the insights gained from the literature study, as well as the conclusions I had drawn with regard to this project. After looking into literature and comparing the potential of different input devices and methods, certain control units appeared more suitable than others. Therefore, I found it necessary to include feedback from a potential user at this stage. The day before the interview I also had the possibility to try some of the input devices myself, such as simple buttons as well as a few applications that rely on eye tracking for interaction. As described in Chapter 4.3, the interview was conducted in a semi-structured manner. It was focused around the following questions:
Is the list of assistive tools complete, or is something missing?

What are the most common tools that are used to control computers, tablets, or smartphones?

Are there any situations during use you have the most trouble with?

Do you use workarounds?

Do you see any real-life limitations that could affect the use of the chosen tools for watching 360-degree videos?

Do you see any drawbacks of watching 360-degree videos in general (such as fatigue or simulator sickness)?

The interviewee herself uses a combination of different devices and input methods. That includes a lip mouse (a stick that is controlled with the lips, and in turn controls the cursor), a sip-and-puff switch to issue clicks, and speech control for writing, as she cannot use her hands on an ordinary keyboard. She has an eye tracker but is not used to it yet, having a hard time to generate the correct input. Throughout the interview she mentioned several times that eye tracking can be very difficult to use. Also, she is bound to a wheelchair that is controlled with a joystick: in addition to driving there are other functions available through a “mode” switch. Due to the variety of tools she uses and already is familiar with, she is a good example of a user who could benefit from a multimodal approach.

Subsequently, we went through the list of assistive tools I brought with me and discussed preferences and difficulties. The most common tools, according to the interviewee, are buttons, joysticks, eye tracking, and devices controlled by head movements. For her, a joystick would be the preferred tool, because she is already used to drive the wheelchair with it. Due to the middle button that is used to change between modes, the device is quite expressive. She stated that she could imagine using the wheelchair joystick to control a video, similar to the Octopus Launcher application (Ahn et al., 2014).

Using head movement for input is not feasible for her, neither is input through an ordinary keyboard. She mentioned that single switches would be “a bit limited”, and that a lot of them would be required for the purpose of the control system. Voice recognition can be used to a limited extent, depending on the noise level in the environment. When voice recognition is unavailable, the interviewee needs to rely on eye tracking. At the time of the interview, she had not started to make commands using that method. Again, she mentioned that “eye tracking is really hard”. Also, she expressed concern that it would be difficult to use eye tracking technology in combination with VR glasses. While she could see a certain potential for interaction with both eye tracking and speech recognition, the usability of those methods boils down to getting them to work. In theory, they do allow for a wide variety of commands, but their practical application comes with its own set of difficulties.

We briefly discussed real-life limitations that could affect both the use of input devices, as well as the general viewing setup. The interviewee wears glasses, so putting any kind of headset on top is problematic. She had already participated in a testing session with my classmates who used an actual VR headset for their test. Wearing said headset together with the glasses was difficult for her. Another drawback from using VR glasses could be nausea as a symptom of cybersickness (Davis et al., 2014). The interviewee also mentioned limited movement as a potential problem, although I would not consider this an issue since
the goal is to implement a variety of different input methods that work with minimal mobility.

Further, we went through the different movements and actions listed in Chapter 5.1. In the interviewee’s opinion, navigating right, left, and forward should be enough depending on the kind of video, or alternatively “go” and “turn around”. Backtracking was regarded a “maybe” useful feature, but not a must. She considered the option to adjust walking speed nice but not necessary, even though she stated the experience could feel more realistic if adjustment was possible. Being able to select a point in time would be good, for example if she took a longer break from the video and would like to continue in the same spot. An option to fast-forward or backward appeared unnatural to her; she said that winding the video backwards would be “strange”.

Overall, this short evaluation phase yielded insights into different aspects of the problem that this thesis tries to cover. The goal of evaluating my conceptual assumptions for interaction requirements was reached with the interview. It yielded overall similar results to the assessment done for the requirements analysis. Also, the interview confirmed some of my expectations, including the use of eye tracking software. The interviewee described similar difficulties to those I experienced in my own testing session, namely inaccuracy and frequent mistakes. While eye tracking is a powerful technology in terms of its dimensions of expression and reaches a high MNE (see Table 2), it takes a decent amount of training and getting-used-to in order to operate a device or application smoothly. Yet, I would argue that the difficulty also depends on the number and size of targets on the screen – e.g., having large control panels for video control versus a keyboard interface with many options. Therefore, eye tracking should not be disregarded for its difficulty, as it is the only viable input method for some people. Against that, the joystick appears to be a solid tool to begin development with, as it is widespread amongst the target population. Its number of input options are sufficient to meet the core functional requirements outlined in Chapter 5.1, with room for customisation.

5.5 Selecting Control Units

One of the main objectives of this thesis was to gather information about different alternative control units and to select those that are qualified for use in the Experience Library project. In Chapter 2.3, a preliminary selection was already made in regard to the suitability of certain input technologies, while all other alternatives were kept for closer examination. Considerations were made with particular respect to input limitations, resulting in the need to prioritise the most important commands and allowing the combination of multiple tools. Another aspect that needs to be considered for selection is familiarity with the device. If possible, common control methods should be prioritised to reduce learning effort, even if this might mean a lower MNE score and required the combination of multiple devices. For example, the target user I interviewed stated that joysticks are widely used in the target population (see Chapter 5.4), making them a preferred choice. User interviews conducted by Ahn et al. yielded the same preference (Ahn et al., 2014). Other factors, like cost or aesthetic quality were not considered as they lie outside of the scope of this thesis. It is to be noted that the alternatives assessed in Table 2 are examples – some of them in the early stages of development and are unlikely to be available by standard to every user.
Due to this, it is necessary for the concept to be very flexible. Since there is no single solution that fits all, the goal is to offer as many options as possible so that the control system can be adapted to individual properties and preferences. The decision was made not to exclude any of the examined control units, but instead to shift focus toward designing the basis for the control system using a modular approach, to make it work like a construction kit. While the minimum requirement of seven commands stays relevant, it should be possible to issue those commands using a combination of different input tools. Also, a single control unit that fulfills the minimum requirements for input could be expanded with another tool to reach extended or even full functionality.

Figure 13: Simplified spectrum of the user’s freedom of movement.

Both the user’s freedom of movement and the expressive power of different control units can be viewed as a spectrum. The representation is simplified and cannot speak for individual combinations of existing abilities. The spectrum of movements is illustrated in Figure 13. Projecting the spectrum of Dimensions of expression onto it reveals a possible choice of control units suitable to the user’s physical properties (Figure 14). Although generalised, this representation shows the accessibility of certain tools and possible options for combination.

Figure 14: Combined spectra of the Dimensions of expression of different devices and the user’s freedom of movement.

Finally, the decision was made to use joysticks for the control system prototype. This was due to several reasons. For one thing, joysticks offer different dimensions of expression that satisfy all core requirements for controlling a 360-degree video experience. They usually reach a medium to high MNE score and therefore leave enough room to implement both necessary and additional commands. Another reason was the availability of alternative control units. While I did not have direct access to assistive devices, it was possible to borrow different types of regular joysticks. Even though the models I used were made for gaming, they are operated in the same manner as, for example, wheelchair joysticks. The
interview results outlined in Chapter 5.4 suggest that this is beneficial – the interviewee stated that she would prefer a joystick because she is already familiar with it. As the most important aspect of the prototype was to build something that is operable as a general proof of concept, the devices were sufficient to emulate the idea.
6 Prototyping the Control System

In this chapter, I will describe the structure, design and implementation of a proof-of-concept control system for the Experience Library 360°. The objective was to build a basic API that facilitates the connection of different input devices for controlling aspects of the 360-degree video experience, such as acts of looking or moving. The main problem was to map the input received from a control unit to trigger the desired commands used to manipulate a 360-degree video application. In the following sections I will describe the development process from the initial sketches to the final prototype.

6.1 Sketches and Design Considerations

The prototyping process consisted of a short sketching and ideation phase and a longer building phase. This was due to the nature of the prototype setup in that the main objective was to tie the input from an external control unit to the intended behaviour of the video, hence the biggest challenge lay in the development process. The user interface was not of much importance in this stage. Instead, the focus was on finding a backend solution for the problem at hand to facilitate basic interactive qualities. As argued by Hudson and Mankoff, implementing at least critical aspects of a system in a proof-of-concept comprises the necessary groundwork to evaluate an idea that is created from scratch (Hudson & Mankoff, 2014). Accordingly, my goal was to prove that the envisioned solution is feasible and to prepare a first setup for future development to build on. In addition to that, the solution serves as a software prototype for the control API, as it demonstrates a way of processing input data in order to create the desired output. The hardware part of the prototype was necessary to make the connection possible, and to get real input data in an easy way.

![Diagram of the prototype setup](image)

**Figure 15: First sketch of the prototype setup.**

The initial idea was to have an intermediate device to connect the different control units to, up to eight channels at a time. In the sketch in Figure 15, it is represented by The Box, in this case an Arduino microcontroller. It was selected due to its simplicity, as it is designed...
to facilitate physical prototyping on a low technical level. *The Box* is intended to receive, and interactively process input from the different control units listed on the right side of the sketch, including both digital and analogue sources. Then, the input is passed through the controller API. It contains a set of control codes (examples listed in Figure 15) that would be re-mapped depending on the device’s input. The control codes are, in turn, associated with specific video events such as turning the field of view, or selecting a path at a decision point, which is defined in the main application. During the whole process, the Arduino is connected to a computer via USB. The computer hosts the application with the attached controller API and the screen serves as the output medium. VR glasses were not considered in this version of the prototype.

The visual part of the prototype was supposed to be kept basic. As 360-degree video is still a new medium, manipulating an existing application appeared difficult, especially with limited programming skills. Instead, the easiest solution was to write an own simple video player application that could be adapted to the project’s requirements. The idea was to limit the user interface to very few elements, possibly similar to a basic heads-up display. Actual UI design was put off to a later development stage.

Besides an Arduino microcontroller, the development environment Processing was chosen for the prototyping process. Arduino and Processing work well together, as it is made simple to establish a serial port connection. This makes it easy to read input and use it in the application. Additionally, a USB Host Shield was required to extend the Arduino, as it does not come with an inbuilt USB port. A gaming joystick and a gamepad were chosen as example control units. This was mainly due to practical reasons, since the prototype should be made operable before including more specialised devices that might be harder to obtain. Also, this meant that it would be possible to use a pre-existing library that offers code examples for the chosen control units, reducing the time and effort put into the prototype.

### 6.2 Building

While building was not the most important activity of my research project, the development of an operable system took a good amount of time and effort. Prior to the building phase, I had to get acquainted to the tools that were chosen for the prototype. Having only used an Arduino once in a previous course, I did not have enough knowledge to start right away. Therefore, some time was spent learning the basics of physical prototyping and writing simple Arduino code. It was in this learning process that I decided to use the Processing environment for the software side of the prototype. Another chunk of time was spent getting used to the environment, its possibilities and limitations. As I could not cover all of Processing’s features before getting started on the prototype, a part of them was discovered and learned in the process. Because of that, the different iterations of the prototype were experimental to an extent. Even though the target outcome was defined I did not know beforehand if my ideas were actually achievable in the way I planned.
The first step towards the actual prototype was to set up the Arduino for use with the **USB Host Shield**. As mentioned in the previous section, a library\(^9\) could be used to parse input from the connected USB device. The library supports a number of common control units, including the joystick and gamepad chosen for this project. Hence, the respective piece of code only needed to be uploaded to the Arduino to produce first output. The output contains a list of coordinates of all controller sticks and buttons and is continuously printed on the Arduino’s serial monitor, as shown in Figure 16.

![Arduino serial output](https://example.com/arduino-output.png)

**Figure 16:** Screenshot of Arduino serial output.

The format originally uses labels such as X or Y to denote the different dimensions, and the coordinates are printed in hex notation. Therefore, the next step was to figure out the corresponding set of coordinates for each joystick feature. Then, a Processing sketch was set up and connected with the Arduino’s serial. The formatting of the output was changed in the Arduino code so that the coordinates would be printed without labels and only separated by commas, to make them easier to process in the main application.

**FIRST ITERATION**

The first iteration of the prototype was focused on reading serial output correctly and on enabling simple interaction. On the visual side, a static, two-dimensional, panoramic image was used to simulate full 360-degree view (Figure 17). The image could be translated up, down, left and right depending on joystick movement. While this worked to demonstrate the re-mapping of commands, there was a considerable lag due to the large size of the image. It was being moved in pixel intervals and I did not succeed in making the field of view shift smoothly. Also, this approach did not offer much potential for interaction and the immersive quality of real 360-degree video could not be demonstrated.

---

\(^9\) **USB Host Shield Library 2.0**, developed by Oleg Mazurov et al.  
[https://github.com/felis/USB_Host_Shield_2.0](https://github.com/felis/USB_Host_Shield_2.0)
SECOND ITERATION

For the second iteration, a more sophisticated visualisation was built. I learned about the possibility to include videos in Processing sketches using a library\(^\text{10}\), and that they can be used as the texture of three-dimensional objects. Therefore, the new approach was to use a video in equirectangular format (filmed with a 360-degree camera) and to play it on the mantle of a sphere (see Figure 16). To create immersion, the viewpoint was moved into the centre of the sphere. The field of view could then be shifted by rotating the sphere along the X and Y-axis with the help of the joystick. This setup turned out to run smoothly despite the use of large video files.

Subsequently, more video controls were connected to the joystick’s features to meet the functional requirements specified in Chapter 5.1. I used the `Video` library for Processing that contains functions for common video player commands such as play, pause, and jumping to a point on the timeline. Once the video player was operable, I added in the second example control unit. It was necessary to modify the Arduino code slightly to make it possible for the Processing application to recognise which control unit is connected. Due to the modular structure of the application only one block of code was needed to re-map the output of the second controller.

\(^\text{10}\) `Video` library, developed by The Processing Foundation

6.3 Final Prototype

The outcome of this work is a high-fidelity prototype that can be run on a computer. It is shown in Figure 19. The prototype allows for user interaction through selected USB controllers and can be adapted to suit the interaction needs and preferences of the user. As there is no previous solution for controlling 360-degree video with alternative devices, this prototype presents a first approach of re-mapping common commands for a simple video player.

HARDWARE AND SOFTWARE SETUP

The current version of the prototype consists of the following parts:

- An Arduino Uno microcontroller, extended by an Arduino USB Host Shield to enable the connection of control units with an USB 2.0 plug.
- An example control unit to test the connection and communication between input and output device. For this prototype, a joystick and a generic gamepad were used.
- A corresponding Arduino sketch from the USB Host Shield 2.0 library, slightly altered to simplify processing of input data in the main program.
- The main program written in the Java-based, visual development environment Processing. It is used to demonstrate a 360-degree video experience with adaptable controls, as well as it contains a basic API that regulates the communication between input from the controller and output on the computer screen.

The hardware setup of the prototype is simple: the controller is plugged into the Host Shield’s USB hub that is, in turn, plugged into the Arduino board. Then, the corresponding library sketch is uploaded to the board, as it can only hold one sketch at a time. The sketch is necessary to process the original input from the control unit in that it translates it.
into an easily readable format that is written out on the Arduino’s serial monitor. The serial output contains the coordinates and states of the different sticks and buttons of the control unit and is constantly read while the Processing code is running.

In the following, I will outline the main program and give an idea of what it does. The overall workflow is documented in Appendix B. Appendices C and D contain all parts of both Arduino and Processing sketches including comments. The code is structured into four files that cover different aspects of the prototype.

![Diagram of file structure of the main program.](image)

The main file contains library imports, variable and object declarations. Its only own functions are setup() and the looped function draw() that are integral parts of every sketch written in Processing. Serial communication with the Arduino board is initialsed and the 360-degree video player is set up. The main file contains calls to the functions associated with input mapping, video acts, and contents of the overlay. The subfile “mapping.pde” can be considered the core of the API, as it contains information about how each input parameter (Figure 16) is mapped to a corresponding command. Mapping was done separately for each control unit. It is fairly simple, as the only task necessary was to determine the value or range within which an action should be triggered. The subfile “acts.pde” describes what video events are connected to the execution of the mapped commands. Events associated with controlling the video on a meta level (e.g. play, pause) are included in the Video library for Processing. Finally, “overlay.pde” holds information about visual elements of the video overlay, such as arrows or pointers. This is the place where any additional information or instructions related to the user interface are described.

RESULTS

The prototype was made adaptable to different devices, in that the one connected will be automatically recognised by the application. While the current version only works with two different control units, it is structured in a way that facilitates further adaptation. This is what the proof-of-concept implementation was intended to validate. The code follows a modular concept, so that there are only two places where changes need to be made. The API part of the prototype can be considered as fulfilled with the “mapping” and “actions”
file – input is run through the controller-specific mapping block, and the correct video commands are returned.

As mentioned beforehand, all input parameters are compared to a fixed value or range specific to the connected control unit. In the building process, differences in regard to those parameters became obvious, for example, different value ranges had to be normalised depending on the original values (e.g. 0–1023 for the joystick, 0–255 for the gamepad). If a new device were to be connected, a short calibration process would be necessary. Also, it would require both a working Arduino sketch that produces serial output in a certain format (see Appendix C), as well as an own block of code to ensure correct mapping. Therefore, the flexibility of the code is limited, even though the adaptation process is rather simple.

The video experience can be controlled with eleven commands on each control unit. This includes the manipulation of the field of view (FOV) both horizontally and vertically, choosing the next video at a decision point from up to four paths (left, right, straight, back), as well as video commands on a meta level such as play/pause and play from beginning. Depending on the expressive power of the control unit and the individual properties of the user, more or only a selection of commands can be implemented. This means that the final prototype meets 11 out of the 14 requirements for movement and actions specified in Chapter 5.1, including all primary and most secondary and tertiary acts.

<table>
<thead>
<tr>
<th>Movement or activity</th>
<th>Logitech joystick controls</th>
<th>Generic gamepad controls</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rotate FOV right</td>
<td>turn stick right/twist right</td>
<td>right stick right</td>
</tr>
<tr>
<td>Rotate FOV left</td>
<td>turn stick left/twist left</td>
<td>right stick left</td>
</tr>
<tr>
<td>Rotate FOV up</td>
<td>turn stick up</td>
<td>right stick up</td>
</tr>
<tr>
<td>Rotate FOV down</td>
<td>turn stick down</td>
<td>right stick down</td>
</tr>
<tr>
<td>Play from beginning</td>
<td>button 2 (secondary fire)</td>
<td>left stick up</td>
</tr>
<tr>
<td>Move forward (Play)</td>
<td>button 1 (fire handle)</td>
<td>left stick down</td>
</tr>
<tr>
<td>Stop moving (Pause)</td>
<td>button 1 (fire handle)</td>
<td>left stick down</td>
</tr>
<tr>
<td>Choose path to the left</td>
<td>hat left</td>
<td>X</td>
</tr>
<tr>
<td>Choose path to the right</td>
<td>hat right</td>
<td>B</td>
</tr>
<tr>
<td>Choose path straight ahead</td>
<td>hat up</td>
<td>Y</td>
</tr>
<tr>
<td>Choose path back</td>
<td>hat down</td>
<td>A</td>
</tr>
</tbody>
</table>

Table 3: Distribution of commands on the example control units.

Both example controllers count as joysticks, but there are variations among their input options. The Logitech joystick has an MNE score of 24, the gamepad reaches 12 (several buttons do not work with the Arduino library). Different dimensions of expression were used to host the available commands for each control unit (see Table 3). Every dimension
allows for one or more commands depending on the nature of the feature: generally, simpler commands such as (de)activating or choosing an option were mapped onto buttons, and looking acts were mapped onto sticks. While this convention is helpful in that it goes into the direction of natural mapping (Norman, 2013) and mimics spatial relationships of looking and navigation acts, it should not be enforced at any cost. For example, an exception to the button “rule” was made when deciding the re-mapping of the gamepad. With two sticks but only four functioning buttons, the vertical dimension of the left stick was used to host two commands. “Pause” is toggled by moving the stick down, “Play from beginning” is chosen by moving the stick up. This was the only option, because the A, B, Y and X buttons were already used for choosing the video path. The hat switch did not produce an own set of input, making it useless.

Figure 21: User interface with pointers, arrows and time counter.

Since the focus of this thesis is on the control system itself, not much time was allocated to designing a sophisticated user interface. Currently, the overlay includes pointers near the edge of the screen to indicate the possibility of 360-degree movement, and arrows that indicate possible paths to choose from, shown at the ending of each video clip (Figure 21). Other visual elements include a time counter as well as an indicator of the video’s status, displayed when the clip is paused. Further development of the video player interface would exceed the scope of this work and needs to be left for future development. Another aspect of the user interface was explored by my classmates Ruba Abou Torab and Julio Tapia, who worked on the transition between video clips in order to create an immersive, forking path (Abou Torab & Tapia, 2018).

REMAINING ISSUES

For this prototype, two common game controllers were used to build and test the application. The benefit of this approach is that a pre-existing library with example sketches could be used to process input data, so most of the effort could be focused on writing the API code. The prototype was not tested with or tailored to control units such as single buttons, a wheelchair joystick or eye tracking, since this would have added the task of writing device-specific Arduino sketches or even changing up the method of processing input. This means that the prototype in its current form is not ready to be used by people with major motor impairments and more specific device requirements.

While the API works as intended, a point of criticism could be that it is too hard coded. I attribute this mainly to my limited programming knowledge and experience. The parts
that need adaptation to a new control unit could be separated from the rest by introducing a configurations or settings file. That way, only the parameters in the file would need to be changed, but not the code of the application. Yet, due to time constraints, this could not be implemented.

Further, an effort was made to transfer the command mapping into an own library specific to the project. While the mapping generally worked, conflicts arose from the communication between the Arduino’s serial output and the Java library. Input was not processed properly and caused major lag and errors in the application, which is why the library approach was discarded. Even though the combination of Arduino and Processing worked well for a quick, yet high-fidelity prototyping process, it is likely not the best environment to write sustainable code, or to build a stable and smoothly-running application.
7 Discussion

In this chapter, I will summarise the work that has been done over the course of this thesis. I will discuss how the research questions were answered, give a brief overview of the methods used, and state my contribution to the field. Reflecting back on the scope discussed in Chapter 1.2 I will address the limitations of my work. Finally, I will discuss possible implications and reasonable steps for future work.

7.1 Conclusion

I have presented a working prototype of a control API for the connection of alternative control units that will be used as a part of the Experience Library 360° project. The development of the prototype was based on extensive literature research of existing assistive tools and guided by my own concepts of control requirements. My methodology consisted of a literature study to gain understanding of the subject area and to explore possible control units, a conceptualisation phase for idea generation, a target user interview to evaluate my concepts, and a prototyping phase. In the latter phase, a rapid software prototype was built that demonstrates the intended functionality of the API and visualises how the 360-degree video reacts to manipulation by the user. It is a proof-of-concept implementation that was intended to validate our assumptions about how the control system could be built.

The motivation for this thesis was to contribute to an application that aims at overcoming the exclusion of people with certain impairments from exploring interesting locations in the world from a first-person perspective. Despite physical restrictions, the viewer should be enabled to move freely through a network of interactive and immersive video experiences that lend the feeling of “being there”. By providing initial concepts as well as a prototype of an adaptable control system, the work of this thesis contributes to making the experience accessible to people with varying physical abilities. Although the system is not fully developed yet, it serves to demonstrate the feasibility of adapting to different input devices of choice.

ANSWERING RESEARCH QUESTIONS

This thesis had the goal to answer three different research questions that build on each other. In the following, I will refer back to those questions and discuss how they were answered using the methods discussed previously.

1. What alternative, suitable input devices and assistive tools are available for controlling 360-degree videos?

To be able to answer this question, I created an overview of existing technologies and control units. The overview was based on a systematic review of different access technologies and supplemented with a free search in the ACM Digital Library as well as Google Scholar. A total of 57 journal articles and conference papers were categorised and analysed to assess the suitability of the different options. The range of available control alternatives and input tools includes mechanical switches, joysticks, touchpads, gestural interfaces, interfaces based on head movement, voice-based interfaces (speech and humming), and eye-based interfaces (gaze and winks). Several examples of each input option were discussed to
present their characteristics, availability depending on the user’s abilities, and advantages as well as drawbacks.

2. What are appropriate criteria to determine the suitability and applicability of such input devices?

To be able to answer the second research question, I started with an analysis of functional requirements for controlling 360-degree videos. A full list of commands was made and, under consideration of possible limitations, classified by importance. They were categorised into primary, secondary and tertiary commands. Out of fourteen total commands, seven comprise the minimum requirements for successful interaction with 360-degree videos. As the control alternatives in question allow for different numbers of input options, I decided to base the criteria for suitability and applicability on their expressive power. In order to describe expressive power, I introduced the terms Dimensions of expression to judge the flexibility for implementing a variety of commands, and Maximum number of expressions as a way of assigning a concrete number of possible commands to each controller. This helped me estimate which control alternatives could stand alone, and which were insufficient to accommodate all necessary commands. As expressive power varies between devices and the availability of sufficiently expressive controllers depends strongly on the kinds of impairments a person has, the findings led me to concluding that a multimodal approach would be the smartest way to handle the different options. Enabling a connection of multiple devices at once would be an easy way of making the system adaptable to the individual user’s needs, without requiring a highly individualised tool.

3. How can different input devices be connected in practice to make the control system flexible and adaptable?

In order to answer the third research question, I needed to put my theoretical considerations into practice. To achieve flexibility and adaptability of the control system, an API was written that can be calibrated for the control unit of choice. It maps input data to a set of commands with which the video is manipulated. The current version of the prototype allows for the connection of two example controllers of which both fulfil the minimal requirements for interaction. Implementation was made possible through an Arduino microcontroller with a USB extension, acting as an intermediate connection box between control unit and a computer on which the video application runs. The API that handles the re-mapping of commands is included in the main application. As the successful proof-of-concept implementation validates the idea for the adaptable control system, the third research question is answered.

APPLICATION OF FRAMEWORKS

The frameworks relevant to my research area are Non-Excluding Design, Universal Design and Ability-based Design. As discussed in Chapter 3, the Universal Design framework was the first attempt at designing for inclusion, and it is applicable to different disciplines (Center of Universal Design, 2008; Connell et al., 1997). The authors promote the creation of universal solutions which, as I would argue, is impossible to achieve when taken literally. It needs to be interpreted in terms of state-of-the-art software and adaptability to be a reasonable goal. From the perspective of an individual, a system is only as good as the support it provides for their personal needs. Besides that, I would like to point out the special relevance of the Ability-based Design framework (Wobbrock et al., 2011). On one hand, it emphasises making use of the abilities the user actually possesses, framing design
for people with impairments in a positive way. While it is almost congruent with the Non-Excluding Design framework in that aspect (Oestreicher, n.d.), it also includes concrete instructions on how the fundamental principles can be put into practice in systems design.

The proof-of-concept built for this thesis follows the first three principles of Ability-based Design, namely Ability, Accountability, and Adaptation. It was designed with the intention to make the most of the individual user’s abilities by considering a variety of input devices and the combination of multiple control units. While it would be desirable to satisfy all of the seven principles, the system at hand is rather limited in its functionality and only represents a part of the overall project. As for example the user interface is still very rudimentary, the principles of Transparency and Commodity could not be reflected in the current implementation.

CONTRIBUTION TO THE FIELD

360-degree video is still a young technology that has only become accessible to consumers over the last few years. While research has been done both on the use of 360-degree video for virtual exploration as well as on a variety of control units for people with physical impairments, no connection has been made between those areas. This fact led to a first question about general feasibility. My work contributes to the existing body of research in Non-Excluding Design in that it focuses on connecting available technologies to lay a foundation for creating enjoyable virtual experiences regardless of the user’s physical abilities and limitations. As the Experience Library 360° is still in the early stages, the efforts made over the course of this thesis focused on an implementation of the critical aspects of that idea. The building process was not directly aimed at producing (parts of) a finished product – instead, the outcome should be considered a by-product of the research process that was needed to validate the concept, as described by Hudson and Mankoff (2014). Together with the work of my colleagues Ruba Abou Torab and Julio Tapia, a first attempt was made to investigate how 360-degree video technology can be used to facilitate an immersive exploration of places in the world for users who physically cannot access them.

LIMITATIONS OF THIS THESIS

While I concluded that the research questions posed in chapter 1.2 could be answered over the course of this thesis, I would like to reflect on some aspects in more detail. A comprehensive taxonomy of control units and input tools has been made in an attempt to create an overview over the technologies relevant to the project. However, the question remains if the list is complete, or if I have missed some controller types. I tried to be very thorough in my literature search and used the subsequent interview to evaluate the completeness of my findings, but I cannot say with absolute certainty that every existing possibility was included. As technology advances fast, the list is more of a work in progress.

Furthermore, it is currently impossible to predict how each alternative will perform in practice. On the one hand it is to be noted that not every example mentioned in Chapters 2.3 and 5 has been evaluated with users from the target population (see Appendix A for more information). As I have not used any of them directly in the building process, I can neither make a statement about their technical implementation for this specific project. On the other hand, the operability and usefulness of a control unit is highly dependent on the individual user. The classification that I provide can be used as a guideline to estimate which alternatives can be worth trying, but no predictions can be made. In order to know how the system can be adapted to an individual user, it is inevitable to sit down with them.
and experiment with different controllers. Due to this, user testing will not produce generalizable findings. As discussed in Chapter 1.3 (Delimitations and Scope), it would not be reasonable to aim at measures like efficiency, as it is more important to find a solution that works at all.

For example, a user could have muscular weaknesses that do not allow them to move at all, leaving gaze-based control systems as the only option. They are intricate and take an amount of training to get used to. Also, the use of dwell time as a selection mechanism makes them rather slow and recovery from mistakes takes effort. Compared with other input tools, gaze-based control would not reach a high score in terms of efficiency or usability. To give another example, someone could be able to use a part of the functions of a joystick comfortably, but not enough to satisfy the necessary functional requirements. Even though the joystick by itself is considered a sufficient and fairly efficient tool, the assigned Maximum number of expressions does not apply in that specific case. The joystick would need to be extended by another control unit to reach the necessary functionality. Such cases are not exceptions, but instead they are the rule.

Lastly, I would like to point out that a hard measurement through the Dimensions of expression or Maximum numbers of expression is not possible, nor does it make sense. The two concepts were intended to be used for assessment and classification, in order to estimate if and how different control units are worth considering for usage in the Experience Library 360°. As both the target user group as well as the available tools are very heterogeneous, I would argue that there is no point in assigning exact measurements to their characteristics.

### 7.2 Future Work

To conclude, I would like to discuss implications and starting points for future efforts. As stated in the previous section, the results of my work answer the research questions posed in the beginning. Yet, new questions came up in the process that cannot be answered in this thesis. As the prototype described here was developed as a proof-of-concept, it only addresses a small part of the overall system. One task that needs to be left for future development is the integration of the control system with the visual interaction mechanisms that my classmates explored (Abou Torab & Tapia, 2018). It needs to be investigated how the user can physically interact with the visual metaphors, and what adaptations have to be made in order to make that possible. We have worked separately and have not synchronised our efforts in that respect. Therefore, it will be necessary to match the commands that are required for their interaction model with the functional requirements for the control system discussed in this work. It needs to be evaluated if the complexity of interaction can be realised through the control API. If that is not the case, a reasonable compromise will need to be found that satisfies the findings of both theses.

Another necessary task would be to transfer the ideas for technical implementation to a more sustainable environment. My choice of materials was based on their convenience for rapid prototyping, and not their ability to handle a complex solution. While an advantage of the current setup lies in its comparably low cost, it will be worth replacing the Arduino Uno with a stronger microcontroller such as a Raspberry Pi. That will make it possible to connect more than one control unit at a time.
Another interesting step would be to try out the control system with different users. As mentioned before, individual implementations can be tested, but not the prototype as it is. User testing would require the production of parser code to process input from new control units. Over time, the possibility to reuse parts of the code will increase, but there is no point at which the range of supported controllers can be considered “complete”. Similarly, command mapping always needs to be calibrated individually. To simplify that process, the system could be modified to reduce the effort for calibration and eliminate the need to make changes directly in the application code.

Yet another task that needs to be left for future development is the design of an unobtrusive user interface that guides navigation and aids the forming of correct mental models for interaction. Ultimately, those aspects need to be integrated into the overall system which is supposed to be a library of interconnected video experiences. Aside from that, I would like to mention a few smaller ideas that came up during the work process. It could for example be interesting if the Experience Library 360° could be used in motivational training, if the control framework could be reused for video games, or if meta commands could be recycled for other multimedia applications. While it is too early to make any statements at this point, 360-degree video technology is a promising field that will attract more attention in the future. My personal conclusion is that it has potential to level the playing field from a perspective of Non-Excluding Design.
References


for augmentative communication. *IEEE Transactions on Rehabilitation Engineering*, 7(2), 174–182. https://doi.org/10.1109/86.769408


## Appendix A: Overview of Control Alternatives

<table>
<thead>
<tr>
<th>Publication</th>
<th>Type of Controller</th>
<th>Properties it is suitable for</th>
<th>Summary</th>
<th>Annotations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>“Navigating a 3D avatar using a single switch”</strong>&lt;br&gt;(Folmer et al., 2011)</td>
<td>Single switch with hold-and-release mechanism</td>
<td>▪ Paralysis of the lower extremities&lt;br&gt;▪ Speech problems&lt;br&gt;▪ Stiffness of the neck</td>
<td>The device was developed as an alternative controller for video games. As most modern games use non-linear control patterns, a simple switch for on/off is not enough. This solution is based on hold-and-release of a single switch. Switches are not as expensive, and there exist different types (“pull, push, squeeze, blink or sip or puff”).</td>
<td>So far, the controller has not been evaluated with participants with impairments. Tests have only been done with a simulator.</td>
</tr>
<tr>
<td><strong>“Mouse Manipulation Through Single-Switch Scanning”</strong>&lt;br&gt;(Blackstien-Adler et al., 2004)</td>
<td>Single switch with scanning mechanism</td>
<td>▪ Paralysis from the neck downward&lt;br&gt;▪ Tremor in the extremities&lt;br&gt;▪ Absence of fine motor skills in the upper extremities</td>
<td>The study evaluated four different single-switch scanning strategies for mouse emulation with ten participants. Participants used a variety of switch placements. The authors found differences in accuracy and speed, with Cartesian scanning being preferred by the participants.</td>
<td>Not a specific device, but interesting in case a switch scanning mechanism is considered.</td>
</tr>
<tr>
<td><strong>“Use Octopus Launcher Like Your Hands: Joystick-based Smartphone Control Solution for Motor Impaired People in Electric Wheelchairs”</strong>&lt;br&gt;(Ahn et al., 2014)</td>
<td>Wheelchair joystick used to control smartphone</td>
<td>▪ Paralysis from the neck downward&lt;br&gt;▪ Tremor in the extremities&lt;br&gt;▪ Absence of fine motor skills in the upper extremities&lt;br&gt;▪ Muscle weakness</td>
<td>This approach uses the eight directional controls of an electric wheelchair joystick to operate touch gestures and hardware buttons on a smartphone. This answers to the study participants’ wish to use the same apps as people without motor impairments. Benefits of the joystick framework are that users do not need to get used to a new device, and that there are no additional costs.</td>
<td>The system has not yet been evaluated with target users in a real-life context.</td>
</tr>
<tr>
<td>Publication</td>
<td>Type of Controller</td>
<td>Properties it is suitable for</td>
<td>Summary</td>
<td>Annotations</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------</td>
<td>-------------------------------</td>
<td>---------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| “Text Entry from Power Wheelchairs: EdgeWrite for Joysticks and Touchpads” (Wobbrock et al., 2003) | Wheelchair joystick, wheelchair touchpad | ▪ Spastic movements  
▪ Stiffness | In this study, the authors connected two power wheelchair control devices to an on-screen keyboard that uses EdgeWrite for text entry. The setup was evaluated with seven participants with motor impairments. While the two wheelchair controls had higher average error rates than a comparison tool, participants preferred the touchpad option over the rest. | While text entry is not the goal of the Experience Library project, this paper demonstrates an interesting way of recycling a tool that some users already have. |
| “Controlling Mouse Pointer Position Using an Infrared Head-Operated Joystick” (Evans et al., 2000) | Joystick controlled by head movement | ▪ Tremor in the extremities  
▪ Absence of fine motor skills in the upper extremities  
▪ Spastic movements | The paper presents a head-operated joystick that uses LEDs and photodetectors to identify the position of the user’s head. With the input a mouse cursor can be emulated. The device was evaluated with both people with and without impairments. The participants with impairments preferred joystick interaction over “mouse-style” devices, possibly due to the fact that the joystick's deadband offers a rest position. | The device is only accessible for users who can control their head movement. |
<p>| “Personalized, Wearable Control of a Head-mounted display for Users with Upper Body Motor Impairments” (Malu &amp; Findlater, 2015) | Switch-based wearable touchpads | ▪ Paralysis of the lower extremities | The touchpads were tested for controlling a head-mounted display (in this example Google Glass). They can be affixed to either the body or wheelchair. Preferences for placement and size depended on each participant’s motor abilities. | The control alternative was evaluated with a group of people with different upper body motor impairments. |</p>
<table>
<thead>
<tr>
<th>Publication</th>
<th>Type of Controller</th>
<th>Properties it is suitable for</th>
<th>Summary</th>
<th>Annotations</th>
</tr>
</thead>
<tbody>
<tr>
<td>“A machine vision-based gestural interface for people with upper extremity physical impairments” (Jiang et al., 2014)</td>
<td>Gestural interface</td>
<td>• Paralysis of the lower extremities                                                             • Absence of fine motor skills in the upper extremities • Speech problems</td>
<td>Participants stated that the alternative would provide for increased independence in using a head-mounted display. They did however express concerns about social stigma due to the visibility of the device.</td>
<td>The authors mention people with tetraplegia as their target group. Does the condition not imply that a person cannot move any body part below the neck? Also, this device is only available for people who can move their hands.</td>
</tr>
<tr>
<td>“OK Glass?” A Preliminary Exploration of Google Glass for Persons with Upper Body Motor Impairments” (Malu &amp; Findlater, 2014)</td>
<td>Head-mounted display with gesture and voice commands (Google Glass)</td>
<td>• Paralysis of the lower extremities                                                             • Limited mobility of the left arm • Stiffness of the neck</td>
<td>This study evaluated the accessibility of Google Glass with people with upper body motor impairments. All five participants had difficulties to command the device with tapping and swiping gestures, partly due to limited mobility. While the participants appreciated the hands-free approach, more accessible input alternatives are required for independent and satisfying use.</td>
<td>The evaluation included five participants.</td>
</tr>
<tr>
<td>“Adaptive Noncontact Gesture-Based System for Augmentative Communication” (Reilly &amp; O’Malley, 1999)</td>
<td>Gestural interface</td>
<td>• Paralysis from the neck downward • Tremor in the extremities</td>
<td>This alternative uses movement of any body part to control cursor movement as well as click actions. It is non-contact, that means no device needs to be placed on the user. Evaluation was conducted with a group of target users who successfully completed the assigned tasks. The accuracy depends on how much control the user has over their head movement as well as training.</td>
<td></td>
</tr>
<tr>
<td>Publication</td>
<td>Type of Controller</td>
<td>Properties it is suitable for</td>
<td>Summary</td>
<td>Annotations</td>
</tr>
<tr>
<td>----------------------------------------------------------------------------</td>
<td>-------------------------------------------</td>
<td>------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>“The Camera Mouse: Visual Tracking of Body Features to Provide Computer Access for People With Severe Disabilities”</td>
<td>Gestural interface</td>
<td>§ Absence of fine motor skills in the upper extremities  § Stiffness</td>
<td>This alternative uses a video camera to track user movements which are then translated into mouse pointer movements on the screen. Different body parts can be used for tracking, depending on the user’s ability to move, for example the nose, lips, or thumb. The system can be used for different applications, e.g. surfing the internet or playing games, and was tested with both people with and without impairments. One additional benefit is that no device is attached to the user.</td>
<td></td>
</tr>
<tr>
<td>(Betke et al., 2002)</td>
<td></td>
<td></td>
<td></td>
<td>To use the Camera Mouse, the user needs to be able to move at least one body part reliably, as the system is limited in regard to eye tracking.</td>
</tr>
<tr>
<td>“Click Control: Improving Mouse Interaction for People with Motor Impairments”</td>
<td>Camera-based mouse-replacement system, controlled by head movements</td>
<td>§ Paralysis from the neck downward  § Tremor in the extremities  § Absence of fine motor skills in the upper extremities  § Muscle weakness  § Spastic movements</td>
<td>This alternative is a mouse cursor that can be controlled by head movements, for people who cannot click with their hands. It is supposed to counteract accidental clicking, to give the user more control. The authors state that clicking aids do normally not allow for a satisfying experience, as they are inconvenient and time-consuming.</td>
<td>This alternative is only available for people who can move their head to follow a virtual path.</td>
</tr>
<tr>
<td>(Kwan et al., 2011)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“Designing a game controller for motor impaired players”</td>
<td>Combination of head-mounted and speech controls</td>
<td>§ Paralysis from the neck downward  § Tremor in the extremities</td>
<td>The controller was developed in cooperation with a user with certain motor impairments that result in difficulties to press buttons. The device should replace a joystick or keyboard. It consists of two parts: an accelerometer on a cap captures head movements. Depending on the user’s ability to move, for example the hand, arm, or face, the system can be used for different applications, e.g. surfing the internet or playing games, and was tested with both people with and without impairments.</td>
<td>The controller was developed under consultation of only one target user and has not been evaluated with a larger group of participants.</td>
</tr>
<tr>
<td>(Pereira et al., 2011)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Publication</td>
<td>Type of Controller</td>
<td>Properties it is suitable for</td>
<td>Summary</td>
<td>Annotations</td>
</tr>
<tr>
<td>---------------------------------------------------------------------------</td>
<td>----------------------------------------</td>
<td>-------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| “Application of Tilt Sensors in Human-Computer Mouse Interface for People With Disabilities” (Chen, 2001) | Head-operated computer mouse          | - Paralysis from the neck downward  
- Tremor in the extremities  
- Absence of fine motor skills in the upper extremities | This study investigated a head-operated mouse that uses tilt sensors to detect horizontal and vertical movement. Additionally, a touch switch next to the user’s cheek is used to click, double click, and drag. The device is worn like a pair of headphones. The system was evaluated with people with spinal cord injuries as well as people without disabilities. | The device is only accessible for users who can control their head movement.                                                                                                                                 |
| “FaceMouse: A Human-Computer Interface for Tetraplegic People” (Perini et al., 2006) | Mouse pointer controlled by head movement | - Paralysis from the neck downward  
- Tremor in the extremities  
- Absence of fine motor skills in the upper extremities | For this study, the authors used a low-cost webcam and computer vision techniques to track parts of the user’s face. The nose is generally the easiest feature to track due to its shape. The user chooses points that represent the four directions and the central position they can assume to interact with the system. Tests were conducted with a group of people with tetraplegia. | The device is only accessible for users who can control their head movement.                                                                                                                                 |
| “Hands Free Mouse: Comparative Study on Mouse Clicks Controlled by Humming” | Cursor controlled by humming sounds    | - Paralysis from the neck downward  
- Tremor in the extremities | This alternative comprises a mouse cursor that is moved through head movements, while commands are executed though humming. The commands are distinguished by their pitch and length. The controls allow for left, right Subjective accuracy and comfort of the alternative were ranked worst by test participants. With regard to |
<table>
<thead>
<tr>
<th>Publication</th>
<th>Type of Controller</th>
<th>Properties it is suitable for</th>
<th>Summary</th>
<th>Annotations</th>
</tr>
</thead>
</table>
| (Poláček & Mikovec, 2010) | Interface controlled by silent humming | ▪ Absence of fine motor skills in the upper extremities  
▪ Muscle weakness  
▪ Spastic movements  
▪ Stiffness | and double click, as well as dragging and scrolling. A benefit of this option is that it is independent of language and culture. On the other hand, humming controls are uncommon and require getting used to. Input speed was faster than for speech and gesture input. | possible stigmatization, it is questionable if audible humming is an idea worthy to follow. |
| “Humming Control Interface for Hand-held Devices” (Won et al., 2007) | Voice-based interface | ▪ Paralysis from the neck downward  
▪ Tremor in the extremities  
▪ Absence of fine motor skills in the upper extremities  
▪ Muscle weakness  
▪ Spastic movements  
▪ Stiffness | This alternative uses subvocal (silent) humming as an input. For this method, the user only needs to tense the vocal cords without making a sound, like when reading silently. A contact microphone receives sound and vibration from the bone. So far, the authors have tested algorithms for pitch detection and string matching. User tests still need to be done. | As the authors have not yet conducted an evaluation with real users, there is no information available about experiences or comfort with this type of control interface. |
| “The Vocal Joystick: A voice-based human-computer interface for individuals with motor impairments” (Bilmes et al., 2005) | | ▪ Paralysis from the neck downward  
▪ Tremor in the extremities  
▪ Absence of fine motor skills in the upper extremities  
▪ Muscle weakness | This alternative interprets different vocal parameters (“pitch, loudness, vowel quality”) into commands. Different applications can be controlled, such as browsing the web. The controls are based on a predefined set of discriminable sounds instead of words or syllables, to make them easy to recognise for the device and easy to memorise for the user. Both the technical details of the engine and prototype as well as preliminary evaluation results are described. | The interface has only been tested in a preliminary study without participants of the target population. Therefore, no information about comfort or possible stigma is available. |
## APPENDIX A · OVERVIEW OF CONTROL ALTERNATIVES

<table>
<thead>
<tr>
<th>Publication</th>
<th>Type of Controller</th>
<th>Properties it is suitable for</th>
<th>Summary</th>
<th>Annotations</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Non-speech input and speech recognition for real-time control of computer games” (Sporka et al., 2006)</td>
<td>Humming input</td>
<td>- Paralysis from the neck downward</td>
<td>This study compares user performance in the game Tetris for speech commands and humming input, as alternatives to keyboard control. Humming detection includes pitch and length of the sound. There were four commands (right, left, drop, turn). Humming input was found to allow for faster gameplay than speech recognition. Overall, participants’ responses to humming controls were positive.</td>
<td>The evaluation was not conducted with people with impairments. Therefore, information about perceived comfort and possible stigma is missing.</td>
</tr>
<tr>
<td>“Voicedraw: A Hands-Free Voice-Driven Drawing Application for People with Motor Impairments” (Harada et al., 2007)</td>
<td>Drawing software that uses voice commands</td>
<td>- Paralysis from the neck downward</td>
<td>The paper describes the development of a voice-based interaction system for drawing without hands. The paintbrush is controlled with nine different vowel sounds and loudness; for sound recognition the Vocal Joystick engine was used. Additionally, full word commands (e.g. “erase”) are available. The software was evaluated with an experienced voice painter. Findings include the reliable production of sound, and an increase in drawing speed.</td>
<td>The application is a software specifically developed for drawing.</td>
</tr>
<tr>
<td>Publication</td>
<td>Type of Controller</td>
<td>Properties it is suitable for</td>
<td>Summary</td>
<td>Annotations</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------</td>
<td>-------------------------------</td>
<td>---------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| “Voice as Sound: Using Non-verbal Voice Input for Interactive Control” (Igarashi & Hughes, 2001) | Voice-based control interface | - Paralysis from the neck downward  
- Tremor in the extremities  
- Absence of fine motor skills in the upper extremities  
- Muscle weakness  
- Spastic movements  
- Stiffness | This paper discusses several prototypical approaches for voice-based interface control. Examples are “control by continuous voice” in which the user’s voice works as an on/off switch for a system response, as well as input from pitch and discrete sounds. Benefits of using sounds as input are speed, simplicity and language independence. | The authors do not report user studies to verify or compare the functionality of the system. |
| “Interaction Techniques Using Prosodic Features of Speech and Audio Localization” (Olwal & Feiner, 2005) | Voice-based control interface | - Paralysis from the neck downward  
- Tremor in the extremities  
- Absence of fine motor skills in the upper extremities  
- Muscle weakness  
- Spastic movements  
- Stiffness | This study investigates the usefulness of nonverbal features of speech to supplement speech-based interaction. The authors used different aspects of verbal sentences, such as speech rate, duration, volume and the position of the speaker (approximated through audio localization). They could show that speech analysis provides additional interaction possibilities, yet the user would need to use speech with a different intention than normal (which is to convey emotion, rather than to control interaction). Also, the authors report technical limitations of the prototype. | At the time of publication, the prototype had not been evaluated with users. The position metric is not available to users who cannot shift or rotate their bodies. |
| “Automatic Eyewinks Interpretation System Using Visual control interface based on the recognition of eye winks” | Visual control interface | - Paralysis from the neck downward  
- Tremor in the extremities | The alternative uses face recognition to locate the eyes and determine if they are open or closed. Eye winks are then transformed into a binary code (1=open, 0=closed). The interface contains 65 possible commands. The paper | The interface has not been tested with people with impairments yet. Therefore, no information about feasibility |
### APPENDIX A · OVERVIEW OF CONTROL ALTERNATIVES

<table>
<thead>
<tr>
<th>Publication</th>
<th>Type of Controller</th>
<th>Properties it is suitable for</th>
<th>Summary</th>
<th>Annotations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face Orientation Recognition For Human-Machine Interface”</td>
<td></td>
<td>▪ Absence of fine motor skills in the upper extremities</td>
<td>mainly describes the technical aspects of the solution, including frameworks and algorithms used.</td>
<td>or the overall user experience is available.</td>
</tr>
<tr>
<td>(Mamatha &amp; Ramachandran, 2009)</td>
<td></td>
<td>▪ Muscle weakness</td>
<td></td>
<td></td>
</tr>
<tr>
<td>▪ Stiffness</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>“The Eye Wink Control Interface: Using the Computer to Provide the Severely Disabled with Increased Flexibility and Comfort” (Mamatha &amp; Ramachandran, 2009)</td>
<td>Visual control interface based on the recognition of eye winks</td>
<td>▪ Paralysis from the neck downward</td>
<td>This alternative comprises a pair of eye glasses that are equipped to register winks using infrared technology. Commands are issued through closing or opening one or both eyes. Test participants found the device to be unobtrusive and comfortable. The possibly biggest challenge lies in filtering “accidental” winks from actual commands. Also, the user needs to be able to wink distinctly with both eyes.</td>
<td>The prototype was not tested with people with impairments, although it was thought to be used to operate electric wheelchairs in the future.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▪ Tremor in the extremities</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>▪ Absence of fine motor skills in the upper extremities</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>▪ Muscle weakness</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>▪ Stiffness</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“Communication via eye blinks and eyebrow raises: video-based human-computer interfaces” (Grauman et al., 2003)</td>
<td>Visual control interface based on the recognition of eye movements</td>
<td>▪ Paralysis from the neck downward</td>
<td>This paper presents two visual interaction tools that can be used to issue mouse clicks. BlinkLink automatically detects eye blinks and measures their duration; a voluntary long blink triggers a mouse click. Additionally, blink patterns are coupled to a small vocabulary (using a type of binary code). EyebrowClicker triggers clicks through eyebrow raises. The systems were tested with simple games and spelling software, demonstrating the accuracy of detection. Recognition does not depend on lighting or skin colour and the computational resources required are limited.</td>
<td>These alternatives are only available for users who can control their eye blinks or eyebrow movement.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>▪ Tremor in the extremities</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>▪ Absence of fine motor skills in the upper extremities</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>▪ Muscle weakness</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>▪ Spastic movements</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>▪ Stiffness</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Publication</td>
<td>Type of Controller</td>
<td>Properties it is suitable for</td>
<td>Summary</td>
<td>Annotations</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------</td>
<td>--------------------------------</td>
<td>---------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| “EyePhone: Activating Mobile Phones With Your Eyes” (Miluzzo et al., 2010) | Visual control interface based on the recognition of eye movements and gaze | - Paralysis from the neck downward  
- Tremor in the extremities  
- Absence of fine motor skills in the upper extremities  
- Muscle weakness  
- Spastic movements  
- Stiffness | This paper presents a system for hands-free interaction with mobile applications, controlled by the user’s gaze and eye movements. For example, eye winks can be used to emulate clicks. To track the user’s eyes, the inbuilt front camera of the phone was used. While the system is simple, the accuracy of the technology used depends on conditions such as lighting and movement of the device. | The authors have tested the technical behaviour of the prototype, but its usability has not yet been evaluated with target users. |
| “NeuroPad: Use Cases For A Mobile Physiological Interface” (Matthies et al., 2012) | Neuronal headset controlled with eye and head movements | - Paralysis from the neck downward  
- Tremor in the extremities  
- Absence of fine motor skills in the upper extremities  
- Muscle weakness  
- Spastic movements  
- Stiffness | This paper presents a system for hands-free interaction with an iPad. It uses a commercially available neuronal headset that reads physiological signals. The user can issue commands through eye winks and head movements. However, the authors report limited data and signal quality, making the device less of a stand-alone option than a source of supplemental input. | The system is not targeted specifically towards people with impairments and has not been tested with that user group. |
| “EyeDraw: Enabling Children with Severe Motor Impairments to Draw with Their Eyes” | Drawing software that uses eye tracking for input | - Paralysis from the neck downward  
- Tremor in the extremities | This alternative uses eye gaze and dwell time for drawing lines. The user defines starting and end point of a line by fixating them for a certain time. There are two modes, looking and drawing which are indicated by the colour of | The application is a software specifically developed to facilitate drawing for children |
<table>
<thead>
<tr>
<th>Publication</th>
<th>Type of Controller</th>
<th>Properties it is suitable for</th>
<th>Summary</th>
<th>Annotations</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Hornof &amp; Cavender, 2005)</td>
<td></td>
<td>▪ Absence of fine motor skills in the upper extremities</td>
<td>the <em>eye cursor</em> that follows the user’s gaze. The paper mentions two versions of the software; one with minimal commands and one with additional tools. Both were evaluated successfully with target users. The authors reported the biggest issue to be initial difficulties with learning the drawing technique.</td>
<td>with severe physical impairments.</td>
</tr>
<tr>
<td>“GazeSphere: Navigating 360-Degree-Video Environments in VR Using Head Rotation and Eye Gaze”</td>
<td>Head-mounted display using rotation and eye gaze</td>
<td>▪ Paralysis from the neck downward</td>
<td>This paper presents a navigation system for virtual reality applications displaying 360-degree videos. The system tracks eye gaze and head movement while the user is wearing a head-mounted display. It is described as unobtrusive and intuitive, yet the authors do not report user tests.</td>
<td>This navigation concept has not been developed for people with impairments as a target group, neither has it been evaluated with them.</td>
</tr>
<tr>
<td>(Pai et al., 2017)</td>
<td></td>
<td>▪ Tremor in the extremities</td>
<td></td>
<td></td>
</tr>
<tr>
<td>“Use of Eye Movements for Video Game Control”</td>
<td>8Visual control interface using eye tracking</td>
<td>▪ Paralysis from the neck downward</td>
<td>This study investigates the use of a commercially available eye tracker for video game control. The authors evaluated it with different gaming genres (shooter, role playing game, arcade). They found that using eye gaze for control can increase immersion and can, in some cases, be quicker and more comfortable than using the computer mouse.</td>
<td>The system is not targeted specifically towards people with impairments and has not been tested with that user group.</td>
</tr>
</tbody>
</table>
Appendix B: Application Workflow

1. Initialize video player
2. setupOverlay()
3. check for type of input device
4. prepare input data and re-map it
5. displayPointers()
6. displayPlaytime()
7. displayArrows()
8. lookingActs()
9. movementActs()
10. metaActs()

> 10s remaining in video clip?

Yes

- displayArrows()
- lookingActs()
- movementActs()
- metaActs()

No

- draw() loop

Data available?

Yes

- draw video sphere and overlay background
- draw video and overlay layers

No

open serial communication

Application stopped by user

function of the Video library
**1: initialise video player**

- Start
- create new movie object for each video clip
- set first video clip as "currentVideoClip"
- start playing "currentVideoClip"
- change camera coordinates to put the viewer’s viewpoint into the video sphere
- create the video sphere object and set "currentVideoClip" as its texture
- apply changed camera coordinates
- End of sub-process

**2: setupOverlay()**

- Start
- set style of pointer object
- set anchor points of pointer object
- set style of arrow object
- set anchor points of arrow object
- End of sub-process
3: check for input device type

Start

write index position of the first occurrence of substring "le3dp" into "controllerType1"

write index position of the first occurrence of substring "USBHIDJoystick" into "controllerType2"

if the substring is found, the first value is 0

Is controllerType1 = 0?

Yes

set "logitechJoystick" to true

set value range to 0–1023

No

set value range to 0–255

Is controllerType2 = 0?

Yes

set "genericGamepad" to true

No

End of sub-process
4: prepare input data for mapping

Start

split the string that contains the input values and write them into an array

create a new array that will hold the input as integer values

create a new array that will hold the input as float values

\[ i = 0 \text{ to } i < \text{length of the array} \]

\( \text{translate hex to int values} \)

\( \text{write into int array} \)

\( \text{map unhexed values to standardized range (0–1)} \)

\( \text{write into float array} \)

\( \text{mapControllerInput} \)

End of sub-process
4a: mapControllerInput()

Start

Is the device a Logitech joystick?

Yes

Normalized Y value < 0.45?

Yes

set "lookUp" to true

No

Normalized Y value > 0.55?

Yes

set "lookDown" to true

No

Normalized Y value > 0.45?

Yes

set "lookUp" to false

No

Normalized Y value < 0.55?

Yes

set "lookDown" to false

No

Normalized X value < 0.45?

Yes

set "lookLeft" to true

No

Normalized X value > 0.55?

Yes

set "lookRight" to true

No

Normalized X value > 0.45?

Yes

set "lookLeft" to false

No

Normalized X value < 0.55?

Yes

set "lookRight" to false

No

Flowchart is continued on next page…
5: displayPointers()

Start

translate each pointer object to its intended position on the overlay

draw pointer object

End of sub-process

6: displayPlaytime()

Start

transform current playtime in seconds into mm:ss format

transform total playtime in seconds into mm:ss format

draw current and total playtime on the overlay

End of sub-process

7: displayArrows()

Start

translate each arrow object to its intended position on the overlay

draw arrow object

End of sub-process

8: lookingActs()

Start

Command to look up?
Yes
rotate video sphere downwards
No

Command to look down?
Yes
rotate video sphere upwards
No

Command to look left?
Yes
rotate video sphere to the right
No

Command to look right?
Yes
rotate video sphere to the left
No

End of sub-process
9: movementActs()

Start

Command to go left?

Yes

play chosen video clip and stop the previously playing clip

No

Command to go right?

Yes

play chosen video clip and stop the previously playing clip

No

the same behavior can be implemented for selecting the top or bottom arrow

End of sub-process

10: metaActs()

Start

toggle pause

on

off

pause video clip

display "Video paused"

play video clip

Play from beginning?

set playtime to 00:00

End of sub-process
Appendix C: Arduino Code Documentation

Logitech Extreme Pro 3D

le3dp.ino

/* Simplified Logitech Extreme 3D Pro Joystick Report Parser */
#include <usbhid.h>
#include <hiduniversal.h>
#include <usbhub.h>

#include "le3dp_rptparser.h"

// Satisfy the IDE, which needs to see the include statement in the ino too.
#if !defined(__MIPSEL__)
#include <spi4teensy3.h>
#endif
#include <SPI.h>

USB Usb;
USBHub Hub(&Usb);
HIDUniversal Hid(&Usb);
JoystickEvents JoyEvents;
JoystickReportParser Joy(&JoyEvents);

void setup()
{
    Serial.begin(115200);
#if !defined(__MIPSEL__)
    while (!Serial); // Wait for serial port to connect - used on Leonardo, Teensy
    and other boards with built-in USB CDC serial connection
#endif
    Serial.println("Start");
    Serial.println("le3dp");

This file needs to be uploaded to the Arduino in order to connect the Logitech Joystick.

The command to print “le3dp” was added as a means to identify the connected device in the Processing sketch.
```cpp
if (Usb.Init() == -1)
    Serial.println("OSC did not start.");

delay(200);

if (!Hid.SetReportParser(0, &Joy))
    ErrorMessage<uint8_t>(PSTR("SetReportParser"), 1);

void loop()
{
    Usb.Task();
}
```

---

**le3dp_rptparser.cpp**

```cpp
#include "le3dp_rptparser.h"

JoystickReportParser::JoystickReportParser(JoystickEvents *evt) :
    joyEvents(evt)
{
}

void JoystickReportParser::Parse(USBHID *hid, bool is_rpt_id, uint8_t len, uint8_t *buf)
{
    bool match = true;

    // Checking if there are changes in report since the method was last called
    for (uint8_t i=0; i<RPT_GAMEPAD_LEN; i++) {
        if (buf[i] != oldPad[i]) {
            match = false;
            break;
        }
    }

    // Calling Game Pad event handler
```
if (!match && joyEvents) {
    joyEvents->OnGamePadChanged((const GamePadEventData*)buf);
    for (uint8_t i=0; i<RPT_GAMEPAD_LEN; i++) oldPad[i] = buf[i];
}

This method handles the output of controller data to the serial. Here, it is defined in which order the commands are printed and in which format.

I added the comma as a separator between values and decided to keep the hexadecimal format. If the format should be changed, the Processing sketch needs to be changed as well (hex to int conversion is done in the “main file” block).

### le3dp_rptparser.h

```c
#include <usbhid.h>

struct GamePadEventData {
    union {  // axes and hat switch
        uint32_t axes;
    }
}
```

In this header file, the input report of the different analog controls is broken down. More information about this by the library author can be found in (Mazurov, 2014).
struct {
    uint32_t x : 10;
    uint32_t y : 10;
    uint32_t hat : 4;
    uint32_t twist : 8;
};

uint8_t buttons_a;
uint8_t slider;
uint8_t buttons_b;
};

class JoystickEvents
{
  public:
    virtual void OnGamePadChanged(const GamePadEventData *evt);
};

#define RPT_GAMEPAD_LEN sizeof(GamePadEventData)/sizeof(uint8_t)

class JoystickReportParser : public HIDReportParser
{
    JoystickEvents *joyEvents;

    uint8_t oldPad[RPT_GAMEPAD_LEN];

public:
    JoystickReportParser(JoystickEvents *evt);

    virtual void Parse(USBHID *hid, bool is_rpt_id, uint8_t len, uint8_t *buf);
};

#endif // __HIDJOYSTICKRPTPARSER_H__
Generic USB Gamepad

USBHIDJOYSTICK.ino

```c
#include <usbhid.h>
#include <hiduniversal.h>
#include <usbhub.h>

// Satisfy IDE, which only needs to see the include statement in the ino.
#ifdef do bogus include
#include <spi4teensy3.h>
#endif
#include <SPI.h>

#include "hidjoystickrptparser.h"

USB Usb;
USBHub Hub(&Usb);
HIDUniversal Hid(&Usb);
JoystickEvents JoyEvents;
JoystickReportParser Joy(&JoyEvents);

void setup() {
  Serial.begin(115200);
  #if !defined(__MIPSEL__)
    while (!Serial); // Wait for serial port to connect - used on Leonardo,
    Teensy and other boards with built-in USB CDC serial connection
  #endif
  Serial.println("Start");
  Serial.println("USBHIDJoystick");

  if (Usb.Init() == -1)
    Serial.println("OSC did not start.");
  delay(200);
  if (!Hid.SetReportParser(0, &Joy))

This file needs to be uploaded to the Arduino in order to connect the Logitech Joystick.

The command to print “USBHIDJoystick” was added as a means to identify the connected device in the Processing sketch.
```cpp
ErrorMessage<uint8_t> (PSTR("SetReportParser"), 1);

void loop() {
    Usb.Task();
}

hidjoystickrptparser.cpp

#include "hidjoystickrptparser.h"

JoystickReportParser::JoystickReportParser(JoystickEvents *evt) :
    joyEvents(evt),
    oldHat(0xDE),
    oldButtons(0) {
    for (uint8_t i = 0; i < RPT_GEMEPAD_LEN; i++)
        oldPad[i] = 0xD;
}

void JoystickReportParser::Parse(USBHID *hid, bool is_rpt_id, uint8_t len, uint8_t *buf) {
    bool match = true;

    // Checking if there are changes in report since the method was last called
    for (uint8_t i = 0; i < RPT_GEMEPAD_LEN; i++)
        if (buf[i] != oldPad[i]) {
            match = false;
            break;
        }

    // Calling Game Pad event handler
    if (!match && joyEvents) {
        joyEvents->OnGamePadChanged((const GamePadEventData*)buf);
        for (uint8_t i = 0; i < RPT_GEMEPAD_LEN; i++) oldPad[i] = buf[i];
    }
```
```cpp
uint8_t hat = (buf[5] & 0xF);

// Calling Hat Switch event handler
if (hat != oldHat && joyEvents) {
    joyEvents->OnHatSwitch(hat);
    oldHat = hat;
}

uint16_t buttons = (0x0000 | buf[6]);
buttons <<= 4;
buttons |= (buf[5] >> 4);
uint16_t changes = (buttons ^ oldButtons);

// Calling Button Event Handler for every button changed
if (changes) {
    for (uint8_t i = 0; i < 0x0C; i++) {
        uint16_t mask = (0x0001 << i);
        if (((mask & changes) > 0) && joyEvents) {
            if (((buttons & mask) > 0)
                joyEvents->OnButtonDown(i + 1);
            else
                joyEvents->OnButtonUp(i + 1);
        }
    }
}
oldButtons = buttons;
}

void JoystickEvents::OnGamePadChanged(const GamePadEventData *evt) {
    PrintHex<uint8_t> (evt->Z2, 0x80); // X2
    Serial.print("","");
    PrintHex<uint8_t> (evt->Z2, 0x80); // Y2
    Serial.print("","");
    PrintHex<uint8_t> (evt->X, 0x80); // X1
    Serial.print("","");
    PrintHex<uint8_t> (evt->Y, 0x80); // Y1
}
```

Order and formatting of the printout was changed slightly in order to standardise it for further processing. Again, commas were chosen to separate the values.
With the example controller, input values from buttons and hat switch overlap. This may be different for a similar controller of this type and needs to be tested individually.
virtual void OnGamePadChanged(const GamePadEventuserData *evt);
virtual void OnHatSwitch(uint8_t hat);
virtual void OnButtonUp(uint8_t but_id);
virtual void OnButtonDn(uint8_t but_id);

#define RPT_GEMEPAD_LEN 5

class JoystickReportParser : public HIDReportParser {
  JoystickEvents *joyEvents;
  uint8_t oldPad[RPT_GEMEPAD_LEN];
  uint8_t oldHat;
  uint16_t oldButtons;

public:
  JoystickReportParser(JoystickEvents *evt);
  virtual void Parse(USBHID *hid, bool is_rpt_id, uint8_t len, uint8_t *buf);
};

#endif // __HIDJOYSTICKRPTPARSER_H__
Appendix D: Processing Code Documentation

MAIN FILE

```java
import processing.serial.*;
import processing.video.*;

Serial myPort;
Movie videoClip1;
Movie videoClip2;
Movie videoClip3;
Movie currentVideoClip;
Movie previousVideoClip;

PGraphics video;
PGraphics overlay;

PImage img;
PShape globe;
PShape pointer;
PShape arrow;
PFont myFont;

int minValue, maxValue;
int[] intInputList;
int pauseCounter = 0;

float globeRadius = 100;
float eyeX, eyeY, eyeZ, centerX, centerY, centerZ, upX, upY, upZ;
float playbackSpeed = 1;
float scaleArrow = 1.5;
float scalePointer = 0.4;
float[] normalizedInputList;
```

The *Serial* library is required for communication with the Arduino. The *Video* library is needed to play video files within a Processing sketch.

Next, a *Serial* object is initialised to enable the connection. *Movie* objects (of the *Video* library) are initialised to hold the different video clips that are played in the application.

Each PGraphics object equals a transparent layer in the sketch. A separate layer is needed for the overlay, because the video viewer works with a distorted perspective.
String controllerInput;
String[] inputValues;
boolean logitechJoystick, genericGamepad = false;
boolean lookUp, lookDown, lookLeft, lookRight = false;
boolean goStraight, goLeft, goRight, goBack = false;
boolean pause, playFromBeginning, speedUp, speedDown, jumpForward = false;

void setup() {
  size(600, 600, P3D);
  //fullScreen(P3D);

  println("Available serial ports: ");
  println((Object[])Serial.list());
  myPort = new Serial(this, Serial.list()[1], 115200);

  video = createGraphics(width, height, P3D);
  overlay = createGraphics(width, height, P3D);

  videoClip1 = new Movie(this, "sharks.mp4");
  videoClip2 = new Movie(this, "bahamas.mp4");
  videoClip3 = new Movie(this, "jellyfish.mp4");
  currentVideoClip = videoClip1;
  currentVideoClip.play();

  eyeX = width/2;
  eyeY = height/2;
  eyeZ = globeRadius;
  centerX = width/2;
  centerY = height/2;
  centerZ = 0;
  upX = 0;
  upY = 1;
  upZ = 0;
}

In the setup() method, serial communication is established, and the video player is created (a globe with a video clip as its texture).

The correct serial port number needs to be found and inserted manually.

The changed eyeZ coordinate puts the viewer’s eye position into the middle of the sphere. All other coordinates keep their default values.
Camera coordinates are set with the variables created previously.

In this method, all overlay elements are created (pointers, arrows, other information on the screen).

In the draw() method, serial output is read and translated into individual values. This method runs as a loop. All the elements that are continuously updated need to be drawn here.

Each Arduino file contains a keyword at the very beginning that indicates the device. Depending on the device, the range of input values is set.

The Arduino output is read as a single string and needs to be split up into individual values.
for (int i = 0; i < inputValues.length; i++) {
    int intValues = unhex(inputValues[i]);
    intInputList[i] = intValues;
    float normalizedInput = map(intValues, minValue, maxValue, 0, 1);
    normalizedInputList[i] = normalizedInput;
    mapControllerInput();
}

} 

} catch (Exception e){}

video.beginDraw();
video.pushMatrix();
video.translate(width/2, height/2);
video.scale(-1, 1);
video.shape(globe);
video.popMatrix();
video.endDraw();

overlay.beginDraw();
overlay.clear();
overlay.backgroundColor(200, 200, 200, 0);
overlay.endDraw();

displayPointers();
displayPlaytime(); // included for testing, may be deleted later

if (currentVideoClip.time() >= currentVideoClip.duration()-10) {
    displayArrows();
}

lookingActs();
movementActs();
metaActs();

image(video, 0, 0);
image(overlay, 0, 0);

The values come in hexadecimal format and need to be translated into integers. Another array holds normalised values, because part of the application works with ranges. Lastly, mapControllerInput() is called to re-map the values to their corresponding commands.

As the viewer is inside the sphere, the video needs to be flipped horizontally.

At the end of the current video clip, arrows are displayed to indicate different possible paths that the viewer can choose from.

These two lines need to come last. Here, the actual layers are drawn. Only what is mentioned before them gets drawn to the screen.
void movieEvent(Movie m) {
    m.read();
}

From the Processing reference: “This event function is run when a new movie frame is available. Use the read() method to capture this frame.”

void setupOverlay() {
    overlay.beginDraw();
    overlay.shapeMode(CENTER);
    overlay.pushStyle();
    overlay.fill(255, 255, 255, 150);
    overlay.noStroke();
    pointer = overlay.createShape();
    pointer.scale(scalePointer, scalePointer);
    pointer.beginShape();
    pointer.vertex(0, 50);
    pointer.vertex(50, 0);
    pointer.vertex(100, 50);
    pointer.vertex(85, 65);
    pointer.vertex(50, 30);
    pointer.vertex(15, 65);
    pointer.vertex(0, 50);
    pointer.endShape(CLOSE);
    overlay.popStyle();
    overlay.pushStyle();
    overlay.fill(255, 255, 255, 170);
    overlay.stroke(255);
    overlay.strokeWeight(2);
    arrow = overlay.createShape();

In this method, the pointers and arrows for the overlay are defined. The vertex() function is used to create those custom shapes.
In this method, the pointers defined above are created on the overlay. They are drawn to the overlay once the method is called in draw().

```java
void displayPointers() {
    overlay.beginDraw();
    overlay.noStroke();

    // top pointer
    overlay.pushMatrix();
    overlay.translate(width/2, 32.5*scalePointer+10);
    overlay.shape(pointer);
    overlay.popMatrix();

    // bottom pointer
    overlay.pushMatrix();
    overlay.translate(width/2, height-32.5*scalePointer-10);
    overlay.rotateX(radians(180));
    overlay.shape(pointer);
    overlay.popMatrix();

    // left pointer
    overlay.pushMatrix();
    overlay.translate(32.5*scalePointer+10, height/2);
}
```
In this method, the arrows defined above are created on the overlay. Just as the pointers, the actual drawing takes place once the method is called in draw().
In the current version, the bottom arrow is not created because it is not being used.

In this method, current and total playtime are calculated. They are displayed on the overlay once the method is called in draw().
overlay.popStyle();
overlay.endDraw();

void displayVideoStatus() {
    overlay.beginDraw();
    overlay.pushStyle();
    overlay.fill(255);
    overlay.textAlign(RIGHT);
    overlay.text("Video paused", width-10, height-10);
    overlay.popStyle();
    overlay.endDraw();
}

This method is called once the video is paused and displays the video’s status on the overlay.

// This method contains an own block of commands for each type of input device
void mapControllerInput() {
    if (logitechJoystick == true) {
        if (normalizedInputList[1] < 0.45) lookUp = true;
        if (normalizedInputList[1] > 0.55) lookDown = true;
        if (normalizedInputList[1] > 0.45) lookUp = false;
        if (normalizedInputList[1] < 0.55) lookDown = false;

        if (normalizedInputList[0] < 0.45) lookLeft = true;
        if (normalizedInputList[0] > 0.55) lookRight = true;
        if (normalizedInputList[0] > 0.45) lookLeft = false;
        if (normalizedInputList[0] < 0.55) lookRight = false;

        // Alternatively, twist can be used to rotate left and right
    }
}

In this method, each input value is mapped to a specific command. It depends on the device which array index corresponds to which dimension or button.

For movement on the X and Y axis, the normalised joystick values are used (range from 0-1).

Only one alternative for horizontal rotation should be active at a time. The twist feature is specific to the joystick used in the example.
APPENDIX D · PROCESSING CODE DOCUMENTATION

//if (normalizedInputList[3] < 0.1)  lookLeft = true;
//if (normalizedInputList[3] > 0.13)  lookRight = true;
//if (normalizedInputList[3] > 0.1)   lookLeft = false;
//if (normalizedInputList[3] < 0.13)  lookRight = false;

if (intInputList[2] == 0)               goStraight = true;
if (intInputList[2] == 6)               goLeft = true;
if (intInputList[2] == 2)               goRight = true;
if (intInputList[2] == 4)               goBack = true;
if (intInputList[2] == 8) {
  goStraight = false;
  goLeft = false;
  goRight = false;
  goBack = false;
}

if (intInputList[5] == 1 && pauseCounter == 0) {
  pause = true;
  pauseCounter++;
}
if (intInputList[5] == 0 && pauseCounter == 1) {
  pauseCounter++;
}
if (intInputList[5] == 1 && pauseCounter == 2) {
  pause = false;
  pauseCounter++;
}
if (intInputList[5] == 0 && pauseCounter == 3) {
  pauseCounter = 0;
}
if (intInputList[5] == 2) {
  playFromBeginning = true;
} else {
  playFromBeginning = false;
}

Paths are chosen with the hat switch. With those buttons, it is not viable to use a range. Instead, fixed values are used.

To give the command to pause and unpause the video, the fire button at the back of the handle is used.

The button to the side of the handle can be used to play the video from the beginning.
if (genericGamepad == true) {
    if (normalizedInputList[1] < 0.45)     lookUp = true;
    if (normalizedInputList[1] > 0.55)     lookDown = true;
    if (normalizedInputList[1] > 0.45)     lookUp = false;
    if (normalizedInputList[1] < 0.55)     lookDown = false;
    if (normalizedInputList[0] < 0.45)     lookLeft = true;
    if (normalizedInputList[0] > 0.55)     lookRight = true;
    if (normalizedInputList[0] > 0.45)     lookLeft = false;
    if (normalizedInputList[0] < 0.55)     lookRight = false;
    if (intInputList[4] == 31)            goStraight = true;
    if (intInputList[4] == 143)           goLeft = true;
    if (intInputList[4] == 47)            goRight = true;
    if (intInputList[4] == 79)            goBack = true;
    if (intInputList[4] == 15) {
        goStraight = false;
        goLeft = false;
        goRight = false;
        goBack = false;
    }
    if (intInputList[3] == 255 && pauseCounter == 0) {
        pause = true;
        pauseCounter++;
    }
    if (intInputList[3] == 128 && pauseCounter == 1) {
        pauseCounter++;
    }
    if (intInputList[3] == 255 && pauseCounter == 2) {
        pause = false;
        pauseCounter++;
    }
    if (intInputList[3] == 128 && pauseCounter == 3) {
        pauseCounter = 0;
    }
    if (intInputList[3] == 0) {
        In some cases, the generic gamepad uses different values than the joystick. Here, again, integers had to be used instead of the normalised values, to match the exact number value.

On the gamepad, the pause command is issued by moving the left stick down.

Moving the left stick up gives the command to play the video from the beginning.
```java
playFromBeginning = true;
} else {
    playFromBeginning = false;
}
}

// Alternatively, the arrow keys can be used as input
// Left this in for testing purposes
void keyPressed() {
    int k = keyCode;
    if (k == UP) lookUp = true;
    if (k == DOWN) lookDown = true;
    if (k == LEFT) lookLeft = true;
    if (k == RIGHT) lookRight = true;
    if (k == BACKSPACE) playFromBeginning = true;
    if (k == ALT) speedUp = true;
    if (k == CONTROL) speedDown = true;
    if (k == TAB) jumpForward = true;
    if ((k == ENTER || k == RETURN) && pause == false) {
        pause = true;
    } else if ((k == ENTER || k == RETURN) && pause == true) {
        pause = false;
    }
}
```

The function keyPressed() is called once a key is pressed, and the respective key is stored in the keyCode variable.

The video player commands were first tested with the arrow keys, and other common keys that can be detected by the keyCode variable.

This part is not vital for the application, but was left in case it could be useful for testing.

```java
void keyReleased() {
    int k = keyCode;
    if (k == UP) lookUp = false;
    if (k == DOWN) lookDown = false;
    if (k == LEFT) lookLeft = false;
    if (k == RIGHT) lookRight = false;
}
```

This method is the counterpart to keyPressed().
if (k == BACKSPACE) playFromBeginning = false;
if (k == ALT) speedUp = false;
if (k == CONTROL) speedDown = false;
if (k == TAB) jumpForward = false;}

The lookingActs() method handles the rotation of the field of view. This is done by rotating the video globe by one radian each time input is received from the control unit.

If the FOV should rotate faster or slower, the number of radians could be increased or decreased.
// Enables the user to choose on which video path to continue
void movementActs() {
    if (goStraight == true) {
        // not implemented in current version
    }
    if (goLeft == true) {
        previousVideoClip = currentVideoClip;
        currentVideoClip = videoClip2;
        currentVideoClip.play();
        previousVideoClip.stop();
        globe.setTexture(currentVideoClip);
    }
    if (goRight == true) {
        previousVideoClip = currentVideoClip;
        currentVideoClip = videoClip3;
        currentVideoClip.play();
        previousVideoClip.stop();
        globe.setTexture(currentVideoClip);
    }
    if (goBack == true) {
        // not implemented in current version
    }
}

// Mouse click option kept for testing purposes
void mousePressed() {
    // middle arrow
    if (mouseX >= width/2-50 && mouseX <= width/2+50 &&
        mouseY >= height/2-50 && mouseY <= height/2+20) {
    }
    // left arrow
    if (mouseX >= width/2-210 && mouseX <= width/2-80 &&
        mouseY >= height/2-30 && mouseY <= height/2+40) {
        previousVideoClip = currentVideoClip;
    }
}

The movementActs() method handles what happens once the user selects one of the arrows displayed towards the end of the current video.

In this version, they can choose to either go left or right. Then, a new video clip is played and the previous one stops.

The mousePressed() function is built into Processing like keyPressed() and keyReleased(). In this case, it is called when the mouse is clicked in a certain area on the screen.

This part is not vital to the application and was left only for testing purposes.
currentVideoClip = videoClip2;
currentVideoClip.play();
previousVideoClip.stop();
globe.setTexture(currentVideoClip);
}

// right arrow
if (mouseX >= width/2+80 && mouseX <= width/2+210 &&
    mouseY >= height/2-30 && mouseY <= height/2+40) {
    previousVideoClip = currentVideoClip;
currentVideoClip = videoClip3;
currentVideoClip.play();
previousVideoClip.stop();
globe.setTexture(currentVideoClip);
}
}

// Controls typical video features such as play/pause, jump, playback speed
// Only play/pause toggle is implemented for controllers so far
void metaActs() {
    if (pause == true) {
        currentVideoClip.pause();
displayVideoStatus();
    } else {
        currentVideoClip.play();
    }
    if (playFromBeginning == true) {
        currentVideoClip.jump(0.0);
    }
    if (speedDown == true && playbackSpeed > 0.25) {
        playbackSpeed = playbackSpeed-0.25;
currentVideoClip.speed(playbackSpeed);
    }
    if (speedUp == true && playbackSpeed < 3.75) {
        playbackSpeed = playbackSpeed+0.25;
currentVideoClip.speed(playbackSpeed);
}

This method contains video-specific commands. They are executed with the help of different functions of the Video library.
if (jumpForward == true && currentVideoClip.duration() - currentVideoClip.time() >= 10) {
    currentVideoClip.jump(currentVideoClip.time() + 10);
}