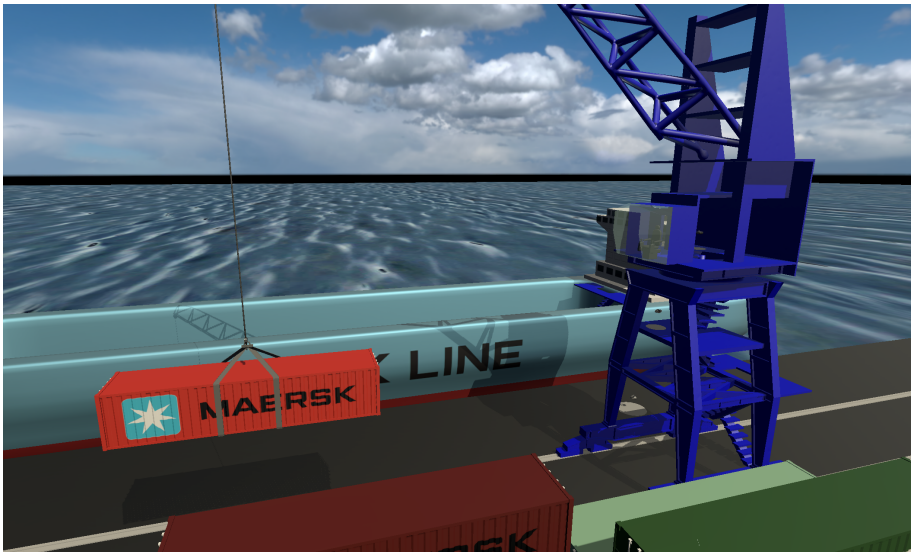
Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Robert Eric Maikher

How to create accessible Virtual Reality (VR) experiences to aid young job seekers with career guidance

Master's thesis in Computer Science
Supervisor: Monica Divitini
Co-supervisor: Ekaterina Prasolova-Førland
July 2021



NTNU
Norwegian University of
Science and Technology

Robert Eric Maikher

How to create accessible Virtual Reality (VR) experiences to aid young job seekers with career guidance

Master's thesis in Computer Science
Supervisor: Monica Divitini
Co-supervisor: Ekaterina Prasolova-Førland
July 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

Growing unemployment among young adults in Norway has been a problem for decades. As the country's demographic gets older, it is crucial that as many youths as possible become part of the workforce. A collaboration between NAV and IMTEL at NTNU was created in order to engage young adults using applications made using new Virtual Reality technology and use game elements to inform them about career choices and motivate them into working life. However, the accessibility of these applications is a problem, as current apps require expensive Virtual Reality equipment connected to powerful computers, and the pandemic places further limitations on use of this equipment due to health concerns.

This master project investigates two potential improvements to accessibility: one being a cheaper, standalone VR device that does not require a computer to function, and the other omitting the VR equipment entirely to run on computers as regular computer games do. The development process of applying those solutions to an existing application is documented in order to support applying the solutions to other educational applications. The conversion of the VR variant of the game from the SteamVR framework to the Unity XR framework is especially noteworthy, as the framework provides support for all currently available VR devices and the process has yet to be explored for educational apps. The technologies and resulting application are evaluated through user testing and qualitative developer analysis, and ultimately lay the groundwork for future efforts of creating a unified codebase across IMTEL's educational applications to support development on existing and new applications.

Video

A video showcasing the final application can be seen here:
https://youtu.be/ijF93XrwbQ4

Sammendrag

Arbeidsledigheten blant unge i Norge har økt de siste tiårene. Det er kritisk at flest mulig unge kan bidra med arbeidskraft mens demografien i landet blir eldre. Et samarbeid var startet mellom NAV og IMTEL hos NTNU for å skape apper i Virtuell Virkelighet som bruker spillelementer for å lære unge om arbeidsplasser og motivere dem til å søke jobb. Disse appene har blitt godt mottatt av unge, men har fortsatt problemer med tilgjengelighet. De krever dyrt utstyr for Virtuell Virkelighet som må kobles til kraftige datamaskiner. Pandemien som startet i 2019 har også satt begrensninger på bruk og deling av dette utstyret.

Denne masteroppgaven undersøker to måter å gjøre appene mer tilgjengelige. En av mulighetene er nytt Virtuell Virkelighet-utstyr som er billigere og ikke trenger en datamaskin for å fungere. Den andre muligheten er å lage en app som ikke krever Virtuell Virkelighet-utstyr i det hele tatt, og heller kjører på en datamaskin som et vanlig dataspill. Utviklingsprosessen til disse løsningene dokumenteres slik at den kan brukes for utvikling av andre pedagogiske apper. Konverteringen av applikasjonen fra SteamVR-rammeverket for Virtuell Virkelighet til Unity XR-rammeverket er spesielt merkverdig, siden Unity XR-rammeverket støtter all moderne Virtuell Virkelighets-utstyr og konverteringsprosessen ikke har vært dokumentert for pedagogiske apper tidligere. Den resulterende applikasjonen og teknologiene rundt den blir evaluert gjennom brukertesting og kvalitativ analyse fra utvikleren, og legger grunnarbeidet for å ha samme kodebase bak alle IMTELs pedagogiske apper.

Video

En video som viser frem sluttresultatet av applikasjonen kan ses her:
`https://youtu.be/ijF93XrwbQ4`

Acknowledgements

A huge thanks to Ekaterina for the invaluable support during this research. Their guidance about VR, feedback on work, writing advice, and giving many opportunities to conduct user tests with students and counsellors have been crucial to this thesis. A big thanks to Heidi Fossen and Arild Kristensen at NAV for feedback and arranging user tests with young job seekers. Thanks to Jobbhuset and the other NAV personnel that helped host many user tests. Thanks to Trondheim Havn for doing their best to help with evaluation despite the difficulties. Many thanks to Monica Divitini for their assistance in the research. For Mikhail and all others at IMTEL, thank you for the seminars and hardware that has made this work possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

The Norwegian job market is facing a crisis. The number of unemployed young adults has been growing since 1993 (Ekelund 2018, p. 2), and the current COVID-19 pandemic is making it even worse (NAV 2020, nr. 1, nr. 2). The Norwegian Ministry of Labour and Social Affairs concluded in 2016 that too many young adults require welfare support (sosialdepartementet 2016), and that more resources should be focused on getting young job seekers into the workforce.

The *Virtual Internship* project was created as a collaboration between IMTEL at NTNU (Norwegian University of Science and Technology) and NAV (The Norwegian Labour and Welfare Administration) in order to engage with young job seekers using immersive and interactive experiences (Prasolova-Førland et al. 2019). One of the project's contributions is the concept of *Immersive Job Taste*, which describes XR (eXtended Reality) games set in real workplaces intended to inform young job seekers about career possibilities and motivate them to seek out jobs. Multiple *Immersive Job Tastes* have been developed and put through user tests in collaboration with real workplaces, detailed in Table 1.1. These are set in Virtual Reality, containing tasks based on the ones performed at the workplaces alongside 360-degree video recordings of the workplace. Both young job seekers and career counsellors have shown enthusiasm about the usage of these applications for career guidance in job centers and schools (Prasolova-Førland et al. 2019; Henrichsen 2019).

Immersive Job Tastes are short experiences designed to show off a workplace. They are not meant to be thorough simulations of a workplace as projects of that scale require large teams and budgets to complete. A full simulation of a workplace would also take too long to play through for a young job seeker to be able to try out many career paths. Instead, *Immersive Job Tastes* are a vertical slice of a profession, showing off a few important tasks required to understand the profession and what skills are required. The tasks given to the player should be reasonably accurate in order to represent the workplace correctly and allow the young job seeker to attain a sense of mastery of the tasks, but they are not intended as a general training tool for the profession. This allows a young job seeker to experience professions they believe they might like in a short period of time, before receiving further guidance from a career counsellor.

Immersive Job Tastes still have room for improvement, however. Prasolova-Førland et al. notes the need for creating a coherent methodology for development of *Immersive Job Tastes*, in order to create new virtual workplaces quicker and make the user experience more consistent (Prasolova-Førland et al. 2019, ch. 5.2). Current *Immersive Job Tastes* have not reached a consensus on which technologies to base their development on and tend to reimplement functionality they share in common. Henrichsen claims at the conclusion of his development of the Wind Turbine Technician application that a common codebase to share between applications will be crucial for developing a methodology for creation of Immersive Job Tastes. New virtual workplaces cannot start their development from scratch every time (Henrichsen 2019, p. 187).

Application	Description	Status
FiskeVR	A breeding facility and processing plant for salmon	Finished
Wind Turbine VR	Performing maintenance as a wind turbine technician	Finished
Interview Training	Practicing the interview training	Finished
Road Construction	Road construction and driving a digger	Finished
Car mechanic VR	Perform basic maintenance as a car mechanic	Finished
Lager VR	Order fulfilment at a warehouse	Finished*
Blikkenslager VR	Metal sheet worker	Finished
Apotektekniker VR	Pharmacy worker	Finished
Snekrer VR	Carpenter	Finished
Heisekran VR	Working at a harbor and driving a crane	Under development

Table 1.1: Immersive Job Taste applications developed or under development for the Virtual Internship project.

Prasolova-Førland et al. also recommends ensuring *Immersive Job Tastes* have sufficient accessibility, both for supervisors and players of the applications (Prasolova-Førland et al. 2019, ch. 5.2). Installation and operation of *Immersive Job Tastes* must be sufficiently documented or self-explanatory that welfare personnel and schoolteachers can use them. Without the ability to be deployed to schools or job centres, the *Immersive Job Tastes* would serve no purpose. Recent technological advances in the VR sphere has led to new "standalone" VR devices that are cheaper and easier to use for people without technical backgrounds, but the current applications do not support them. Methodology would have to be created for development targetting these standalone devices.

There should also be alternative variants of *Immersive Job Tastes* that run on computers without immersive VR. Many people suffer from VR cybersickness, which is motion sickness induced from applications in Virtual Reality, and having a desktop variant available is important to avoid excluding them from the Virtual Internship program. The current COVID-19 pandemic additionally highlights another strength of desktop variants of *Immersive Job Tastes*: young job seekers could get career guidance without requiring extra VR equipment, as most households have access to computers with internet connections (SSB 2020) and most computers should be able to run Immersive Job Tastes (Wallossek and Angelini 2020).

Creating standalone and desktop variants of *Immersive Job Tastes* would be a benefit to the *Virtual Internship* project, but it is crucial that the effort required to develop and maintain these variants is minimised. The budget available to the Virtual Internship project is limited, and many applications are created by students during research projects or similar. Some exploratory work has been performed on some applications to add standalone and desktop compatibility as seen in Table 1.2, but the existing solutions are limited and separated from the original application. If these applications are to be updated in the future, the same work will have to be performed multiple times for each copy. There is a risk that the copies will diverge from the originals, meaning there is no guarantee the same solutions will apply for both. This could further increase the work required to maintain all variants. The same application should support all platforms in order to minimise the maintenance required.

The Heisekran VR (Crane VR) application was developed as a student project which the author took part in. The result was a prototype *Immersive Job Taste* with 360-degree videos and a single task for the player to complete (Hesle et al. 2020). The limited scope of the application lends itself to experimentation, thus Heisekran VR will be used as the basis for development during this research project.

Application	Framework	Platform support				
		Oculus(standalone)	Oculus	Steam	WMR	Desktop
FiskeVR	SteamVR	–	✓	✓	✓	–
Wind Turbine VR	VRTK	–	✓	✓	✓	–
Blikkenslager VR	SteamVR	–	✓	✓	✓	–
Blikkenslager VR*	Unity XR	✓	✓	–	–	–
Lager VR	Unity XR	✓	✓	–	–	–
Lager VR*	–	–	–	–	–	✓
Career Labs VR	?	–	✓	✓	–	–
Heisekran VR	SteamVR	–	✓	✓	✓	–

Table 1.2: State of related applications. Shows which platforms they are compatible with. Most applications are Immersive Job Tastes, with exception of the Career Labs VR application.
* Modified variant of original that needs to be maintained separately

1.2 Research Goals

This thesis will investigate methods of improving the accessibility for *Immersive Job Tastes*. Heisekran VR will be one of the first applications within the *Virtual Internship* project to use the Unity XR software framework in order to support standalone VR devices. The framework will be evaluated for its usefulness in developing future *Immersive Job Tastes*. A desktop variant will also be created in order to investigate its value and whether it is worth the development time to create desktop variants of other, completed *Immersive Job Tastes*. The development process will be documented in order to support creating methodology for future development. Finally, this thesis and the resulting application will provide the groundwork for implementing a common codebase for functionality shared by different *Immersive Job Tastes*.

1.2.1 Research Questions

RQ 1: What value does Heisekran VR provide as an Immersive Job Taste?

RQ 2: What value do desktop variants provide for development and use of Immersive Job Tastes?

RQ 3: What value do standalone HMDs, Unity XR and its surrounding technologies provide for development and use of Immersive Job Tastes?

RQ 4: How would one develop or port an *Immersive Job Taste* to be compatible with Unity XR, standalone HMDs and desktops?

1.3 Duration

This research will be performed in the span of a year, starting from August 2020 and ending in June 2021. The work performed in 2020 is pre-thesis work, resulting in a preliminary report. Chapters 5 through 8 covers work that was performed during the pre-thesis period. Chapters 9 through 10 covers work that was performed for the thesis in 2021.

Chapter 2

Theory

2.1 Virtual Reality

Virtual Reality (VR) describes an experience where players are put in a computer-simulated reality different than their own. VR technology replaces the player's senses of the outside world with virtual equivalents in order to make them feel they are truly present in the computer-simulated world. Such a sense of "presence" is key to the definition of Virtual Reality (Steuer 1992). Advanced displays and head tracking technology are often used to replace the player's sense of vision and hearing in order to put the entire virtual world inside the player's field of view. VR additionally tends to employ naturalistic controls using sensors or special controllers in order to track the player's head and hands and allow them to interact with the world using hand gestures or motions. The end result is "A computer-generated digital environment that can be experienced and interacted with as if that environment is real" (Jerald 2016).

The term VR has broad usage in literature with different meanings. To avoid confusion, a distinction must be made between *immersive* VR as described above, and *non-immersive* (or *Desktop*) VR. Non-immersive VR has been used to describe computer simulations that are intended to mimic real life environments. These environments are interacted with using traditional computer interfaces like computer monitors, keyboards, mice and gamepads. Non-immersive VR environments cannot evoke the same sense of presence, and it is therefore important not to conflate immersive VR experiences with non-immersive ones (Johnson-Glenberg 2018). Any further usage of the term VR will therefore refer exclusively to the immersive form of VR unless otherwise specified.

2.1.1 Immersion & Presence

The term immersion is often defined differently in different literature, and the terms immersion and presence often become conflated and used interchangeably. It is therefore important to define what they mean in the context of this thesis.

(Mel Slater and Wilbur 1997) defines immersion as an objective property of a system based on how well it allows for natural movement and perception. Under this definition, a system will have higher levels of immersion as its technology improves. Better gesture and movement tracking, higher visual fidelity, higher field of view, etc, give the system a higher level of immersion. (M. Slater 2018) makes a comparison between a high-immersion system using VR technology and a low-immersion system using a computer monitor. In a high-immersion system, the player can turn their head to look around and bend down to reach under a table. In a low-immersion desktop system, if the player turns their head around they will simply no longer see the game. Higher immersion systems could entirely simulate lower immersion ones. This definition of immersion is used for the "non-immersive" and "immersive" separation of VR games mentioned above.

Another definition of immersion tends to be used in literature and informal settings in regard to

video games. For this definition, immersion is an experience where a person is so engrossed in an activity that they lose the sense of their surroundings, themselves and the passing of time. Unlike Mel Slater and Wilbur's definition, this definition of immersion is not a unique objective property of the application but a result of technological and emotional factors of a game (Dalgarno and Lee 2010). Players get immersed in games that are of sufficiently high quality (see Section 2.4), and this form of immersion is considered crucial for engaging the player and providing opportunities for learning. This type of immersion can be seen in both non-immersive (desktop) and immersive VR games, though due to the nature of the technology used in VR it is easier for players to experience immersion in VR games.

Presence is defined as a sense of "being there" in a virtual environment (Mel Slater and Wilbur 1997). The player feels as if they are present in the virtual environment and will act and react accordingly. (M. Slater 2018) notes that the players know the environment is not real, but their perception still creates the illusion of presence. This concept is not exclusive to immersive VR systems and can be seen in desktop games, but those experiences are usually fleeting and short-lived (Brown and Cairns 2004). In contrast, "in a full immersion headset experience, the feeling of being in a different location is systematic and usually instantaneous" (Johnson-Glenberg 2018).

To define immersion, (Brown and Cairns 2004) separates immersion into three levels: engagement, engrossment, and total immersion. Engagement is the lowest level of immersion, and the simplest one to attain. Engagement requires access and an investment of time from the player. The game must be accessible enough for the player to attempt to play the game in the first place. The game's genre and control scheme are examples of factors affecting access. Personal preferences and experiences may alter the player's perception of the game, and therefore its accessibility. Once a player invests enough time into the game, they become engaged in it. An engaged player is interested in the game and wishes to keep playing it but lacks the emotional attachment of higher levels of immersion. If the game's construction is sufficiently good, a player will become emotionally invested once they have spent enough time and effort in a game and attain engrossment. In this state, they lose awareness of self and their surroundings. The game takes centrepoint in the player's attention, and directly affects the player's emotions. Reaching Total Immersion is equivalent to attaining presence. The player's reality is entirely replaced by the game, and the game is the only thing that matters. The player feels a sort of detachment from self, and instead feel like they inhabit their character in the game. Achieving this state requires even more from the game's construction, as well as an ability to empathise with the character or team the player is controlling. Brown and Cairns notes that the feeling of presence granted by games at the time was fleeting and uncommon. Later empirical research supports separating immersion into these three levels (M.-T Cheng et al. 2015).

The effect of immersion on learning outcomes is uncertain. It is generally agreed upon that engagement is necessary for learning and significantly increases learning gains from serious games. While higher levels of immersion show an improvement on the gaming experience, the effect on learning outcomes is uncertain. Some studies show no correlation between learning outcomes and higher levels of immersion (Hamari et al. 2016), while others show improvements (Meng-Tzu Cheng et al. 2017). The most common concern is that higher levels of immersion risk the players focusing more on the game itself rather than the learning elements (Fernandes et al. 2016). These studies also focus on learning outcomes in the context of science learning. One study has been performed on the effects of higher levels of immersion on persuasive games, serious games that have the intention of persuading players to change their views or take specific action, where higher levels of immersion show promise (Hafner and Jansz 2018). Further studies must be made before conclusions can be made, however.

2.1.2 Motion Sickness

One of the major challenges faced by VR is motion sickness, also known as cybersickness (Jensen and Konradsen 2018). It is the most common negative health effect resulting from VR usage (Jerald 2016). (Sharples et al. 2008) found that 60-70% of participants experience an increase in symptoms after using an HMD, worse than the viewing of desktop screens. Jerald identifies two causes of motion sickness in VR: *scene motion* and *vection*.

Scene motion occurs when the environment moves in ways that are unnatural compared to the real world. Jerald splits it into intentional and unintentional scene motion, where intentional scene motion is done "to make the virtual world behave differently than the real world", and unintentional scene motion is the consequence of technological shortcomings. Intentional scene motion covers movement and change of perspective separate from the player's physical movements. Of note is unintentional scene motion, which tends to happen only when the user moves their head and the technology fails to follow. It is unclear how scene motion causes motion sickness, but it is known that too much will cause motion sickness, so it should be avoided where possible.

Vection is "an illusion of self-motion", caused when the scene moves independently of how a user is physically moving. Vection does not necessarily cause motion sickness, with linear motion usually being a smaller factor in motion sickness. This is important, as some form of vection is often necessary to traverse a large virtual environment. Acceleration, rotation, and especially angular motion should be avoided when possible, as they are more likely to cause motion sickness than linear movement.

Jerald summarises multiple theories as to why motion sickness occurs, some of which are particularly relevant to the design of the Heisekran VR application. The *rest frame hypothesis* states that the brain chooses a stationary part of a scene as a reference point, named the *rest frame*, and judges movement of objects relative to the rest frame. Having stationary objects as part of a VR scene to work as a rest frame is shown to potentially reduce motion sickness caused by vection (Duh et al. 2001). Additionally, the *eye movement theory* states that motion sickness occurs if the eyes need to move differently than they must move in the real world in order to keep an image stable on the retina. Having stationary points to focus your eyes on may alleviate motion sickness due to eye movement, and the brain might use the focus point as a rest frame (Jerald 2016).

The *postural instability theory* suggests that motion sickness occurs when an animal has not learned strategies for maintaining postural stability. When the visual scene moves differently from reality, users will attempt to compensate for the visual movement by moving their body, causing postural instability and motion sickness (Jerald 2016). One study suggests that seated users experience less cybersickness than standing users (Pölönen 2010), which Jerald links to this theory. Multiple popular VR games with intense visual movement also suggest playing while seated to mitigate motion sickness (Zero 2020; McWhertor 2020). Jerald also mentions that users report discomfort when their view in the game is lower than their usual height.

(Jerald 2016) notes that experience with VR will cause the user to be more resilient to cybersickness, letting the user get their "sea legs", so to say. As most users of *Virtual Internship* applications will be entirely new to VR, they will not have had the opportunity to build a resistance to cybersickness. It is therefore important to reduce the factors that can cause motion sickness in applications created for the *Virtual Internship* project.

2.1.3 Different Levels of Virtuality

Entirely replacing the player's view with a virtual environment is not the only use of VR-related technology. One may wish to bring in elements of the real world into the virtual or add some virtual elements to the real world to enhance the player's understanding of the objects they are seeing.

(Milgram et al. 1995) coins the Mixed Reality term and the Reality-Virtuality continuum, which shows a continuous scale between Virtual Reality (VR) and Augmented Reality (AR), shown in Figure 2.1. On one end, Virtual Reality is entirely devoid of elements of the real world, being an exclusively virtual environment. Adding more elements of the real world to the virtual one brings an application closer to Augmented Reality. Once elements of the real world are included, it becomes part of Mixed Reality (MR). To reduce confusion, the term eXtended Reality (XR) was created to encompass VR and MR and has popular use in commercial and informal settings (Henrichsen 2019). Their relations can be seen in Figure 2.2.

Most literature and legacy technology uses the "VR" moniker, while more recent commercial technologies use "XR" to indicate they can be used for equipment across the Reality-Virtuality

continuum. The "MR" term is also occasionally used commercially. As the work described in this thesis relates to VR, the term VR will generally be used throughout. The term XR will be used in relation to technologies explicitly named after XR, but with a focus on VR.

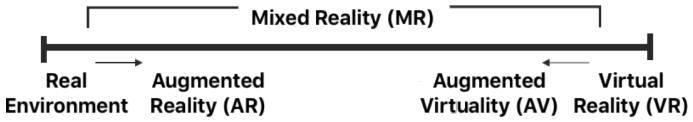


Figure 2.1: The Reality-Virtuality continuum. Taken from (Milgram et al. 1995)

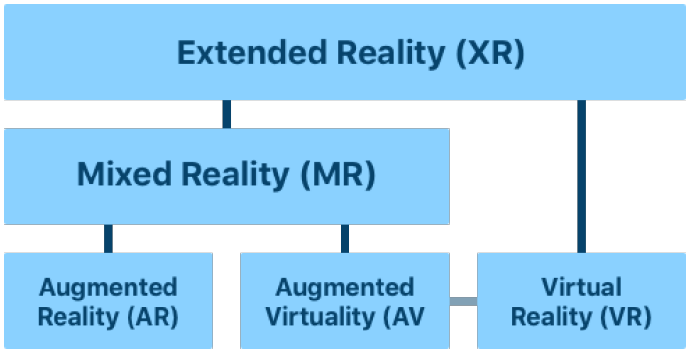


Figure 2.2: The eXtended Reality hierarchy. Taken from (Henrichsen 2018)

2.2 Immersive VR technology

(Steuer 1992) focuses on the importance of not defining Virtual Reality by the technology or hardware it uses, but by the experience it provides. The principles discussed relating to VR are not tied to the technology currently in use. Nevertheless, understanding the state of current immersive VR technology is crucial to understanding the kind of applications that can be created. The accessibility, usability and cost of the technology affects how the applications can be deployed.

2.2.1 World-fixed Displays

World-fixed displays use stationary real-life physical surfaces in order to present the virtual world to the user. The virtual world is commonly displayed using monitors or projectors, and the user can interact with the world using some form of head tracking and controllers. The advantage of world-fixed displays is that they do not place as strict requirements on the accuracy and latency of head tracking, as the stimuli on the user is not as dependent of head motion (Jerald 2016). The disadvantage is that fixed world displays are expensive and require a lot of dedicated space, preventing widespread deployment. The most common example is the CAVE system seen in Figure 2.3, which was the subject of much research before recent head-mounted displays were available (Freina and Ott 2015).

Figure 2.3: A user inside the CAVE VR system. Taken from (Davepape 2021)

2.2.2 Head Mounted Displays

Head-mounted displays use displays and speakers that are mounted to the user's head in order to present the virtual world. See-through HMDs exist and allow the user to see the real world through the HMD to overlay the virtual world above it for Augmented Reality purposes. The focus of this thesis is enclosed HMDs, however. These entirely obscure the user's vision and replace it with the virtual world. As much of the stimuli on the user now comes from the device, it is critical that the head tracking is accurate. When the user moves their head to look around, the virtual world should follow. Otherwise, it could lead to cybersickness, as described in Section 2.1.2.

HMDs are not a new technology, but recent HMDs are more accessible and affordable (M. Slater 2018). The commercial release of HMDs in 2013 at a low price point has caused a lot of excitement within the field (Freina and Ott 2015). Since 2016, commercial HMDs have also employed controllers that track the user's hands, providing the potential for use in training application. These technologies have led to a surge in VR-related research in recent years (Renganayagalu et al. 2021).

Commercial HMDs use motion tracking allows applications to provide "room-scale" experiences. Compared to "standstill" experiences where the player is forced to stay in one spot, like a chair or the centre of a room, a room-scale experience allows the player to walk physically across short distances in the virtual environment. To avoid accidentally running into obstacles, an area of the room must be designated as safe before use. When the player walks close to the edge of the designated area, barriers appear in the virtual world to warn the user and will usually overlay a video feed of the real world if the HMD has any cameras. This allows HMDs to create large virtual spaces for the player to interact with, even compared to solutions like CAVE.

Most HMDs are *tethered*, which is to say they cannot work on their own. They connect to a computer through a cable, and the computer is responsible for running the applications shown on the HMD. The system requirements for the computers are relatively high, requiring mid-range gaming computer hardware to run at acceptable levels (Oculus 2020b). Recent technological developments have led to *standalone* HMDs, which do not require a computer to work. VR applications run straight on the device, so they are entirely self-sufficient. They use the same hardware as smartphones internally and run the Android OS, but still contain the same sensors and viewport technology as tethered HMDs.

Despite the improvement provided by recent HMDs, the biggest hurdle facing VR is still accessibility (Jensen and Konradsen 2018; Renganayagalu et al. 2021). Motion sickness is one factor, as described in Section 2.1.2. The other factor is usability. Since recent HMDs are commercial ventures, they are primarily entertainment devices and are commonly built upon proprietary systems. The software and support surrounding them assume that they are personal devices tied to specific users, and that users own at most a couple of devices. Support for multi-user contexts like classrooms or NAV offices is lacking, and assuming the user can bring their own device raises issues of impracticality and equity (Jensen and Konradsen 2018). They require frequent software updates which take time and create opportunities for the software to break. Moving an HMD to another location requires a decently long period of setup, especially for HMDs that require

external sensors. The software installation and setup process differ significantly from ecosystem to ecosystem, meaning familiarity with one ecosystem might not be transferable to another. The technical knowledge required to operate an HMD will be a challenge to most instructors (Jensen and Konradsen 2018). Most controllers are tied to the device they came with, and cannot be used with another model of HMD. If controllers are compatible with multiple HMDs, it is almost certainly limited to HMDs within the same ecosystem.

2.3 Learning in Games

Games have been a popular tool for education and training for many years. The term "Serious Games" was created to refer to the kind of games whose primary purpose is learning and not entertainment (Schollmeyer 2006). (Wouters et al. 2009) proposes dividing the learning outcomes of games into four categories: cognitive, motor skills, affective and communicative. Cognitive learning outcomes cover knowledge and problem-solving skills. Motor skills cover training to perform a task by attaining procedural knowledge and practising to perform the task faster and with fewer errors. Affective learning outcomes describe changes in attitude and motivation. Attitudes are described as "internal states that influence the choices or actions of an individual", meaning one could get a positive attitude towards learning a subject, change their behaviour in their daily life or ease an anxiety. Motivation is required to initiate the learning process in the first place. Finally, communicative learning outcomes affect communication and collaboration skills. (Tennyson and Jorczak 2008) theorises that games are uniquely suitable as learning tools by simultaneously engaging subjects' cognitive and affective processes. A meta-analysis of literature surrounding learning games supports that theory with findings that they increase self-efficacy by 20%, declarative knowledge by 11%, procedural knowledge by 14%, and retention by 9%, at least compared to traditional passive teaching methods (Sitzmann 2011). Previous studies relating to the Virtual Internship program showed that young job seekers usually play video games and are receptive to learning through video games, making the Immersive Job Taste a good fit for educating them about career opportunities (Prasolova-Førland et al. 2019). Additionally, welfare personnel mention self-efficacy as one of the biggest issues facing young job seekers, another issue that learning games are fit to fix (Prasolova-Førland et al. 2019).

Three-dimensional games, both with and without immersive VR technology, provide a set of affordances that 2D games do not (Dalgarno and Lee 2010). A 3D representation of a world is more consistent with our own, facilitating transfer of knowledge and skills to real situations and development of spatial knowledge of the explored domain. This also allows for learning and training in tasks that would be impractical or impossible to undertake in the real world due to costs or inherent dangers. Given that the game allows sufficient interactability with the environment, they can also facilitate increased motivation and engagement, on top of better collaborative learning than 2D alternatives. Dalgarno and Lee notes that "technologies themselves do not directly cause learning to occur but can afford certain learning tasks that themselves *may* result in learning [...]". The affordances provided by 3D must be used to be worthwhile, as 3D and immersive technologies are more costly to develop. This is reflected in studies that compare 2D and 3D variants of Serious Games where both variants are mostly identical other than the added dimension. They find that the 2D variants are more appealing and less taxing than the 3D ones (Richards and Taylor 2015; Ak and Kutlu 2017). A review of studies of 3D Serious Games from 1999-2009 shows that they are effective (Mikropoulos and Natsis 2011), though most studies used non-immersive technologies in this timeframe, likely due to the low availability of immersive VR technology.

Immersive VR technologies provide further potential to support learning over desktop serious games. Because of the stereoscopic view and controllers, VR applications are useful for training cognitive skills related to spatial memory, procedures and psychomotor skills, and have potential for their learning outcomes to transfer to real life scenarios (Renganayagalu et al. 2021). (Johnson-Glenberg 2018) describes "the two profound affordances" of immersive VR: The feeling of presence, and the player having agency of embodied actions. As explained in Section 2.1.1, presence occurs in non-immersive video games, so it is not a unique property of immersive VR. The feeling of presence in immersive VR games are systematic and usually instantaneous, however, compared to the short and fleeting occurrence of presence in non-immersive games.

Proponents of embodied learning hold that the body should be in movement for learning and retention to be improved. Many studies show that gestures and movement may help learners memorise and generalise what they learn better. Gestures and movements can also offload cognition during the learning task. Some researchers place high value on the congruence of the movements. Congruent gestures are gestures that match the learning content. When learning about the direction and speed of spinning gears, it is more congruent for the gear's spin to match the learner's hand movements in direction and speed (Johnson-Glenberg, Birchfield et al. 2015). Pushing a keyboard button in order to make the gear spin, or worse, moving a hand in another direction than the gear spins is an incongruent action. The actions do not need to map perfectly in magnitude and speed if the general movement matches the representation. (Johnson-Glenberg and Megowan-Romanowicz 2017) proposes a taxonomy for embodiment in education based on amount of movement, the congruency of the movement and the immersion and presence of the learner. Modern immersive VR technology allows accurate hand and gesture tracking at a low cost, making it easy to create learning tasks requiring movements high in magnitude and congruence. As mentioned earlier, VR technology also makes attaining presence and immersion easier, meaning VR simplifies the creation of embodied learning tasks.

The learner should also feel they have agency, a term describing when players experience they have control over the environment (Minocha et al. 2017). Giving the player agency allows them to pace their learning to a degree by deciding where to put their attention and for how long. The freedom of movement given by immersive VR technology also allows the player freedom to experience and manipulate the world in personalised ways, also supporting the player's agency. The player having the agency to perform actions in their own personalised way and order is also deemed important for their learning outcomes.

2.4 Designing games

A game must be fun in order for people to play it (Penelope Sweetser and Wyeth 2005). Many frameworks and heuristics have been created in order to attempt to quantify player enjoyment and provide guidelines on how to design games in order to make them fun. (Malone 1980) was early in providing a taxonomy for what makes a game fun, and therefore fun to learn. (Penelope Sweetser and Wyeth 2005) is a model for game enjoyment that was created later and has been widely used since its inception (Penny Sweetser and Johnson 2019). Brown and Cairns's definition of immersion, described in Section 2.1.1, also ultimately provides good guidelines for the design of games.

Firstly, the game's controls need to be sufficiently high quality that the player feels they have control over their own actions. They should be accurate, intuitive and well-explained (Penelope Sweetser and Wyeth 2005). If the controls are insufficient, there is a risk that the player feels their skills are irrelevant for the result of the game and will not find it fun (Malone 1980). Without proper controls, it is likely the player will not want to engage with the game at all (Brown and Cairns 2004).

Good game controls are especially important when designing VR games. One should assume that everyone who plays a VR game has no experience in VR (Johnson-Glenberg 2018). Players might not know how to move in the game and interact with objects within it, making it important to provide proper guidance. Many players might not even realise they can look, move and turn around physically. Johnson-Glenberg recommends introducing these concepts slowly to allow players to get used to them. VR controls should also be embodied and allow the player to have agency in movements, as described in Section 2.3.

The game should provide the right amount of challenge for players. The players should receive challenges in the form of goals that players are uncertain to attain (Malone 1980). The challenge cannot be too easy, as the player will feel their skills are irrelevant for attaining the goal. The challenge must simultaneously not be too difficult, otherwise the player might perceive the goal as impossible and give up. In order to be enjoyable, the goals must be clear to the player. The player's actions must provide immediate feedback whether they are progressing the goals or not

(Penelope Sweetser and Wyeth 2005). The game should guide the player to their goals to avoid the player getting lost and being unsure how to proceed (Johnson-Glenberg 2018).

Setting the player up for early failures is a good learning tool if the feedback is constructive (Johnson-Glenberg 2018). Low-impact failures are a good way of teaching the player what not to do. It is important not to make the feedback too negative to avoid hurting the player's self-efficacy too much. An abundance of negative feedback could demotivate the player and affect whether they believe the goal to be attainable or not (Malone 1980). This is especially important to consider for Immersive Job Tastes, as NAV finds many young job seekers to have poor self-efficacy (Prasolova-Førland et al. 2019).

Challenge can be adjusted with various factors. The difficulty of the goals given to the player can be adjusted automatically or based on the player's selections (Malone 1980). Challenges can also be made easier through good use of guidance and tutorials. (Penelope Sweetser and Wyeth 2005). Introducing challenges and concepts one step at a time can reduce the perceived difficulty of the game, otherwise the player could get overwhelmed (Johnson-Glenberg 2018). Providing good guidance through the goals is crucial for Immersive Job Tastes, as there is a limit to how easy the tasks can be made without misrepresenting the workplace.

A good setting for the game can contextualise goals and actions to make them more intuitive and easier to learn, in addition to engaging the player. Malone describes the backdrop of the game as a fantasy, split into *extrinsic* and *intrinsic* fantasies. Extrinsic fantasies cover when the setting of the game is simply a backdrop and not intrinsically tied to the tasks given to the player, like Hangman or making a rocket fly using trivia questions. These fantasies could be replaced by another without a major effect on the tasks themselves. Intrinsic fantasies have the setting of the game match the game's learning outcomes, i.e. learning arithmetic to aim a cannon or in the case of the Virtual Internship project, performing tasks as a worker in different professions.

The end goal is to construct a game of sufficient quality to let the player become immersed in the experience, by the non-objective measure of immersion (see Section 2.1.1). The GameFlow model states that once all the above criteria have been met, a player can reach (Csikszentmihalyi 1990)'s state of *Flow*, an experience "so gratifying that people are willing to do it for its own sake, with little concern for what they will get out of it" (Penelope Sweetser and Wyeth 2005). It also draws some parallels to Brown and Cairns's definition of Total Immersion and presence. How immersive a game can be depends on the aforementioned factors and many others. The factors include avoiding technical issues, having high visual fidelity, avoiding outside distractions, and a good deal more beyond the scope of this thesis.

These guidelines focus on making the users engage with the game and learn from it, but it is important to not make Immersive Job Tastes too fun or too simple. The games should be well designed in order to avoid any unnecessary frustrations, but they should not embellish the workplace they are representing. Better tutorials and explanations should be employed before lowering the difficulty of the tasks that are given to players to avoid misrepresenting the difficulty of participating in the workplace. Portraying professions as simpler or more enjoyable than they are risks young job seekers aiming for a profession that does not suit them in reality, leading to them dropping out after getting a job. Young job seekers already complain about misrepresentation in the job market, and previous Virtual Internship initiatives have raised concerns about gamifying Immersive Job Tastes too much. Additionally, Immersive Job Tastes already face difficulty portraying certain aspects of the workplace like communication, physical exertion and tiredness. Therefore, it is important for Immersive Job Tastes to represent professions as accurately as possible despite the limitations. (Henrichsen 2019; Prasolova-Førland et al. 2019)

2.5 Related work

The use of Virtual Reality and HMDs for learning and training has become common since the release of affordable commercial HMDs. A review of the use of HMDs in professional training found 60 studies using VR training applications in various fields (Renganayagalu et al. 2021). Most of these applications focus on education and training for tasks that are not practical to

perform in reality due to safety concerns or costs. Industrial applications are the most common, where users learn to perform assembly or maintenance within a controlled, simulated environment. Other common uses are firefighter training, safety/emergency training and aviation. Training these skills in real environments can be both dangerous and costly, so a virtual simulation that can provide spatial learning is appealing. The final common use for VR in training is for health and medical professions. There are many risky medical procedures that require practice to reduce the chance of failure but practising on real people is both dangerous and unethical.

These applications all seem to make use of the affordances provided by 3D games and VR described in Section 2.3 by focusing on cognitive skills, spatial knowledge and motor/psychomotor training in difficult or dangerous situations (Renganayagalu et al. 2021). The most comparable applications to workplace training and the Immersive Job Taste are the Industrial training applications. These have less of a focus on dangerous situations and more on cost and practicality, and train users for a task to prepare them for a profession. Internships and company visits are costly to arrange and performing too many can have a negative effect on young job seekers (Prasolova-Førland et al. 2019). Immersive Job Tastes can lower the barrier of entry and ease the burden on young job seekers, providing a similar value to the aforementioned training applications.

One notable example is an immersive VR training tool for coal miners (Grabowski and Jankowski 2015). This application was designed to train young coal miners in order to avoid workplace accidents which were common among new employees. The application modelled the steps required to perform a dangerous task of blasting work. The study tested the application using multiple types of VR equipment for interactions, including a high immersion variant that used an HMD and controllers comparable to current commercial HMDs. Miners in a training role with many years of experience were used to evaluate the application's usefulness for training. The trainees found the application useful and worthwhile even after three months, with the high immersion variant coming out on top. Many lines can be drawn to the Heisekran VR application, though it is worth noting that Immersive Job Tastes have less of a focus on skill retention after training.

Another example is an immersive VR training tool to teach welding (Fast et al. 2004). This application used an HMD and a custom controller based on an actual welding tool with 6DOF tracking on both devices and haptics built into the controller. High importance was placed on the accuracy of the result, with the authors of the study simulating the result of the welding using neural networks and accounting for the position and rotation of the controller. The tool was evaluated by a wide range of users from non-welders to welding students and welders with 30 years of experience, and most found it to be good or better in a questionnaire.

The most comparable applications to the Virtual Internship program are a set of applications designed to provide workplace training created in Canada by *Career Labs VR* (VR 2021b). Their applications run on recent commercial VR systems and are available for workplace training. The CLVR applications focus on training the user in industrial tasks like stick welding and professions like sheet metal workers. Their development process is performed in the same way as the Virtual Internship project. They interview experts within professions and academia and visit workplaces to gather information about a profession and their work tasks. They gather photographs and videos from the workplace in order to support creation of the application. A VR development team proceeds to make a training simulation based on the gathered information while collaborating closely with the experts to ensure the application is accurate.

Compared to the Virtual Internship project, CLVR has a clear advantage in budget. Their applications have greater graphical fidelity and environmental detail than Immersive Job Tastes can achieve with their resources. Immersive Job Tastes need to compromise on visuals in certain cases, which some users have noted as a negative in past research (Henrichsen 2019). Likely due to the budget, the goals of Immersive Job Tastes differ slightly from CLVR. CLVR's applications are intended to provide actual workplace training in the fields they cover. Immersive Job Tastes only intend to inform the users of the workplace and its tasks in order to motivate them to seek jobs. Training the user's skills is secondary to teaching them about the workplaces for these applications. Developing games for Virtual Reality is still an expensive prospect that takes a lot of time (Renganayagalu et al. 2021), so Immersive Job Tastes need to compromise if they wish to cover a breadth of professions.

The focus on education and variety allows Immersive Job Tastes much greater variety than CLVR. The current CLVR lineup focuses entirely on training for industrial tasks while Immersive Job Tastes cover a wider range of professions. Examples are pharmacy workers, car mechanics, road construction workers and workers at a fishery. The variety is useful for the Virtual Internships' purpose of career guidance. If every application focused on an industrial task then it would exclude any young job seekers without an interest in those fields. The graphical fidelity of CLVR also causes some issues for accessibility. The application is only compatible with tethered HMDs running on decently powerful computers (VR 2021a). The lower detail of Immersive Job Tastes should have no issues running on standalone HMDs. The technologies in use for different Immersive Job Tastes and CLVR can be compared in Table 1.2.

Of course, Heisekran VR also needs to be compared to other Immersive Job Taste applications. Its goals are identical in providing career guidance to young job seekers and give them a taste of their various professions. Special attention can be paid to the Wind Turbine VR application due to its robustness and the good documentation of its development and evaluation (Henrichsen 2019). The Wind Turbine VR application invites the users to try out being a wind turbine technician by climbing up a wind turbine and performing maintenance on it. The application has all the common aspects of Immersive Job Tastes, with a tablet for tasks and points-scoring, 360-degree videos recorded at a wind turbine, and tasks from the workplace.

When it comes to desktop VR applications, there have been countless applications made for education and training. (Dalgarno and Lee 2010), described in Section 2.3, covered many of their potential affordances. A useful example could be the water pump assembly application described in (Boud et al. 1999) due to its comparison between various non-immersive and immersive solutions. The users of the application were tasked with completing a water pump assembly in non-immersive and immersive forms of VR (and AR), before being tasked to complete the water pump assembly for real. Their performance in the assembly of the real water pump formed an evaluation for the different technologies. The stereoscopic and non-stereoscopic desktop experiences were found to perform equally to the immersive VR experience, and vastly outperforming the traditional manual.

There is a lot of variety within desktop experiences. Traditional training and guidance applications are not the only ones that could be relevant to look at. Mozilla Hubs is a desktop VR application designed for collaborative learning experiences. Users using desktop or immersive VR devices can take control of a virtual, customisable avatar and join a room with many other users. Here they can interact using text-based, voice-based and gesture-based communication, create their own objects and props for interaction and share their own media. It has found good use in classroom/presentation settings as an alternative for gathering physically for classroom teaching (Yoshimura and Borst 2021). Using the props and room editing tools, it is also possible to create environments that are suitable for learning or collaboration, the effect of which is described in (Gomes de Siqueira et al. 2021). The team who developed the Heisekran VR prototype used Mozilla Hubs for collaboration and planning during its creation process (Hesle et al. 2020).

Chapter 3

Method

This chapter describes the research process, starting with the strategy used for research and development, and following with data generation and data analysis used for evaluation.

3.1 Research process

A research project is performed in order to reach an informed conclusion regarding a problem. To support the research, a strategy must be selected beforehand consisting of certain activities and actions, and methods to collect and analyse data must be chosen in order to evaluate the research.

The research was motivated by the author's experiences with the Virtual Internship project and the current problems facing the program. The research questions seen in subsection 1.2.1 were constructed after a review of relevant literature and the state of the various Immersive Job Taste applications. The Design & Creation strategy was chosen based on the nature of the research focusing on artefacts and new technological solutions. Data would be gathered from participants through observation, interviews and questionnaires. An overview of the research process is depicted in Figure 3.1, based on the research model described in (Oates 2006).

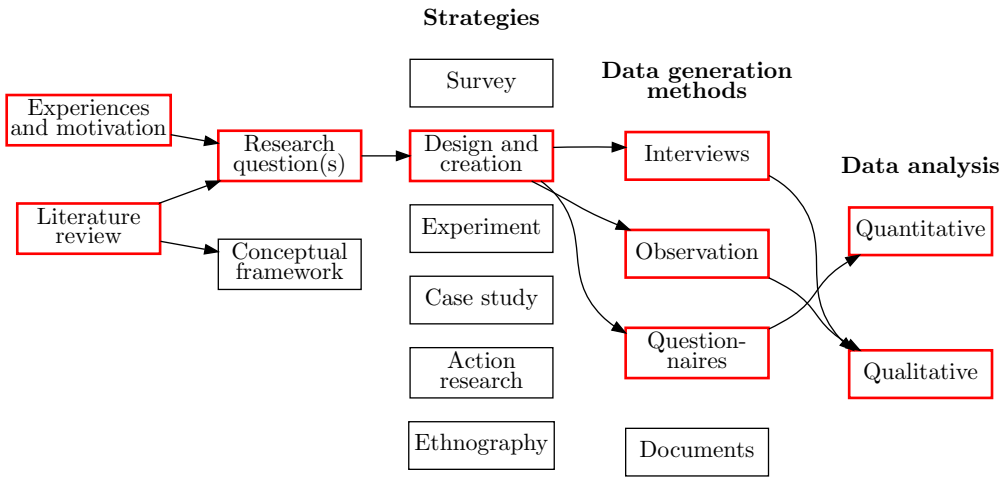


Figure 3.1: A research model describing the research strategy and data generation methods of the thesis. Red squares and arrows denote which properties are used for the thesis. Based on the Oates research model (Oates 2006).

3.2 Research Strategy - Design & Creation of Heisekran VR

The design & creation research strategy provides knowledge through the development of new IT products, also called *artefacts* (Oates 2006). The focus of this strategy is "learning via making". New technologies and methods for development are explored through an iterative process of creating an artefact with them. The strategy is a problem-solving approach split into five activities: Awareness, suggestion, development, evaluation and conclusion. *Awareness* is about recognising and articulating a problem. The *Suggestion* stage attempts to find a tentative idea for solving the problem. *Development* is where the suggested idea is implemented into the artefact. *Evaluation* assesses the developed artifact to find whether it matches expectations and whether the solution has value. *Conclusion* gathers the result from the process, writing up any knowledge gained and identifying unexpected results that could be the basis of future work. These activities are not a rigid step-by-step guide. As the design & creation strategy is iterative, each activity is performed multiple times in various order. Development and evaluation of an idea might shed light on the problem, uncover additional problems or lead to new ideas for solutions. In such a way, the experience gained through development is constantly helping create new knowledge. Such a research process is fitting for development of VR applications and Immersive Job Tastes, as many aspects surrounding the applications such as technological solutions and methodology are still uncertain. Other research projects in this domain have used the same strategy, both within the Virtual Internship project with the Windmill VR application (Henrichsen 2019) and other educational VR games (Fernandes et al. 2016).

The design & creation process will follow a specific development methodology. An initial set of requirements will be constructed based on the research questions and an examination of the initial state of the Heisekran VR application. Development of the artefact will be separated into iterations. In each iteration, the requirements with the highest priority will be chosen and solutions to fulfil the requirements will be identified and implemented. Issues encountered and knowledge gained during the development process will be considered during the development process, potentially changing the priority of requirements within an iteration. User tests will be performed using the final product of each iteration where data is gathered using observations, questionnaires and interviews where applicable. User feedback and data analysis will be used identify any issues with the application and potential solutions, changing the requirements and influencing the goals of the next iteration. Ideally, each iteration would set aside a specific amount of time for development before looking for user feedback, as seen in (Henrichsen 2019). This ensures that too much time is not spent on development without taking the time to evaluate the value and knowledge it provides. However, user tests are generally unavailable due to restrictions surrounding the pandemic as this thesis is written. Opportunities for user testing are scarce and spontaneous. Therefore, each iteration is finalised when opportunities for user testing arise. Once sufficient data has been collected, another iteration is started. Each iteration is described in its own chapter, in chapters 5-10.

The resulting artefact in this thesis will be an *instantiation*, a working system that demonstrates constructs, models, methods, ideas, genres or theories through an implementation in a computer-based system (Oates 2006).

3.3 Methods for Data Generation

This section describes the methods for data generation employed in this thesis.

3.3.1 Observation

During user tests, the author will be present to observe the players as they play through the application. This is defined as *overt* observation, where the researcher is present and the participants know they are being observed (Oates 2006). This allows the researcher to talk with the participants and uncover their thoughts on the application as they play. The main purpose of observation is to see what people are doing and how they are doing it in order to find issues within the application.

Some issues that users face might only be visible to a researcher with more knowledge about the subject matter. Players might not realise an issue exists without experience in other immersive VR games, and other issues might only be visible when observing multiple users play through the same application. Observations might uncover technical issues with the application, aspects of the game that lack feedback or user control (see section 2.4), aspects that cause motion sickness (see subsection 2.1.2), and other unexpected results. The observations are not systematic, meaning they only intend to provide qualitative data and there are no pre-defined events they record. Anything that the author deems relevant are recorded in field notes and described in the results of the user tests. Special attention should be paid when users seem perplexed, as those are usually the breakpoints (Johnson-Glenberg 2018).

3.3.2 Questionnaire

A questionnaire is a pre-defined set of questions assembled in a pre-determined order (Oates 2006). They are mostly used to obtain standardised quantifiable data about simple topics where the questions are short and unambiguous, and the answers are not controversial. To simplify gathering quantifiable data it is common for the questions to be *closed*, meaning they have predetermined answers that the respondent can choose between. These questions often use a numerical scale to measure respondents' attitudes directly. The most widely used is the Likert scale (McLeod 2019), which is a linear scale commonly ranging from 1-5 that allows the respondent to express how much they agree or disagree with a particular statement.

The questionnaire given to young job seekers will be based on the questionnaire developed for Immersive Job Tastes as described in (Prasolova-Førland et al. 2019, ch. 3.3). This questionnaire begins with a section asking about the user's background. The second section asks the user about their opinion of the individual job taste applications. The third section is about the Immersive Job Taste system's usability, containing a set of 10 standard questions based on the System Usability Scale (Brooke 1986). The fourth captures the perceived usefulness of the Immersive Job Taste concept adapted to the user group being asked. Additionally, section 2 and 4 contain *open* questions where respondents can write freeform text responses, in order to gather qualitative feedback.

3.3.3 Interview

An interview is a conversation with the specific purpose of eliciting information from the participants (Oates 2006). Interviews are commonly scheduled in advance, with specified topics for the conversation. The interviewer usually prepares questions in advance, or at least steers the course of the conversation to the topics that were agreed upon. In some cases, the questions can be provided beforehand to allow the participants to prepare answers. Interviews are used to gather information that is too complex to be gathered through a questionnaire. The questions might be too complex or open-ended, or the answers might be too complex to write as a response to a questionnaire.

Semi-structured interviews will be performed to gather detailed feedback about the application. Semi-structured interviews have questions and topics prepared beforehand that the participants will be asked during the interview but allow for re-ordering of questions and additional topics to be added if they come up during conversation.

3.3.4 Performance

In order to measure the performance of a video game, quantitative measures will have to be taken while the game is being played. The common measure of performance is Frames Per Second (FPS), which measures how many frames the system was able to produce in the span of a second on average. A *frame* is a 2D picture representing the game world that is continuously updated on a display in order to give the viewer a sense of motion. If frames are rendered quickly enough, the game world appears to be in continuous motion in the same principle as videos.

FPS averages are not ideal as they obscure the time it takes to render each single frame by creating a rolling average over a second. Instead, this thesis will use the concept of *frame time* to measure performance. The frame-time is how long it took for the game to prepare each individual frame to show the display. The ideal is to have the frame-time be lower than the time it takes the screen to render a frame, meaning there is always an image ready to be shown. Any time a frame cannot be prepared in time, the same frame that was shown last time will have to be shown to the user again. Drops in frame rate are generally not noticeable if the frame-time is not significantly longer than the display's refresh time, as the display can show every other frame and still get a decently smooth result. Immersive VR headsets even have technology to slightly shift the resulting image in order to keep tracking the user's head movement and avoid nausea, called "reprojection". Once frametimes are more than double the target frametime, the user will perceive the application as stuttering and not comfortable to play. Reprojection also starts failing once frames take too long to render, as there is simply not enough data to reproject correctly.

Frame times for each rendered frame will be gathered while performing a normal playthrough of the application. One playthrough will be performed on each relevant platform. Care will be taken to play through the applications in the same way for all platforms.

3.3.5 User tests

Table 3.1 provides an overview of user tests performed with the application.

Date	Users	Count	Section
24th September 2020	Students	10±	5.3
30th September 2020	Young job seekers	5	6.2
5th November 2020	Students	5	7.2
17-18 November 2020	NAV Personnel	4	8.2
2nd December 2020	Young job seekers	6	8.3
3rd February 2021	Students	8	9.2
7th May 2021	Young job seekers	3	11.5
3rd June 2021	Crane professionals	2	11.8
18th June 2021	Young job seekers	6	11.6
18th June 2021	NAV personnel	1	11.7

Table 3.1: An overview of the user tests performed with the Heisekran VR application. Participants were only counted if they tried the application. A count was not performed for the first user test and is therefore an estimate based off of memory.

3.4 Ethics

When performing research, the ethics of the research must be considered. This research project deals with people directly by involving them in user tests and data gathering, and the finished artefact has the potential to affect young job seekers when it is brought into use.

Participants involved in user tests will be informed of the goals of the research project and the purpose of the user test when starting. They will additionally be informed of what kind of data will be collected from their participation. They will be reminded that participation is voluntary, and that they might at any point withdraw from the user test if they wish. This will be communicated both verbally and through a consent form that can be seen in appendix C. The consent form has the contact information of the project's supervisor and the data protection officer at NTNU. Immersive VR technology has no known long-term side effects but can commonly cause discomfort in the form of cybersickness and nausea (Jerald 2016). Participants testing the application in a VR HMD will be warned of these risks and will be recommended to withdraw from the user test immediately if they feel any sign of discomfort during play.

Users will be asked to volunteer to answer the questionnaire after trying the application. They will be able to respond to the questionnaire at the location of the user test using a computer, or from home in their spare time using a URL they can enter into their browser. The questionnaire will start by explaining that all collected user data should be anonymous and request the user to contact the project's supervisor should they feel the questionnaire asks for personally identifiable information. Interview subjects will also be given the explanation and consent form in good time before the interview starts, and permission will be requested to record the interview before any recording is performed. The project has approval to collect user data from the Norwegian Centre for Research Data (NSD). All collected personal data will be treated in a secure manner and destroyed by the date given in the consent form.

As the application is intended to be used for career guidance, it is important that it provides an accurate representation of the profession to avoid misinformation. As noted in (Prasolova-Førland et al. 2019), leading young job seekers down the wrong career path can be disastrous for their ability to find employment and continue working in the long term. Effort will be put in to ensure the application provides accurate and up-to-date information.

3.4.1 Health & Safety measures

This research will be performed while the global COVID-pandemic is still underway. It is important that proper precautions are taken to prevent the spread of disease. The recommendations made by the Norwegian Health Institute will be followed during user testing. Users will be instructed to wash or disinfect their hands before and after touching any equipment. Researchers will wear masks during testing, and masks will be mandatory for users as well when they are recommended by NHI and local authorities. Equipment will be thoroughly disinfected between each user. Only a small amount of people will be present at a time. Going beyond the recommendations given by FHI, anyone using immersive VR HMDs will additionally need to wear a disposable VR mask as seen in Appendix B to prevent direct contact between their face and the equipment. A machine designed to disinfect HMDs using UV will also be used between each user for any user tests held by IMTEL where the machine is available. The number of users using the same equipment will be minimised.

3.5 Evaluation

Evaluation of the standalone VR and desktop variant will be performed based on user experience and perceived usefulness of the participants of the user tests, based on guidelines for the *Virtual Internship* project (Prasolova-Førland et al. 2019, ch. 5.2).

Evaluation of technologies and their use for development will be done qualitatively based upon the experience gained during the design & creation process.

Chapter 4

Software and Hardware

This chapter begins by introducing the crane which is modelled by the Heisekran VR ("Crane VR") application. It proceeds to describe the Heisekran VR application in its prototype state at the beginning of development. Descriptions of the Immersive Job Taste tutorial and Yrkeskatalogen applications follow. Finally, it describes the game development and Virtual Reality technologies in use during development.

4.1 The Crane

The crane in the application is modelled based on the crane at Trondheim Havn, the harbour in Trondheim. A picture of the crane can be seen in Figure 4.1. This is a type of gantry crane, designed to lift goods in and out of ships moored at the harbour. It is important to understand how the crane works in order to understand how it is simulated inside the application.

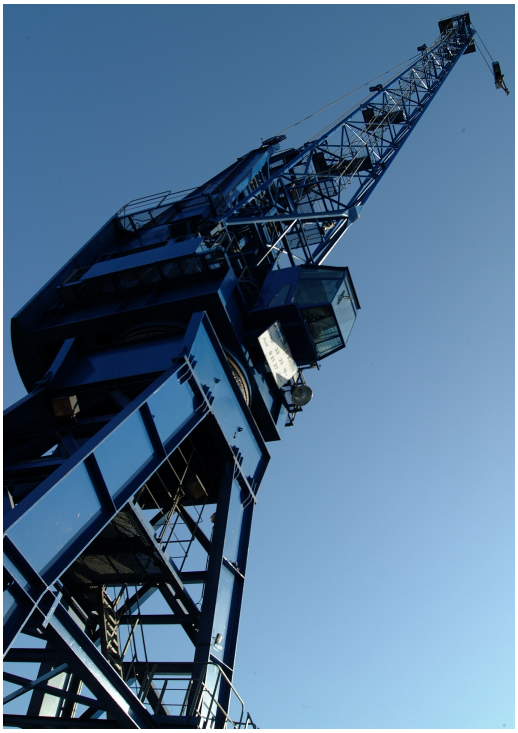
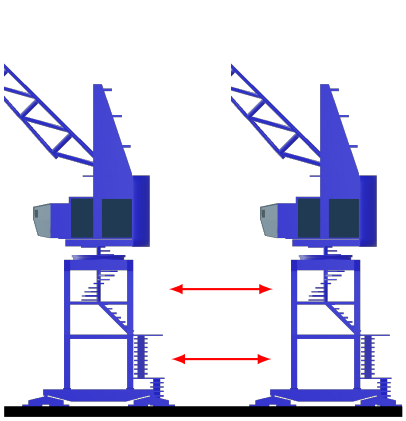
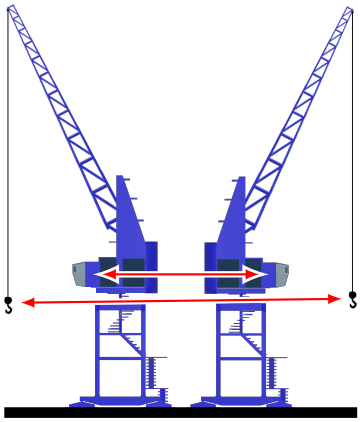


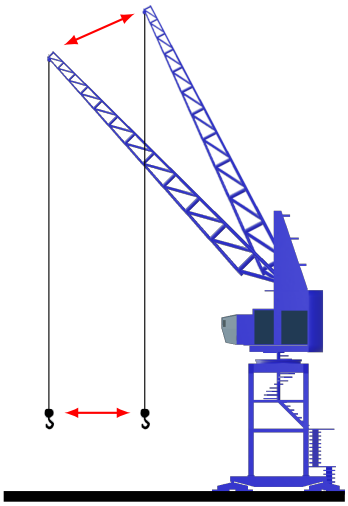
Figure 4.1: The gantry crane at Trondheim Havn. Taken from (Havn 2003)



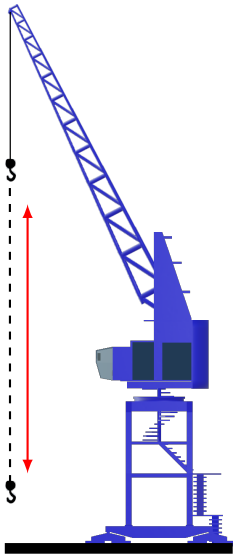
(a) Driving on rails



(b) Rotating the body



(c) Lifting and lowering the beam



(d) Extending and retracting the wire

Figure 4.2: An illustration of the crane's different capabilities.

The crane consists of three main parts. The legs of the crane form the bottom and can be considered the base of the crane. The legs are placed upon a set of rails that allow the crane to drive along the length of the harbour as illustrated in Figure 4.2a. The second part is the crane's body, situated atop the base. The crane's body contains the crane driver's cabin where the crane is steered from. The body can rotate around the legs in order to allow the crane to face any direction on the harbour as illustrated in Figure 4.2b. The crane's beam protrudes out of the crane's body. A wire is drawn out of a spool inside the crane's body and hangs from the end of the crane's beam. Unlike some other cranes, this wire is securely fastened to the end of the beam and cannot travel alongside it. Moving the wire is accomplished by lifting and lowering the beam as illustrated in Figure 4.2c, which moves the tip of the beam closer and further away from the crane and therefore the wire alongside it. The crane can extend and retract the wire using the spool inside its body, allowing it to lower and lift objects attached to the wire. This can be seen in Figure 4.2d.

4.2 Heisekran VR Prototype

The player starts in the middle of the harbour seen in Figure 4.3. The crane is the centrepiece, towering over the rest of the scene. A group of containers is stacked on one side of the crane, and a container ship is docked to the harbour on the opposite side. The crane is placed onto a set of

rails, with a group of video platforms at the end of them, as seen in Figure 4.4.

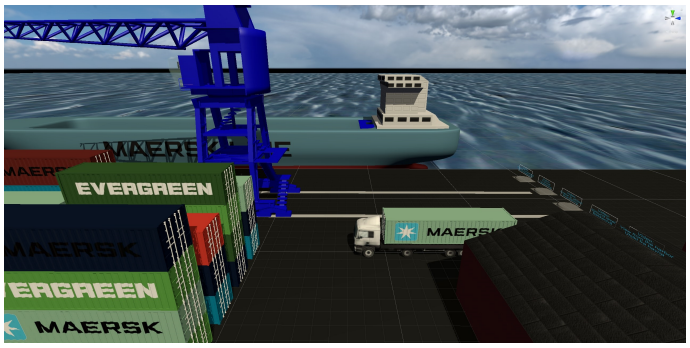


Figure 4.3: A picture of the harbour in the Heisekran VR application.

Standing on the video platforms replaces the harbour with their respective 360-degree video, recorded at Trondheim Havn. There are five 360-degree videos available. The first shows the safety equipment required before one can enter the workplace, namely a helmet, safety boots, and high-visibility clothing. The second is an explanation and showcase of the crane's controls by the crane driver, recorded from the crane's cabin. The third video shows the process of lifting a container out of the ship and onto a truck, viewed from the cabin. The fourth video shows the same process from the ground level, emphasising the ground crew receiving a load from the crane and driving it away. The last video is optional and shows the harbour and the ship from another point of view. Leaving the platform places the player back at the harbour.

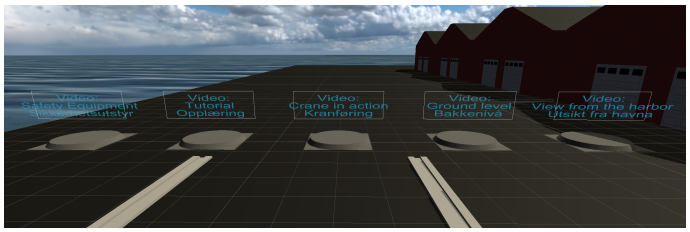
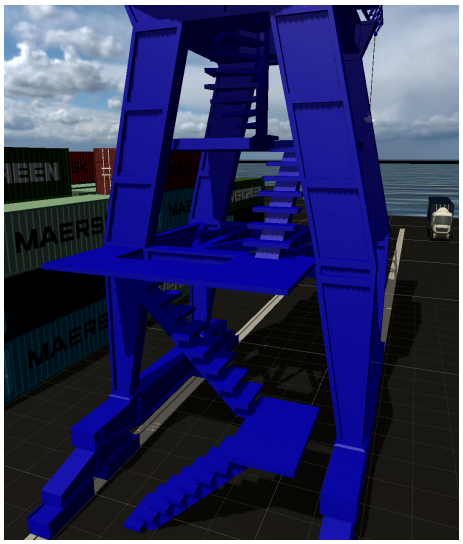


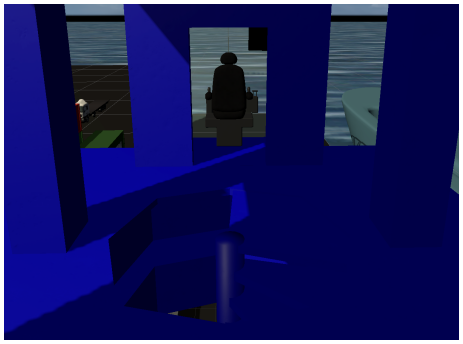
Figure 4.4: The video platforms in the Heisekran VR application.

Climbing the staircases seen in Figure 4.5a leads the player to the crane driver's cabin, shown in Figure 4.5b. From here, the player can control the crane to perform their tasks. A cable with a hook attached to it hangs from the tip of the beam. Lifting the beam moves the hook closer to the crane, while lowering the beam moves it further away. The crane automatically compensates for the difference in height as the beam moves, keeping the hook at the same level off the ground. The body of the crane rotates around the centre, moving the hook on the horizontal axis from the view of the cabin. This allows the crane to position the hook above wherever it needs to be to lift an object. The crane can extend or shorten the cable by using a spool inside the body, which grants the crane its lifting capabilities. Finally, the crane can drive along the rails on the ground, letting it position itself where it needs to be along the entire stretch of the dock.

Unlike other Immersive Job Tastes, Heisekran VR does not have explicit tasks or scoring. The player is free to interact with the application but has no goals to accomplish. The supervisor must provide the goals verbally to the player. To keep user tests consistent, the same goals have been provided to all players. A regular playthrough of Heisekran VR begins with explaining the controls to the player and letting them get comfortable. The first goal is watching the videos in order, guiding the player towards the video platforms if necessary. Afterwards, the player is told to take control of the crane and lift a container off the harbour and place it into the ship. With the stack of containers close by, the player does not need to familiarise themselves with moving the crane itself and can focus on moving the hook. If successful, the player is offered to challenge themselves by picking up a container from a truck and placing it into the ship. The crane cannot reach the containers on a truck from its starting position, so the player must drive the crane along the rails, adding another mechanic for the player to learn. Otherwise, they are finished with the application and are directed to quit.



(a) The stairs up to the crane in the Heisekran VR application.



(b) The crane driver's cabin from the outside.

4.2.1 Immersive VR controls

The controls in VR are situated on the controllers as illustrated in Figure 4.6. The player can move across the harbour by teleporting. This is done by pushing the thumbstick of a controller forward. This creates an arcing line, projected out of the player's hand in the direction they are pointing. If the player is allowed to teleport where they are aiming, a circle is shown where the line hits and the line is coloured white. This circle shows the destination the player will teleport to. When the player proceeds to release the thumbstick, their screen will flash black, and the player will be instantly moved to their destination. If the player aims somewhere they are not allowed to go, like the roof of a building or the water around the dock, the line will be coloured red and the player will not move when releasing the thumbstick. The player can also move physically across their room but will be limited by the size of their play area. In order to assist the player in looking around, they can also turn their viewpoint by pushing one of their thumbstick to the left or right to perform a "snap-turn". Their screen will flash black, and their viewpoint will be turned 30 degrees in the respective direction.

Climbing up the stairs is done by teleporting up the steps. Once at the top, teleporting into the control room will place the player into the crane driver's seat. The player starts the application standing but will have to be seated in order to reach the crane's controls. A supervisor should provide a chair at this point and guide the player to sit down. From there, the player has access to the three control sticks controlling the crane. The player holds on to a stick by hovering a hand over it and pressing the grip button on the side of the controller. The left joystick is responsible for moving the hook along the ground. Pulling the left joystick backwards lifts the beam, moving the hook closer. Pushing it away will lower the beam, moving the hook away. Dragging the left joystick to the left or right will move the hook in the same direction by rotating the body of the crane. The right joystick lifts the hook when pulled backwards and lowers the hook when pushed forwards. The small lever on the right drives the crane along the rails, forward to go towards the tip of the boat, backwards to drive towards the back.

As the crane applies rotational vection, it is important for the supervisor to monitor the player for motion sickness. At first, requiring the player to sit down when reaching the control room was an oversight during development, but it was kept due to it potentially alleviating motion sickness from giving the user better balance. The hook should provide the player with a point to focus their eyes, and the crane cabin alongside the seat and screen should provide a rest frame for the harbour to rotate around. Nevertheless, rotational movement is tough for first-time players, especially one that accelerates as the player moves the left control stick.

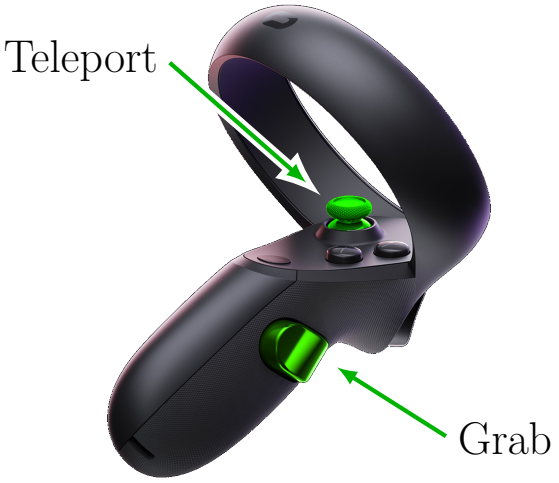


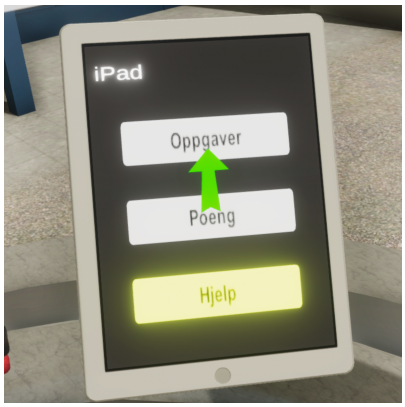
Figure 4.6: The Immersive VR controls for the prototype of Heisekran VR. Illustrated on the left Oculus Quest 1 controller. The right controller has an identical control scheme. Controller taken from (Kerman 2021)

4.3 The Immersive Job Taste tutorial

The Immersive Job Taste tutorial is a short application designed to teach players step-by-step how to play immersive VR games. Each Immersive Job Taste can assume that players have been taught the basics using the tutorial, instead of each application needing to explain the same controls to the player. Through playing the application, the player is taught how to use a virtual tablet, move around in a VR world, pick up and interact with objects, and watch 360-degree videos. These are all elements that occur in Immersive Job Tastes, and the knowledge should allow players to succeed in playing any other job taste.

The application puts the player in front of a building in a setting that appears to be a city. A text prompt in front of the building gives the player their next goal and explains how to accomplish it. It is accompanied by a floating VR controller seen in Figure 4.7a which shows the player how to use the controller to accomplish their task. The controller flashes the button the player needs to press and is animated when the game wishes to teach the player a motion to perform with the controller. The first task given to the player is opening the tablet seen in Figure 4.7b. The tutorial explains the purpose of the tablet, which is to provide the player with tasks and explain where they need to go next. Once the player has successfully selected a task and read about its requirements, the player is tasked to enter the building through a sliding door that opens automatically when the player gets close. To successfully enter the building, the player needs to learn how to teleport. The game shows this by animating the controller button presses and the aiming process. Once inside the building, the player is tasked with picking up a hammer from a table and dropping it inside the box. After succeeding, the player is brought to a 360-degree video platform and taught how to watch videos in the applications.

(a) The controller visible in the Immersive Job Taste tutorial. In this moment, the trigger flashes yellow to show the user which button to press to proceed.



(b) The Immersive Job Taste tablet in the tutorial. It has three buttons for tasks, points and additional assistance.

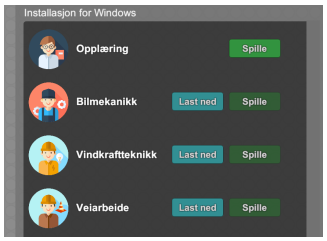
Figure 4.7: Pictures from the Immersive Job Taste tutorial.

4.4 The NAV catalogue - Yrkeskatalog

The NAV catalogue *Yrkeskatalog* (Profession Catalogue) is a desktop application simplifying the download and installation of Immersive Job Tastes to computers and standalone devices. An installer for the application can be downloaded with one click from IMTEL's website. Running the installer installs the application and allows the catalogue to run on the computer like any other program.

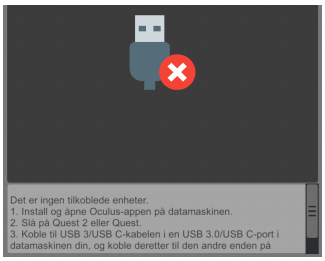
Opening the application shows two columns that list the Immersive Job Taste applications available through the catalogue, in addition to a text box providing the user with feedback in a corner. The left column seen in Figure 4.8a is dedicated to tethered VR and desktop job tastes that run on the same computer as the catalogue. The right column seen in Figure 4.8b is dedicated to standalone variants of applications. The user can click the Download ("Last ned") buttons in order to download the respective job tastes. The download progress will be seen in the text box in the corner of the application. Once the application is downloaded, the user can either click the Play ("Spille") button to start the job taste if it is a computer application, or the Install button in order to install the application to a standalone device.

If no standalone device is connected to the computer, the right column will not list standalone applications. Instead, the catalogue will show an icon telling the user that no standalone device was found. The text box will be filled with instructions explaining how to prepare a standalone device to install applications and connect it to a computer. This can be seen in Figure 4.8c.



(a) Left column with tethered VR and desktop apps.

(b) Right column with standalone apps.



(c) Right column when no standalone device is connected, alongside text box used for feedback.

Figure 4.8: The NAV job taste catalogue - Yrkeskatalog

4.5 The Unity Game Engine

A game engine is both a platform and a tool for creating videogames. It includes technologies and implementations necessary for videogames to function, like processing game logic, rendering graphics and shadows, simulating physics, dealing with assets and handling human-device input. Developers can use a game engine to quickly create an experience without spending a long time implementing the basics that are common for almost every video game.

Unity is the game engine used to create *Immersive Job Tastes*. It is the most popular game engine at a 50% market share and roughly 60% market share for VR games in 2019 (Unity 2020c; Peckham 2020). Due to its userbase, Unity has many plugins, ready-to-use solutions and good community support when it comes to questions and answers. Unity also provides the Asset Store, a storefront where users can sell their technical solutions or assets like models, sounds and animations to other users for use in their projects.

There are of course alternative game engines, the largest competitor being the Unreal Engine. They are largely the same with the same capabilities, and both are free for educational and non-profit use. Choosing a specific engine is difficult when starting at a new project, but as previous *Immersive Job Tastes* have been made for Unity, it is likely that all *Immersive Job Taste* applications will use Unity for the foreseeable future. Game engines have their own formats and idiosyncrasies that make transferring work from one engine to another a significant undertaking, so choosing another game engine is not viable for the *Virtual Internship* project at this stage.

4.5.1 Unity Objects

Objects in most game engines are arranged in a hierarchy in order to make them easier to work with. An example of some objects in a Unity hierarchy tree can be seen in Figure 4.9. An object can be the "parent" object of any number of "child" objects. Child objects are placed in the world relative to the parent object, meaning that moving the parent object also affects its children. This hierarchy is recursive, meaning child objects can be the parent of any number of their own children. This hierarchy does not only make it easier to transform and translate objects, but also makes it easier to structure code. A crane can be represented as an object which has the crane base as a child. The crane base can then have the crane housing as a child. Moving the crane object along the railings moves the entire crane but rotating just the housing does not rotate the legs with it. Additionally, any code attached to the crane object can find the base and housing just by checking its children. This is more efficient than being forced to check every object in the game world and makes it possible to create a second crane by cloning the first one without them conflicting with each other.

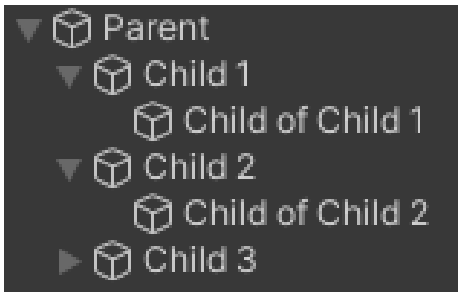


Figure 4.9: Some objects in the Unity hierarchy.

An object in the Unity hierarchy is called a "GameObject". Each GameObject has certain basic properties: position, rotation and scale. Any amount of "Components" can be attached to a GameObject in order to give it further properties. Common choices would be models and materials to give a GameObject a visual representation in the game, a "RigidBody" component to simulate physics on the GameObject, or self-written code attached as a script component. Components tend to have fields that are configurable by the developer, allowing a single component to easily have different effects on different objects. The Container GameObject can be seen in Figure 4.10.

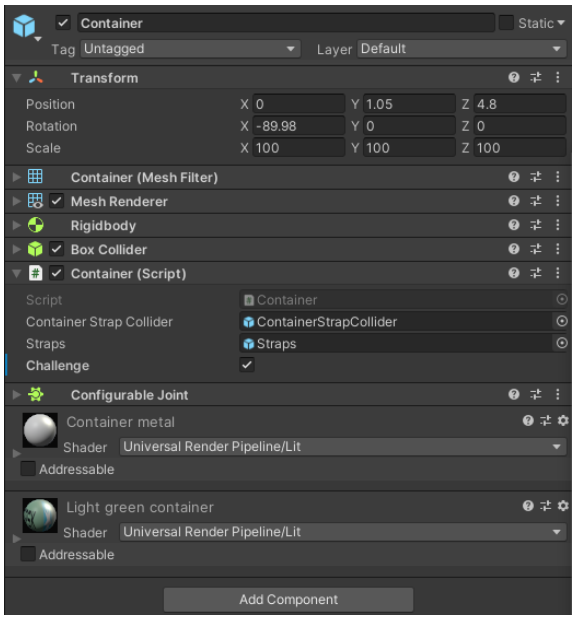


Figure 4.10: A Unity GameObject with many added components.

A collection of GameObjects with components in a hierarchy can be saved to a "Prefab" representing a finished thing in the game world. A prefab can be used in order to easily create instances of said thing. Examples of prefabs could be a crane, a video platform for showing 360 videos or a representation of the player's hand. Turning a finished thing into a Prefab makes it easy to manage multiple copies of it, as a change to the prefab is reflected in every instance of the object. There is no added effort to supporting multiple instances of the item, and there is little risk of forgetting to apply the change to one of the instances. A prefab could even be maintained outside of any specific application, allowing a single prefab to be used in multiple Immersive Job Taste applications.

4.6 VR frameworks

A VR framework is an abstraction layer between the game and the VR hardware it uses. It allows developers to focus on creating the game logic and player interactions without spending time and resources supporting specific hardware devices. The game uses abstract representations for VR devices like HMDs and controllers, and the framework takes responsibility for communication between the game and the physical devices. Multiple frameworks exist, and choosing the correct one is important, as the choice of VR framework defines the development process and which devices the game will support.

VR frameworks consist of many complex layers of software and the specifics vary from framework to framework. For simplicity's sake, one could divide a framework into a front-end and a back-end. The front-end defines the development process by creating the abstractions the application works with and various shortcuts surrounding them. Common functionality in the front-end would include animated hands and controller models, the ability to grab objects and move across the game world using teleportation or smooth locomotion. The back-end is mostly invisible to the developer and defines which devices the application is compatible with and which software it requires in order to run.

In the past, VR frameworks were single, monolithic entities. The front-end was mostly inseparable from the back-end. The choice of framework came down to which ecosystem of devices the back-end could support, and the front-end came with. This can be seen in the legacy Oculus and SteamVR plugins for Unity, which included both front-end and back-end with a single installation (Valve 2021; Oculus 2021c). It is also reflected in past literature describing VR frameworks as singular entities to be chosen between (Henrichsen 2019).

Recently, the focus in VR frameworks has shifted to modularity. Unity stopped supporting the monolithic legacy frameworks and instead focuses on a plug-in-based system (Unity 2021g). The idea is separating the front-end to the back-end, allowing the developer to target a front-end regardless of which hardware platform they wish to target and easily install back-end plug-ins based on which back-end they wish to support. A large number of companies have also focused on creating a back-end that is compatible with a multitude of platforms named OpenXR (Khronos 2021). The shift in strategy is likely due to the growing amount of VR hardware platforms and the VR platforms becoming more mature in general. Targeting specific devices and platforms becomes less viable as more options become available.

As mentioned in (Henrichsen 2019), the Virtual Internship project has yet to choose which framework will be used for future projects. Heisekran VR will be ported to Unity XR and the XR Interaction Toolkit to explore the potential of using them for future Immersive Job Tastes.

4.6.1 Unity XR & XR Interaction Toolkit

Unity XR is a first-party framework built into the Unity game engine, designed to replace the legacy Oculus and SteamVR frameworks. Unlike the legacy frameworks, Unity XR provides little functionality of its own. Instead, functionality and device support is handled through plugins. Device manufacturers are expected to provide back-end plugins to support their devices, and any number of them can be installed depending on which devices the application needs to support. The developer then chooses which front-end to install. The Unity XR Interaction Toolkit is a first-party package still in development that provides interactions and locomotion. Due to being in early development, there are few existing solutions and documentation regarding the use of the package. Since it is still marked as a preview package, there is also a risk that changes could be made to the front-end that break existing solutions. Being a first-party front-end for a first-party framework gives it good potential for future development with low complexity and widest possible hardware support. In the worst case, the front-end could be replaced with another one like VRTK.

4.6.2 VRTK

VRTK is a third-party front-end for the Unity game engine designed as a wrapper around other VR frameworks. Developers use VRTK to add interaction and movement, and VRTK translates it to a supported backend like SteamVR or Oculus VR in the background, making it compatible with multiple frameworks without a significant addition to the developer workload. VRTK also comes with the widest variety of existing solutions for interaction and locomotion. The Windmill VR application used the VRTK wrapper, which is further described in (Henrichsen 2019, ch. 9.6).

Henrichsen mentions that VRTK significantly simplified the development process due to example code, bundled software solutions and community support, as they were not an experienced VR developer. They also mention the downsides of VRTK: increased complexity by relying on a third-party library and the risk of backend framework changes breaking VRTK. VRTK is slated to use Unity XR in the future instead of the other backends to eliminate the risk of backend changes breaking existing application. As many developers of Immersive Job Tastes are new to developing for VR, it could be relevant for future applications once its transition to Unity XR is completed.

4.6.3 SteamVR

SteamVR is a game engine-agnostic framework designed to be compatible with the widest array of tethered HMDs. SteamVR only requires the installation of Steam, a launcher and storefront for games, and it is compatible with Windows, MacOS and Linux, covering the major computer operating systems. With approx. 1.7 million users and 4800+ VR-compatible games on Steam, it is currently the framework with most community and developer support (Lang 2020; *VR-compatible games on Steam* 2020). It also has a good choice of bundled solutions and samples, with highly detailed hand models and hand posing tools, a multitude of locomotion options and teleport graphics, and sample interactables to simplify implementation in your applications. For applications running on a computer, it is a natural choice for both quick initial prototyping due to the included solutions and future robustness due to support and compatibility. While no justification for the choice has been previously documented, it is likely why most existing Immersive Job Tastes were made on SteamVR.

The downside of SteamVR is its requirement of a desktop application running in the background. It is not compatible with popular standalone HMDs, so it is not a viable choice for development of future applications unless another version is simultaneously developed with another framework to cover standalone HMD support. The initial version of Heisekran VR was created using SteamVR.

SteamVR started out using the OpenVR back-end created by the same developer (Steam 2021b) but announced supporting OpenXR as a goal in June 2020 (Steam 2021c). In February 2021, SteamVR transitioned to support OpenXR by default, allowing any application targeting OpenXR to run on computers with SteamVR installed (Steam 2021a).

4.6.4 Oculus VR

Oculus VR is a proprietary framework supporting Oculus' own HMDs. It is game engine-agnostic and compatible with both the tethered and standalone HMDs made by Oculus. It has good documentation created by Oculus and decent community support with development tutorials and examples, but the front-end solutions available with the framework are poor. The framework provides different locomotion options including teleporting, but there is no built-in way to restrict where players can teleport. The provided interactables are also limited, with no good samples available.

A Heisekran VR prototype was created to explore the Oculus VR framework. With no restrictions to teleporting built-in to the framework, players would have been able to teleport onto any flat surface, landing on roofs and containers. Sufficient development time would compensate for the lack of existing teleport destination restrictions, but the product would be limited to Oculus devices. Needing to support an Oculus VR version on top of the original SteamVR version would further

add to the development time required. Exploring Oculus VR further was therefore deemed of little use to the Virtual Internship project, even if it could run on the Oculus Quest.

4.7 Commercial VR hardware

Commercial HMDs have reached a consensus on features the last five years. Commercial VR hardware tends to consist of one enclosed Head-Mounted Display (HMD) and two hand-held controllers. The HMD contains a high-resolution screen and two lenses, giving the player a window into the virtual world while obscuring the real one. They often contain speakers or headphones, and sometimes a set of cameras to show the outside world. The controllers tend to have a joystick or trackpad, two face buttons, one trigger and one grip button as a minimum. Some controllers implement more advanced finger tracking for use in gestures and more precise gripping of objects in virtual space.

The HMD and controllers can track their position and rotation in the space they are used. The industry calls this "6DOF", 6 Degrees of Freedom, as it tracks the HMD and controllers' position along three axes in addition to rotation around those three axes. This allows for precise enough head tracking to allow the player to move and look around like they would in the real world, without causing nausea (see Section 2.2.2). It also gives the player the agency to manipulate the environment using their hands in their own way and allows for the player's actions to embody their effects (see Section 2.3). The techniques used to perform the tracking vary between manufacturers but tend to involve optical recognition with the aid of dots or patterns of light. Older HMDs tend to require external sensors in addition to the HMD and controllers, while newer HMDs tend to be self-sufficient, with all necessary sensors built into the HMDs and controllers themselves. The latter tends to be called "inside-out" tracking in commercial contexts. HMDs without external sensors tend to be easier to use and configure, at a slight loss to tracking accuracy.

Tethered HMDs are easier to develop and distribute applications for. They have more powerful hardware available to them compared to standalone devices, so limitations regarding processing capabilities, memory and storage are less of a concern. Computers also use operating systems with significantly fewer restrictions on the applications that are allowed to run. A VR application targeting a tethered HMD can simply be started on the computer, and assuming the necessary software is installed and configured, it will work. Distributing the application is as simple as putting the application's files on the computer through a memory stick, online storage, a launcher (Section 4.4), etc. Developing and distributing an application for a standalone HMD is more complicated as they are akin to mobile devices like smartphones. They are more restrictive on the software they will allow to run. Installing an application requires it to either be published to the device's official app store, or the device to be put into developer mode before the application can be copied to the device using a computer with developer tools installed. The user also has to grant the application permission to use features like device storage.

The advantage of standalone HMDs is that they do not require any software installation or external devices in addition to the HMD and controllers themselves. Once an application is installed to an HMD, it should function without any problems. Computers and tethered HMDs are complex and require various software to be installed before they will work. Installing this software is more difficult than installing applications to the standalone HMD, even considering the case where developer mode has to be enabled on the standalone HMD. If there is an incompatibility between the software the tethered HMD requires and the computer it runs on, figuring out the cause and solution requires significant technical knowledge that few instructors will have. An example would be a brand of laptops that could not run tethered HMDs in the IMTEL labs unless a specific graphics driver version was installed (I-LOVE-CHICKEN 2021). Only online forums had any form of documentation regarding this error. Updates to the operating system and other software are significantly more likely to cause issues to tethered HMD solutions than more controlled standalone HMD solutions. The aforementioned laptops would overwrite the working graphics driver installation when newer versions became available, necessitating the old ones be installed again.

While initial setup of standalone HMDs should be simpler than tethered ones, the currently avail-

able devices come with their own set of issues due to the expectations that they be used as personal entertainment devices. The standalone devices used in this thesis, the first and second iteration of the Oculus Quest, require a personal Facebook account to be connected to the device in order to use them. Setting up the device requires a smartphone with the Oculus app and a Facebook account logged in to the app. The device will attempt to pair with the smartphone and connect to the Facebook servers, requiring both the headset and smartphone to be connected to the internet. The device is then irrevocably linked to the account until a factory reset is performed on it. Unless this step is performed correctly, the device will not run any applications. There is a risk this process can fail until the owner of the device can contact Oculus support. Even if the setup is successful, having a personal social media account connected to a multi-user device that the owner of the account might not be in control over raises privacy and security concerns for the owner of the account. This is especially true in cases where the owner of the account also owns a personal device, where purchases and sensitive details like payment methods might be inadvertently shared between both devices. Creating accounts for the devices that are not connected to a person is not an option, as it is against Facebook's Terms of Service and the accounts can become closed at a moment's notice, making the devices inoperable (Ehrhardt 2021). As of writing, IMTEL and NAV have been unable to find an alternative solution to this issue.

4.7.1 "Cardboard" Virtual Reality

Cheap HMDs and older prototypes are 3DOF and only track rotation, not movement. They are largely out of favour as they are not as comfortable as 6DOF devices and do not allow for interactive experiences like *Immersive Job Tastes*. Without tracking movement, they are liable to cause motion sickness from unintentional scene motion, when the player moves their head but the environment does not follow (see Section 2.1.2). They also tend to lack controllers entirely or use basic controllers without the motion tracking and sensors of the higher end devices.

Nevertheless, they have some use in non-interactive settings like 360-degree videos which make movement impossible even for 6DOF devices. Their cost is their main advantage, making some researchers consider them for school contexts where concerns of budget and equity are important. 3DOF devices are cheap to produce due to their limited functionality. More importantly, most modern smartphones can be put in a cheap head mount to temporarily turn the smartphone into an HMD. These mounts can be made of plastic or even cardboard, like the one shown in Figure 4.11.

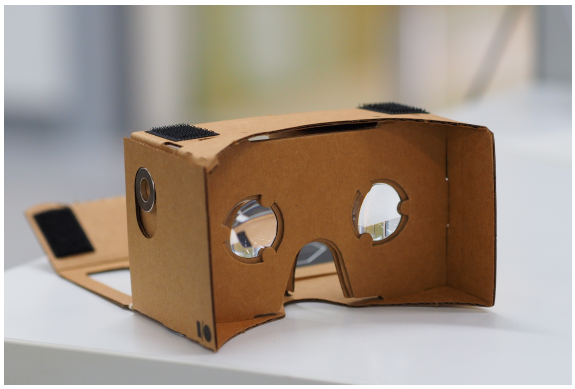


Figure 4.11: An HMD mount made out of cardboard, allowing a smartphone to be mounted inside to view VR content with some limitations. Taken from (othree 2020)

4.7.2 The HTC VIVE Cosmos

The HTC VIVE Cosmos is a tethered HMD based on the SteamVR platform that will be used for testing of the application before it runs on standalone hardware. It uses "inside-out" tracking, meaning it does not require any external sensors, only the HMD and controllers themselves. During use, the strips on the controllers light up and are used for tracking. The cameras on the HMD use

these lights to track the positions of the controllers. Due to this, high ambient lighting is required in the room for the tracking to perform well.

Setting up the HMD will require installation of Steam, SteamVR and Vive Software. When starting the HMD, it will make the user look around the play area until it has enough information to recognise its position in the room. If it is not the same room as last time it was set up, it will require the user to designate a new play area. This is done by first lowering the controller to the floor to set the floor height, then standing up and tracing the borders by pointing the controller around the play area. Once the play area is set up, one can download and launch applications and the computer will automatically start the HMD and necessary software. Steam, SteamVR and the Vive Software will update themselves automatically if the computer is connected to the internet.

It was acquired for the piloting of *Immersive Job Taste* applications by NAV before the decision was made to go with standalone devices instead. It is not very popular, being only 1.25% of HMDs in use amongst users of the Steam game platform (Steam 2020). The reason is likely due to prohibitive cost ($700 compared to $400 of equivalent devices from competitors) and poor tracking quality.

Figure 4.12: A HTC Vive Cosmos HMD and its two controllers. Taken from (Lang 2021).

4.7.3 The Oculus Quest 1 & 2

The Oculus Quest models are standalone HMDs that will be the focus of this thesis's development efforts. The first iteration of the Oculus Quest was the company's first standalone HMD with two controllers and tracking quality equivalent to inside-out tethered HMDs. The second iteration of the Oculus Quest is largely identical in functionality but at a lower price point.

Tracking is performed using cameras on the HMD and non-visible light, meaning they place less of a requirement on ambient lighting for the tracking quality.

When starting the HMD, it will try to recognise the room and play area, the same way as the HTC Vive Cosmos. However, the Oculus Quest is capable of remembering multiple rooms and play areas and will switch between them automatically. Room recognition also tends to be significantly faster. The play area is designated in the same way as the HTC Vive Cosmos, by first setting the floor level, then drawing the borders by pointing the controller at the edges of the play area.

Installing applications is much harder than tethered HMDs if they are not published to the Oculus Store. The HMD must first be set into developer mode to allow running of unpublished applications, and then rebooted. Afterwards, Android debugging software must be installed on a computer, and the HMD must be connected to the computer with a cable. A prompt asking whether the device should connect to the computer will pop up inside the HMD, which must be accepted. Afterwards, one can use the Android debugging software to transfer the application to the HMD. Once this is done, the HMD can be disconnected from the computer, and the application can be started from inside the HMD. Updating the app will require reconnecting the device and performing the Android debug steps again.

Figure 4.13: An Oculus Quest HMD and its two controllers. Taken from (Tirman 2021).

Figure 4.14: An Oculus Quest 2 HMD and its two controllers. Taken from (Daniel 2021).

4.7.4 The HP Reverb G1 & G2

The HP Reverb models are tethered HMDs that run on the Windows Mixed Reality platform. The first model is officially named just the "HP Reverb", but has recently been called the HP Reverb G1 in unofficial settings to differentiate it from its successor, the HP Reverb G2. They use inside-out tracking to track the player's head and controllers. The controllers themselves have a strip of LEDs around a ring emitting visible white light, placing some requirements on the ambient light level in the room.

The first HP Reverb came with a set of unique controllers without the common ABXY face buttons, as seen in Figure 4.15a. Instead, it had both a touchpad and joystick coupled with a single menu button per controller. This controller layout is unique amongst commercial VR controllers, meaning it tends to require explicit support to work for VR applications. The second model of HMD foregoes the touchpad on its bundled controllers, and instead uses the same common button layout as most other VR controllers. Nevertheless, the G1's controllers still cause occasional issues in applications that were created before the G2's release and assume its controllers still have the touchpad. The HP Reverb G2's controllers can be seen in Figure 4.15b.

The HP Reverb G1 is the tethered HMD used by the NAV personnel in Trondheim. Support for that device is still recommended for newer Immersive Job Taste applications, so that all Immersive Job Tastes are playable by the same device. The HP Reverb G2 was acquired by the author in order to have a device available that runs on the WMR platform and ensure the Heisekran VR application's compatibility with more HMD platforms than just the Oculus standalone devices.

(a) Old WMR controller (b) New WMR controllers

Figure 4.15: The design of old and new controllers for the Windows Mixed Reality platform. Old controller taken from (Bonasio 2021). New controllers taken from (Pertzborn 2021)

Chapter 5

Iteration 1

This chapter starts by defining the requirements that should be met after development of all iterations. It then focuses on the changes and user testing performed on the first iteration.

The focus of this iteration is on performing the first user tests of the application. The application was mostly unchanged during this iteration, with only necessary stability and usability issues fixed. Results from user tests would discover flaws that were not found during development of the prototype.

User testing had never been performed on their final product. Only early prototypes had been tested with family members.

5.1 Initial Requirements

The main purpose of this thesis is to make the application compatible with Unity XR, standalone HMDs and desktop play. The desktop variant should allow the player to complete the same amount of workplace tasks as the immersive VR variants. The final application should be compatible with tethered HMDs, standalone HMDs and desktops within the same application, as having to maintain feature parity between different copies of the application for each platform is not viable long-term. The application must also be optimised to run on each platform. Stuttering and low framerates take desktop players out of immersion (Penelope Sweetser and Wyeth 2005) and have the potential of causing cybersickness for players in immersive VR (Jerald 2016).

The game should be as accessible as possible for young job seekers to avoid excluding anyone from the Virtual Internship program. As described in section 4.2, the application does not currently provide any goals or guidance to the player. As described in section 2.3, the application's controls should be intuitive and allow the player to feel in control. The application should provide clear goals with appropriate challenge and feedback, so that players feel rewarded when they complete them. The application should also guide the players to the goals avoid players being lost and unable to find where to go. Having goals and guidance within the application itself is also important to allow the application to be deployed to NAV offices around the country, where the career counsellors will have minimal knowledge of the application. Cybersickness is also a factor to consider (see subsection 2.1.2). Measures against cybersickness should be taken in order to minimise exclusion.

NAV provided a set of requirements for the original prototype of the application that were described in (Hesle et al. 2020). Most of the requirements were implemented: a drivable crane with the possibility of lifting containers in and out of a ship, a 360-degree video showing employees completing a work task from the profession, and an element of workplace Health & Safety. These elements should be evaluated for accuracy by professional crane drivers to ensure they are not misleading. Some requirements planned for the prototype were not implemented. NAV wished for Immersive Job Tastes to have a form of gamification with points or achievements as feedback to

the player. They also wished for Immersive Job Tastes to model multiple work tasks, which the prototype does not.

The initial requirements have been listed in Table 5.1 based on the above. Additional requirements may be identified as development progresses in the iterations, as described in section 3.2.

ID	Requirement
R1	The application should use Unity XR and the XR Interaction Toolkit for immersive VR
R2	The application should support the Oculus Quest standalone HMD
R3	All of the application's tasks and features should be playable in desktop mode without immersive VR
R4	Both tethered and standalone immersive VR and desktop play should be supported by the same application
R5	The players should feel in control of the application's control scheme
R6	The application should minimise the occurrence of cybersickness
R7	The application should provide clear goals to the player to complete
R8	The application should guide the player to the goals
R9	The application should provide the player with points or other feedback for completing their goals
R10	The application should run at a minimum of 60 frames per second in desktop mode
R11	The application should run at minimum the refresh rate of the VR HMDs in immersive VR mode
R12	The application should provide accurate information and tasks from the profession
R13	The application should implement another work task from the profession

Table 5.1: The initial requirements for the Heisekran VR application.

5.2 Changes

The first iteration of Heisekran VR was mostly unchanged from the prototype described in section 4.2. Some issues regarding stability and usability were fixed, described in Table 5.2.

ID	Change	Section
C1.1	Containers would occasionally be sent flying when the player performed a teleport.	–
C1.2	The crane's beam was prevented from lifting or lowering into a position that should be impossible.	–

Table 5.2: Changes made in this iteration. Changes that require further explanation will refer to a later section.

5.3 User testing: Students

The first user test was conducted 24th September 2020 on students after an introductory lecture by Prasolova-Førland about VR. A testing station was set up with a laptop and a tethered HTC Vive Cosmos in a corner of the classroom, and students were asked to volunteer. After a short introduction to wearing the Vive Cosmos and using its controllers, students were guided through the application.

Data was only gathered through observations during this test. Students were encouraged to speak their mind and think out loud. The exact number of participants was not recorded but estimated at around ten.

5.3.1 Observations

Table 5.3: Observations made in this user test. Changes that require further explanation will refer to a later section.

ID	Observation	Section
O1.1	Players would often teleport off the stairs onto the ground, making climbing to the top of the crane difficult.	5.3.2
O1.2	Players would get tangled in the cable when going up the stairs due to multiple counterclockwise turns	–
O1.3	Players would struggle to find the direction to look on 360-degree videos, as the focus point was not in front of them when they entered the platform	–
O1.4	Players would occasionally be unable to teleport in certain directions	–
O1.5	The crane's control sticks would occasionally fly around and become unusable until the game is restarted	–
O1.6	Some users struggled to use the crane's control sticks, moving their hands far forwards or backwards without effect.	5.3.3
O1.7	The container would occasionally not attach to the hook, or fall off while being lifted	–
O1.8	Many users struggled to see the screen in the cabin, reporting it was too small or too dark.	–
O1.9	One user reported feeling nausea from the 360-videos due to how short the player appears	–
O1.10	Most users needed an explanation on how to stop viewing the 360-videos.	–
O1.11	A handful of users reported experiencing a fear of heights but chose to continue.	–
O1.12	Inexperienced VR users would often snap-turn by accident. Only the experienced VR players would use the feature intentionally.	–

5.3.2 Teleporting off the stairs

Players had no issue understanding how to teleport around the harbour, but teleporting up to the crane was difficult for players without experience in VR. The problem laid in the gaps between each step of the staircase, and the lack of railing at the edges. Experienced VR players would aim at the steps and only release the teleport stick when they made sure it was correct, getting up effortlessly. Inexperienced VR players would aim at the general direction of the staircase and flick the teleport stick without confirming their aim, sometimes hitting the ground through the gaps in the staircase. Additionally, inexperienced VR users were more likely to teleport twice or accidentally trigger snap-turning, often teleporting onto the ground off the edges of the staircase. One user ended up feeling nausea while attempting to climb the stairs, while another quit after falling off the stairs twice and getting frustrated.

5.3.3 Unintuitive control sticks

Players had issues steering the crane due to unintuitive controls as seen in Figure 5.1. The sticks were programmed to have the same rotation as the player's hands. Rotating their hand left would turn the stick left. The player rotating their hand forwards would turn the stick forwards, etc. The position of the hand grabbing the stick was not accounted for, meaning the player moving their hands around would have no effect. Roughly half of the testers would turn their hands as the game expected, while the other half would hold their hands straight and move them around and therefore achieving nothing. Some players would eventually figure it out after an explanation, but the rest struggled to control the crane effectively and got frustrated.

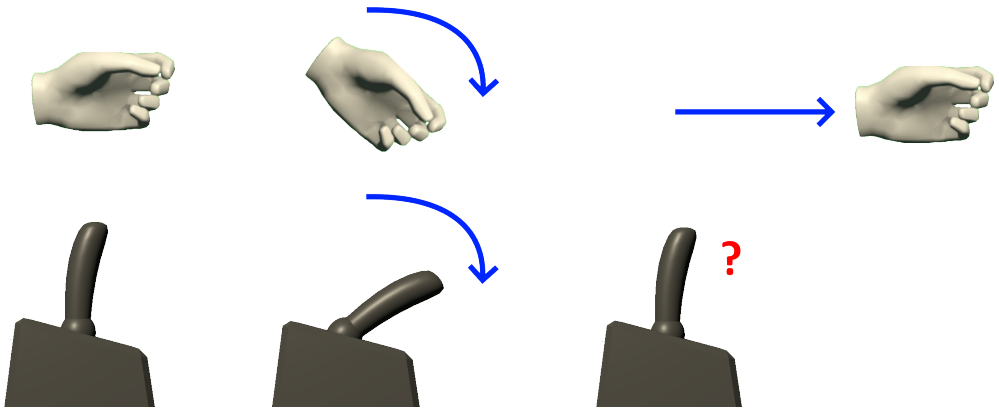


Figure 5.1: How to move the controller sticks of the crane. The stick has the same rotation as the hand holding it. Moving the hand has no effect.

Chapter 6

Iteration 2

This chapter describes the changes performed during development of the second iteration, as well as the results of user testing this iteration.

6.1 Changes

This iteration focuses on fixing the usability issues preventing many players from reaching the crane's control room. This allows user tests to provide more useful data regarding the crane driving itself. The changes made in this iteration are listed in Table 6.1. Further explanation is given in the following sections.

ID	Change	Section
C2.1	Invisible walls and flooring were added to the crane's stairway.	6.1.1
C2.2	Control scheme was changed to match the Immersive Job Taste tutorial.	6.1.2

Table 6.1: Changes made in this iteration. Changes that require further explanation will refer to a later section.

6.1.1 Invisible railings

subsection 5.3.2 showed that many players would struggle teleporting up the staircase due to the lack of barriers preventing them from accidentally hitting the ground instead. A good number of players quit because they got frustrated or convinced they were not going to succeed, preventing them from trying the crane driver portion of the game. Players will not engage with the game if they perceive the task is too difficult or if the control scheme is not good enough (Malone 1980), and it should be assumed the players have little previous experience in VR (Johnson-Glenberg 2018). There is nothing about the profession that makes climbing the crane difficult, so it should be simplified to be attainable for new players.

Invisible flooring was added to the staircases and set as a teleportable surface, allowing players to navigate up the stairs without requiring precision control. Invisible walls were added around the staircases to prevent users from falling off by accident. This avoided the issue spotted in observation O1.1. The barriers are highlighted in brown in Figure 6.1 but are normally invisible to the player.

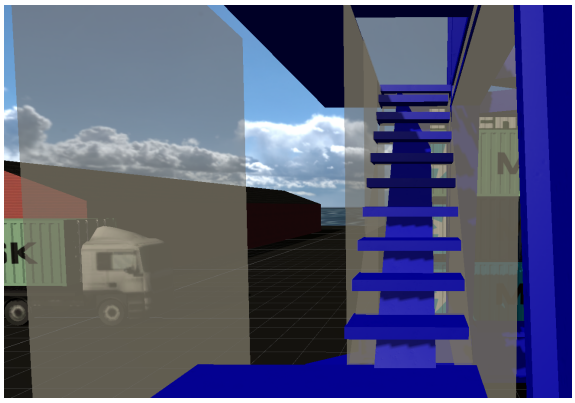


Figure 6.1: The barriers highlighted in brown surround the staircase to prevent the player from teleporting off.

6.1.2 Consistent controls

The Immersive Job Taste tutorial taught the players to teleport by clicking in the thumbstick and had no form of snap-turn. For consistency's sake, Heisekran VR's controls were changed to be identical. This control scheme has another advantage over the previous one, in that it is more difficult to perform any actions accidentally. Pushing the stick inwards gives off a distinct click, making it less likely for players to unintentionally teleport multiple times as observed in O1.1. Observation O1.12 showed that snap-turn was a feature that players without experience in VR struggled to use. They would push the stick to the side while attempting to teleport, turning them around and causing disorientation. As most players will not have the VR experience or understanding of the controls to use snap-turn intentionally, it is better off disabled.

6.2 User testing: NAV users

This user test was arranged by NAV on 30th September 2020, inviting several young job seekers to try out Immersive Job Tastes. Multiple IMTEL developers set up stands with tethered HMDs and showed off the Immersive Job Taste applications they were familiar with. Field notes were taken during playthroughs, and players were invited to fill out a questionnaire at the end of the test session.

6.2.1 Observations

The observations made during this user test can be seen in Table 6.2 Additionally, O1.3, O1.4, O1.5, O1.6, O1.8 were observed again.

ID	Observation	Section
O2.1	Players found it difficult and uncomfortable to push the sticks all the way in one direction.	6.2.2
O2.2	Players would attempt to go to the left video platform first, when the correct order was right-to-left.	–
O2.3	The tutorial video explaining the crane's controls was difficult to hear, requiring a supervisor to explain the controls.	–
O2.4	Containers get unhooked as observed in O1.7 when players try to hook the container from the side instead of bottom.	–

Table 6.2: Observations made in this user test. Changes that require further explanation will refer to a later section.

6.2.2 Uncomfortable control sticks

Players would find the rotation-based control sticks uncomfortable, on top of the issues identified in 5.3.3. The most intuitive way for the player to grip the control sticks would be to put their elbows down, as if they were resting on top of the virtual armrests on the chair, and then grab the control sticks. It is not a natural movement to rotate one's hand without moving their arm in that position. It becomes virtually impossible to push the stick all the way forwards or backwards, and any rotation of the hand causes a significant amount of strain on the wrist and forearm. One tester commented (translated) "This has to be wrong, surely it's not this uncomfortable for real crane drivers?". The most intuitive way to move a joystick is to move one's entire arm with only some amount of tilt on the hand, but the application would only register the slight tilt. This was enough for some players to steer the crane, while others became confused.

The player only needs to touch the stick when they grab hold of it. Afterwards, the player can move their hands as far away from the stick as they wish if they do not let go of the grab button on the side of the controller. This makes the most effective way to actuate the control stick to ignore the visuals of the virtual stick and the proximity of one's hand to the stick in the virtual world. Instead, the player should grab onto the stick and then use the entire length of their arm and their body to rotate their hand in a comfortable fashion. This is not intuitive for a first-time player, and inaccurate to the controls of a crane in real life. Additionally, while the motions are large, they are not congruent with their results in the game, further hampering learning. Only the developers, who knew the details of how the sticks registered player movement, were able to consistently actuate them as far as possible.

6.2.3 Questionnaire

Two users that had tried the Heisekran VR application answered the questionnaire. Below are the questions related to the Heisekran VR app. Additionally, the questionnaire contained a freeform text field question asking for comments or suggestions for improvement. A short version of the responses, containing only questions related to the application, is shown in Table 6.3. The entire questionnaire will be summarised in chapter 11.

One responded commented in the text field, saying "It should have been easier to use the sticks [...] without them bouncing all around or being loose when you touch them".

ID	Question	Mean
Q1	I liked using the Crane app	4
Q2	The Crane app was easy to use	4.5

Table 6.3: The Heisekran VR-specific questions and results to the questionnaire given out during this iteration's user testing.

Chapter 7

Iteration 3

This chapter describes the changes made during development of the third iteration, as well as the results of this iteration's user testing.

7.1 Changes

This iteration focuses on making the application use Unity XR and the XR Interaction Toolkit instead of SteamVR. It also makes the application compatible with the Oculus Quest standalone HMD.

ID	Change	Section
C3.1	Video order was changed to left-to-right to fit user behaviour seen in O2.2.	–
C3.2	The top-down hook screen was enlarged and would no longer be darkened by shadows.	–
C3.3	Switch VR framework to Unity XR.	7.1.1
C3.4	Create hands for use with Unity XR.	7.1.2
C3.5	Designate teleport areas again for Unity XR	7.1.3
C3.6	Add other miscellaneous graphics for Unity XR.	–
C3.7	Unity project was changed to Universal Rendering Pipeline.	7.1.4
C3.8	Change joystick controls to be more intuitive.	7.1.5
C3.9	Make the application compatible with the Oculus Quest standalone HMD.	7.1.6

Table 7.1: Changes made in this iteration. Changes that require further explanation will refer to a later section.

7.1.1 Unity XR

Switching the VR framework to Unity XR required a lot of work due to the lack of existing solutions. Unity XR's interaction toolkit included no graphics of any kind, meaning the player's hands were invisible and there was no teleport marker. Teleport points, as used for 360-degree videos and the crane's chair, also had no graphics. Support for limiting teleportation to specific areas was included by default but was intended only for surfaces level with the ground, needing a modification to support the slopes on the staircases. On top of that, the interaction toolkit only included an abstract interaction framework to support developers in implementing interactions, but no concrete examples of interactable objects. The control sticks of the crane would need reimplementation from scratch.

7.1.2 Giving the player hands

The main challenge was adding hands to the player. No existing solutions for the Unity XR framework were available. While Oculus VR and SteamVR provided their own hands and hand implementations, they were bound to their own frameworks and would not work correctly with Unity XR. Creating a new set of hands would not be a problem for an experienced modeller and animator, but this project had no one available who fit the bill. The easiest solution seemed to be converting one of the other frameworks' hands to work with Unity XR.

Hands in Unity consist of three components in a hierarchy: the hand model itself and associated animations, an animation controller that plays the hand's animations and blends multiple animations together, and a hand controller that tells the animation controller which animations to play. The first attempt focused on SteamVR's hands, as they were more detailed than the Oculus VR hands and the crane's control sticks already had hand poses for SteamVR. It was quickly found out that Steam VR's hand controllers were too tightly knit to their own framework, so we would have to write our own. Looking into Steam VR's animation controller, it was found to be too complex, and attempts to write a hand controller for it failed. Turning our sights to the Oculus VR hands, we found that OVR's hand controller was simple and easy to modify. Changing the Oculus VR hand controller to read hand positioning and gestures through Unity XR was quick, allowing us to reuse their animation controller and hand models. The project was left with working hands with the same look as the Oculus VR framework.

The basic animations required to animate grabbing was implemented in these hand visuals. There were further animations available that allowed for gestures like pointing and finger guns, but they were left out of this iteration as these animations would only work for select Oculus controllers. There was also a "pose" system available in the Oculus VR framework that would force the hands into assuming a certain gesture, designed for when the user holds objects with esoteric shapes. As the Heisekran VR application only required the user to hold joysticks, this was also left out of this iteration for time.

7.1.3 Teleportation areas

Once the hand models were added, the areas the player was allowed to teleport to had to be designated again. In SteamVR, you would mark an area as a valid teleportation target by adding a TeleportationDestination component to the object. A similar component would make the player teleport to a specific point instead of wherever they aimed and was used to put the player in the centre of the chair.

Unity XR had analogous components, so porting the teleportation areas to Unity XR was theoretically as simple as replacing every SteamVR component with the equivalent one from Unity XR. In practice, the Unity XR teleportation destinations were more simplistic than SteamVR's. SteamVR ensured the player was standing upright after teleportation with the world as a reference point, while Unity XR would put the player normal to whatever surface the player aimed at. This caused no issues when teleporting around the harbour, but attempting to travel up the invisible slope representing the stairs would cause the player to get tilted sideways, causing huge disorientation and motion sickness.

Luckily, modifying the components was an easy task, as they were designed to be extended. Their simplicity compared to SteamVR turned out to be an advantage for development. After altering the components to turn the player upwards compared to the world, the teleportation worked as intended.

7.1.4 Universal Rendering Pipeline

The project needed a teleport destination marker for Unity XR. Initial attempts were made to use the teleport markers from SteamVR, but it was later found that the Blikkenslager Immersive Job Taste had one already that could be used for Unity XR. Attempting to use this teleport marker

led to the discovery that it was incompatible with the render pipeline of Heisekran VR. This raised further questions as to which render pipeline the different applications used. After investigating the current Immersive Job Tastes, it was found that no consensus was reached as to which render pipeline to use. Some used the standard pipeline like Heisekran VR, while others like Blikkenslager used the newer Universal Rendering Pipeline (URP).

A rendering pipeline is responsible for converting the objects in the game to a visual representation that can be shown on a screen. It decides how materials, models, lights and other effects interact with each other to produce a two-dimensional image and has to do it in an efficient way in order to produce images fast enough that we perceive it as smooth motion. The standard rendering pipeline in Unity is older and not particularly extensible (Lira 2021). The new Universal Rendering Pipeline is designed to replace the standard pipeline in the future, and has better performance and support for a variety of platforms, including VR and AR. It is also more extensible, allowing artists to create assets easier than with the standard pipeline. The performance improvements and platform support are appealing to the Virtual Internship project focus on VR and AR experiences. Due to the relative simplicity of applications in the Virtual Internship project, it is likely that assets that take advantage of the extensibility of the pipeline will be acquired from the asset store rather than developed specifically for the applications.

The current issue with URP is community support. As of writing, none of the VR frameworks support URP out of the box. SteamVR, Oculus VR and VRTK all come with assets that are designed for the standard rendering pipeline. In fact, checking the development log of Heisekran VR, it was found that the initial prototype attempted to use URP at one point but reverted to the standard pipeline "due to steamvr shader issues". Unity XR does not come with any assets, avoiding this issue. The XR Interaction Toolkit performs some shading in situations like hovering your hand over an interactable item, which had to be slightly altered to work in URP. A cursory search of the Unity asset store for VR and XR assets also finds that most assets available only support the standard rendering pipeline and would have to be converted in order to display correctly in URP.

Heisekran VR was converted to URP during this iteration, partly to use the assets from Blikkenslager, partly to investigate the effort required to convert a project's assets from one rendering pipeline to another. The incompatibility between the pipelines was found to be the shaders selected for materials. Shaders from one rendering pipeline are not compatible with the other, causing objects to be rendered with a pink colour instead of their correct look. At first, the conversion process seemed daunting, as one would have to select every material in the world and manually change the material to a matching one. Further investigation found an option included in the URP package to automatically change from standard shaders to URP ones based on a reasonable mapping, which trivialises the conversion from standard pipeline to URP for any applications that do not have custom shaders (Unity 2021f). Custom shaders are only necessary if the included shaders cannot provide a certain type of visual effect for an application, and the creation of custom shaders requires some knowledge. The author finds it unlikely that an Immersive Job Taste will have issues converting assets to URP unless they acquire assets from the asset store that happen to use a custom shader. In such a case, the developers should request that the creator of the asset provide an URP variant.

7.1.5 Better joysticks

The first joystick implementation was based on a SteamVR "throwables" script, designed to provide objects players could throw around the world. The sticks were throwables glued to the seat, rotating to match the rotation of the player's hands. Unity XR had no equivalent component, so the control sticks would have to be reimplemented. Reworking SteamVR's "throwables" script to function with Unity XR instead was considered, but ultimately shelved, due to the existing issues with that implementation observed in O1.6 and O2.1. Instead, it would have to be implemented from scratch.

Reimplementation started by investigating existing solutions in SteamVR's interactable examples. Even if they could not be used for Unity XR, the principles they work on could be used for our own

implementation. The result yielded no concrete implementation, as SteamVR's examples only had implementations for joysticks along one axis and wheels, but the investigation was not fruitless: it led to a debugging script that would significantly simplify development. Developing interactables used to be difficult as there was no easy way to see how interactables reacted to your hands while testing. One would have to put on the HMD, interact with the joystick, then take the HMD off and hope to match the text logs to the actions you attempted to perform. SteamVR's example interactables had a debugging option that put tiny spheres where your hands were and where the output of the interactable was, making it easy to connect hand movement to stick movement during development and try new ways of interacting. Using this script for developing Unity XR interactables proved to simplify development by getting immediate feedback about the correctness of the calculations performed. Such a solution might seem obvious to developers with experience in Unity development, but it might nevertheless prove helpful to any less experienced developers that might work on future Immersive Job Tastes.

The control scheme of the sticks was developed by splitting the task into multiple problems and solving one at a time, until the debug circles appeared where expected. First, the sticks were made to point towards the hand when they were held. The debug spheres would appear at the centre of the hand's grip, and the stick's rotation would follow. Then, a position a set magnitude away from the base of the joystick was calculated, signified by the red spheres in Figure 7.1. This position was projected down to the surface plane intersecting the base of the joystick, signified by the green spheres in Figure 7.1. The x and y magnitude was taken from this projection was used as the output to the crane. This control scheme still allows controlling the crane by rotating your hands, but it also allows moving your hands to move the stick, as shown in Figure 7.2.

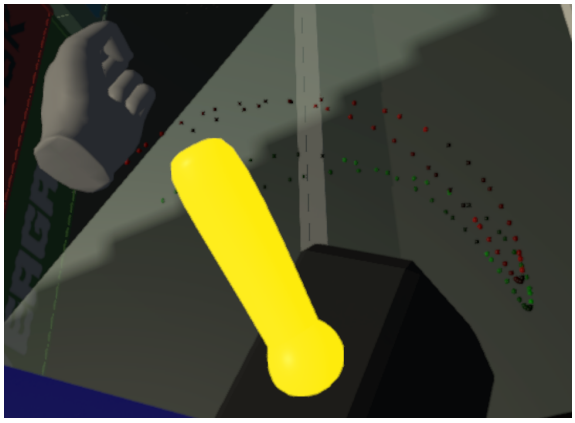


Figure 7.1: The debug spheres of a joystick during its development. The red spheres appear at the top of the joystick, while the green spheres are the result of the projection onto the plane.

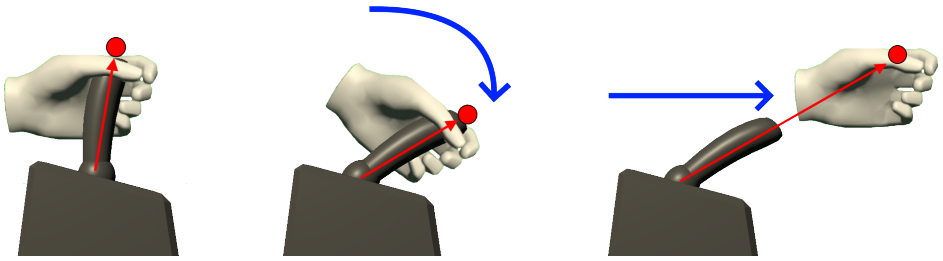


Figure 7.2: How to move the controller sticks of the crane in this iteration. The stick follows the hand when rotating. Moving the hand also works.

7.1.6 Compatibility with standalone devices

With the above changes, the application ran on Unity XR on a tethered HMD. Further changes were required to make it run on a standalone HMD. Oculus has instructions for configuring Unity to

run applications for the Quest (Oculus 2021a). This includes configuring Unity to target Android, installing standalone Oculus support for Unity XR, and configuring other settings to make the application look and feel proper on an Oculus Quest device. Following these instructions got the application close, but some issues remained, as standalone HMDs are more restrictive about the applications they will run.

The general development process for standalone HMDs requires building an APK (Android Application Package) and copying it to the device. These files have a restriction on total size, with Oculus limiting the file to 1GB for their storefront and Google limiting it to 100MB for Google Play (Oculus 2020c; Google 2020). Video files are significantly larger than this, with Heisekran VR's 360-degree videos taking up greater than 2GB of space in this iteration. Attempting to load an application file this large on the Oculus Quest resulted in a crash upon start-up. The videos would have to be split apart from the application file itself and loaded onto the device separately, and luckily, Oculus describes the process on their website (Oculus 2020a). Their recommended Unity technology for separate assets is AssetBundles, but that technology was deprecated in favour of Addressables since Unity version 2018.2 (Unity 2020a).

Addressables allow developers to tag assets and game objects in Unity as "Addressable", and the tagged items would get separated into a separate asset file instead of being part of the application when built. By default, this asset file would be in the same directory as the application. This works for the desktop application as computers do not place restrictions on the asset file's location. The standalone HMD requires the application to be installed to the application directory, and the assets to be in a separate asset directory. After some attempts, it was found that creating an Addressables profile with an asset loading path of `"/sdcard/Android/obb/com.IMTEL.HeisekranVR"` and copying the asset file to the same folder on the device would allow the standalone application to run. To build the standalone or desktop variants, the developer would just have to select the right Addressables profile beforehand.

Splitting the video files onto its own path needed some extra logic on the video platforms. Separate video files will not be loaded by default and need to be explicitly loaded by a script somewhere. At first an attempt was made to load the video when the player went onto the platform, but the loading times turned out to be exceedingly long. Videos would only play after half a minute, with no feedback to the player. A loading screen could be devised to provide feedback, but the wait would still be frustratingly long. Instead, all video files were loaded immediately upon the start-up of the application. The loading time ended up being longer as all the files are being loaded simultaneously, but as the user will start the application figuring out how to teleport, it is unlikely they will reach the platform before it is finished. There was a risk of the videos exceeding the device's memory, but the concerns turned out to be unfounded, with the application running fine. The loading time is still a few seconds long, longer than on tethered HMDs, so the reactions to the load times will have to be gauged during future user tests.

Some further care has to be taken when attempting to play video files on Android, the operating system used by standalone HMDs. As detailed by Unity on their VideoPlayer page, "Playback from asset bundles is only supported for uncompressed bundles, read directly from disk" (Unity 2020d). This limitation is deceptive, as there are multiple stages where Unity might apply compression. First, the file itself must obviously be uncompressed. Second, compression must be disabled for the Addressables group containing the video, which is hidden away under the group's "Advanced Options". Finally, Unity compresses assets by default when loading them from asset files, which would also break playback of the video files. This must be disabled before loading the video by setting `Caching.compressionEnabled` to `false` in a script but can be reenabled after the video has finished loading.

7.2 User testing: Students

This user test was performed the 5th November 2020. Students volunteered to visit the IMTEL VR Lab to experience VR technology and IMTEL's new applications. The application was tested running on an Oculus Quest HMD. Five students participated in testing.

7.2.1 Observations

The observations written down in the field notes are listed in Table 7.2.

Players gave positive feedback, with one player mentioning they felt like a crane driver and had a lot of fun. "It feels productive to move such a big object around".

More players reacted to a slight fear of heights but continued up and did not have an issue once in the crane driver's seat.

ID	Observation	Section
O3.1	There is no way to see what the user sees in standalone HMDs by default, making observation and guidance difficult.	–
O3.2	The lack of a wire makes the constant counterclockwise movement of the staircase smoother.	–
O3.3	The spiral staircase caused a user to flinch on the way up. It is easier without a wire, but is likely still too steep and claustrophobic.	–
O3.4	One player reported falling out of the cabin and onto the ground before going back up.	–

Table 7.2: Observations made in this user test. Observations that require further explanation will refer to a later section.

Chapter 8

Iteration 4

This chapter describes the changes made through the fourth iteration, and the results of multiple user tests.

8.1 Changes

This iteration was focused on getting the application running in desktop mode. Table 8.1 gives an overview of the changes required to make the application playable on desktop. The following sections describe the changes required in detail.

ID	Change	Section
C4.1	Make a desktop player object capable of moving across the game world	8.1.1
C4.2	Make the desktop player able to sit down in the crane seat and control the crane	8.1.2
C4.3	Make 360-degree videos viewable in desktop mode	8.1.3
C4.4	Turn the player to face the focal point of 360-degree videos in both desktop and immersive VR	8.1.3
C4.5	Allow the developer to swap between immersive VR and desktop mode through the Unity editor	8.1.4

Table 8.1: Changes made in this iteration. Changes that require further explanation will refer to a later section.

8.1.1 Desktop Movement

Character movement was simple to implement. Movement in a 3D game environment is a solved issue with endless tutorials and solutions available. Unity has a Character Controller class with a robust implementation of all the important functionality of a character in a 3D environment (Unity 2020b). Using that component allows the character to move in any direction while colliding with obstacles and walls. The component ensures the character does not phase through solid objects or flooring. The developer only needs to create a component that takes input from the player and signals the Character Controller class when the character wishes to move. That component defines the game's control scheme, meaning which buttons the player must press in order to make the character move. Additionally, the developer still needs to give the player a way to change where they are looking, on top of keyboard/mouse buttons for functionality unrelated to character movement. A cursory search found a "FPS Character Controller" script of decent quality and expandability, which was used and expanded for Heisekran VR (Coder 2020).

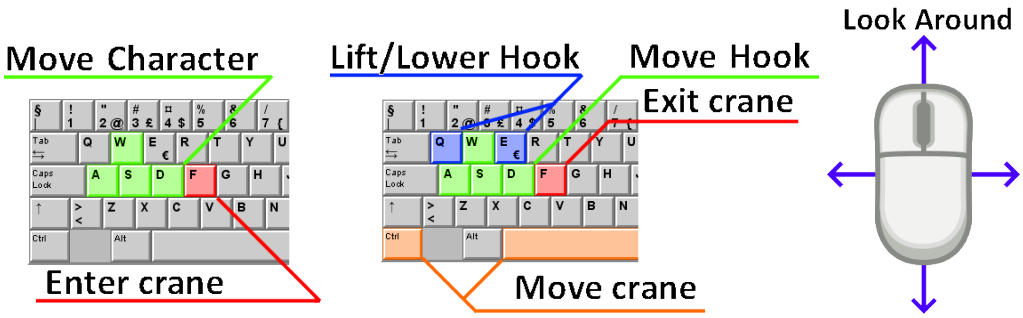
Certain precautions had to be taken to allow freeform movement. Previously, the player was restricted to teleporting to areas designated by the developer. Allowing the player to move freely would allow them to get into unintended areas and struggle to figure out what to do. Invisible railings had to be placed around the harbour to prevent the player from falling into the ocean. The flooring and teleport destinations of the crane had to be redone, as the player in desktop mode would bump their head into the teleport destination covering the upper floors and be prevented from moving further. These changes are only the obvious issues that arose during initial testing, and user testing might discover further areas that need improvement.

Using that FPS Character Controller class, the player was allowed to move across the harbour in a continuous motion using a keyboard & mouse (or trackpad) as illustrated in Figure 8.1. Pressing the WASD keys moved the player based on the direction they were looking without turning their viewpoint, which is standard first-person 3D game convention. For those unfamiliar with 3D video games, W moves the player forwards, A moves the player leftwards, S moves the player backwards, and D moves the player rightwards. Moving the mouse changed the direction the player's viewpoint is looking. The player could freely choose where to look by moving the mouse upwards and downwards and turn around by moving the mouse left and right. No extra effort was required in order to climb the crane's stairs. The player character automatically climbs up or down the steps of the staircase when moving towards them.

8.1.2 Desktop Crane Controls

After movement was implemented, the player needed a way to control the crane. A common archetype for 2D/3D games is to have an "activate"/"use" button for interactions with other objects. When the player is looking at or near an interactable object, pushing the button starts the interaction. A prototype of such a button was implemented into the FPS Character controller. A robust interaction button should be generic and event-based, sending an activation event to the object the player is interacting with and letting the object handle the event. This would allow any number of interactions to be defined without a lot of effort on the developer's side. In this iteration, the Character Controller was hardcoded to interact with the crane seat only and had no easy way of extending it.

The player could "activate" the crane by looking at the seat in the crane's control room and pressing the activation key (F) on their keyboard. This would play an animation which moves the player into the seat. Once the player was seated, they were no longer allowed to move around. Instead, their keyboard was set to control the crane with the control scheme seen in Figure 8.1. Their movement keys would control the crane's hook by moving it along the ground, like the left joystick in VR mode. W would lower the beam and move the hook further away, while S would lift the beam, moving the hook closer. A and D would rotate the crane in the respective direction. Four additional keys were enabled which allow the player to lift and lower the hook, as well as move the crane along the rails. The Q and E keys lower and lift the hook, respectively. Spacebar and Ctrl allow the crane to drive along the rails, with Spacebar moving the crane towards the front of the ship and Ctrl moving the crane towards the back. The crane functions the same as in the VR variant, allowing the player to pick up containers by lifting the centre of the rope with the hook. Once the player is done driving the crane, they can press the activation key in order to leave the crane seat and regain control of their character. Once out of the seat, the normal control scheme is reinstated.



(a) Normal controls (b) While driving the crane (c) Always active

Figure 8.1: How to control the player character and crane in desktop mode.

8.1.3 360-degree videos for desktop

The 360-degree video platforms were originally designed for the VR character, so the desktop character would not be able to watch the videos. Allowing the FPS Character Controller to trigger the platforms was a simple task, but the smaller FoV of the desktop screen made it even harder to find the right section of the video to focus on. That led to modification of the 360-degree video platforms to allow the developer to choose a "forward" direction, turning the player in that direction on entry. To avoid the player getting disoriented, and in the worst case running right into another platform when exiting the video, the player was teleported to the front of the platform when they were done watching the video. This change benefited both the immersive VR and desktop modes of the application, fixing the issue observed in O1.3 where players would struggle to figure out where to look on 360-degree videos.

8.1.4 Supporting desktop and VR simultaneously

As mentioned in previous sections, the application needs to support both desktop- and VR mode with minimal effort to change between them. Otherwise, the development and maintenance of both modes will become significantly higher. A basic "Player Manager" was implemented, with a checkbox in the Unity Editor deciding whether the desktop or VR mode is to be used. When desktop mode is enabled, the VR framework is disabled to save resources. This implementation works as a prototype, but ideally the selection between desktop and VR would be fully automated by detecting whether the player had an HMD available or not. A benefit from automatically toggling between the modes would be that a single desktop release could support both keyboard & mouse controls, as well as playing with a tethered HMD.

8.2 User testing: NAV Personnel

Two user tests were arranged by NAV at the IMTEL VR lab. Four career counsellors took part, including Heidi Fossen, the lead of the Virtual Internship project at NAV's side. One tester had previous experience as a crane driver. The first user test was performed the 17th November 2020, and the other the next day, 18th November 2020.

Users were invited to try the application in immersive VR mode using an Oculus Quest standalone HMD, and in desktop mode on a laptop. The view of the Oculus Quest was streamed to the author's phone during testing to allow for observations of the application itself.

8.2.1 Observations

ID	Observation	Section
O4.1	One player struggled to figure out the control sticks, moving their hands upwards or not sufficiently far in a direction to move the hook at a reasonable speed.	–
O4.2	Another player forgot to let go of the sticks when they were finished with them, moving the hook out of alignment while trying to lift or lifting/lowering it while trying to align.	–
O4.3	One player could not figure out where to go, as they were facing away from the ship after they took the headset on.	–
O4.4	It was difficult to see in the 360-degree videos showing the crane driver's perspective according to one user.	–
O4.5	Few people attempted to play the desktop version after trying the VR version.	–
O4.6	It was difficult for players without previous PC game experience to learn using WASD. They succeeded at finishing the game, but they did not take the controls intuitively and had to be reminded.	–

Table 8.2: Observations made in these user tests. Observations that require further explanation will refer to a later section.

The observations made during these user tests are listed in Table 8.2.

The testers gave valuable feedback at the end of these sessions. One tester suggested using haptic feedback in form of vibrations when touching the sticks, as otherwise it was difficult to tell if their hands were in contact. This change could potentially help with O4.1 and O4.2 which were observed during these sessions.

Heidi Fossen said that Heisekran VR's 360-degree videos were more interesting than other Immersive Job Tastes, perhaps due to the variety of locations they were filmed at or the insight they gave into the real work tasks performed at the workplace. Another tester mentioned that the video from the crane's cabin was interesting but found it difficult to see what the crane driver was doing. They wished to move the viewpoint closer to the crane driver's shoulder. This is impossible to fix for the current video but should be considered when recording future 360-degree videos.

The tester who previously worked as a crane driver found the controls intuitive and thought the application was fun, lifting a few extra containers and driving the crane around. No explanation was needed for the crane's control sticks for them to start lifting containers. They suggested playing an alarm sound when the crane was moving along the tracks, as the application was missing the sound. Other users mentioned the lack of sound while controlling the crane as a negative.

Crane containers unhooking themselves during lifting was observed once again (O1.7). One tester, even being a career counsellor and not a job seeker, said "I know what career I'm not going to be" after dropping a container onto the harbour, and had to be encouraged to try again.

8.3 User testing: NAV users

NAV arranged a user test the 2nd December 2020, inviting young job seekers to try out Immersive Job Tastes. The focus was on new Immersive Job Tastes created by IMTEL this year, but older job tastes were also displayed. Multiple stations for user testing were set up, one with a standalone HMD, one with a tethered HMD, and two with laptops running application in desktop mode. At the end of the testing session, users were invited to fill out a questionnaire about their experience. Six participants tried the Heisekran application in immersive VR mode, and roughly the same the desktop version.

8.3.1 Observations

The observations made during this user test are listed in Table 8.3.

ID	Observation	Section
O4.1	All testers with no previous VR experience were impressed with how real the application seemed. They all expressed reactions of wonder and surprise.	–
O4.2	The above comes with a downside, as one tester quit after the 360-degree video from the crane's point of view due to fear of heights.	–
O4.3	Some users noticed the screen without guidance. The rest seem fixated looking downwards and to the left (at the left joystick?) and do not notice the screen until it is pointed out.	–
O4.4	Some users were engrossed by the 360-videos, watching them intently and multiple times.	–
O4.5	The users who had tried the desktop mode tend to skip the videos in VR mode and go immediately up the crane.	–

Table 8.3: Observations made in this user test. Observations that require further explanation will refer to a later section.

8.3.2 Questionnaire

The questionnaire used for this testing session was identical to the previous one apart from the additional Immersive Job Taste applications added to the applications section. Heisekran in immersive VR mode and Heisekran in desktop mode had separate questions in the applications section of the questionnaire. Each mode had three answers each. Table 8.4 shows the questions related to the Heisekran application itself. The entire questionnaire will be summarised in chapter 11, Results.

One respondent suggested a "checkpoint" system with clear tasks would help with the feeling of mastery and learning what the profession is about. It is unclear whether they meant a checkpoint system similar to games where you can go back in time to certain states of the world, i.e. before dropping a container onto the ground, or if it simply means a checklist of objectives similar to other immersive job tastes.

ID	Question	Mean
Q1	I liked using the Crane app with VR glasses	3.66
Q2	The Crane app was easy to use with VR glasses	4.33
Q3	I liked using the Crane app on desktop (without VR glasses)	4
Q4	The Crane app was easy to use on desktop (without VR glasses)	4.33

Table 8.4: The Heisekran VR-specific questions and results to the questionnaire given out during this iteration's user testing.

Chapter 9

Iteration 5

This chapter describes the changes made during development in the fifth iteration and presents the result of the user testing.

9.1 Changes

This iteration focused on improving the usability of the application. Long-standing issues with crane driving were fixed. The video prefabs were made consistent with the other Immersive Job Tastes. The application was also prepared to be distributed through NAV's catalogue. The overview of the changes can be seen in Table 9.1, with additional explanation in their respective sections.

ID	Change	Section
C5.1	Use the same 360-degree video prefab as other Immersive Job Tastes.	9.1.1
C5.2	Make the control room screen view rotate intuitively alongside the crane.	9.1.2
C5.3	Make the hook move the same distance regardless of position and framerate.	9.1.3
C5.4	Only hook onto the container when lifting the hook.	9.1.4
C5.5	Reduce the size of the 360-degree video files included in the application	9.1.5
C5.6	Make the application compatible with distribution through Yrkeskatalog.	9.1.5

Table 9.1: Changes made in this iteration. Changes that require further explanation will refer to a later section.

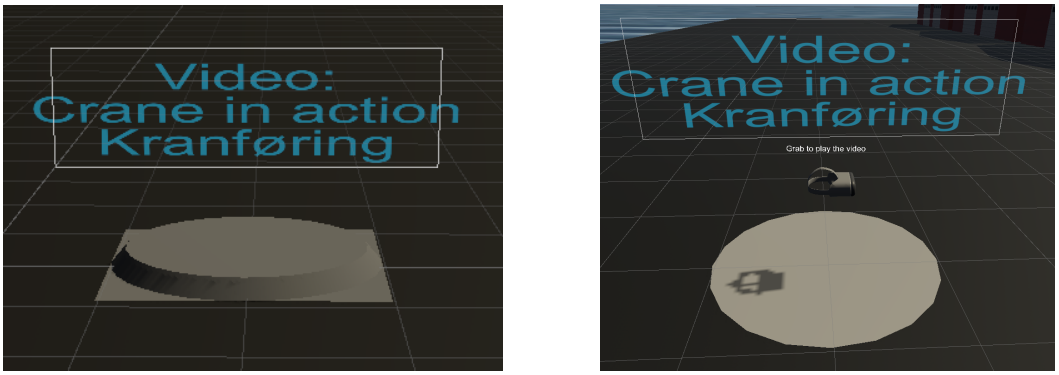
9.1.1 360-degree video prefab

Heisekran VR added support for 360-degree videos before a 360-degree video prefab had been finalised for Immersive Job Tastes. The 360-degree video platforms in the application had a different design and functionality from the final 360-degree video prefab which was in use by the Immersive Job Taste tutorial and other applications. The older prefab required the user to teleport onto the platform to play the video and teleport off to stop playback. The new one has a virtual HMD that plays the video as long as the device is held. The prefab has better affordance than the previous solution, as a floating VR HMD is more obviously an interactable object than a platform on the ground. The virtual HMD also has floating text above it explaining that the player must grab it to play a video. The HMD would also be present in the player's hand while watching the video, with the text changing to tell the player that letting go stops video playback. This should be an improvement compared to the previous prefab having no hints or explanation on how to start and stop viewing the video, often requiring guidance to stop viewing a video (O1.10). The prefab

would also be consistent with the other applications that are part of the catalogue. Especially important is being consistent with the Job Taste Tutorial described in section 4.4.

The new prefab had a drawback in that it was more simplistic than the 360-degree video platforms used in Heisekran VR previously. The improvements made to Heisekran VR's video platforms in subsection 8.1.3 were not included in the new prefab. The author considered adding the improvements to the prefab themselves, but the developer of the prefab requested no changes to be made to the prefab. The risk of updating the prefab is that it might break the prefab in other applications, especially if they have made some changes to it. There are many tools available for handling these kinds of changes but not all developers are experienced in their use. Developer guidelines for using Git to update existing prefabs in applications should be created in order to facilitate future updates of the prefab.

The new prefab also did not support viewing video in the desktop mode. In lieu of a standardised desktop player prefab for Immersive Job Tastes, it would not be possible for the prefab to add support for all applications. Instead, a copy of the 360-degree video prefab used in Heisekran VR was modified to function similarly to the old prefab, playing video as long as the player is stood on the platform. Once a desktop player prefab has been created, the 360-degree video prefab should be updated to support it for use in other applications, perhaps when the floating HMD is activated by the player.



(a) Old design (b) New design

Figure 9.1: The design of both video platforms side by side.

9.1.2 Screen rotation

Players were observed struggling with aligning the hook with the containers' ropes, even after they were aware of the top-down view through the screen. The issue was that the orientation of the view through the screen did not match up with the surroundings. As the crane rotated, the screen view would still face the same direction. Looking out the cabin and attempting to move the hook forward would intuitively move it directly away from the player. Looking at the screen, one would expect the hook to move upwards when moving directly away from the crane, but unless the crane was facing its original direction, it would not match up. Using Figure 9.3a as an example, one would expect the hook to move closer to the centre of the green container when looking at the screen, but the hook would move away from the player towards the red container. Figure 9.3a illustrates this disparity.

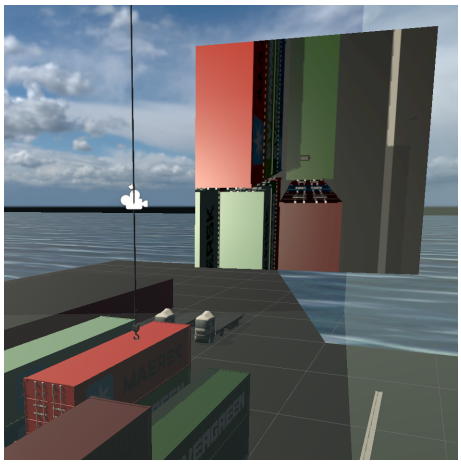
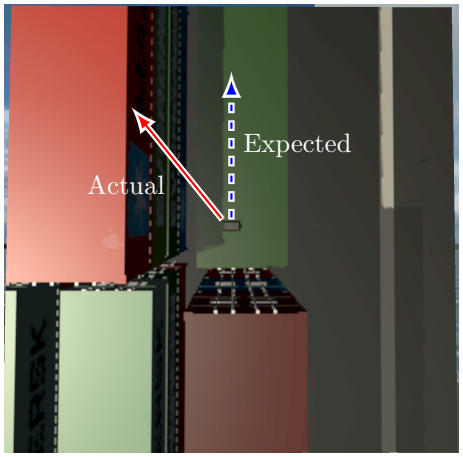


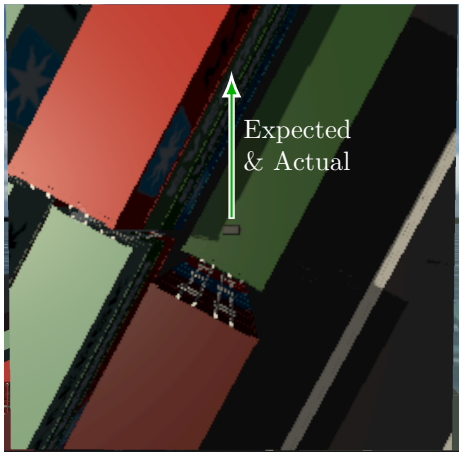
Figure 9.2: The view from the cabin. Note the red container is behind the hook from the cabin's view, while the green container is above the hook on the screen.

This was made especially obvious when observing players using the desktop variant. As the field-of-view through a computer monitor is smaller than an HMD and a computer monitor cannot provide depth perception, the top-down view from the screen becomes even more crucial for aligning the hook with the container. Unlike joysticks, keyboard WASD controls are limited to movement in eight directions, and players with less computer game experience are unlikely to use the diagonals. Additionally, previous iterations had much faster hook movement for players on the desktop variant. Altogether, this caused players to struggle to align the hook with the container, frequently overshooting or moving it in wrong directions, even as the top-down screen hinted that an upwards movement would be enough.

Changing the screen to rotate its view to match the crane's rotation should alleviate this issue. The resulting view is illustrated in Figure 9.3b.



(a) Earlier iterations. Players would expect the hook to move towards the top of the screen when they moved it forwards, but it would move at an angle depending on where the crane was facing.



(b) Current iteration. The screen rotates with the crane, making the direction the hook moves intuitive.

Figure 9.3: Close-up of the screens in both iterations.

9.1.3 Consistent hook speed

The crane's controls in previous iterations were simple and did not account for the hook's position. Moving the hook back and forth would move the crane's beam a set amount of degrees, and left/right movement would rotate the crane a set amount of degrees as well. The consequences are that the speed the hook moves is strongly dependent upon its position due to simple trigonometry. Moving the beam would barely move the hook at low elevations, while moving it incredibly quickly at high elevations. Conversely, the hook would move quickly sideways when the beam was low, and significantly slower when the beam was high. Users would get confused attempting to learn the controls, as the container would move too slow in certain positions, and they lacked any other form of feedback.

Additionally, as noted in the previous subsection, the hook would move significantly faster when playing on the desktop variant. The issue was that the crane would move a set amount every rendered frame. A VR game will automatically limit its framerate to match the refresh rate of the HMD in use, while a PC game will render as many frames as possible unless the game or PC itself is explicitly configured to limit it. Additionally, desktop mode is less resource intensive than VR mode, allowing more frames to be rendered within the same time span. If the hook speed was calibrated to feel natural in VR, it would be extremely fast and difficult to control on desktop. Alternatively, hook speed that felt natural on a fast desktop would be somewhat slow on a laptop and unusable in VR.

Instead of using a set number of degrees to rotate the crane and beam, a value was defined for hook speed per second. Formulas were derived to compensate for the position of the crane to make the hook cover the same distance per second regardless of its position. The speed is multiplied with the time in seconds since the last frame, making the movement identical regardless of hardware and mode of play.

9.1.4 Container hooking

Many players have been observed struggling to pick up containers or having them fall off the hook after lifting them. As seen in observation O2.4, it tends to happen when the player hooks the container from the top or sides instead of the bottom. Further investigation showed that the container would push against the container below it when not hooked from the bottom, accelerating it off the hook. As a simple measure against it, hooks were made to not attach to the containers unless they were being pulled upwards.

9.1.5 Yrkeskatalog

The application was made available through the NAV Immersive Job Taste Catalogue, Yrkeskatalog, which is described further in Section 4.4

Releasing a Unity application for use by others requires the developer to "build" the application. Building an application creates a bundle of files that can be ran from a computer or put on a standalone HMD in order to play it. The application had to be built and uploaded twice for computers, as the application still did not support automatic detection of VR HMDs. The process was not particularly difficult, as switching between immersive VR and desktop mode only required switching the setting within the Player Manager described in subsection 8.1.4.

Distributing it for standalone HMDs turned out to be more complicated. As described in subsection 7.1.6, the video files of Heisekran VR were too large for the Oculus Quest to accept them, meaning they had to be split into a separate file using the Addressables system. Yrkeskatalogen expected standalone HMD distributions to consist of only a single APK file. The Addressables solution had to be reverted, and another solution had to be found.

The previous 360-degree video platforms required two video feeds to be stacked on top of each other as seen in Figure 9.4a, with one video feed per eye. Investigating the new 360-degree video

prefab showed that it could support videos with only one feed, as seen in Figure 9.4b. Cutting the existing videos in half would also roughly halve their file size. It was unknown whether halving the video size was enough to make it run on the Oculus Quest if the video files were packed into the APK, but halving the size of the video would nevertheless make the application faster to download for the variants running on computers.

(a) A single frame from the original video

(b) A single frame from the modified video

Figure 9.4: 360-degree videos before and after modification. Both the upper and lower copy of the video feed in the original videos were identical, so they could be modified with no loss of information.

The open-source software FFmpeg is free and capable of converting and modifying most video and audio formats (ffmpeg 2021). This software was used with the following command in a terminal in order to crop the bottom half of the video, based on the command found at (Logix 2021). `input.mp4` and `output.mp4` was replaced with the name of each video file.

```
ffmpeg -i input.mp4 -filter:v "crop=iw:1920:0:0" output.mp4
```

The total size of the videos before conversion was 2197 megabytes. After conversion, the total size of the videos was only 691 megabytes. This is even less than half the original size and is likely because FFmpeg had a more efficient compression algorithm than the software that made the original video. Post-conversion, the application would successfully run after packing the video files directly into the APK. The resulting APK was uploaded to Yrkeskatalogen.

9.2 User testing: Students

A user test was arranged the 3rd February 2021. Students were invited to try out various VR applications, including Heisekran VR. Eight students volunteered to test the application.

O1: One player quit because of nausea after watching the video from the crane driver's perspective. One player quit because of fear of heights after climbing the stairs. One player mentioned slight nausea and fear of heights while climbing the stairs but finished lifting a container regardless.

O2: One player learned the controls quickly, proclaiming "Seems like I was born a crane driver!". Another player had a few containers fall off the hook, got frustrated and quit. "Crane driving was not for me."

Chapter 10

Iteration 6

This chapter describes the changes made in the last iteration of Heisekran VR, then proceeds to describe user tests and interviews that it was used for.

10.1 Changes

This iteration focuses on implementing prefabs that will be shared between future Immersive Job Tastes. It also implements some improvements to the player experience, as well as ensuring Quest 2 support.

ID	Change	Section
C6.1	Attach containers to the hook using a rotating joint. Detach containers only when they are placed on a surface.	10.1.1
C6.2	Make holding the side joystick more intuitive.	–
C6.3	Add the tablet package with tasks and navigation.	10.1.2
C6.4	Add a system for changing between hands and controllers when needed.	10.1.4
C6.5	Create a "NavMesh" to let the tablet package find a path to navigate the player to their goals.	10.1.5
C6.6	Use the Unity InputSystem for controllers to easier support more devices.	C6.6
C6.7	Add support for Windows Mixed Reality devices.	10.1.3
C6.8	Set up haptics for teleporting and controlling the crane in VR.	10.1.6
C6.9	Play sounds when initialising and finishing teleportation.	10.1.6
C6.10	Merge Desktop rig into XR rig.	10.1.7
C6.11	Add an event-based activation system for the desktop mode.	10.1.8
C6.12	Add a manual with the controls to the desktop mode	10.1.9
C6.13	Rearrange the 360-degree videos to be more intuitive.	10.1.10

Table 10.1: Changes made in this iteration. Changes that require further explanation will refer to a later section.

10.1.1 Hook joint

While the change to only hook containers when lifting in C5.4 lessened the occurrence of containers dropping, it did not fix it entirely. It was clear that the current design to hook containers did not work and would have to be redesigned.

To redesign the hooking mechanism, the issue with the existing one had to be identified. The current system uses two collision spheres, one attached to the hook, and one attached to the

container's rope. When the hook's collision sphere intersects with the rope's, the rope will appear to show the player the container can be lifted. When the hook gets close enough to the centre of the rope's attachment point, the container becomes attached to the hook. Specifically, the container is "parented" to the hook, meaning it follows the hook's movements. The container also ignores gravity while attached, and its velocity is set to zero to make it match the hook's velocity when the hook moves. This resulted in a container that is rigidly attached to the hook when lifting and moving, without the possibility of rotation. As containers do not rotate when lifted by a crane in real life, this approximation does not look out of place when playing. The container will move when it hits other objects, and if it hits another object hard enough or gets pushed down onto the ground, the hook will move far enough away from the rope that they will detach.

The first suspicion was that the hook detached due to being attached at the border of the attachment sphere and rounding errors during movement caused the game to occasionally believe the container was pushed off the hook. It was found that the distance needed between the hook and rope was already greater than the distance needed to hook, so the possibility was eliminated. On a hunch, the possibility to detach the container was temporarily disabled, to observe what happens when a container was lifted from the top. It was found that after the container was attached to the hook, it would not slow down after any force was applied to it. Like a container in space, it would fly at a constant speed away from the hook if it touched another surface during lifting. The effect was obvious when the lifted container was intentionally collided with another one, launching the container into the sky. However, it would also happen when the container was lifted during normal operations, from lightly touching the surface of another container before being lifted off it. The movement from such a small force was not visible from the perspective of the crane driver, but after leaving the container lifted above the ground for a short period of time, it was floating far enough away from the hook that the game would normally detach it. This was the cause of containers dropping off the hook seemingly at random. Hooking the container from the sides or top would push the container harder against the surface below it, giving the container a higher velocity and shortening the time it would take for it to fall off, causing the frustration for some players during earlier testing.

To fix this, lifted containers needed their velocity to slow down continuously, instead of setting it to zero in the moment they were hooked. A simple solution would be to simply set the velocity to zero every frame or apply a significant amount of air drag to the containers to simulate the same effect. The issue would be that the containers would end up in impossible positions if they were tilted or fell to the side, and no amount of player input would be able to fix it. Efforts could be made to righten the containers while on the hook, acting as if it was hanging off a joint, but it would take time designing such a system. At that point, the effort could be spent on a more robust system instead than trying to fix the current one. Attaching two objects with a joint should be a common issue with common solutions, and after some searching, the Unity Joints system was found.

A Unity Joint is an attachment point between two objects in one point of space. A joint does nothing by default but can easily be configured to prevent movement or rotation, and apply forces to objects in an effort to righten their orientation. Some specialised variants exist that simplify configuration, mainly targeted at creation and animation of characters or the creation of springs. As none of those alternatives were applicable for hooking the container, the universal type of joint was used. To create a joint, the Configurable Joint component was applied to the containers. The anchor point was set to the middle of the rope, where the hook would intuitively attach. Joints apply their restrictions even if they are not attached to another object, so the joint on the container needs to be unrestricted to begin with. If the joint always had restrictions applied, then containers would pivot around the rope even before being attached to the hook. When the hook gets close enough to the rope, it gets attached to the container and the restrictions are applied. Firstly, X, Y and Z motion is restricted to prevent the container from detaching from the hook. Secondly, a dampener is applied to its angular velocity to slow it down if the player hits another object and the container starts spinning. Finally, the joint is configured to attempt to orientate the container along the length of the ship in case the player manages to rotate the container by accident. This last configuration is a compromise between realism and usability. In actuality, the crane driver would be unable to rotate a container after it has been lifted. Rotating the container would have to be done through cooperation with a co-worker on the ground if needed. To prevent the players from

losing confidence and locking themselves out of progressing the game, automatically realigning the container is necessary, even if it is unrealistic. If a multiplayer component is ever added to the application, the task of rotating the container could be done by another player instead.

Previous iterations were dependent on the less-than-ideal physics interactions in order to detach the containers from the hook. When lowering a container down onto a surface, the container would be held in place from being pushed against the surface, and the hook would eventually get far enough away from the rope to detach. The Unity joint is more rigid and will not detach unless a significant force is exerted onto it. By default, it would only detach if the container hit the ship at full speed. Attempting to detach the container by forcefully lowering it onto the ground would likely spin it around the joint instead. Lowering a container onto another container would catapult both containers in different directions rather than detach the container from the hook. A system for detaching containers would have to be more deliberate in order to support hooking containers with Unity Joints.

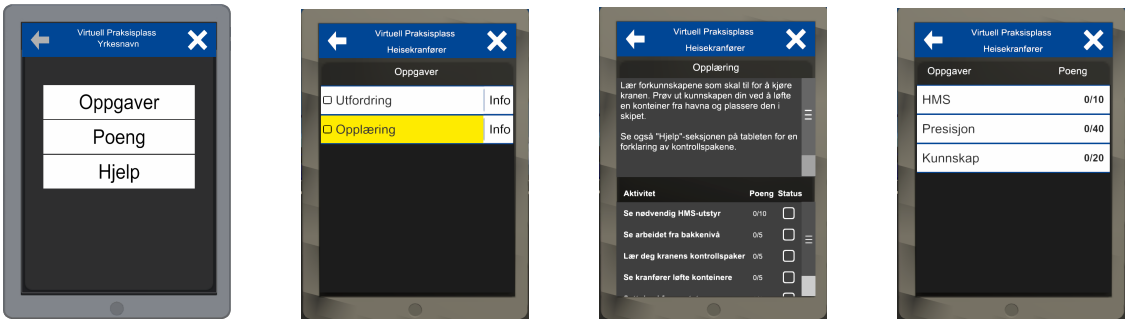
A simple solution was devised in order to keep complexity down. A collider tagged as "Container-Bottom" was added to the bottom of the container prefab, protruding beyond the physical bottom of the container. This collider had its physical collisions disabled, so it would intersect with the surface when the container is placed down. A basic script was created to mark an object as a valid surface for the container to be detached upon. When a collider enters or exits the object the script is attached to, it checks whether the collider is tagged as the bottom of a container and if yes, tells the container that it just entered or exited a valid surface. This keeps the complexity of the interactions between container and surfaces to a minimum, allowing the container script itself to determine how to use this information. The docks, truck beds and deck of the container ship was set as a container surface. Additionally, a collider was added inside the top of the container prefab and tagged as a container surface, allowing the crane driver to stack containers. This solution allows the crane driver to consistently place containers on any intended surface while preventing the containers from detaching in weird places, like the edge of the ship or the roof of a building. When a container is picked up, the container is likely intersecting with a valid surface already. The container script will prevent the container from detaching until it has left the surface it was placed on. Having the surface tell the container when it intersects is efficient, as code is only ran when a container actually intersects with a surface instead of constantly checking for container placement. The possibility of tagging any surface as a valid detachment point also allows the container to know where it has been placed, which is useful for the future when creating tasks telling the player to place a container within the ship, onto another container, or onto a truck.

10.1.2 The tablet package

The tablet package is a tablet interface intended to support the player through the application. It includes the tablet itself alongside its interface, but also contains supporting components like a task system, point scoring and navigation helper. Together they provide the player with goals to complete, rewards for good performance, and clarifies where to go and what to do next. This is especially useful when the application is presented by NAV career counsellors without close familiarity with the details of the game, making it difficult for them to guide players through it step-by-step. Most players should have experience with tablets or smartphones equipped with touch screens, requiring players to push the buttons that appear on the screen. The tablet is also explained in the Immersive Job Taste tutorial and is in use by other job tastes, so its usage should be familiar for the player. A tablet screen also has the bonus of being large, making it easily readable in VR devices.

Once the tablet package is properly implemented, it will be brought out by pressing a button on the controllers. Bringing out the tablet turns the player's hands into VR controllers with a bright laser coming out of the end. The player interacts with the tablet by pointing the laser at it and pressing the trigger. Pointing the laser at a button on the tablet and pressing the trigger naturally activates the button, and dragging scroll-bars with the laser scrolls the text if there is a lot of content to read. The player starts with the menu as seen in Figure 10.1a. Pressing the Tasks button (Oppgaver) takes the player to a list of tasks as seen in Figure 10.1b. Here the player can see the available tasks and select which task they want to do. Pressing a task sets it as the active

task and starts navigating to the first sub-task. Pressing the info button brings the player to the page seen in Figure 10.1c which gives further information about the task, with a description, list of sub-tasks and the points the sub-tasks are worth. The Points (Poeng) button on the main menu shows a list of points per skill as seen in Figure 10.1d, each of which can be selected to see a further explanation about the skill's relevancy to the profession. The Help (Hjelp) button shows a detailed text to the player that could consist of an explanation of the application, the workplace, the controls of the game, etc.



(a) Main screen (b) Task list (c) Task information (d) Skill information

Figure 10.1: Pictures of the tablet's different pages. The user can click different buttons on the main page to reach the different screens, go back using the top left arrow, or close the tablet with the cross in the top right.

Implementing the tablet package requires some work from the developer. Naturally, the names and descriptions of tasks, skills and the help menu must be written to match the workplace in the application. The tablet comes in form of a prefab as seen in Figure 10.2 split into the visual model (Mesh), the underlying systems (Managers) and the UI shown on the tablet itself (Canvases). The TabletManager component contains the name of the profession. The skill manager (FerdighetManager) component contains the name and description of each skill the workplace requires. The Task Manager (OppgaveManager) component does not require configuration in-and-of itself, but requires instances of Tasks created using a supporting script included in the package. These instances can be placed anywhere, but gathering them under the Task Manager component in the hierarchy is intuitive and makes it easy for other developers to find where all the tasks are located. These Tasks consist of a name, description and sub-tasks, alongside a navigation target to navigate the player towards once the task is selected. Tasks may also specify a prerequisite task that must be completed before it will appear on the tablet. Finally, the help text in the canvases can be edited to contain a relevant help page for the application.

Once the content of the components is written, extra attention must be paid to the tasks. The maximum points attainable to each task must be set in order to make the total point count in the task- and skill pages show correctly. The tablet package adds a button to the Unity editor to generate, backup and recover a text file which contains the maximum point count each sub-task can grant within each skill. Changing the maximum requires editing the file using a text editor. One could edit the file so that watching a 360-degree video grants up to 5 points for a player's knowledge and stacking a container onto another grants a maximum of 10 points to precision and 5 for speed. Once the maximum points have been defined, it is up to the developer to grant these points during gameplay. Granting points when a container is lifted and placed down would be a natural time to grant points.

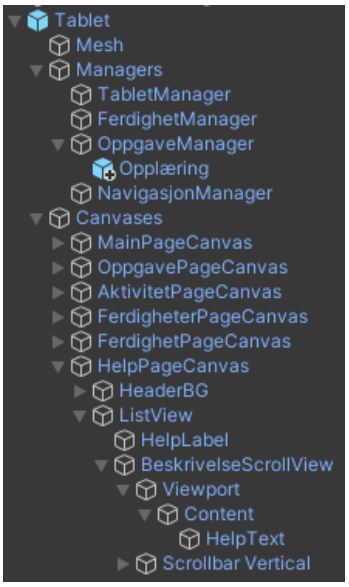


Figure 10.2: The hierarchy of the tablet prefab.

Additional work had to be done in order to support the tablet package outside of the work needed to configure the package itself. Out of the box, the tablet model would float at the bottom of the player's field of view when inactive and require a button press on a VR controller to activate. This was found to be distracting during development, so the tablet package was modified to be hidden while inactive. Support for changing hands to the laser pointer or hiding the hand model while the tablet is held also had to be implemented, further detailed in Section 10.1.4. Furthermore, a navigational mesh had to be created in order for the tablet to find a path from the player to the task's goal and help lead the player to it. This endeavour is further documented in Section 10.1.5.

10.1.3 Unity InputSystem

Unity's InputSystem framework is "intended to be a more powerful, flexible, and configurable replacement for Unity's classic Input Manager" (Unity 2021b). The legacy Input Manager is tedious to configure, requiring one to manually create listings for XR controllers corresponding to the axes documented by Unity (Unity 2021e) as seen in Figure 10.3. The axes in question are arbitrary numbers from 0 to 28 which Unity reports predefined button presses to. It is tedious enough that some of Unity's developers have taken to sharing a pre-configured Input Manager file to shorten the time it takes other developers to configure it (StayTalm_Unity 2021). Even once configured, the Input Manager is limited to polling specific axes for changes continuously and cannot simply react to a button being pressed. Furthermore, applications need to figure out which buttons to read based on the type of controllers that are connected, as some controllers lack certain buttons or report them to different axes than the ones Unity expects.

The InputSystem framework allows the developer to configure "actions", which have multiple advantages over the Input Manager's axes. Firstly, an action may consist of multiple buttons, any of which will trigger the action. Moreover, an action can be defined globally and referred to from anywhere, allowing a complex action to be defined and changed in one location to take effect across the application. One can add functions as callbacks to the action, allowing input to be handled in an event-based manner instead of continuously polling the buttons. Furthermore, one can enable or disable actions at will, easily preventing actions from being called when not needed i.e. teleporting while viewing a video. There is also a possibility to merge different button states into the output of a single action, apply a filter to the buttons' value and more.

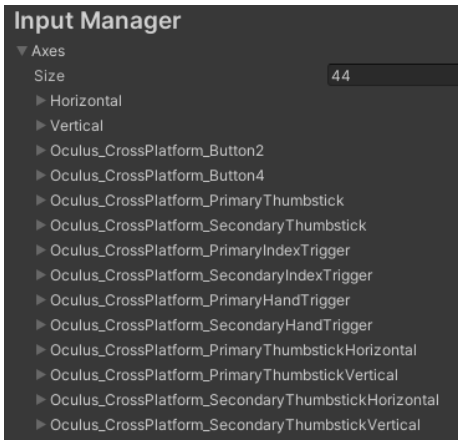


Figure 10.3: An excerpt of the legacy Input Manager window.

The Unity XR framework attempted to work around the limitations of the Input Manager before the release of the Input System by introducing "usages". These would represent certain buttons or joysticks, removing the need to configure the Input Manager and simplifying support of different controllers. The most important kind of usages are the CommonUsage, representing inputs that most controllers are expected to have. Common examples are Primary2DAxisClick representing pressing in the primary joystick, and Grip simply mapping to the grip button. Unity documents the full mappings between Usages and buttons for various devices on the same page as the axes for the Input Manager (Unity 2021e). Inputs specific to certain devices can be used by importing their plugins containing device-specific usages, like the Oculus plugin having a Usage for the touch sensor on the joystick. Using the Usages system directly does not alleviate the Input Manager's issues, however. It still requires the application to poll for inputs continuously, and the developers need to make a system for handling different types of controllers themselves.

Previous iterations of Heisekran VR used the Usages system directly for input from XR devices. An example would be the teleport action mapping to the Primary2DAxisClick usage, teleporting whenever the primary stick was pressed in. This worked fine while targeting the Oculus Quest device but testing the application with other devices showed issues. The HP Reverb G2 did not map its buttons in a standard way. Instead, its sole joystick reports to the Secondary2DAxis usage, and the face A / X / B / Y buttons are not reported at all. This issue has been reported to Unity and marked as "Won't fix" (Unity 2021a). This could be worked around by detecting the type of device connected. A temporary solution was tested which would swap the teleport button to the secondary axis if the connected device was a WMR controller. The solution was found to be tedious and not particularly robust, as the workaround simply set the required Usage to teleport in various components in different locations in the hierarchy. Changing which devices the workaround supported or the functionality of the application itself would be difficult for somebody without prior knowledge of the application. As new VR devices may become available after the original developers are unavailable, needing to know the specifics of the application to add support for new devices is unviable. A solution that could be changed in one location would be preferred.

The pre-release of version 1.0 of the Unity XR Interaction Toolkit switches to the Input System's Actions instead of Usages directly. There are two ways to setup these actions that can be seen in Figure 10.4. The developer can either configure the action within the component itself or refer to a global action. Global actions are defined using an Input Actions Asset. Opening an Input Actions Asset shows a menu like the one shown in Figure 10.5. The file may contain multiple Action Maps, each of which contains multiple Actions. As mentioned earlier, each Action may consist of multiple inputs, which Unity calls Bindings. The Bindings consist of either XR Usages or other inputs, like keyboard buttons, mouse movements and game controller buttons. In total, they tell Unity what kind of inputs should trigger the actions and give other Unity components a unique way to refer to specific actions. With just the name of the Action Map and Action, the Unity component in Figure 10.4 can refer to the teleport action in Figure 10.5. If all components that want to know whether the player is attempting to teleport refer to that action, a developer would only have to

change the bindings in one location to affect the entire application.

Figure 10.5 shows an action map containing actions for teleporting with the left and right controller, as well as bringing up the tablet with a button press. The left teleport action contains a binding for pressing in the joystick with a WMR controller, an Oculus controller and three common Usages representing joysticks for any XR controller. The right teleport action has the same bindings, but for the opposite hand. These bindings were selected as neither the WMR- nor the Oculus controller's joystick press would trigger with only the common usages bound to the actions. The tablet can be triggered by the primary face button (A or X) on most controllers, and the Menu button for WMR controllers as their primary buttons are not picked up by Unity. If the WMR controllers' face buttons are ever fixed in Unity, they will trigger the tablet action as well.

The above allowed the application to support Windows Mixed Reality without requiring strange workarounds or encountering any bugs. OpenXR controllers could also be added to these action maps in order to support HMDs running on the SteamVR platform. While this was developed, there was no OpenXR plugin supporting SteamVR for Unity XR, but once the plugin is created it should be trivial to add OpenXR support as well.

Figure 10.6 shows another action map containing actions for the Desktop rig. The Movement action is configured as a 2D vector composite created by four different keyboard buttons. The Look action is configured as a 2D vector from mouse movement. Other actions are configured to match Figure 8.1.

The current iteration only has one action map. Creating multiple Action Maps allows the developer to create multiple control schemes for the same Actions. If the Actions within the Action Maps are named the same thing, it would be possible to swap action maps while using the same actions, allowing a player to choose whether they want to teleport by pressing a stick or by pushing it forward. Alternatively, Action Maps can simply be used for organisation, allowing XR related bindings to stay in one Action Map, and Desktop bindings in another.

Note that configuring an Input Action asset and pointing other components to it is not enough to make things work. Actions and Action Maps are disabled by default, and a component must explicitly enable them in order to trigger the Actions. Unity's quick-start guide recommends the PlayerInput component (Unity 2021c). The PlayerInput component links to an Input Action Asset file, automatically enables all the actions within it, and makes it easy to add events that will happen when the Actions are triggered. This is convenient when the Input Action Asset represents a control scheme that the player always has access to, which is common in Desktop games. It is less convenient in XR games where the control scheme is spread across multiple components and is dependent on context. For that reason, this component was used for the Desktop rig. The XR rig would instead enable and disable the events manually using the Hand Manager described in the following section.

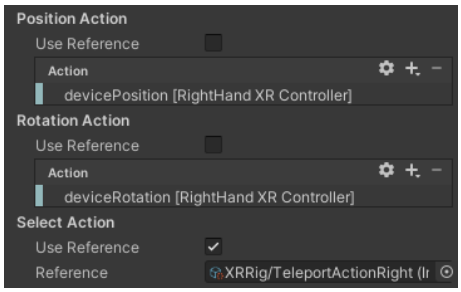


Figure 10.4: The setup of actions within a Unity component that requires them. One may either configure an action specific to the Unity component, or one may refer to an action contained in an Input Action asset.

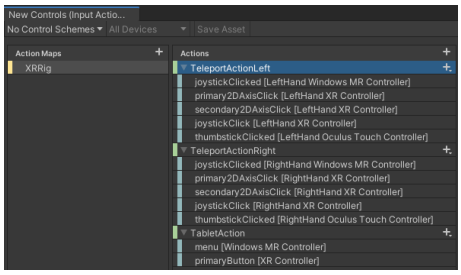


Figure 10.5: The configuration window showing actions created for the application inside an Input Action Asset using the Input System. This asset configures the controls for XR, defining which buttons initiate a teleport and bring out the tablet.

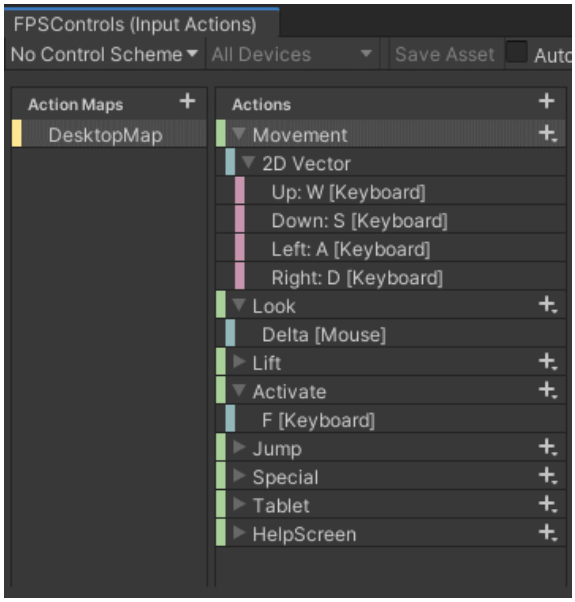


Figure 10.6: The configuration window showing actions created for the application inside an Input Action Asset using the Input System. This asset configures the controls for Desktop mode. The movement action is a 2D vector composite created from simple button presses, while simpler actions like Activation only require a single button binding.

10.1.4 The Hand Manager

An XR controller in Unity XR consists of a Controller component, an Interactor component and optionally a visual model to represent a hand or game controller. The Controller component is responsible for tracking the controller's position and defining which buttons perform actions. The Interactor component is responsible for communicating with other objects in the game world and performing the actions related to them. There are mainly two types of Interactor components: A "Direct" interactor which interacts with objects that are touching the hand itself, and a "Ray" interactor that draws a line from the controller and interacts with the object the ray hits. These components can be further customised by i.e. filtering the objects an Interactor can interact with, changing the size of the Direct Interactor collision box, or the curvature of the line drawn by the Ray Interactor.

A Unity project will likely have multiple XR controllers per physical controller. An intuitive way to grab an object would be to touch it with one's hand and press the grip button on the side of the controller. However, teleporting is more intuitive through pressing a button or stick on the controller's face and aiming a ray interactor at the destination. Each XR controller supports a maximum of one Controller and Interactor component, and Unity XR has no support for swapping

between components or controllers out of the box. It would be very unfortunate if the player had to aim a ray at one of the crane's sticks to grab them, or worse, have to touch the floor and press the grip button to move around! Unity does support having multiple controllers enabled simultaneously, but the consequence of having a Ray Interactor enabled is that the ray it is drawing will always be visible. Having the teleport line visible when not pushing a button would be distracting for the player and make the controls more difficult to understand. The problem would be exasperated if multiple Ray Interactors were needed, like with the tablet, as multiple lines would suddenly be coming out of the player's hands in different directions.

The common solution is to have a script attached to the player which enables and disables XR controllers when they are needed. Previous Immersive Job Tastes used a basic script to enable the XR controller responsible for teleportation when the joystick was pressed down and hide it when the stick was released. This made the teleportation ray only show up when needed and disappear once the player was done teleporting. This solution would have to be expanded upon after adding the tablet interface, which required yet another XR controller. The solution would also need to be reworked to use Input System actions as defined in Section 10.1.3, instead of using the joystick Usage directly.

The Hand Manager component can be seen in Figure 10.7. It has a reference to the hands, teleportation rays, virtual controller models and tablet. It also references the teleportation and tablet actions. On start-up, the Hand Manager hides the controller models and teleport rays while enabling the hands. It also enables the teleportation and tablet actions. When the player presses in their joystick, the hand manager will enable the teleportation ray for the corresponding hand and prevent other actions from triggering, like teleporting with the other hand or bringing out the tablet. When the player lets go of the stick, the player will teleport, and the Hand Manager will disable the teleportation ray afterwards. Bringing out the tablet replaces the hands with controller models and a straight ray, disabling all other actions until the tablet is put away. This allows the player to interact with objects, teleport and use the tablet when needed without being overwhelmed by all functions at once.

The tablet prefab itself was previously responsible for activating itself. It would poll the Primary-Button usage continuously until the player pushed the button, then pop out. This iteration removed that piece of code from the tablet prefab and gave responsibility to the HandManager instead. This allows using the TabletAction configured in the Input Actions Asset for bringing out the tablet and disabling the Tablet action while other interactions are in progress.

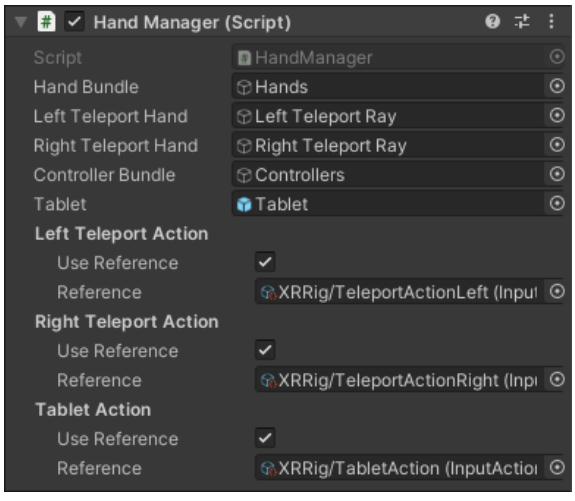


Figure 10.7: The Hand Manager component in Unity. Configuring it requires specifying the XR controller objects, the Tablet object, as well as which Actions would be needed to activate a teleport or the tablet.

10.1.5 NavMesh: A pathfinding tool

A Navigation Mesh, or NavMesh for short, is a tool used for pathfinding purposes. It allows the game engine to know where an agent is capable of traversing and finding the shortest path from a point to another. The Unity NavMesh is easy to build, only requiring the developer to mark which objects in game are traversable and configuring the size of the agent seen in Figure 10.9. Creating the finished NavMesh afterwards is called "baking" the NavMesh, and the end result can be seen in Figure 10.8.

The tablet prefab requires a NavMesh to navigate the player to their next objective. When a task is selected, it will create a multitude of green arrows on the floor that move along the NavMesh towards the destination. The NavMesh agent was configured to be identical to the desktop player rig, with the values seen in Figure 10.9. The surface of the docks was set as walkable to cover most of the docks before stationary obstacles like the trucks and buildings were set as non-walkable. That way, the application would never attempt to navigate the player through an immovable obstacle. Certain obstacles like the crane and containers cannot be marked as non-walkable, as they could potentially be moved out of the way and placed in another location. It is worth noting that this does not compromise the navigational capabilities of the tablet, as agents moving along the NavMesh in Unity will move out of the way of obstacles when they get close enough even if the NavMesh covers the surface underneath them. Finally, the crane's staircases and surfaces were marked as walkable, and the invisible railings around the staircases were marked as unwalkable. This made the NavMesh successfully cover the pathway up to the control room.

The above solution seemed to work, but a fatal flaw was discovered after some testing. The regular Unity NavMesh workflow assumes that all objects are static and will not move after the application has started. The NavMesh is baked during development before the game is built and distributed, and cannot change while the game is running. When the crane is rotated or moved along the tracks, the NavMesh will not follow the crane and will instead remain in place as seen in Figure 10.10. The application needs to alter the NavMesh while the game is running in order to make it successfully follow the crane's movements.

Unity has a different solution to baking NavMeshes that can bake during runtime. The NavMeshComponents package uses Unity components added to objects to bake NavMeshes instead of using the editor to mark objects, and it allows baking meshes during runtime. Installing the package requires downloading it from Unity's GitHub page (Unity 2021d) and manually adding it to the project. The NavMeshComponents package is deemed a beta release and is not available to download through Unity's usual package manager. Most likely this is to avoid developers installing the package when they do not require its more advanced features.

The NavMeshSurface component is responsible for baking the NavMesh when using the component-based workflow. Unlike the editor-based workflow, it will attempt to include every object in the game world when baking NavMeshes by default. This is easy to work around when dealing with an object like the crane, as the NavMeshSurface component can be configured to only include its children in the Unity hierarchy. The NavMeshSurface component can then be attached to the top-level Crane object and will only bake NavMeshes for the crane. To avoid marking every flat surface of the crane as potentially walkable, the surfaces that are intended for the player to traverse were put in the "CraneSurface" layer, and the NavMeshSurface component was further configured to only use objects that are part of that layer. The results differed slightly from the editor-based workflow, but the resulting NavMesh fit the crane properly.

Using the NavMeshSurface component to bake a NavMesh for the entire game world is more difficult. The surface of the docks and the buildings on it do not have a parent in common, so the solution of collecting only the component's children cannot be used. Additionally, one cannot simply configure every component in the game world with a certain layer. Some components already have a custom layer for other purposes, so it would ruin the entire purpose of the layer system. Instead, the component can be configured to consider every object as *non*-walkable by default, and another component named NavMeshModifier can be used to manually set objects as walkable. The object with that component attached will be considered walkable in addition to all the object's children. Utilising this solution, the NavMeshSurface was attached to the docks,

and the NavMeshModifier attacked to the surface of the docks itself. This made the NavMesh cover the docks but avoid navigating through solid objects like buildings and trucks. This solution would also avoid putting a NavMesh underneath movable obstacles like the containers and crane, but the NavMeshModifier component also has an option to mark objects to be ignored during NavMesh baking. Ignoring the containers, crane and 360-video platforms for NavMesh-baking led to a similar NavMesh as the editor-based solution.

Now the application had two separate NavMeshes, one for the docks, and one for the crane itself. Linking these two NavMeshes requires another component reasonably named a NavMeshLink. Adding this component to an object allows one to create a link between two NavMeshes, requiring only a startpoint, endpoint and width of the link. Attaching the component to the crane's bottom staircase allows the link to follow the crane even as it moves along the surface of the docks.

Finally, the application must rebuild the NavMesh at certain points during gameplay. Baking a NavMesh is an expensive process, so one cannot rebuild the NavMesh continuously during gameplay without ruining the performance of the application. Identifying the right time to rebuild the NavMesh is needed. The crane will only move when a player is controlling the crane from the control room, so baking the NavMesh is only needed while the crane is being driven. Unity does not support navigating agents on moving platforms, so rebuilding the NavMesh while the crane is in motion is pointless. One could rebuild the NavMesh after the crane has been idle for a certain amount of time, but there is a simpler solution. One can assume that the player will not need navigation while they are still in the control room's chair. Ergo, the right time to bake the NavMesh is when the player leaves the chair and goes out of the control room. With this, the application has a working NavMesh for the tablet to navigate the player, even after the crane has moved from its original position.

It is important that the baking process does not take too long, even when it happens rarely. The application pausing for even a split second would take the player out of the application and potentially induce nausea. Only including the crane's necessary surfaces in the baking process speeds up rebuilding the NavMesh. The docks never change, so its NavMesh does not require rebuilding. The baking process can also be sped up significantly by configuring the NavMeshSurface component to perform its calculations using the physics collision model rather than the visual model. There is no downside to this for Heisekran VR, as the player will be interacting with the crane's collision model in the first place.

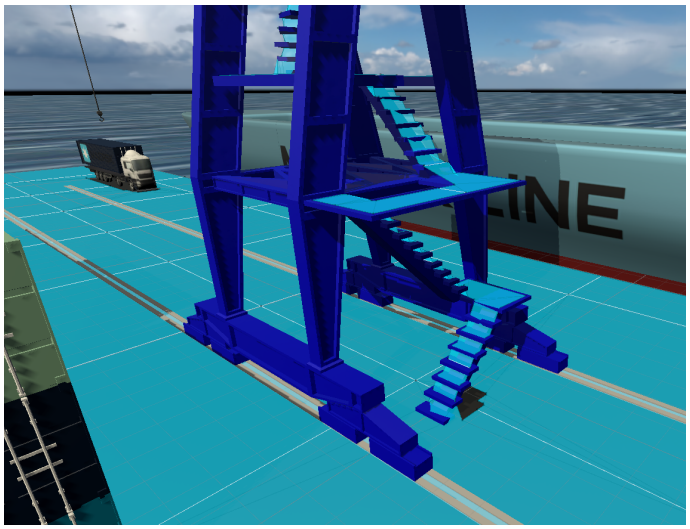


Figure 10.8: The NavMesh baked on top of the Heisekran VR application. All light blue surfaces are denoted as walkable by the player.

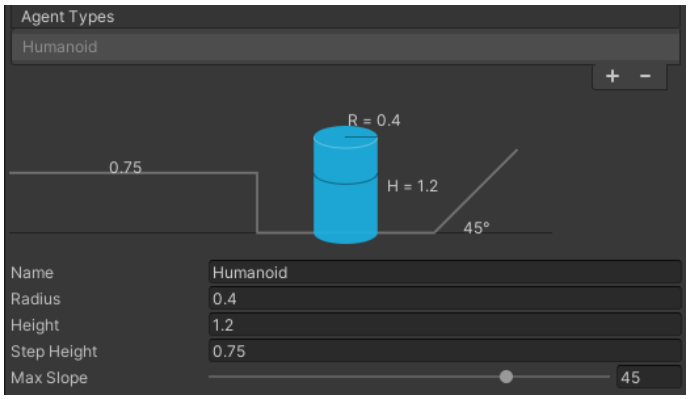


Figure 10.9: The configuration of a NavMesh Agent is simple and has a self-explanatory illustration that automatically updates based on the specified values.

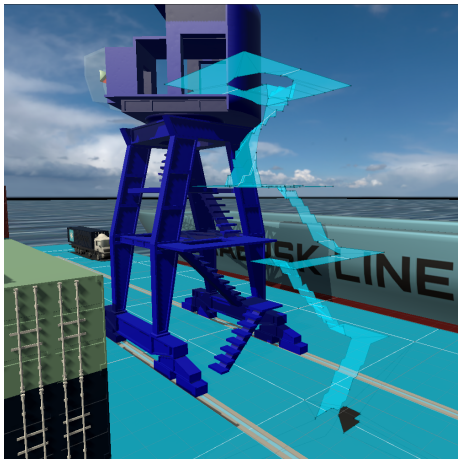


Figure 10.10: The NavMesh in the classic Unity workflow will remain where it started even as the crane moves away. The result is that the NavMesh will float in mid-air, and it will be impossible for the tablet to navigate to a target on the crane.

10.1.6 Control Feedback

Some players struggled with the VR controls in the crane as seen in O4.2, with players never letting go of the control sticks or moving their hands in the wrong directions. Players from the same testing sessions requested haptics as a form of feedback while controlling the crane.

Haptics were added to the crane's control sticks to make players more aware of how they were steering the crane. Hovering a hand over an interactable object will lightly vibrate the controllers to indicate to the player that they can interact with it. Grabbing the object will give the controller a stronger vibration. Holding one of the crane's control sticks will vibrate the controller periodically depending on how far down the control stick is currently being held. Holding the stick upright at its original position vibrates roughly once every two seconds. Pushing it all the way down vibrates the controller roughly twice a second. This gives the player feedback when they are holding the control stick, which should alleviate cases where the players forget to let go when they are finished with a stick O4.2. It also gives the player immediate feedback when they actuate the control stick correctly, hopefully avoiding cases where the player struggles to move the sticks correctly (O4.1).

Users were occasionally disoriented or confused by teleportation. To give more feedback when they push the stick, a sound is played when the teleport is initiated, and another sound is played once the stick is released and the teleport is performed.

10.1.7 Merging the Desktop and XR rig

As mentioned in the requirements (Section 5.1), the application should ideally support Desktop and XR mode with a single project. The PlayerManager created in that iteration supported toggling between XR and Desktop in the editor, but it simply toggled either the XR or Desktop rig on startup. The other functionality of the application still had to separate between the XR and Desktop rig. Ideally, the same player object would be in use for both XR and Desktop mode, allowing most of the code to be reused across both modes. The only code that should differ is code related to input that differs between the rigs, i.e. XR controllers, HMD positioning and keyboard & mouse controls.

This iteration merged the Desktop rig into the existing XR rig. The PlayerManager class now needs to disable multiple components based on which mode is selected. XR mode disables the FPS Player Controller to prevent accidentally moving using the keyboard and mouse and causing nausea. It also disables the large collision capsule representing the Desktop player's body to avoid any accidental interactions with the world. Finally, it disables any 2D overlays, as they cause performance issues for XR displays. Desktop mode disables the XR Rig component, the Hand Manager, and all the XR controllers. In XR mode, the player camera starts on ground level before the HMD's position is added to it by the XR Rig component. In Desktop mode, the offset is set to always be on eye level, as there is no HMD to provide an offset. This merged Player Rig allows packages like the Tablet that references a player object work in both Desktop and XR mode without any major reconfiguration.

The hope was that a merged Player Rig would also allow a single build to run both XR and Desktop mode. It turned out to be a more complex issue than first anticipated. Most of the objects in the application attempt to configure themselves on startup, searching for objects they require to function and setting their initial state. After the application has started, they will not attempt to search for objects again. If XR mode is initialised after the start of the application, most objects that require XR interactions will not find the XR components and will fail to function. Switching from Desktop mode to XR mode causes additional issues with the 2D UI preventing the XR controllers from interacting with the tablet. Switching to Desktop mode after initialising XR had no issues, making it appear as if a solution is not far off.

After finishing this solution, it was clear that it was not ideal. The main player object now has a large number of components, some of which have unclear dependencies on each other. The PlayerManager class especially grew in complexity without a major benefit in usability. Most likely the right choice would be to split the players again, and instead make a Player class that has the functionality required of both the desktop and immersive VR player object. This class could then have subclasses for immersive VR and desktop variants that has functionality specific for those platforms. Objects like the 360-degree video platform and tablet that do not have a need to differentiate between the variants could simply use the top-level Player class for its functionality. Objects that need specific functionality per player type like the crane driver's seat could instead check for which subclass of the Player class activated it and perform different actions based on that. This would likely require changes in the prefabs, meaning the player classes should likely become a prefab of their own to stay consistent between applications.

10.1.8 Event-based Desktop activation button

The desktop activation button was changed in order to avoid the button only working for the crane. Instead of tracing a line in the direction the player was looking to find the crane seat, pressing the activation button on the keyboard would check every object close to the player whether it had an ActivationEvent component on it. If such an object was found, the events registered to the closest object to the player would trigger. Using the proximity to the player makes it easier to use than having to aim at objects in order to activate them. It would not be obvious that aiming at objects is required or possible without a good feedback system, and it would also be difficult for users that lack experience in first-person games and users using a laptop trackpad.

The ActivationEvent component was made for this purpose with inspiration from the teleport

surfaces that the XR Interaction Toolkit used. The component has only one configurable field, which is a list of events that should happen when the player activates it. These fields are universal and common in different places in Unity, allowing the developer to specify arbitrary things that should happen when the event is triggered. It could set the property of an object in order to move it or change its behaviour, or it could trigger a certain function in code. Attaching one such component to the crane's seat would allow the desktop player to enter and exit the seat. Attaching another to the 360-degree video prefab would allow the desktop player to watch and stop watching each player's respective video.

10.1.9 Adding a manual for desktop players

An explanation of the desktop controls was not provided to desktop players. The control scheme is not intuitive, and there are many buttons across the keyboard with different functionality. To at least provide some explanation to players, a manual was added explaining what each keyboard button does.

The player will see this manual when they start the game. They can open and close this screen by pressing the button that toggles the manual. The same screen will appear when they enter the crane driver's seat, except showing the crane's controls instead of the character's. Dumping the controls on the player in this way is not ideal. The controls should ideally be introduced to the player during play in intuitive steps, rather than all at one time. This solution is more of a quick stopgap to provide some documentation rather than a solution that should be used long-term. It was adapted from a similar solution in the Lager VR job taste, only changing the trigger opening the manual to use the InputSystem rather than polling keyboard buttons directly.

10.1.10 Rearranging the video platforms

The previous arrangement of video platforms being adjacent to each other seemed to overwhelm the players. Many players had asked which video to start watching and which ones were important, and many other players had issues watching them in the right order. Some would start watching them left-to-right, while others would start watching them right-to-left, meaning there was no good way to place them. The tablets were originally placed together because it would be difficult to navigate the player to video platforms if they were strewn all over the application, especially for NAV personnel with less experience in the application. After implementing the tablet, it would be possible to have the game navigate the user automatically to videos, meaning that arrangement was no longer necessary. The videos could be put in different positions on the harbour.

This had two benefits. Firstly, the videos could be put in natural positions related to their importance and the content of the video. The Health and Safety video could be put adjacent to the player's initial position and next to the building. The video showing the ground crew could be put at ground level, not too far from the crane. The crane's tutorial was put halfway up the crane to give the player a respite from climbing. The crane driving session was put outside the crane driver's cabin to allow the player to see an example of crane driving before entering. The optional video showing the view from the harbour was put at ground level, slightly out of the player's way in case they feel like exploring. For an additional benefit, the player would be given time between each video to reflect, as recommended by (Johnson-Glenberg 2018) in Section 2.3.

Chapter 11

Results

This chapter starts by describing the Heisekran VR artifact in its final state, after all the development performed in different iterations. It proceeds to present the data gathered during the iterations and on the final product. Both quantitative data from questionnaires and performance benchmarks and qualitative data from user evaluations are summarised.

11.1 Video of the application

A video showcasing the final application can be seen by opening the following URL. If the URL is no longer valid, contact IMTEL or the author for a copy of the video.

https://youtu.be/ijF93XrwbQ4

11.2 The final Heisekran VR application

This section summarises the final Heisekran VR application without going into details about how different parts of the application were made. Sections with more detail will be referred to as changes are mentioned.

The application features a virtual harbour inspired by the harbour at Trondheim Havn, seen in Figure 11.1. The player starts in the middle of the harbour, in the location denoted by a green cross. The crane is the centrepiece of the harbour, towering over the rest of the scene. A group of containers is stacked on one side of the crane, and a container ship is docked to the harbour on the opposite side. The crane is placed onto a set of rails that cover the length of the far side of the harbour. A group of buildings are located at the end of the rails with a pair of trucks carrying containers parked in front of them.

Once the player has gotten their bearings, they can pull out their virtual Immersive Job Taste tablet (see subsection 10.1.2). This tablet is the player's main source of information while playing the application. It gives them access to a list of tasks to perform on the workplace, explains the important skills of the profession, and provides a help page with further explanations about the application. The player can view further information about the tasks and skills, and performing tasks grants them points to the relevant skills, showing the player their progression as they play through the application. Tasks consist of multiple sub-tasks to perform in sequence. The player can select a task to focus on, which causes the tablet to guide the player towards their next sub-task

using arrows that move along the ground, as seen in Figure 11.2. Once the player has finished a sub-task, the tablet will guide the player towards the next one (see subsection 10.1.5).

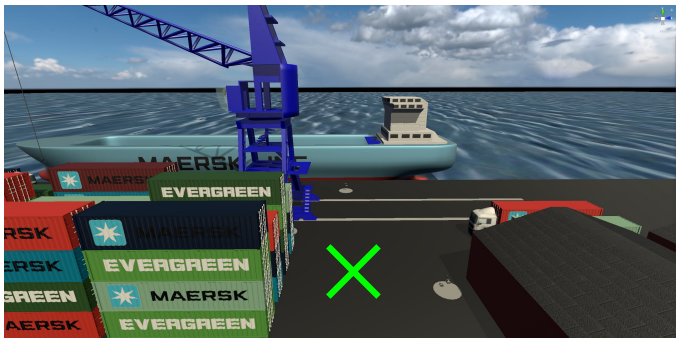


Figure 11.1: A picture of the harbour in the final version of the Heisekran VR application.

The application has five video platforms scattered around the area, one of which can be seen in Figure 11.3. Each video platform has a floating VR HMD in the centre, with the title of the video floating above it in huge letters (see subsection 9.1.1). Interacting with the floating HMD plays its respective 360-degree video. The virtual harbour is hidden from the player while the video is playing, replaced with 360-degree footage recorded at Trondheim Havn. Only the floating HMD follows the player from the virtual world. Interacting with the HMD again stops playing the video and puts the player back onto the platform on the harbour. Viewing these videos is part of the tasks the tablet gives the player, and it will help guide the player from platform to platform in an intuitive way (see subsection 10.1.10)

Figure 11.2: Navigational arrows guiding the player to their next video.

The videos are designed to introduce the player to the workplace and tasks performed there, as well as teach them the necessary skills and requirements to perform the tasks modelled in the application. The first video platform is located next to the building showing a student equipping the safety equipment required to enter the workplace, namely a helmet, safety boots, and high visibility clothing. The second platform is located next to the crane at the ground level, showing the ground crew at Trondheim Havn receiving cargo from the crane and driving it away using a forklift. The third one is optional, located near the ship at ground level. It shows a worker and a student at another location in the harbour, with the crane lifting cargo out of the ship on the horizon. The fourth video is mid-way up the crane, showing the crane driver inside the driver's cabin. The crane driver explains how to drive the crane using the control sticks found in the cabin. The fifth and final video is found at the top of the crane, next to the driver's cabin. It shows the crane driver lift a container out of the ship and place it on a truck with the help of the ground crew.

Figure 11.3: A video platform in the final Heisekran VR application

The player can easily climb the staircase (see subsection 6.1.1) to reach the top of the crane. Once there, they can enter the crane's cabin and sit down on the seat to start controlling the crane. The inside of the cabin can be seen in Figure 11.4. The crane's beam can lift and lower to move the hook closer and further away from the crane. The crane's house can be rotated to move the hook sideways. The wire hanging from the edge of the crane's beam can be retracted and extended in order to lift and lower the hook, respectively. This way, the crane can lift and move objects attached to the hook. The crane can also drive alongside the rails in order to get closer to objects it needs to lift.

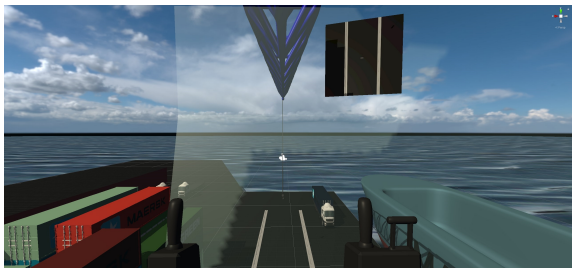


Figure 11.4: The crane driver's view from the driver's seat inside the cabin.

When the hook is moved towards a container, a rope will appear around the container as seen in Figure 11.5, signifying to the player how to lift it. Once the crane's hook is centred on the rope, the player can lift the hook to automatically attach it to the rope and lift the container with it (see subsection 9.1.4). The easiest way to align the hook with the rope is by using the screen present in the cabin as seen in Figure 11.4 (see subsection 9.1.2 for details). It shows a top-down view through a camera mounted at the tip of the crane's beam. Using the screen and the view from the cabin, the crane driver can get an overview of the hook and its surroundings.

Figure 11.5: The crane about to pick up a container from a truck. The rope fades in when the hook nears the container, showing the player how to lift it.

Once a container is lifted, it is securely fastened to the container's hook. Just like lifting the container, detaching it is also automatic. The player can simply place the container down on a surface, and it will be freed from the container's hook (see subsection 10.1.1).

The first task given to the player is lifting any container from the stack close to the crane and placing it within the ship. This can be done without driving the crane along the rails, as the container and ship are close by. Their second task is a challenge that asks the player to lift a container from a truck and place it inside the ship. Finishing this task requires the player to drive the crane along the rails, as the trucks are out of reach of the crane from the position it starts in. Some trucks are parked on each end of the rails, so either side will work to finish the tasks. Once all the tasks are done, the player can check their tablet to find all their tasks are done.

The application runs on tethered VR HMDs, standalone VR HMDs (see subsection 7.1.6) and desktops (see subsection 8.1.1) from a single development project. The supported platforms can be seen in Table 11.1. The controls for tethered and standalone VR HMDs are identical and can be seen in subsection 11.2.1. The desktop controls are described in subsection 11.2.2

Application	**Framework**	**Platform support**				
		Oculus(standalone)	Oculus	Steam	WMR	Desktop
FiskeVR	SteamVR	–	✓	✓	✓	–
Wind Turbine VR	VRTK	–	✓	✓	✓	–
Blikkenslager VR	SteamVR	–	✓	✓	✓	–
Blikkenslager VR*	Unity XR	✓	✓	–	–	–
Lager VR	Unity XR	✓	✓	–	–	–
Lager VR*	–	–	–	–	–	✓
Career Labs VR	?	–	✓	✓	–	–
Heisekran VR	Unity XR	✓	✓	–	✓	✓

Table 11.1: Comparison of related applications. Shows which platforms they are compatible with. Most applications are Immersive Job Tastes, with exception of the Career Labs VR application.
* Modified variant of original that needs to be maintained separately

11.2.1 Immersive VR Controls

The immersive VR variant of Heisekran VR is played using two immersive VR controllers, one for each hand. The control scheme is illustrated in Figure 4.6. The player can move across the harbor by teleporting. This is done by pushing the thumbstick of a controller inward until it clicks (see subsection 6.1.2). This creates an arcing line, projected out of the player's hand in the direction they are pointing. If the player aims at a location they are allowed to teleport to, a circular reticle is shown where the line hits and the line itself is coloured white. This circle shows the destination the player will teleport to. When the player proceeds to release the thumbstick, their screen will flash black and the player will be instantly moved to their destination. If the player aims somewhere they are not allowed to go, like the roof of a building or the water around the dock, the line will be coloured red and the player will not move when releasing the thumbstick. The player can also move physically across their room but will be limited by the size of their play area.

The player can bring up the tablet by pressing the lower face button. For most controllers, this is the X button for the left controller or the A button for the right controller. The exceptions are HMDs using the Windows Mixed Reality platform, which only have one face button on their controllers (see subsection 10.1.3. Bringing up the tablet swaps the player's hands for models of the VR controllers with straight lasers pointing out of them (see subsection 10.1.4). Pointing the laser at buttons on the tablet's User Interface and pressing the trigger activates the buttons on the tablet. This way, the player can navigate the tablet and choose which tasks to perform.

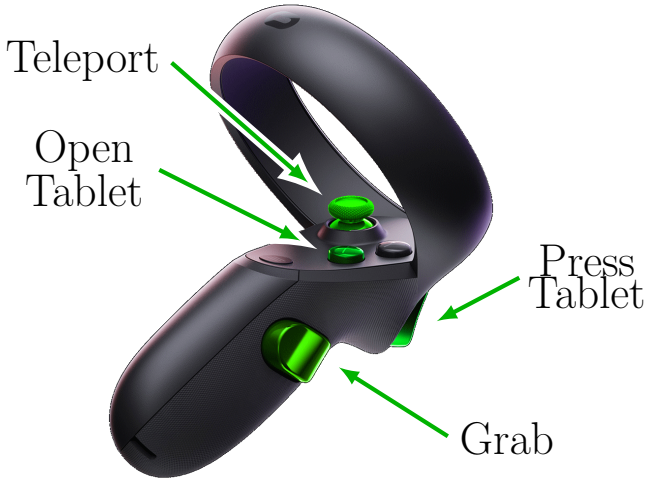


Figure 11.6: The Immersive VR controls for the final release of Heisekran VR. Illustrated on the left Oculus Quest 1 controller. The right controller has an identical control scheme. Controller taken from (Kerman 2021)

The 360-degree video platforms will play a video while the virtual HMD associated with the video is being held. The player can hold an HMD by touching it with their hand and pressing the grip button on the side of the controller. The video will continue playing until the player lets go of the HMD by releasing the grip button.

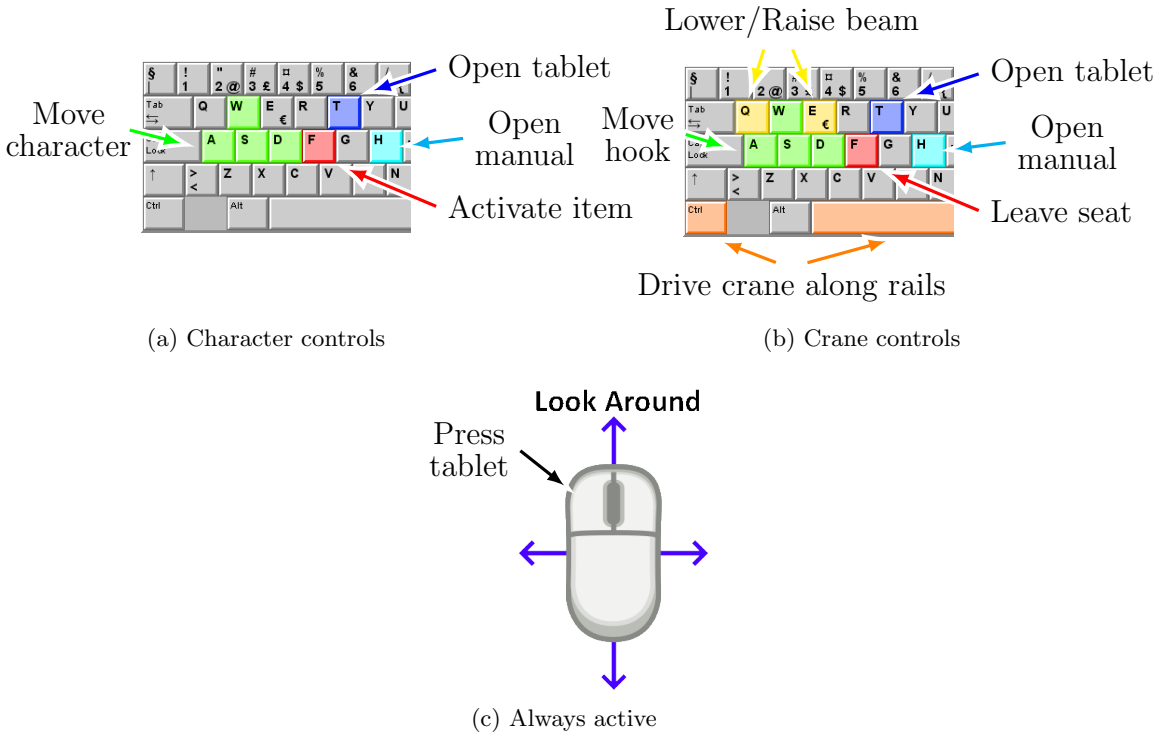
Climbing up the stairs is done by teleporting up the steps. Once at the top, teleporting into the control room will place the player into the crane driver's seat. From there, the player has access to the three control sticks controlling the crane. The player holds on to a stick by hovering a hand over it and pressing the grip button on the side of the controller. Once the player has gripped a joystick, the player can move the joystick by moving their hand as if they were holding a real joystick (see subsection 6.1.2). The left joystick is responsible for moving the hook along the ground. Moving it along the forward axis lowers and lifts the beam, and moving the joystick sideways rotates the crane in the same direction. This way, the hook moves in the same direction as the left joystick. The right joystick is responsible for the height of the hook, lifting the hook when pulled backwards, and lowering the hook when pushed forwards. The small lever on the right drives the crane along the rails, forward to go towards the tip of the boat, backwards to drive towards the back.

11.2.2 Desktop Controls

The player moves across the harbour in a continuous motion using a keyboard & mouse (or trackpad) as illustrated in Figure 11.7. Pressing the WASD keys moves the player based on the direction they are looking without turning the viewpoint, as standard first-person 3D game convention. For those unfamiliar, W moves the player forward, A moves the player left, S moves the player backwards, and D moves the player rightwards. Moving the mouse changes the direction the player's viewpoint is looking. The player can freely choose where to look by moving the mouse upwards and downwards, and turn around by moving the mouse left and right.

The player starts the game with an instruction manual covering the screen, explaining the game's controls (see Section 10.1.9). The player can close and re-open the manual using the H key on their keyboard. The tablet can be brought out by pressing the T key (see subsection 10.1.2). A mouse cursor appears when the tablet is opened, allowing the player to move their mouse and click the buttons on the tablet to use it. When the player clicks the cross on the tablet, the tablet is put down. The video platforms can be viewed when the player is adjacent to them by pressing the universal activation key, F (see subsection 10.1.8). The video will continue playing until the player presses the activation button again. The player cannot move while the video, manual or tablet

are visible, to prevent them from unintentionally moving their character while their vision of the game world is obscured.



(a) Character controls

(b) Crane controls

(c) Always active

Figure 11.7: How to control the player character and crane in desktop mode.

The player will automatically climb the crane's steps when moving towards them. Once the player is close to the crane driver's seat, they can press the activation key F to sit down and activate the crane. Once in the seat, another page of the manual opens on the screen, explaining the controls of the crane. While seated, the player loses the ability to move their character with WASD. Instead, WASD moves the hook along the ground, similar to the left control stick in VR mode. Four additional keys are enabled while the crane is active. The Q and E keys lower and lift the hook, respectively. Spacebar and Ctrl allow the crane to drive along the rails, Spacebar moving the crane towards the front of the ship, and Ctrl moving the crane towards the back. The tablet and manual can be opened as normal, using the T and H key respectively. Once the player is done driving the crane, they can press F to leave the seat and regain control of their character.

11.3 Performance

The performance benchmark was performed by opening the tablet view, selecting the first task, then performing the sub-tasks in order until the task is finished. The tablet would then be opened again, and the second task would be selected and performed. Once both tasks are performed, the benchmark is stopped. Performance benchmarks were performed in desktop mode using the CapFrameX software (CXWorld 2021). One benchmark was performed on the desktop variant using a high-end desktop computer with an AMD RX 3700X CPU, 32GB of memory and NVidia GTX 1080 GPU, and ran at a 2560x1440 resolution. Another benchmark was performed on the desktop variant running on a low-end office laptop with an Intel Core i5-6200U CPU, 8GB of memory and no dedicated GPU at a 1366x768 resolution. The last benchmark was performed on the standalone variant running on an Oculus Quest 2 using the OVR Metrics Tool (Oculus 2021b). The results of the benchmark can be seen in Table 11.2, with further data in Appendix A.

A performance benchmark was attempted on the Oculus Dashboard in order to get a baseline for performance, but no way to export the numerical data was found. A picture was taken of the tool

instead, which can be seen in Figure 11.8.

Device	Target FPS	Average FPS	Time spent below target	Time spent below half of target
High-end Desktop	60.0	1250.0	0.0%	0.0%
Low-end Laptop	60.0	91.6	16.5%	0.9%
Oculus Quest 2	72.0	68.4	34.9%	0.0%

Table 11.2: The performance of two systems running the Desktop variant and the Oculus Quest 2 running the VR variant

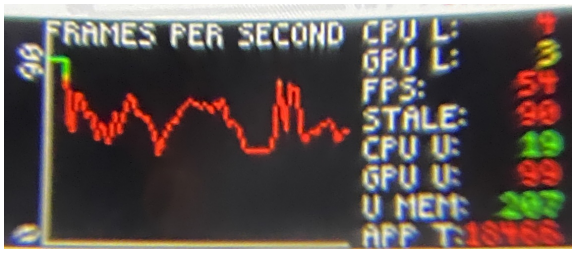


Figure 11.8: A performance graph made by the OVRMetrics tool showing the average FPS inside the Oculus Dashboard.

11.4 Requirements

A set of requirements were created for the application in Section 5.1. Their status at the end of development can be seen in Table 11.3.

ID	Requirement	Complete?
R1	The application should use Unity XR and the XR Interaction Toolkit for immersive VR	✓
R2	The application should support the Oculus Quest standalone HMD	✓
R3	All of the application's tasks and features should be playable in desktop mode without immersive VR	✓
R4	Both tethered and standalone immersive VR and desktop play should be supported by the same application	✓
R5	The players should feel in control of the application's control scheme	✓
R6	The application should minimise the occurrence of cybersickness	✓
R7	The application should provide clear goals to the player to complete	✓
R8	The application should guide the player to the goals	✓
R9	The application should provide the player with points or other feedback for completing their goals	*
R10	The application should run at a minimum of 60 frames per second in desktop mode	✓
R11	The application should run at minimum the refresh rate of the VR HMDs in immersive VR mode	*
R12	The application should provide accurate information and tasks from the profession	✓
R13	The application should implement another work task from the profession	–

Table 11.3: The resulting state of the initial requirements set for the Heisekran VR application.

11.5 User testing, NAV: 2021-05-07

Several young job seekers were invited to a career guidance session by NAV personnel. They were given access to the entire Job Taste catalog, and would choose a couple of professions that they were most interested in. Each young job seeker was scheduled separately, so only one would be present at a time. Three young job seekers ended up participating. The NAV personnel had a gaming laptop with a tethered HP Reverb G1 available, as well as multiple Oculus Quest 1 and an Oculus Quest 2. However, most of their Oculus devices were nonfunctional. Some would not start because of issues connecting to Facebook's servers to verify the accounts connected to the devices. Installing applications through Yrkeskatalog failed for unknown reasons, and the details surrounding the failed installation were reported to Yrkeskatalog's developer. One Quest 1 and one Quest 2 was finally made functional using a personal laptop.

The NAV personnel played through the Heisekran VR application once to learn the ropes, with guidance from the developer. Afterwards, they were responsible for guiding the young job seekers through the applications while the developer observed in the background. Due to the NAV personnel's familiarity with the tethered solution, they opted to use the HP Reverb initially. The second user had issues with the application crashing during play for no discernible reason, so the third user was given a Oculus Quest 1 instead. Neither NAV nor the job seekers were interested in trying the desktop solution.

O3: The tethered VR solution crashed multiple times for one player, while others had no issues.

O4: The players without previous VR experience struggled with the controls. They needed multiple reminders of the control scheme to manage to get through the application, even after playing another VR application first.

O5: The players would grab the video player and immediately let go the first time, seeing the video pop in and out quickly. They would then understand they have to hold the device. This feature turned out to teach the players how the video platforms work using a small failure, as recommended by (Johnson-Glenberg 2018).

O6: One player did not realise the video in the crane showing the crane driver lifting a container existed. They went straight to the crane's control room once they reached the top.

O7: Players would struggle to aim at the correct buttons on the tablet, missing the buttons or pressing the wrong ones. This was especially obvious with the smaller "Back" and "Close" buttons in the header of the tablet UI.

O8: The tablet following the player's face was unintuitive. Players would look up or down attempting to read text on the tablet, but soon realised it would just follow. This is especially bad since the focal point in a VR HMD is very small towards the centre of the screen, making the edges of the tablet blurry.

O9: The players without previous VR experience had some issues steering the crane and would forget which control stick did what. The player that chose to lift the challenge container had no issues lifting their second container, however.

O10: One player had some previous experience in VR games, and had no issues with the controls or steering the crane. On the other hand, they did not go to the video platforms outside of the crane. It is unclear whether they missed the platforms or chose to ignore them.

11.6 User testing: NAV users

NAV invited several unemployed youths to try out their Virtual Reality applications on 18th June 2021. An Oculus Quest 1, an Oculus Quest 2 and a HP Reverb were set up in different rooms to allow multiple people to try applications simultaneously. The author took charge of the two standalone devices, while NAV personnel took care of user tests using the HP Reverb. The author

guided three users using standalone devices and helped another user on the HP Reverb afterwards. The NAV personnel guided 2 users on the HP Reverb on their own. At the end, they were invited to fill out the questionnaire on a laptop.

One tester had poor self-efficacy and lacked the confidence to explore the application. Most of the time, they asked for confirmation before pressing any buttons on the controller. They opened the tablet a multitude of times by accident, and initially selected the second *Challenge* task instead of the *Tutorial* one. Even after selecting the correct task, they did not realise they could climb the crane's stairs. Nevertheless, they were happy with the experience after viewing the ground-level videos.

Another tester got a task selected in a tablet and proceeded to head straight up to the crane while ignoring the videos. It is probable that they also selected the wrong task on the tablet. They ultimately failed to manage to lift a container for unknown reasons before giving up on the application due to fatigue from the VR glasses.

One tester had previous VR experience and took the controls after some explanation. They watched the 360 videos on the way up and managed to learn the crane's controls quickly. They skipped the easy task and went right for the difficult one by picking up a container from a truck. They hit the edge of the ship while moving the container but managed to place it down.

The last tester had extensive experience with video games, and had tried some job tastes previously. They seemed confused about the tablet following their head, and commented it was very close to their face. They watched all the videos that the game navigated them to. Once in the crane's seat, they quickly figured out the crane's controls on their own and turned the crane towards the ship. Due to the video from the crane driver's point of view, they expected their task to also be lifting containers ashore from the ship, rather than the other way around. While driving the crane, they commented that people with less game experience might struggle to learn the game's controls if they have tried the other apps, as older apps use a different control scheme from the new ones. They opened the tablet again after placing a container into the ship to see their progress and start the other task. The tester came with a suggestion to use the empty space in the top left of the crane to host a task list similar to the tablet so the player could see their progress without having to pull out the awkward tablet all the time.

11.7 Interview: NAV Career Counsellor

After the user test in 11.6, a short semi-structured interview was performed with the NAV career counsellor that was present.

They sadly did not have much experience with the job taste system. They had only seen videos and seen a few users play through the application. They had not tried it themselves and had not seen any users play through the Heisekran VR application, making it impossible for them to comment on the application itself.

The NAV counsellor did believe that the job taste system provided excellent opportunities for young people to try out different jobs, especially if they "cannot get out to the ordinary [school or work life]". They identify that many young people need a job that is different than the ordinary jobs of working as a store clerk, and the job taste system could provide an easy way for them to see what they like and what they could succeed in.

When asked about deployment of the job tastes to other NAV offices, they said it would provide a great way for people to test things out. When asked about deploying the applications to high schools, they responded that it was a good idea, as "[the students] are already in a process to find out which direction they want to go".

They believe that NAV offices would be able to use the applications on their own while working with their users if sufficient training is provided beforehand. If NAV personnel were taught how to use the applications and gained sufficient understanding about their purpose, they believe it would be useful to have more NAV personnel capable of using the applications.

11.8 Interview: Crane driver

Trondheim Havn were invited to participate in user testing 3rd of June 2021, to get feedback from professionals in the field. The concept of Immersive Job Taste was explained beforehand, so they could understand the scope and intended use of the crane application. Their operations manager and an experienced crane driver participated in the user test, before taking part of a semi-structured interview. The interview was performed in Norwegian, so any quotes are translated to the author's best ability and paraphrased slightly to keep their original intent.

The user test had to be a remote meeting, as COVID restrictions prevented them from attending any physical meetings. This prevented them from testing the VR variant of the application, as they had no access to VR equipment or anyone capable of setting up the application. They were sent the desktop variant of Heisekran VR and attempted to play it before the meeting but failed to get it to start due to insufficient instructions. As a last resort, they were shown the video demonstrating the Heisekran application and based their feedback from it. This meant they were unable to evaluate the usability aspects of the application.

Their main takeaway was that the application presented the profession reasonably accurately, but in a simplified and more appealing fashion. "That is what we do, but it was a sort of computer game version of it". The application shows the player that they must equip necessary safety equipment, before sending them straight up the crane. The actual profession requires more time spent on maintenance and preparation. Every day before going onto the crane, they must climb the crane and inspect it for faults, making sure it is safe to use. Additionally, they regularly have to lubricate certain parts of the crane to keep it operational, as well as perform other unspecified maintenance. When their upcoming work is non-standard, they also have to fill out a "Safe Work Analysis" form to ensure their tasks can be performed safely and that all precautions have been considered. Most of these steps concern the health and safety of themselves and the workers on the ground. They consider many of these tasks to be boring, while the crane driving is the fun part. "That's not what they want [to do], you know. When they first arrive, what they want is to try [the crane]. They want to drive."

The simplification extends to the crane driving itself. "In reality it's often slightly clumsy and slightly tight and slightly gross". The application gives the player a wide open area to maneuver, and an entirely empty ship to stack containers into. The player only need to load a couple of containers into the ship at most, limiting how precise they have to be. In practice, cargo ships want to maximise the amount of goods they carry on each trip, so the cargo is packed as tightly as possible inside the ship. The application also only tasks the player with lifting regular containers, which are easy to deal with and have a standardised lifting attachment. Real cargo occurs in all shapes and sizes and amounts, which requires more thinking and effort to fasten securely to the crane. There are various straps, wires and other equipment for lifting objects like steel and rebar. Other bulk goods like sand would require a bucket attached to the crane. Fastening the cargo and changing the equipment generally falls to the ground crew while the crane driver observes and assists them as possible from the crane. Nevertheless, it is important for the crane driver to know everything about strapping and hooking cargo. The people from Trondheim Havn describe strapping and hooking cargo correctly to be one of the central aspects of the profession and one of the first things crane drivers are taught. While crane drivers rarely perform the strapping and hooking themselves, the responsibility falls to them to ensure it was performed correctly. They must pay constant attention to ensure that the ground crew has not made any mistakes, as they take the blame if anything goes wrong.

To that end, another skill that is taught immediately is communication with the ground crew and radio contact. Crane drivers must know how to communicate and work together with the ground crew, as they are reliant on their coordination. Not only does the ground crew attach and detach cargo from the crane. They also have the responsibility for driving the cargo to and from the crane using trucks and forklifts, and for relaying their observations to the crane driver. Even with auxiliary camera feeds, the crane driver would struggle to see exactly where to move cargo when it is in a tight spot inside a ship or on top of a truck. Their eyes on the ground have a much better perspective to communicate exactly where to move the cargo to ensure it has clearance and can be placed correctly.

Both of those skills are missing from the application. It only has the container to lift, with no variety of cargo to lift or equipment to lift with. The strapping and hooking process is skipped entirely, as the container automatically hooks on when the crane attempts to lift it. The application also lacks any element of communication, as the player is the only one participating in the game and no ground crew members are simulated. Despite that, the personnel from Trondheim Havn were positive to the application. While they would want tasks added related to maintenance and paperwork, they were also worried that adding too many boring aspects would not motivate the player into seeking the job. They did not find strapping and hooking to be relevant to a crane driver Immersive Job Taste, as crane drivers would rarely perform it themselves. "[The hooking] is where you begin. [The ground crew] stop there. Once you become a crane driver, you have gone a step further, in a way." It would only be relevant if the focus of the application was changed away from crane driving and onto more harbour work. Adding some form of communication was never brought up, perhaps due to its perceived difficulty or usefulness.

The workers at Trondheim Havn recommended adding different types of cranes if the application intends to continue to focus on crane driving. The application only shows the type of gantry crane that they had at Trondheim Havn, but they note the experience differs greatly between different types of cranes. Mobile cranes as seen in Figure 11.9a are small. The crane driver generally sits at the bottom looking up. When somebody wants something from you, they can just walk up to you and knock at your door. For gantry cranes like at Trondheim Havn, the crane driver sits too high up for anyone from the ground crew to speak to them directly. Tower cranes used on construction sites like the ones seen in Figure 11.9b are so tall that "one sits up in the clouds for the most part, and everyone else are like small ants on the ground". Gantry cranes like in Figure 11.9c are common in larger harbours and provide yet another experience. They mention that modelling the cranes might not be necessary. Simply recording some 360 videos from the crane driver's point of view inside some other cranes would do if modelling the cranes is prohibitively time-consuming. Players would be able to experience the other cranes and know they exist, even if they could not try driving them.

(a) A mobile crane. Taken from (G 2021).

(b) A tower crane. Taken from (Cheers. 2021).

(c) A different kind of gantry crane. Taken from (M.Minderhoud 2021).

Figure 11.9: A collection of cranes that differ from the one at Trondheim Havn.

When asked how likely it would be that they would use the application during recruitment, they answered that they do not tend to spend a lot of time looking for workers. They partner with another organisation that provides training for apprentices and refer them to their harbor, and thus rarely spend time recruiting. On the other hand, they mention that a lot of their crane drivers are taught internally, and that they might wish to bring in students from middle and high school to experience crane driving and see if they are compatible with the profession. In these cases, an application like Heisekran VR could be useful to them. The application has enough elements of the real job, from the crane controls, the pile of containers, the ship, the vehicles and the sea on the horizon to give a taste of what the profession is like. They also believe it is interesting and informative enough to start young job seekers on the path of the profession where they can start learning more about it.

They application did not misrepresent the profession or make any egregious mistakes. "One would have to start being pedantic, and that has no purpose in this context." They mention that one

would not lift a container using a hook, but it was not important at the level the application is at.

Trondheim Havn does not have any applications used for training. "Our simulation is three-dimensional and sits at the harbour." The previously mentioned partner that trains apprentices has a thorough crane simulation, however, "and that one is beefy. There you can see how it's really done." They are not aware of other applications designed to train crane drivers in the area but mention that there might be some further south in the country. Other relevant applications that are used for training in this field are simulations by a traffic school intended to train truck drivers, and another one designed to teach how to drive a ship.

11.9 User test summary

Table 11.4 provides an overview of user tests performed with the application.

Summarising the user tests, most users did not struggle with cybersickness (as defined in Section 2.1.2). Some users struggled with cybersickness while watching the 360-degree videos or climbing up the crane's stairs. None suffered cybersickness while controlling the crane.

Most users needed an explanation of the VR controls in order to use the application. All users needed to have the tablet pointed out to them for the iterations containing a tablet. Most users needed guidance to understand their goals and where to go before the tablet. The guidance required after adding the tablet was minimal.

Most users struggled to use the crane's sticks in beginning iterations. Most users would fail to actuate the sticks properly until they were made congruent. Most users afterwards were able to actuate the sticks correctly, but still needed explanations on how to use the sticks and still struggled to pick up containers. This was no longer observed after haptics were added to the application.

Date	Users	Count	Section
24th September 2020	Students	10±	5.3
30th September 2020	NAV users	5	6.2
5th November 2020	Students	5	7.2
17-18 November 2020	NAV Personnel	4	8.2
2nd December 2020	NAV users	6	8.3
3rd February 2021	Students	8	9.2
7th May 2021	NAV users	3	11.5
3rd June 2021	Crane professionals	2	11.8
18th June 2021	NAV users	6	11.6
18th June 2021	NAV personnel	1	11.7

Table 11.4: An overview of the user tests performed with the Heisekran VR application. Participants were only counted if they tried the application. A count was not performed for the first user test and is therefore an estimate based off of memory.

11.10 Questionnaire

This table provides the results for the questionnaire across the user tests. Note that the questionnaire did not solely focus on the Heisekran VR application. It covered all applications that were tried during the user test. A total of sixteen respondents answered the questionnaire, but five responses were omitted as they did not try the Heisekran VR application. Note also that the questions were originally given in Norwegian but have been translated for this thesis.

Table 11.5: A table of questionnaire answers across all user tests.

Background			
ID	**Question**	**Mean**	**Median**
Q1	How much experience do you have with videogames?	3,89	3
Q2	How much experience do you have with VR/AR?	2	2

Applications			
ID	**Question**	**Mean**	**Median**
Q3	I liked using the Crane app with VR glasses.	4,2	4
Q4	The Crane app was easy to use with VR glasses.	4,2	4
Q5	I liked using the Crane app on desktop (without VR glasses).	4	5
Q6	The Crane app was easy to use on desktop (without VR glasses).	4,33	4

Usability			
ID	**Question**	**Mean**	**Median**
Q7	I think that I would like to use the job taste apps frequently.	2,22	2
Q8	I found the job taste apps unnecessarily complex.	2,11	2
Q9	I thought the job taste apps was easy to use.	3,88	4
Q10	I think that I would need the support of a counsellor to be able to use the job taste apps.	2.55	2
Q11	I found the various functions of the job taste apps to be well integrated.	3.88	4
Q12	I thought there was too much inconsistency in the job taste apps.	2,33	2
Q13	I would imagine that most people would learn to use the job taste apps very quickly.	4	4
Q14	I found the job taste apps very cumbersome to use.	1,5	2
Q15	I felt very confident using the job taste apps.	3,77	4
Q16	I needed to learn a lot of things before I could get going with the job taste apps.	1,56	1

Job Taste			
ID	**Question**	**Mean**	**Median**
Q17	I thought the cooperation between me and my counsellor went well when it came to the job taste apps.	4,13	4
Q18	The tasks in the job taste apps are realistic.	4	4
Q19	I feel a sense of mastery of the tasks in the job taste apps.	3,45	4
Q20	I feel that I experience the professions for real.	3	3
Q21	The counsellor mattered a lot for my enjoyment of the job taste apps.	3,25	3,5
Q22	I get to try out my skills in the job taste apps.	3,18	3
Q23	I learn about different professions through job taste apps.	4,27	4
Q24	I get insight into my own vocational competence through job taste apps.	3,33	3
Q25	I get insight into my own resourcefulness through job taste apps.	3.1	3
Q26	I learn about the requirements of different professions through job taste apps.	3,91	4
Q27	I learn a lot about myself through job taste apps.	2,91	3
Q28	I have gotten insight about what professions fit or do not fit for me.	3,36	3

Job Taste			
ID	**Question**	**Mean**	**Median**
Q29	I have learned a bit about what it means to work through job taste apps.	3,81	4
Q30	I have gotten insight about my possibilities on the job market through job taste apps.	3,91	4
Q31	I have learned it is important to explore possibilities in the working life (after trying job taste apps.)	3.81	4
Q32	The job taste apps have inspired me to try out the working life.	3,18	3
Q33	The job taste apps have made me believe I will be able to master the working life.	2,82	3
Q34	I have gotten to know myself better.	2,63	3
Q35	The conversation with my counsellor after trying the job taste apps has given me the courage and determination to find a job.	3,13	3
Q36	I have changed a lot after trying the job taste apps.	2,45	3
Q37	My counsellor and the job taste apps have made me think positively and believe anything is possible for me.	3,13	3
Q38	I think positively about the future (after trying the job taste apps.)	3,45	3,5
Q39	I have more faith that I will master the working life (after trying the job taste apps.)	3,18	3
Q40	This kind of job taste apps should be part of NAV's offer to job seekers.	4	4
Q41	NAV needs more job taste apps for more professions.	4,82	5

Chapter 12

Discussion

The main contribution of this thesis is the Heisekran VR artefact described in Section 11.2. The Design & Creation process has given an *instantiation* as its output, meaning the artefact itself demonstrates the use of new technologies and methods for creating job tastes that could be applied for future development within the Virtual Internship project. A set of requirements were drafted based on the research questions and literature review. Most of these requirements were met with some caveats. Their status and completion are discussed in section 12.1. The circumstances surrounding user testing made it difficult to perform a thorough evaluation of the application as an Immersive Job Taste, but the application had positive reception in the user tests that could be performed. This will be further discussed in Section 12.2.1. The desktop variant was found to be useful during development, but with users having little enthusiasm for a desktop version, its value as an Immersive Job Taste is difficult to determine. The value of desktop variants will be further discussed in Section 12.2.2. Moving the application to Unity XR has allowed the application to support an unprecedented number of platforms compared to previous Immersive Job Tastes, but has also uncovered some drawbacks to the technology. One of those supported platforms is standalone HMDs, which show the potential of increasing the availability of immersive VR technology. However, attempting to use the devices during this thesis uncovered limitations preventing widespread deployment. The benefits and drawbacks of these technologies will be discussed in Section 12.2.3 The process of moving the application to Unity XR and its surrounding technologies has been documented, alongside the process of making an application compatible with standalone HMDs. This documentation alongside the artefact itself provides guidelines and solutions for developing new applications for the framework and updating old ones. Bypassing the use of VR technology altogether was also explored by creating a desktop variant of the application. How these changes were performed and how other developers could perform the same is discussed in 12.2.4.

12.1 Requirements

This section discusses the completion of the application's initial requirements. An overview of their completion can be seen in Table 11.3.

12.1.1 Accessibility

The application has improved its accessibility through supporting additional platforms compared to related work. The platforms supported by the application can be seen in Table 11.1. The application has successfully been ported to Unity XR, using the XR Interaction Toolkit for interactions with the game world. The framework made it simple to support standalone and tethered HMDs using the Oculus platform. With the addition of the Unity InputSystem, the application supports HMDs using the Windows Mixed Reality platform as well. Additionally, the application supports

running in desktop mode without using an immersive VR HMD at all. This covers requirements R1, R2, R3 and R4.

Comparing the application to closely related work shows that the application supports the largest number of platforms. Other applications within the Virtual Internship project had previously experimented with desktop and standalone variants, but these were modified copies of their original applications. Unlike those, the Heisekran VR application only needs one version of the application to be maintained in order to support all the platforms, avoiding duplicate work and the potential mistakes it can cause. Even compared to the higher budget Career Labs VR system, the Heisekran VR application comes out on top by supporting standalone VR, Windows Mixed Reality and Desktop play.

Comparing to other related work is not as simple as just comparing the available platforms, as most of those applications were either too early to use current HMD technologies or do not put much focus on the devices used and their deployment. Nevertheless, one may still note accessibility improvements granted by the Heisekran VR applications compared to the examples brought up in Section 2.5. The welding application described in (Fast et al. 2004) required a specialised VR setup with a custom controller built upon a welding tool attached to a 3DOF haptics device. The setup required additional sensor systems to track the head and controller in 6DOF for use in the application. This is cumbersome to prepare and likely expensive (see Section 2.2.2). Current HMD technologies provide a similar experience at a cheaper cost and easier accessibility as controllers and HMDs support 6DOF movement by default. The drawbacks are the lack of the 3DOF haptic feedback and the controllers not matching a welding tool entirely. The former is likely a worthwhile drawback as current controllers still have access to haptic vibrations without additional tooling. The latter is unlikely to be an issue if the movements are still congruent (see Section 2.3) as HMDs still appear effective for motor- and psychomotor skill training regardless (see Section 2.5).

The Heisekran VR application had one regression, being the loss of the Steam platform. Unity XR had no possibility of supporting SteamVR at the time of development, as described in Section 10.1.3. A Unity XR plugin supporting OpenXR has since been released, and SteamVR has started supporting applications using OpenXR (see Section 4.6.3). This means it should be possible for future applications to add support for the Steam platform if needed. Adding support for OpenXR should not be particularly challenging, being analogous to adding support for WMR (see Section 10.1.3) and its controllers (see Section 10.1.3).

12.1.2 Game Design

The application has had a good deal of improvements regarding game design. One of the most important aspects of a game is an intuitive control scheme that the player can understand, that provides feedback, and performs what the player intends to do (see Section 2.4). Without it, players would not feel in control of the game and would not engage with it at all (Brown and Cairns 2004). Conversely, a good control scheme supports the player's learning. This can be seen during the user tests. Players had no issue moving around the harbour, and after simplifying the staircase in Section 6.1.1, players generally had no issues climbing up the crane either. The 360-degree video prefab provides affordance to the player. While many players would fail to watch the video at first by instantly letting go of the video player, this is simply a low-impact failure that players should learn by, as recommended in Section 2.3. After failing once or twice, players would have no issue watching videos. Unlike the previous 360-degree video solution, they were not dependent on external guidance to exit the video when they were finished.

The crane's control sticks were improved in a similar manner. At the beginning, they would not respond in the way players expected, causing discomfort. Some users quit the game because of it, as Brown and Cairns warned. After making the controls more responsive and more accurate in Section 7.1.5, more users stuck around and eventually succeeded at steering the crane. The improved controls were easier to learn as the movement of the players hands were congruent with the effects they had on the stick, as described in Section 2.3. A good deal of users still struggled to master the controls due to the lack of proper feedback, often keeping the sticks held when they did not intend to, or not moving the sticks far enough to drive the crane effectively. Removing

the false feedback given by the sticks jumping when touched made users realise they needed to push a button to grab hold of them. The addition of haptic feedback both improved the players' mastery of the controls and reduced the amount of guidance needed, with most players learning the controls on their own. To this end, the application fulfils requirement R5.

For a game to be fun, it should provide the player with appropriately challenging goals, and the player should receive feedback when they progress (see Section 2.4). With the addition of the Immersive Job Taste tablet, the application provides the player with clear goals to perform. The goals give the players a natural progression through the application, and the optional challenge task allows the players to somewhat adjust the difficulty to their preference. The application guides the player towards their next goal, preventing them from being lost or unsure where to proceed, as recommended in Section 2.3. This covers requirements R7 and R8. When it comes to feedback from completing goals, the application is somewhat lacking. Opening the tablet shows whether tasks and sub-tasks are completed, and completed tasks give the user points in their respective skills. The issue is that the user needs to open the tablet to see it, as noted by a user in the testing session described in Section 11.6. The feedback is not immediate, like recommended by (Penelope Sweetser and Wyeth 2005). When the sub-tasks are related to watching 360-degree videos, the user gets some form of immediate feedback because the navigational arrows change their target. Once the tasks focus on crane driving, these navigational arrows are no longer used, meaning the user no longer receives any feedback. The aforementioned user suggested putting a task view on a surface in the crane's cabin, which would serve to both provide feedback and additional guidance. Additionally, sound- and visual effects could be used to provide feedback immediately upon completion of a task. The application covers requirement R9 in the strictest sense, but the potential for improvement should be noted.

One of the issues identified with the application was the lack of variety in work tasks to perform, noted in requirement R13. While the app did provide the player with goals as discussed above, there is a risk that the application has too few goals to make the player feel they attained some form of mastery (see Q19 & Q33 in Table 11.5). With few goals and no good way to adjust the difficulty of the tasks through guidance or otherwise, there is also a risk that the difficulty would not match the player's needed level of challenge. The application might be too difficult for some user testers which quit because they did not manage to succeed in picking up a container. Some of those can be attributed to the issue where containers would fall off of the hook (see Section 10.1.1), but others simply failed to figure out how the hook and crane worked. There should be an option for more guidance if the player wants an easier difficulty which shows step-by-step how to pick up and move a container. Such a solution could highlight which stick to grab and how far to move it using visual indicators. It could also show visual indications extending from the hook or container showing which direction it needs to move. More confident players could complete the assisted task quickly, or perhaps skip it. Conversely, it might be too easy to make users believe they could master the tasks in reality and participate in the workplace. Having extra tasks that are more difficult could help users gain a sense of mastery (see Section 2.4). Tasks could be added that require more precision, i.e. lifting or lowering a container through a small opening in a stack of containers or onto a truck. Tasks could also be added that require the player to cooperate, either with other players performing the hooking process and turning the containers like the real workplace, or a fake cooperation with a truck driver and non-player-characters that drive the truck and turn the containers visually.

Users rarely suffered cybersickness (see Section 2.1.2) while playing the application. Only a few cases where users suffered cybersickness were observed, and none of these happened while the player was driving the crane. Only a few cases were observed while players were climbing the stairs or watching 360-degree videos. This is surprising, as Section 2.1.2 identifies vection and especially acceleration as one of the lead causes of motion sickness, and plenty of vection and acceleration occurs while players steer the crane. It is possible that the crane's cabin works sufficiently as a rest frame in order to alleviate the effect of its movement. It is also possible that seated play alleviates cybersickness as some suggest. Identifying why the 360-degree videos cause cybersickness might be easier. The videos were recorded at a set height that is lower than most humans, which some users have linked to discomfort. Because the videos are two-dimensional and recorded from a set position, their content will remain stationary even as the user moves their head. Both fit under the postural instability theory for cybersickness described in aforementioned section. Explaining

why climbing the stairs caused discomfort is difficult. The scale of the stairs is large compared to the rest of the harbour, and the height between each step is unnaturally large. One possibility is that users perceive themselves to be shorter than they are used to when their frame of reference is the staircase, causing the same postural instability issues as for the 360-degree videos. The only reasonable alternative is that the users simply became dizzy due to constantly turning leftwards while climbing the stairs, as the movement while climbing the crane is otherwise identical to everywhere else on the harbour. Either way, with such a low occurrence of cybersickness, the application fulfils requirement R6.

While not explicitly covered by any requirements, the application does have issues with user guidance when it comes to the controls. The immersive VR variant of the application does not explain how to use the VR controls. Teleporting, bringing out the tablet, using the tablet and grabbing hold of objects is not explained within the application. The expectation is that the player has learned the controls from the Immersive Job Taste tutorial application described in Section 4.3, but that application does not support standalone devices as of writing. Even while playing tethered experiences, it is possible the player does not remember all the controls or realise what objects they can interact with. Conversely, the desktop experience explains the controls, but does it through a splash screen at the beginning which simply dumps all the buttons on the player and does not explain their significance. Section 2.4 explains that users should ideally be introduced to concepts one at a time to avoid overwhelming them, which neither variant succeeds in doing. The most egregious example is the tablet. Much of the application's guidance relies on the player using the tablet, yet there is no indication to the player that they should use it when they start the application. Without external guidance, it is unlikely any players would have found it during testing. Without the tablet, they would have struggled to find their way through the rest of the application. The user should ideally be introduced to the tablet immediately so they cannot miss it and prevent themselves from progressing in the application. The user should also see hints about how to interact with other objects in a context-based manner that provides feedback about the affordance of the objects, rather than having to remember a list of buttons. A solution could be to show the immersive VR players a controller in their hands and flashing the button they should press. Desktop players could see a flashing keyboard button on their screen instead. This way, the application could slowly introduce the controls and other concepts like the tablet.

12.1.3 Performance

The results of performance testing can be seen in Section 11.3. The high-end desktop system had no issue running the application without missing a single frame. The highest frame time was reported at 6ms while climbing the crane's stairs but was still significantly lower than the maximum 16.6ms required to hit the 60FPS target. The low-end laptop had a high average frame rate with some occasional stutters, commonly when loading the 360-degree videos and walking up the crane. Even with an average frame rate 50% above the target, around 16,5% of the low end's laptops frames were rendered below the target frame rate, showing the issue of using exclusively average frame rate as a metric to judge the performance of an Immersive Job Taste. Nevertheless, only 0.9% of the frames were rendered lower than half the target frame rate, meaning the game still performed well on a low-end laptop. The graph with the frame times is relatively consistent without many large spikes. Subjectively, low frame rates were not noticed while playing through the application. To this end, the application passes the performance required for the desktop variant in R10.

The Oculus Quest 2 is more difficult to evaluate as the standalone HMD would never render faster than its screen's refresh rate. The average FPS would inevitably end up lower than the target unless the application always met the target FPS. Over a third of the time spent inside the application was below the target frame rate. However, not a single frame took so long to render that reprojection could not take care of the delays. Subjectively, there were no noticeable drops in frame rate or stuttering while playing the application. A strange finding is that the performance of the game was still superior to the numbers reported inside the Oculus Dashboard, which is the menu from where applications are downloaded and started on the Oculus Quest. Subjectively, only occasional stutters were felt while using the Oculus Dashboard. The performance of both the Heisekran VR and Oculus Dashboard appears questionable but are qualitatively sufficient.

The performance of the Heisekran VR application appears sufficient as it is outperforming the first-party application. Further investigation should be performed to find the reasoning behind the seemingly low performance in the quantitative data.

12.1.4 Accuracy

The accuracy of the tasks the application models is deemed accurate enough to fulfil requirement R12. The reasoning is discussed in detail in section 12.2.1. As no other tasks were added during development, it did not fulfil requirement R13.

12.2 Research questions

This section covers this thesis' answers to the research questions.

12.2.1 RQ 1: Evaluating Heisekran VR's value as an Immersive Job Taste

Young job seekers

Analysing the questionnaire shown in Section 11.10 reveals that Heisekran VR was generally perceived positively. Users agreed that the application was fun and easy to use with a score around 4, both in immersive VR and on desktop. The usability score given by the SUS scale was also generally positive, landing at a score of 69. Users agree that the application is realistic and that it has informed them about their career possibilities and their requirements, with those questions scoring around 4. Users also believe the application should be used by NAV going forward, and users overwhelmingly agree that more job tastes should be created. The application received comparable scores in the aforementioned factors to the FiskeVR and Wind Turbine VR applications evaluated in (Henrichsen 2019). While the data provided in (Fast et al. 2004) is limited, Heisekran VR's reception appears to be comparable to the welding application (see Section 2.5), with both receiving positive reception and users believing they would be useful. In this regard, the application appears to also compare well to the mining application (Grabowski and Jankowski 2015). Scaling its questionnaire results from a 7-point Likert scale to a 5-point with a focus on the immersive VR results: The Heisekran VR application gets a 4,2 for ease-of-use while the mining application only gets 3,3 ("The virtual environment is easy to use") and 3,9 (inverted from "Capturing/moving items needed to complete the task was difficult"). The mining application also scores around 3,8 for questions regarding usefulness compared to Heisekran VR's 4. When it comes to informing young job seekers, the application appears to be successful.

The application still has some shortcomings, however. While users felt they were informed about professions and the choices available for them, they did not feel that the applications helped them know whether the jobs were right for them. They were neutral to whether the applications helped them experience the jobs for real with a score of 3, and only slightly positive to whether the application helped them gain insight if the professions were fit for them with a score of 3,36. This is despite users finding the tasks to be realistic (with a score of 4) and finding that they have learned a bit what it means to work (with a score of 3,81). It's possible the users realise there is more to the profession than what the application shows, and therefore do not feel they try it for real. If they do not try it for real, they cannot know if the job is a fit for them. Nevertheless, this is just speculation until further investigation can be performed.

There are also some shortcomings relating to the application's ability to push users to seek jobs. Questions relating to motivation and self-insight have a neutral trend. Users believe they have not changed due to the application, with a score as low as 2,45. They do not believe the app has given them any more insight into themselves and their abilities, nor do they believe the app has made them feel they can master working life (2,82). This differs from the results in (Henrichsen 2019)

where NAV users felt more motivated to seek jobs after playing the application with a score around 4, and NAV personnel overwhelmingly agreed the applications would have that effect with a score close to 5. It also differs from the results from the mining app (Grabowski and Jankowski 2015), which when scaled to a 5-point Likert scale shows an improvement in confidence (3,8), attitude (roughly 3,6-4,0) and ability to perform the job better (3,3).

This might possibly be because of technical issues that caused the players to fail while performing their tasks, with specific focus on the issue that caused containers to fall off the hook until Iteration 6. Many users mentioned that they knew they could not become crane drivers after it happened, despite reassurances that it was not their fault. Dropping a container was seen as a big failure to perform their tasks correctly, and it was irrecoverable once it happened, forcing the player to pick up another container. As (Malone 1980) warned (see Section 2.4), excessive failure states can harm the self-efficacy of the players. A good portion of user tests were performed while this issue was unsolved, which could explain the lower score. Simultaneously, it is possible that the questions simply differ too much to be directly comparable. The questionnaire used for the fishery and wind turbine applications focuses on motivation and confidence in applying for a job, while the questionnaire used for Heisekran VR focuses on self-insight and self-confidence with no direct mention of motivation and applying for a job. There is also a possibility that the low focus on training and the low number of tasks caused the shortcomings. The questionnaire in the mining app focuses on confidence and ability in performing the job and had passable results in that regard. As discussed in Section 12.1.2, the Heisekran VR application only has one work task (lifting a container) which is relatively basic and only has two different difficulties. It lacks the variety and difficulty to give the user a sense of progression. This is compared to the mining app that had a long list of activities to complete in order to finish its work task. Whatever the case, it should be investigated why Heisekran VR seemingly has a worse result than comparable applications in this regard.

Paying further attention to the usability of the application, the results are positive but could still be better. The resulting SUS score of the application is 69, calculated based on (Brooke 1986). An evaluation of the use in SUS places a score around 70 as passable, but with potential room for improvements (Bangor et al. 2008). The results are comparable to the SUS score of 70 the immersive VR variant of the mining application received during its evaluation (Grabowski and Jankowski 2015). Most of the results of the SUS scores for Heisekran VR match the ones for Wind Turbine VR, with one question showing a large discrepancy (Henrichsen 2019). Users disagree that they would like to use the job taste applications frequently, with a score as low as 2,22. Henrichsen asked the users whether they would like to use the job taste applications frequently through NAV or school and received a score as high as 4,86 for Wind Turbine VR and 4,31 for FiskeVR. The only reason the author can find for this discrepancy is the difference is formulation, as enjoyment and usability ratings were otherwise similar. (Bangor et al. 2008) indeed mentions that one should expect large discrepancies in the results of that question as users will wish to use some systems regularly (like cellphones) while only wishing to use others occasionally (like initial setup applications). At the same time, disregarding such an overwhelming difference in sentiment seems difficult to justify. Perhaps one of the other discrepancies mentioned above could be the reason? Future questionnaires should likely use the same formulation as Henrichsen to be clearer.

After the above comparisons, it seems that Heisekran VR provides decent value to its users as an Immersive Job Taste. It does not achieve anything entirely out of the ordinary regarding its effectiveness for education and training when compared to related work, but the additional accessibility it provides described in Section 12.1.1 should make it usable by more NAV users.

Professionals

The interview with personnel from the harbour in Trondheim, Trondheim Havn, in Section 11.8 showed that they had a positive opinion of the Heisekran VR application. They believed it covered enough of their profession to be a good entry-point for young people to learn what they do. They also believed that while it painted their profession in a nice light, it was not misleading in any way. They mainly wished for more types of cranes to be added to properly represent the experience of driving other types of cranes.

As mentioned many times before, Immersive Job Tastes need to avoid misrepresenting professions by making them seem more fun than they actually are. Looking at the results of the questionnaire, the users clearly believe that immersive job tastes are teaching them about different professions. They have no way of verifying it except for the faith they put in NAV. It would have catastrophic results if a young job seeker found a job based on the knowledge they gained from an immersive job taste, then quit because they were misled. It is good that Heisekran VR has finally been tested by professionals after the pandemic prevented them from testing the prototype (Hesle et al. 2020).

Initially, the harbour personnel mentioned that maintenance tasks and paperwork were missing from the application. Those tasks are part of their daily routine. Maintenance and inspection work has to be performed before driving the crane, and a safe work analysis has to be filled out before extraordinary tasks can be performed. The personnel from Trondheim Havn were hesitant about adding those tasks to the application in fear of making the profession seem too boring, but it might be what is necessary for the application to seem more real. It might be an idea to implement those tasks regardless to represent the workplace more accurately. Footage of a worker at Trondheim Havn filling out a form already exists, so adding it to the application would be simple. Adding a maintenance task would require modelling work on the crane, as it is currently somewhat small and awkward to navigate as noted in Section 7.2. It would require more effort than the paperwork task, but it would fulfil Requirement R13. As a counterpoint, if the application is already in an acceptable state, the development time might be better spent on creating another job taste.

On top of teaching players about the workplace, the application should be able to teach the players the basic controls of the crane. Immersive Job tastes are not meant as a training tool, but providing some initial training is still a positive thing for an Immersive Job Taste application (Prasolova-Førland et al. 2019). The kind of spatial and psychomotor skills required to use the sticks are suitable for learning through VR (see Section 2.3). The movement of the joysticks being embodied actions should also help with learning (see 2.3). As long as the control sticks are accurately modelled to the real-life ones, they should be effective as a basic training tool as well as an educational tool. The professionals at Trondheim Havn were positive they could use their application for new recruits. The crane driver that previously tried the application in the user test described in Section 8.2 also had no complaints about the controls, making it seem that they were accurate.

NAV Personnel

Little data was collected from NAV personnel. Their thoughts about the Heisekran VR application were generally positive during user tests and the interview performed in Section 11.7, but without more data, no real conclusions can be drawn.

12.2.2 RQ 2: The value of the desktop variant

There is no proven correlation regarding the effect of immersion on learning, both for the technological and subjective definition of it (see Section 2.1.1). Many applications show no loss in learning for desktop variants compared to immersive variants, like the water pump assembly application described in Section 2.5. Others show large improvements. As described in Section 2.3, immersive Virtual Reality has many affordances that applications need to take advantage of in order to see an improvement, and the technology itself will not automatically manifest into learning games (Dalgarno and Lee 2010). Nevertheless, the desktop variant of Heisekran VR was not expected to be equal to the immersive VR variant, as it was conceptualised with immersive VR technology in mind. The limited user evaluations performed showed it was mostly positively received by NAV users (see Section 12.2.1), and that is sufficient for the scope of this thesis.

The focus of the desktop variant was thought to be accessibility (Prasolova-Førland et al. 2019). Players would be able to play the job taste if they got cybersickness by playing with immersive VR technology or did not have immersive VR technology available at all. In practice, little enthusiasm was shown for the desktop variant of the application. Most users refused to participate in testing the desktop application or did it reluctantly after having tried the immersive VR application

first. In cases where users were affected by cybersickness or fear of heights, they would (perhaps understandably) stop participating entirely rather than switch to the desktop variant. Only three people responded to the questionnaire about having attempted the desktop application, and only one had exclusively played the desktop application. Users would commonly only play the desktop variant if all the immersive VR HMDs were currently occupied and would quit playing to swap to the immersive VR HMD when one became available.

It is likely that low enthusiasm was shown because most of the user tests were arranged with the premise of experiencing immersive VR technology. Analogous to the findings in (Ekelund 2018), no one wants to try "the poor man's version" of the application if they perceive there is a better variant available. It is possible that the desktop variants would be seen as more valuable if user tests were performed on those variants alone without informing the users of the immersive VR variants. It is also possible that simply framing the desktop applications as equivalent would suffice to dispel any preconceived notions about the desktop variants (Ekelund 2018). After all, applications like Mozilla Hubs (Section 2.5) do not have negative reception of the desktop variants. This should be considered for future evaluations of desktop variants of applications once pandemic restrictions are lifted and performing enough user tests becomes realistic again.

Another affordance of the desktop variant that was noted in chapter 1 was the possibility of users running desktop applications remotely. Users could be sent the desktop application through an online download and be asked to play it in their own time. This idea faces some challenges. Users interviewed in (Ekelund 2018) mention that they would want to experience job tastes in a "serious setting" with guidance from NAV. Experiencing them at home without NAV guidance is seen as less appealing. With the Mozzila Hubs application showing that remote collaboration can be successful (see 2.5, (Gomes de Siqueira et al. 2021)), it might be possible to adopt something similar in order to create a serious setting remotely. If NAV users could use Immersive Job Tastes from home in a collaboration with career guidance personnel, it could be possible that users perceive it as a serious setting regardless of the distance. If job tastes gained a multiplayer component, it could be possible for NAV personnel to aid their users while they are using the applications. Alternatively, it could be easier for the developers if NAV adopted another platform for collaboration like the aforementioned Mozilla Hubs.

The user test with Trondheim Havn decribed in Section 11.8 also sheds light on another issue with running applications at home. They failed to get the application to run on their machines before the scheduled remote meeting. Giving only two users technical support to get the applications running during a meeting would have been difficult, so that user test fell back to the video of the application. Providing technical assistance on a large scale to users attempting to run the application from home would be unrealistic unless NAV or IMTEL had people dedicated to technical support available to them. Releasing the application online without any support would limit data gathering to the users that have the technical prowess to get the application running themselves and potentially skew the results. Running the application would have to be simplified to the point that most people can succeed, using technical solutions to simplify starting the application and providing good explanations for how to run them. Inspiration could perhaps be taken from Mozilla Hubs again to run the application in the browser instead of requiring a download. The desktop variant has low system requirements as seen by its performance on a low-end laptop (see Section 12.1.3), and it could likely be optimised further if need be. If the application could run in the browser, all the users would have to do is click a link that could be put on a website, sent through email or sent through another chat service.

All in all, the results for the desktop variant are inconclusive. The limited quantitative data shows promise, but the value of desktop variants depends on how much enthusiasm can be summoned from the user and whether there are cases in the future where only desktop applications can be used. It is the hope of the author that the instantiation provided in this thesis will assist future research on the subject.

The desktop variant did find good use during development. Having a desktop mode available makes it significantly easier to develop and demo solutions in Unity. The developer can test solutions from the same screen they develop on, instead of having to move in and out of immersive VR constantly. Developing on the same screen also allows the developer to print debug information

easier if needed, though there are decent alternatives for immersive VR (see Section 7.1.5). An immersive VR HMD is required for tasks related to VR interactions, but any task that can be performed without VR like hooking containers can be tested easier in desktop mode (see Section 10.1.1). The value of a desktop variant for development purposes depends on how many features can be developed without requiring the developer to test in VR. As the experience of developers working on Immersive Job Tastes varies, it also depends on how easy they find it to implement and what use they can get out of it. Making a desktop variant should not be particularly difficult with the solutions described in this thesis (further described in Section 12.2.4), but it requires the developer to know how to take advantage of it.

12.2.3 RQ 3: The value of Unity XR, XR Interaction Toolkit & Standalone HMDs

Unity XR & XR Interaction Toolkit

Using Unity XR had a large initial start-up cost. Both the framework and the surrounding packages like the XR Interaction Toolkit were new, and therefore lacked the robustness and documentation of more mature frameworks. Fewer developers had used it compared to the legacy frameworks, so online discussion and solutions were rare and difficult to find. Most of the framework had to be learned through reading the framework's code, rather than learning simpler example solutions and building on top of them. Some basic aspects of VR interactions had to be implemented from scratch. It is possible that developers with more experience than the author would have made shorter work of it, but at this point is only speculation.

This issue will be mostly alleviated for future applications that are created for IMTEL using Unity XR and the XR Interaction Toolkit. Many of the basic solutions now exist in Heisekran VR, meaning the application could be studied by newcomers and its solutions applied during the development of new Immersive Job Tastes. This thesis also documents the process of developing those solutions during the iteration process, supporting the implementation of new and complex solutions if needed. Additionally, Heisekran VR has shown how to implement many of the XR Interaction Toolkit-compatible prefabs that IMTEL has created, which should make it easy for all future Immersive Job Tastes to fulfil the requirements of Immersive Job Tastes. The relative simplicity of Unity XR and the XR Interaction Toolkit should make it easier for newer developers to learn the framework as well. The biggest shortcoming of current solutions is the lack of hand poses when grabbing items as described in Section 7.1.2. A robust solution should be created or bought on the Unity Asset Store in the future in case a profession uses tools where hand posing and positioning is crucial for immersion.

There is no question whether Unity XR should be used in the future. The legacy frameworks are deprecated and will no longer be an option once they are removed. There is no choice for IMTEL but to adopt Unity XR if they intend to continue using the Unity game engine for their applications. The real question lies in the front-end used for VR interactions. The XR Interaction Toolkit shows promise as a first-party package, meaning it is developed by Unity themselves and should be supported for the foreseeable future. As discussed above, the XR Interaction Toolkit is basic and requires learning, but solutions have already been created for the common actions that Immersive Job Tastes require. Even more importantly, the prefabs containing common functionality support the XR Interaction Toolkit already. The only alternative currently would be VRTK due to its ease of development as noted in (Henrichsen 2019), but it depends entirely on how VRTK transitions to using Unity XR as a back-end. Moving existing solutions and prefabs from the XR Interaction Toolkit to another front-end would only be worthwhile if it was simple to do and provided a substantial improvement to development time and usability. Henrichsen makes VRTK sound valuable to new developers, but further investigation would have to be made into Unity XR + VRTK before such a decision could be made. If VRTK transitions cleanly to Unity XR and moving existing solutions to VRTK is simple, it could potentially be considered. Until that potential can be explored further, Immersive Job Tastes should likely stay with the XR Interaction Toolkit.

Standalone HMDs

Standalone HMDs have been used for many of the user tests performed during this thesis. Their affordability and ease of transport due to being all-in-one devices has significantly increased the devices available during NAV's user tests that previously only had one or two tethered HMDs available. Their closed ecosystem makes them more predictable and face fewer issues once they have been properly configured. Once they are ready, they can be brought up without fear of potential software issues or crashing that was observed with tethered solutions. This shows that standalone HMDs have potential for future use by the Virtual Internship project and NAV, but they still face many challenges before they can be widely deployed.

The main issue is the difficulty in setting up the devices. As far as the author is aware, not a single standalone HMD was prepared by NAV without requiring help from IMTEL or a researcher. As described in Section 4.7, current standalone devices are designed for entertainment and have strict integration with personal Facebook accounts. No process to set up a large number of devices with anonymous accounts has been found, meaning that each device has to be set up manually with the personal smartphone and Facebook account of NAV personnel. Even discounting the privacy and security issues that arise from this, it is a process that often fails and leaves devices inoperable, as seen during previous user tests. This is entirely unviable if the Virtual Internship project intends to deploy Immersive Job Tastes to NAV offices across the country. A standalone HMD that does not require linking to a smartphone nor a personal social media account needs to be available before that goal can be accomplished.

Even once the devices are configured with an account, installing the Immersive Job Taste applications to them is not easy. As described in Section 4.7.3, the devices must be put into developer mode and connected to a computer. Developer tools must be installed and used from the computer in order to install the application to the standalone HMD. The NAV Catalogue described in Section 4.4 was designed to simplify the installation process by abstracting away the use of the developer tools, but it has a chance of failing as seen during the NAV user test in Section 11.5. If installation using the catalogue fails, one would need to have knowledge of the developer tools and perform the installation manually, which cannot be expected from most NAV personnel.

Widespread deployment of Immersive Job Tastes can only succeed if configuring the device and installing applications can be done by an average worker at NAV. Regular computers within organisations are often configured by an IT department and are delivered to workers in a finished state. If a solution to configure standalone HMDs in the same way is found, it would significantly ease the requirements placed upon NAV workers that wish to utilise the Immersive Job Taste applications in their work. If that is not possible, it should be made easier for NAV personnel to install the applications themselves. Standalone devices tend to have a storefront where they can download applications within the device itself, like the app stores available on smartphones. These storefronts tend to be designed in a way that less technical users are still capable of using them. If Immersive Job Taste applications can be put onto such a storefront, it should be realistic for NAV workers to be able to install and update the applications when necessary.

Another issue is providing guidance to the player when they are using a standalone HMD. The player's view is displayed on the computer when players use a tethered HMD, allowing a supervisor to see what the player is doing and provide advice if necessary. Standalone HMDs do not display their view anywhere by default. This has led to situations where the users fail to progress in the application without the supervisor being able to tell why, an example being the user test described in Section 11.6. The Oculus Quest HMDs can stream their view to an external device, a technique that was used for the user test described in Section 8.2. However, the built-in streaming functionality is limited to the owner of the account linked to the device, making it difficult to use in cases like NAV offices where multiple devices are present with no clear owner. There are alternative ways to stream to other devices like computers that should be investigated. To make standalone HMDs equivalent to the tethered ones, NAV personnel should be able to view these streams without much technical knowledge.

12.2.4 RQ 4: How to develop or port an app for Unity XR, XR Interaction Toolkit, standalone HMDs and Desktop

Unity XR & XR Interaction Toolkit

The Heisekran VR artefact created during this thesis should provide new developers with sufficient example code to aid in development of Immersive Job Tastes using the XR Interaction Toolkit. If there are insufficient tutorials available by the time new developers start working with the framework, they should be able to copy the solutions present in the Heisekran VR application in order to get started. The author is confident that most of the solutions could fit the description of best practice. The XR Player Rig, teleportable areas and XR interactions could be created based on the instantiation of Chapter 7, and the IMTEL prefabs could be implemented based on the instantiation of chapter 10.

Moving an existing job taste from the SteamVR framework to the XR Interaction Toolkit should be largely like developing a new application from scratch. The same solutions mentioned above should be implemented to replace the SteamVR solutions, as was performed in this thesis. The developer should look out for interactables that were implemented using a SteamVR component that has no analogue in the XR Interaction Toolkit and create replacements, similarly to how the crane's joysticks had to be created anew in Section 7.1.5. The control scheme used by SteamVR should be noted and replicated using the Unity Input System described in Section C6.6 to avoid buttons on controllers performing different actions after the change in framework. When moving an application from SteamVR, it might not be necessary to replace solutions that are not tied to the framework like 360-video platforms. The solution for 360-degre videos present in Heisekran VR did not need replacement after the transition. Nevertheless, it is best practice to change the application to the 360-degree video prefab as outlined in Section 9.1.1 in order to have the same interface as other Immersive Job Tastes.

There are some exceptions to best practice in the application that should not be copied without improvements. The main issue is the player object created by merging the desktop and XR player objects in Section 10.1.7. The resulting object has a great deal of components that the Player Manager component (seen in Section 8.1.4) must handle for the top-level player object to work in both desktop and XR modes. The ideal solution would be to revert the change and separate the desktop and XR player objects and instead make prefabs and other application code handle each mode separately. This would reduce the complexity of the player objects and make them easier to understand by new developers. It would also reduce the complexity of the Player Manager component and make it easier to expand the player objects if needed. The recommended solution would be to make a Player class containing the functionality that players of both modes have in common, then create DesktopPlayer and XRPlayer subclasses to implement functionality specific to each mode and differentiate between them. The Immersive Job Taste prefabs could then generalise their code to work with objects that have components of those classes attached, instead of relying on specific implementation details about the objects like current solutions do.

As mentioned in Section 12.2.3, the hand models shown to the user are also not ideal in their current state. Their animations are not as detailed as the hands that come with the legacy SteamVR and OculusVR frameworks, and currently do not have a way to rigidly attach to held objects and attain special poses. It should be relatively easy to implement these features as the code is still present from the OculusVR iteration. Alternatively, a solution could be purchased on the Unity Asset Store if it is suitable for Unity XR and the XR Interaction Toolkit out of the box. Such a solution could be worth the money if the solution is robust and has good enough graphical fidelity. There were no solutions available while the hands were ported from OculusVR to UnityXR during this thesis, but as the framework is starting to be finalised, this might change soon.

Standalone HMDs

Developing for standalone HMDs is simple once the application is running on Unity XR. It is mostly about configuring Unity to match Oculus' recommended settings and avoiding certain

incompatibilities as described in Section 7.1.6. The issue that Heisekran VR ran in most often was supporting the large video files on a standalone device. The video files were separated from the application file in that section using the Addressables package in order to support a large quantity of videos. They were later optimised and bundled back into the application's files due to the NAV catalogue Yrkeskatalog not supporting the distribution of multiple files per application (see Section 9.1.5). Optimising the size of the videos is important, but there is a real risk that 360-degree video heavy applications could exceed the application size restrictions applied by Oculus regardless of optimisation. It might be important for Yrkeskatalog to support separating assets from the application file for standalone devices in the future.

The biggest challenge is not to make the application run on a standalone HMD, but to make the application perform well on the limited hardware. The code and graphical assets that the application uses need to be reasonably optimised. It is beyond the scope of this thesis to discuss how one would optimise an application. Nevertheless, new developers should look at running high-impact code only when it needs to run. Many aspects of this thesis were written in an event-based manner, like the desktop player's activation button in Section 10.1.8. Running performance-intensive code as it is needed instead of constantly while the game is running is an easy way to save resources in many cases. Graphical assets can also be optimised in ways further described in (Henrichsen 2019).

Desktop

Adding basic desktop support is in essence rather simple. As explored in chapter 8, video games have been developed for 3D worlds for a long time and solutions to basic issues are common. Adding support for controlling a character using a keyboard and mouse is simple by either copying the solution created in Heisekran VR or many others. Section 12.2.2 touches upon that much of the game's logic is not dependent upon VR interactions. In such cases, the application is likely to work with the desktop character without much additional effort.

The real issue is how to convert VR interactions into desktop controls. It was easy to convert Heisekran VR into a desktop experience because its VR interactions consisted of moving around the harbour and controlling the crane. Mapping those interactions to a set of buttons on a keyboard was not a difficult task to figure out. Other applications might have more complex interactions where the six degrees of freedom granted by the controllers are important to the tasks performed. Tasks might require the player to rotate an object in a certain way while pressing buttons on the controller and might require the player to use both hands to perform the same task. Some applications like the welding application described in Section 2.5 use every single degree of freedom to create the resulting weld. In such cases, creating a reasonable desktop approximation becomes a challenge in user interface design. Going from two embodied 6DOF controllers to a keyboard and mouse will often involve some form of compromise. Not all games can be converted from VR to desktop in a reasonable manner without redesigning a large part of the game. If VR games could all easily be converted to desktop controls, one could argue that there would not be a point to VR games in the first place. For example, the aforementioned welding application would serve no purpose as a desktop application and could just as well have been replaced by a video when all the complexity of the VR interaction is removed. As mentioned in the section about the value of a desktop variant, one should consider whether the game is a fit for desktop controls before attempting to make it compatible with desktop.

12.3 Limitations

This research has been limited by the COVID-19 pandemic that started in 2019 and continued until the end of the research project in 2021. Restrictions designed to prevent the spread of the disease make user tests difficult to arrange. Gathering users to test applications at large scales has often been impossible, and user tests have still been limited in number when the restrictions have allowed them. One could speculate that even when restrictions have been temporarily lifted, many would hesitate to go out unnecessarily. The kind of user tests performed in research like

(Henrichsen 2019) where the application is set up on a stand and presented to passers-by have also not been realistic.

The issue with COVID restrictions is exasperated in regard to workplaces. Companies would not want to risk key personnel getting infected, so getting NAV personnel and especially professional crane drivers to test the application has been difficult. A possibility to perform a user test with Trondheim Havn did not appear through the entirety of 2021 and had to be done remotely. It is understandable, as a crane driver getting infected could potentially paralyse the harbour. Much appreciation is still given to Trondheim Havn for the support they have given.

The result of the COVID restrictions are a lack of user tests and large periods of time between them. The lack of user testing is reflected in the quantity of the data. One should hesitate to draw concrete conclusions out of the questionnaire. With only eleven responses from young job seekers, the data gathered by the questionnaire is not as thorough as other evaluations of Immersive Job Tastes. This is especially true for the desktop variant of Heisekran VR which only had 3 respondents, likely from the same user test. A positive score with so few respondents could potentially be a fluke and should be verified with further user tests in the future.

Furthermore, the author was guiding users through the application in most of the user tests, with other IMTEL researchers or NAV personnel guiding only a few. Interaction with researchers is known to potentially alter the user's perception of the applications, or at least the answers they give. The risk is that the scores the respondents gave are higher than if they independently tested the application.

Another limitation of this data is that data gathered from NAV personnel was extremely limited and only qualitative. Only young job seekers were given the questionnaire. Without quantitative data, it is difficult to determine the perception of the application without the author's implicit bias. The Immersive Job Taste concept specifies that career guidance personnel should also take part in evaluation. Without evaluating the sentiment of career counsellors, an evaluation of the job taste application is incomplete.

Trondheim Havn were also unable to try either variant of the application for themselves. Their judgement is based only on a video demonstrating the contents of the application. Without having tested the application, they would be unable to comment on issues with the interactions and technical issues. The video could also obscure other issues with the application that are only visible when playing. A previous crane driver trying an older iteration of the application alleviates it to a degree, but ideally multiple crane drivers would be able to try out the final artefact.

In regard to the evaluation of the development process and underlying technologies, it should be noted that this is a subjective and qualitative judgement by a single developer. It is entirely possible that extensive use of these technologies has added some bias regardless of attempts to make an objective judgement. Another developer might have a different experience using these technologies and find them easier or more difficult to use. Without a second opinion about the solutions created for the application, it is also possible that the author overestimates their quality and suitability as "best practices" solutions.

The author's largest deficit is the lack of knowledge of 3D-modelling and animating, which limited the creation of new content to assets available from other sources and applications. This likely increased the time required to develop using any technologies that require graphical prowess and would affect the evaluation of those technologies negatively. The author also has limited experience with development in Unity and using VR technology. Someone with years of experience in Unity development might not have an issue understanding and modifying frameworks more complex than Unity XR. A developer with more experience would also likely have finished the modifications faster, affecting the evaluation of the process and results.

The COVID-19 pandemic also affected productivity during the time spent on this research. It is likely that much of the work could have been performed faster without restrictions and lock-downs affecting society. It would also have made it easier to involve other developers during the design process of the application and find alternative solutions. A common strategy for IMTEL-developed applications has been to spend at least one day at its VR Lab per week in order to foster

cooperation and productivity. That has not been possible during this research.

Chapter 13

Conclusion and future work

13.1 Conclusion

This thesis has investigated relevant literature, Virtual Reality technology and the Virtual Internship project to understand the current state of VR and find improvements to the accessibility of the Virtual Internship project's application. An existing game named Heisekran VR has been improved through multiple iterations of development and testing in order to get it to work on standalone VR devices and on computers with no VR devices at all. The development process of each iteration has been documented to assist future development with these technologies, and the testing process has provided evaluation of the development and the resulting application. Finally, an evaluation of the technologies and their viability for future development has been made.

The Heisekran VR application was found to be equally well received by NAV users compared to other Immersive Job Taste applications. The application was deemed purposeful and accurate by professional crane personnel. Evaluation by NAV-personnel was limited but had promising results. Running Immersive Job Taste applications on standalone devices was found to have potential but faces some roadblocks regarding accessibility and privacy that need to be solved before the technology can be used for deployment at NAV offices across the country. Little enthusiasm was shown by users for the desktop variant without VR. Some positive feedback was received, but further efforts to simplify the distribution and use of the desktop variant is required to make use of its accessibility. The Heisekran VR application has been made available for use by NAV and the solutions implemented within the application will be useful for future development and research of Immersive Job Taste applications.

13.2 Contributions

This research contributes the following to the Virtual Internship project:

- **The Heisekran VR application:** A finished Immersive Job Taste that can be used by IMTEL for further research and development, and NAV for career guidance in one additional profession.
- **An instantiation of processes and methods:** The finished Heisekran VR artefact demonstrates the processes and methods described in this thesis for developing additional functionality for an Immersive Job Taste. The artefact also contains solutions that can be studied and re-purposed by developers of other applications.
- **The widest platform support of a career guidance application:** No other career guidance application within and outside the Virtual Internship project supports tethered VR, standalone VR and desktop play simultaneously. All of this within a single application, simplifying maintenance and preventing different platforms from diverging.

- **Evaluation and guidelines for new technologies:** Being one of the first applications within the Virtual Internship project to use Unity XR, XR Interactive Toolkit and other related Unity technologies proves their viability and provides guidelines for their future use.
- **Process for supporting older applications:** Older job tastes would eventually stop functioning as their technological solutions are deprecated and no longer supported. The artefact and processes provide guidelines on how to update applications and keep them working for the foreseeable future.

13.3 Future work

This section focuses on future work that should be done on the Virtual Internship project and the Heisekran VR app.

13.3.1 Improving standalone device accessibility

Several issues were identified regarding the accessibility of standalone devices. If NAV intends to use standalone devices to deploy the Immersive Job Taste applications across the country, a solution must be found to use the devices in multi-user environments like offices or schools. Current solutions that only allow single users are not viable for the scale of devices that will be required nor their pattern of use. A user-friendly solution for installation and updating of applications has to be found, as current solutions using the NAV catalogue are still too difficult for users without technical knowledge. A way for NAV personnel to observe the application as the user is playing it will also be required, as otherwise they will not be able to assist their users if they have trouble using the applications.

13.3.2 Investigating Desktop applications

While solutions for adding desktop support to applications were found, the viability of the resulting desktop application was not sufficiently explored. Further user evaluations with desktop applications should be performed to assert whether the desktop job tastes provide enough value to justify desktop variants of future job tastes. Efforts to test with users remotely using online downloads, in-browser applications or similar solutions should be explored, as desktop solutions provide unique accessibility when the users have no access to VR equipment. For that reason, exploring evaluations where the users are not made aware of the VR variants might be an option.

13.3.3 VRTK

The VRTK framework for VR interactions is not viable as of writing due to lack of support of Unity XR. (Henrichsen 2019) noted that the framework was helpful for developers without much experience in VR development, which matches most developers working on the Immersive Job Tastes. Once VRTK supports Unity XR as planned, it could be investigated for its potential of further simplifying development of applications.

13.3.4 More consistency and guidance

There are many aspects of Immersive Job Tastes that have room for improvement. While the prefabs used for Heisekran VR and other applications help make the applications more consistent, there are still many aspects that differ between the job tastes. Many applications have different functionality and controls than others as they have not been updated to use the newest prefabs. This results in different ways to view 360-degree videos, different buttons to teleport and similar issues. The most egregious example is the Immersive Job Taste tutorial which is intended to

explain the controls for all other job tastes, but still has some inconsistent aspects to it like the tablet. Older job tastes should be updated to use the same prefabs and control scheme as all others and be made compatible with standalone HMDs. A unified control scheme should be defined and used across all applications, taking care to make it congruent with the actions that are being performed.

Bibliography

Ak, Oguz and Birgul Kutlu (2017). 'Comparing 2D and 3D game-based learning environments in terms of learning gains and student perceptions'. In: *British Journal of Educational Technology* 48.1, pp. 129–144. DOI: https://doi.org/10.1111/bjet.12346. eprint: https://bera-journals.onlinelibrary.wiley.com/doi/pdf/10.1111/bjet.12346. URL: https://bera-journals.onlinelibrary.wiley.com/doi/abs/10.1111/bjet.12346.

Bangor, Aaron, Philip T. Kortum and James T. Miller (2008). 'An Empirical Evaluation of the System Usability Scale'. In: *International Journal of Human–Computer Interaction* 24.6, pp. 574–594. DOI: 10.1080/10447310802205776. eprint: https://doi.org/10.1080/10447310802205776. URL: https://doi.org/10.1080/10447310802205776.

Bonasio, Alice (2021). *Product Review: Dell Visor Windows MR Headset*. URL: https://arvrjourney.com/product-review-dell-visor-windows-mr-headset-fb63d63634fa (visited on 27th June 2021).

Boud, A.C. et al. (1999). 'Virtual reality and augmented reality as a training tool for assembly tasks'. In: *1999 IEEE International Conference on Information Visualization (Cat. No. PR00210)*, pp. 32–36. DOI: 10.1109/IV.1999.781532.

Brooke, John (1986). 'SUS - A quick and dirty usability scale'. In: URL: https://hell.meiert.org/core/pdf/sus.pdf (visited on 7th Dec. 2020).

Brown, Emily and Paul Cairns (2004). 'A grounded investigation of game immersion'. eng. In: *CHI '04 Extended Abstracts on human factors in computing systems*. CHI EA '04. ACM, pp. 1297–1300. ISBN: 1581137036.

Cheers. (2021). *Picture of the under construction Infinity (300 Spear Street) Tower II and the crane*. URL: https://commons.wikimedia.org/wiki/File:Tower_Crane_and_350_foot_tower.JPG (visited on 8th June 2021).

Cheng, M.-T, H.-C She and L.A Annetta (2015). 'Game immersion experience: its hierarchical structure and impact on game-based science learning'. eng. In: *Journal of computer assisted learning* 31.3, pp. 232–253. ISSN: 0266-4909.

Cheng, Meng-Tzu et al. (2017). 'Is immersion of any value? Whether, and to what extent, game immersion experience during serious gaming affects science learning: Does game immersion experience affect science learning?' eng. In: *British journal of educational technology* 48.2, pp. 246–263. ISSN: 0007-1013.

Coder, Sharp (2020). *Unity 3D FPS Controller*. URL: https://sharpcoderblog.com/blog/unity-3d-fps-controller (visited on 3rd Dec. 2020).

Csikszentmihalyi, Mihaly (Jan. 1990). 'Flow: The Psychology of Optimal Experience'. In:

CXWorld (2021). *CapFrameX*. URL: https://github.com/CXWorld/CapFrameX (visited on 19th June 2021).

Dalgarno, Barney and Mark J. W. Lee (2010). 'What are the learning affordances of 3-D virtual environments?' In: *British Journal of Educational Technology* 41.1, pp. 10–32. DOI: https://doi.org/10.1111/j.1467-8535.2009.01038.x. eprint: https://bera-journals.onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8535.2009.01038.x. URL: https://bera-journals.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8535.2009.01038.x.

Daniel (2021). *Oculus Quest 2 VR Headset Takes You into Virtual Reality World with Better Images and 3D Sound*. URL: https://gadgetsin.com/oculus-quest-2-vr-headset-takes-you-into-virtual-reality-world-with-better-images-and-3d-sound.htm (visited on 27th June 2021).

Davepape (2021). *The Cave Automatic Virtual Environment at EVL, University of Illinois at Chicago*. URL: https://commons.wikimedia.org/wiki/File:CAVE_Crayoland.jpg (visited on 27th June 2021).

Duh, Henry, Donald Parker and Thomas Furness (2001). 'An "independent visual background" reduced balance disturbance envoked by visual scene motion: implication for alleviating simulator sickness'. eng. In: *Proceedings of the SIGCHI Conference on human factors in computing systems*. CHI '01. ACM, pp. 85–89. ISBN: 9781581133271.

Ehrhardt, Michelle (2021). *Some Oculus Quest 2 Owners are Getting Banned From Using Their Headsets*. URL: https://www.tomshardware.com/news/oculus-quest-2-users-banned-from-facebook (visited on 10th June 2021).

Ekelund, Oscar Ihlen (2018). 'Virtual work placement: How VR-technology can aid young jobseekers with career counseling and job search - a qualitative pilot study'. In:

Fast, Kenneth, Timothy Gifford and Robert Yancey (2004). 'Virtual Training for Welding'. In: *Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality*. ISMAR '04. USA: IEEE Computer Society, pp. 298–299. ISBN: 0769521916. DOI: 10.1109/ISMAR.2004.65. URL: https://doi.org/10.1109/ISMAR.2004.65.

Fernandes, Luís Miguel Alves et al. (2016). 'Exploring educational immersive videogames: an empirical study with a 3D multimodal interaction prototype'. In: *Behaviour & Information Technology* 35.11, pp. 907–918. DOI: 10.1080/0144929X.2016.1232754. eprint: https://doi.org/10.1080/0144929X.2016.1232754. URL: https://doi.org/10.1080/0144929X.2016.1232754.

ffmpeg (2021). *FFmpeg*. URL: http://ffmpeg.org/ (visited on 16th June 2021).

Freina, Laura and Michela Ott (Apr. 2015). 'A Literature Review on Immersive Virtual Reality in Education: State Of The Art and Perspectives'. In:

G, Sludge (2021). *MOBILE CRANE, GOZO, MALTA*. URL: https://www.flickr.com/photos/28179929@N08/8640303911 (visited on 8th June 2021).

Gomes de Siqueira, Alexandre et al. (2021). 'Toward Facilitating Team Formation and Communication Through Avatar Based Interaction in Desktop-Based Immersive Virtual Environments'. In: *Frontiers in Virtual Reality* 2, p. 33. ISSN: 2673-4192. DOI: 10.3389/frvir.2021.647801. URL: https://www.frontiersin.org/article/10.3389/frvir.2021.647801.

Google (2020). *Create and set up your app*. URL: https://support.google.com/googleplay/android-developer/answer/9859152?hl=en-GB (visited on 30th Nov. 2020).

Grabowski, Andrzej and Jarosław Jankowski (2015). 'Virtual Reality-based pilot training for underground coal miners'. In: *Safety Science* 72, pp. 310–314. ISSN: 0925-7535. DOI: https://doi.org/10.1016/j.ssci.2014.09.017. URL: https://www.sciencedirect.com/science/article/pii/S0925753514002276.

Hafner, Manuela and Jeroen Jansz (2018). 'The Players' Experience of Immersion in Persuasive Games: The Players' Experience of Immersion in Persuasive Games: A study of My Life as a Refugee and PeaceMaker'. eng. In: *International journal of serious games* 5.4, pp. 63–79. ISSN: 2384-8766.

Hamari, Juho et al. (2016). 'Challenging games help students learn: An empirical study on engagement, flow and immersion in game-based learning'. In: *Computers in Human Behavior* 54, pp. 170–179. ISSN: 0747-5632. DOI: https://doi.org/10.1016/j.chb.2015.07.045. URL: https://www.sciencedirect.com/science/article/pii/S074756321530056X.

Havn, Trondheim (Mar. 2003). *Kran på Transittkaia*. URL: https://www.flickr.com/photos/trondheimhavn/2960423713/in/photostream/ (visited on 1st Apr. 2020).

Henrichsen, Jørgen (2018). 'Project Report: Engaging Young Job Seekers with an Internship as a Wind Turbine Technician in Virtual Reality'. In:

— (2019). 'Master Thesis: Engaging Young Job Seekers with an Internship as a Wind Turbine Technician in Virtual Reality'. In: URL: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2626168 (visited on 6th Dec. 2020).

Hesle, Henrik et al. (2020). 'Prosjektrapport Heisekran VR, Eksperter i Team'. In:

I-LOVE-CHICKEN (2021). *HDCP error running SteamVR*. URL: https://www.reddit.com/r/ValveIndex/comments/ddb3o4/hdcp_error_running_steamvr/ (visited on 11th June 2021).

Jensen, Lasse and Flemming Konradsen (July 2018). 'A review of the use of virtual reality head-mounted displays in education and training'. In: *Education and Information Technologies* 23, pp. 1–15. DOI: 10.1007/s10639-017-9676-0.

Jerald, Jason (2016). *The VR Book: Human-Centered Design for Virtual Reality*. ACM Books.

Johnson-Glenberg, Mina C. (2018). 'Immersive VR and Education: Embodied Design Principles That Include Gesture and Hand Controls'. In: *Frontiers in Robotics and AI* 5, p. 81. ISSN: 2296-9144. DOI: 10.3389/frobt.2018.00081. URL: https://www.frontiersin.org/article/10.3389/frobt.2018.00081.

Johnson-Glenberg, Mina C., David A. Birchfield et al. (2015). 'If the Gear Fits, Spin It!: Embodied Education and in-Game Assessments'. In: *International Journal of Gaming and Computer-Mediated Simulations (IJGCMS)* 7.4, pp. 40–65. ISSN: 1942-3888. DOI: 10.4018/IJGCMS.2015100103. URL: https://doi.org/10.4018/IJGCMS.2015100103.

Johnson-Glenberg, Mina C. and Colleen Megowan-Romanowicz (May 2017). 'Embodied science and mixed reality: How gesture and motion capture affect physics education'. In: *Cognitive Research: Principles and Implications* 2.1, p. 24.

Kerman, Brandon (2021). *Best AA Batteries for Oculus Quest Controllers*. URL: https://basereality.co/blogs/guides/best-aa-batteries-for-oculus-quest-controllers (visited on 27th June 2021).

Khronos (2021). *OpenXR Overview*. URL: https://www.khronos.org/OpenXR/ (visited on 16th Apr. 2021).

Lang, Ben (2020). *Analysis: Monthly-connected VR Headsets on Steam Reach Record High of 1.7 Million*. URL: https://www.roadtovr.com/steam-survey-vr-headset-growth-march-2020/ (visited on 29th Nov. 2020).

— (2021). *Vive Cosmos Priced at $700, Pre-orders Open Today for October 3rd Release Date*. URL: https://www.roadtovr.com/htc-vive-cosmos-price-release-date-specs-pre-order/ (visited on 27th June 2021).

Lira, Felipe (2021). *How the Lightweight Render Pipeline is evolving*. URL: https://blog.unity.com/technology/how-the-lightweight-render-pipeline-is-evolving (visited on 14th June 2021).

Logix (2021). *FFmpeg: How To Crop Videos, With Examples*. URL: https://www.linuxuprising.com/2020/01/ffmpeg-how-to-crop-videos-with-examples.html (visited on 16th June 2021).

M.Minderhoud (2021). *Portainer (gantry crane) (Rotterdam harbour)*. URL: https://commons.wikimedia.org/wiki/File:Portainer_(gantry_crane).jpg (visited on 8th June 2021).

Malone, Thomas (1980). 'What makes things fun to learn? heuristics for designing instructional computer games'. eng. In: *Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on small systems*. SIGSMALL '80. ACM, pp. 162–169. ISBN: 9780897910248.

McLeod, S.A. (Aug. 2019). *Likert Scale*. URL: https://www.simplypsychology.org/likert-scale.html (visited on 15th June 2021).

McWhertor, Michael (2020). *New VR game combines tactical stealth action with kayaking*. URL: https://www.polygon.com/2019/5/21/18632753/phantom-covert-ops-oculus-vr-military-stealth-kayaking (visited on 7th Dec. 2020).

Mikropoulos, T. and A. Natsis (2011). 'Educational virtual environments: A ten-year review of empirical research (1999-2009)'. In: *Comput. Educ.* 56, pp. 769–780.

Milgram, Paul et al. (1995). 'Augmented reality: a class of displays on the reality-virtuality continuum'. In: *Telemanipulator and Telepresence Technologies*. Ed. by Hari Das. Vol. 2351. International Society for Optics and Photonics. SPIE, pp. 282–292. DOI: 10.1117/12.197321. URL: https://doi.org/10.1117/12.197321.

Minocha, Shailey, Ana-Despina Tudor and Steve Tilling (July 2017). 'Affordances of Mobile Virtual Reality and their Role in Learning and Teaching'. In: *The 31st British Human Computer Interaction Conference*. URL: http://oro.open.ac.uk/49441/.

NAV (2020). *Utviklingen på arbeidsmarkedet*. URL: https://www.nav.no/no/nav-og-samfunn/kunnskap/analyser-fra-nav/arbeid-og-velferd/arbeid-og-velferd/utviklingen-pa-arbeidsmarkedet (visited on 9th Nov. 2020).

Oates, Briony J (2006). *Researching Information Systems and Computing*. Sage Publications Ltd. ISBN: 1412902231.

Oculus (2020a). *Asset Files to Manage Download Size*. URL: https://developer.oculus.com/documentation/unity/ps-assets/#upload (visited on 30th Nov. 2020).

— (2021a). *Configure Unity Settings*. URL: https://developer.oculus.com/documentation/unity/unity-conf-settings/?device=QUEST (visited on 21st June 2021).

— (2021b). *Monitor Performance with OVR Metrics Tool*. URL: https://developer.oculus.com/documentation/tools/tools-ovrmetricstool/?device=QUEST (visited on 19th June 2021).

— (2021c). *Oculus Desktop package*. URL: https://docs.unity3d.com/Packages/com.unity.xr.oculus.standalone@2.38/manual/index.html (visited on 15th Apr. 2021).

— (2020b). *Oculus Rift and Rift S minimum requirements and system specifications*. URL: https://support.oculus.com/248749509016567/ (visited on 6th Dec. 2020).

— (2020c). *Upload Apps for Oculus Quest and Go*. URL: https://developer.oculus.com/distribute/publish-uploading-mobile/ (visited on 30th Nov. 2020).

othree (2020). *Assembled Google Cardboard VR mount*. URL: https://commons.wikimedia.org/wiki/File:Assembled_Google_Cardboard_VR_mount.jpg (visited on 6th Dec. 2020).

Peckham, Eric (2020). *Unity IPO aims to fuel growth across gaming and beyond*. URL: https://techcrunch.com/2020/09/10/how-unity-built-a-gaming-engine-for-the-future/ (visited on 6th Dec. 2020).

Pertzborn, David (2021). *HP Reverb G2: VR-System nach dem Motto „Best of HP feat. Valve"*. URL: https://www.computerbase.de/2020-05/hp-reverb-g2/ (visited on 27th June 2021).

Pölönen, Monika (Aug. 2010). 'A head-mounted display as a personal viewing device : Dimensions of subjective experiences'. In: URL: https://helda.helsinki.fi/handle/10138/19799.

Prasolova-Førland, Ekaterina, Mikhail Fominykh and Oscar Ihlen Ekelund (2019). 'Empowering Young Job Seekers with Virtual Reality'. In:

Renganayagalu, Sathiya kumar, Steven C. Mallam and Salman Nazir (Jan. 2021). 'Effectiveness of VR Head Mounted Displays in Professional Training: A Systematic Review'. In: *Technology, Knowledge and Learning*. ISSN: 2211-1670. DOI: 10.1007/s10758-020-09489-9. URL: https://doi.org/10.1007/s10758-020-09489-9.

Richards, Deborah and Meredith Taylor (2015). 'A Comparison of learning gains when using a 2D simulation tool versus a 3D virtual world: An experiment to find the right representation involving the Marginal Value Theorem'. In: *Computers & Education* 86, pp. 157–171. ISSN: 0360-1315. DOI: https://doi.org/10.1016/j.compedu.2015.03.009. URL: https://www.sciencedirect.com/science/article/pii/S0360131515000780.

Schollmeyer, Josh (2006). 'Games Get Serious'. In: *Bulletin of the Atomic Scientists* 62.4, pp. 34–39. DOI: 10.2968/062004010. eprint: https://doi.org/10.2968/062004010. URL: https://doi.org/10.2968/062004010.

Sharples, Sarah et al. (2008). 'Virtual reality induced symptoms and effects (VRISE): Comparison of head mounted display (HMD), desktop and projection display systems'. In: *Displays* 29.2. Health and Safety Aspects of Visual Displays, pp. 58–69. ISSN: 0141-9382. DOI: https://doi.org/10.1016/j.displa.2007.09.005. URL: http://www.sciencedirect.com/science/article/pii/S014193820700100X.

Sitzmann, Traci (May 2011). 'A meta-analytic examination of the instructional effectiveness of computer-based simulation games'. In: *Personnel Psychology* 64, pp. 489–528. DOI: 10.1111/j.1744-6570.2011.01190.x.

Slater, M. (Aug. 2018). 'Immersion and the illusion of presence in virtual reality'. In: *Br J Psychol* 109.3, pp. 431–433.

Slater, Mel and Sylvia Wilbur (1997). 'A Framework for Immersive Virtual Environments (FIVE): Speculations on the Role of Presence in Virtual Environments'. In: *Presence: Teleoperators and Virtual Environments* 6.6, pp. 603–616. DOI: 10.1162/pres.1997.6.6.603. eprint: https://doi.org/10.1162/pres.1997.6.6.603. URL: https://doi.org/10.1162/pres.1997.6.6.603.

sosialdepartementet, Arbeids- og (2016). *NAV i en ny tid – for arbeid og aktivitet*. URL: https://www.regjeringen.no/no/dokumenter/meld.-st.-33-20152016/id2501017/ (visited on 4th Dec. 2020).

SSB (2020). *Utstyr internett er brukt på (prosent), etter kjønn, alder, statistikkvariabel og år*. URL: https://www.ssb.no/statbank/table/12349/ (visited on 9th Nov. 2020).

StayTalm_Unity (2021). *XR inputs and WMR*. URL: https://forum.unity.com/threads/xr-inputs-and-wmr.535005/#post-3605240 (visited on 12th Apr. 2021).

Steam (Nov. 2020). *Steam Hardware & Software Survey*. URL: https://store.steampowered.com/hwsurvey (visited on 6th Dec. 2020).

— (2021a). *Introducing SteamVR 1.16 - Now with full OpenXR support*. URL: https://store.steampowered.com/news/app/250820/view/3044967019267211914 (visited on 12th May 2021).

— (2021b). *SteamVR Support*. URL: https://support.steampowered.com/kb_article.php?ref=1131-WSFG-3320 (visited on 24th Mar. 2021).

— (2021c). *Transitioning to OpenXR*. URL: https://steamcommunity.com/games/250820/announcements/detail/2522527900755718764 (visited on 12th May 2021).

Steuer, Jonathan (1992). 'Defining Virtual Reality: Dimensions Determining Telepresence'. In: *Journal of Communication* 42.4, pp. 73–93. DOI: https://doi.org/10.1111/j.1460-2466.1992.tb00812.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1460-2466.1992.tb00812.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1460-2466.1992.tb00812.x.

Sweetser, Penelope and Peta Wyeth (July 2005). 'GameFlow: A Model for Evaluating Player Enjoyment in Games'. In: *Comput. Entertain.* 3.3, p. 3. DOI: 10.1145/1077246.1077253. URL: https://doi.org/10.1145/1077246.1077253.

Sweetser, Penny and Daniel Johnson (2019). 'GameFlow and Player Experience Measures: An Initial Comparison of Conceptual Constructs'. In: *Proceedings of the 31st Australian Conference on Human-Computer-Interaction*. OZCHI'19. Fremantle, WA, Australia: Association for Computing Machinery, pp. 317–321. ISBN: 9781450376969. DOI: 10.1145/3369457.3369486. URL: https://doi.org/10.1145/3369457.3369486.

Tennyson, Robert D. and R. L. Jorczak (2008). 'A conceptual framework for the empirical study of games'. English. In: *Computer games and team and individual learning*. Ed. by H. O'Neil and R. Perez. Erlbaum, pp. 3–20.

Tirman, Sean (2021). *Oculus Quest All-In-One VR Gaming System*. URL: https://hiconsumption.com/oculus-quest-all-in-one-vr-gaming-system/ (visited on 27th June 2021).

Unity (2021a). *[XR] THE A, B, X, AND Y BUTTONS FOR THE HP REVERB G2 CONTROLLERS AREN'T SUPPORTED NATIVELY*. URL: https://issuetracker.unity3d.com/issues/xr-the-a-b-x-and-y-buttons-for-the-hp-reverb-g2-controllers-arent-supported-natively (visited on 12th Apr. 2021).

— (2020a). *AssetBundles Intro*. URL: https://docs.unity3d.com/Manual/AssetBundlesIntro.html (visited on 30th Nov. 2020).

— (2020b). *Character Controller*. URL: https://docs.unity3d.com/Manual/class-CharacterController.html (visited on 3rd Dec. 2020).

— (2021b). *InputSystem - index*. URL: https://docs.unity3d.com/Packages/com.unity.inputsystem@1.0/manual/index.html (visited on 9th Apr. 2021).

— (2021c). *InputSystem - Quickstart Guide*. URL: https://docs.unity3d.com/Packages/com.unity.inputsystem@1.0/manual/QuickStartGuide.html#getting-input-indirectly-through-an-input-action (visited on 13th Apr. 2021).

— (2021d). *NavMeshComponents - Components for Runtime NavMesh Building*. URL: https://github.com/Unity-Technologies/NavMeshComponents (visited on 14th Apr. 2021).

— (2020c). *Our Company*. URL: https://unity.com/our-company (visited on 6th Dec. 2020).

— (2021e). *Unity XR input*. URL: https://docs.unity3d.com/Manual/xr_input.html (visited on 12th Apr. 2021).

— (2021f). *Upgrading your Shaders*. URL: https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@12.0/manual/upgrading-your-shaders.html (visited on 14th June 2021).

— (2020d). *VideoPlayer Reference*. URL: https://docs.unity3d.com/ScriptReference/Video.VideoPlayer.html (visited on 30th Nov. 2020).

— (2021g). *XR Plug-in Framework*. URL: https://docs.unity3d.com/Manual/XRPluginArchitecture.html (visited on 16th Apr. 2021).

Valve (2021). *SteamVR Quickstart*. URL: https://valvesoftware.github.io/steamvr_unity_plugin/articles/Quickstart.html (visited on 15th Apr. 2021).

VR, Career Labs (2021a). *MINIMUM COMPUTER SPECIFICATIONS FOR COMPATIBILITY*. URL: https://careerlabsvr.com/wp-content/uploads/2021/02/CareerLabsVR-Hardware-Specifications.pdf (visited on 24th June 2021).

— (2021b). *The CLVR SYSTEM*. URL: https://careerlabsvr.com/clvr_system/ (visited on 24th June 2021).

VR-compatible games on Steam (2020). URL: https://store.steampowered.com/search/?tags=21978%5C&vrsupport=402 (visited on 29th Nov. 2020).

Wallossek, Igor and Chris Angelini (2020). *Skylake: Intel's Core i7-6700K And i5-6600K*. URL: https://www.tomshardware.com/reviews/skylake-intel-core-i7-6700k-core-i5-6600k,4252.html (visited on 9th Nov. 2020).

Wouters, P.J.M., Erik Spek and H. Oostendorp (Jan. 2009). 'Current Practices in Serious Game Research: A Review from a Learning Outcomes Perspective'. In: *Games-Based Learning Advancements for Multi-Sensory Human Computer Interfaces: Techniques and Effective Practices*. DOI: 10.4018/978-1-60566-360-9.ch014.

Yoshimura, Andrew and Christoph W. Borst (2021). 'A Study of Class Meetings in VR: Student Experiences of Attending Lectures and of Giving a Project Presentation'. In: *Frontiers in Virtual Reality* 2, p. 34. ISSN: 2673-4192. DOI: 10.3389/frvir.2021.648619. URL: https://www.frontiersin.org/article/10.3389/frvir.2021.648619.

Zero, Stress Level (2020). *BONEWORKS*. URL: https://store.steampowered.com/app/823500/BONEWORKS/ (visited on 7th Dec. 2020).

Appendix

A Performance Benchmarks

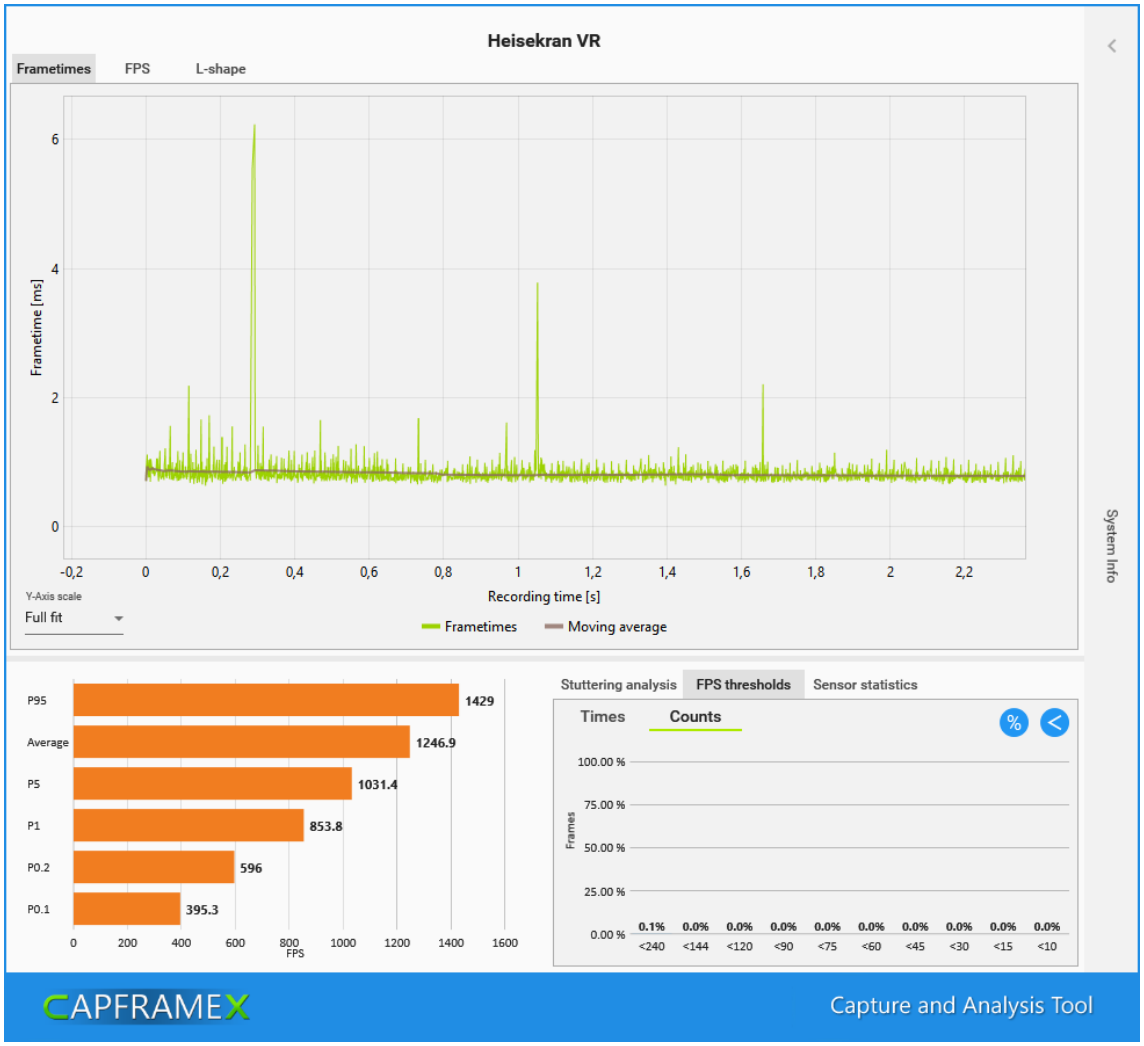


Figure 1: Benchmark on high-end desktop computer of the final desktop variant.

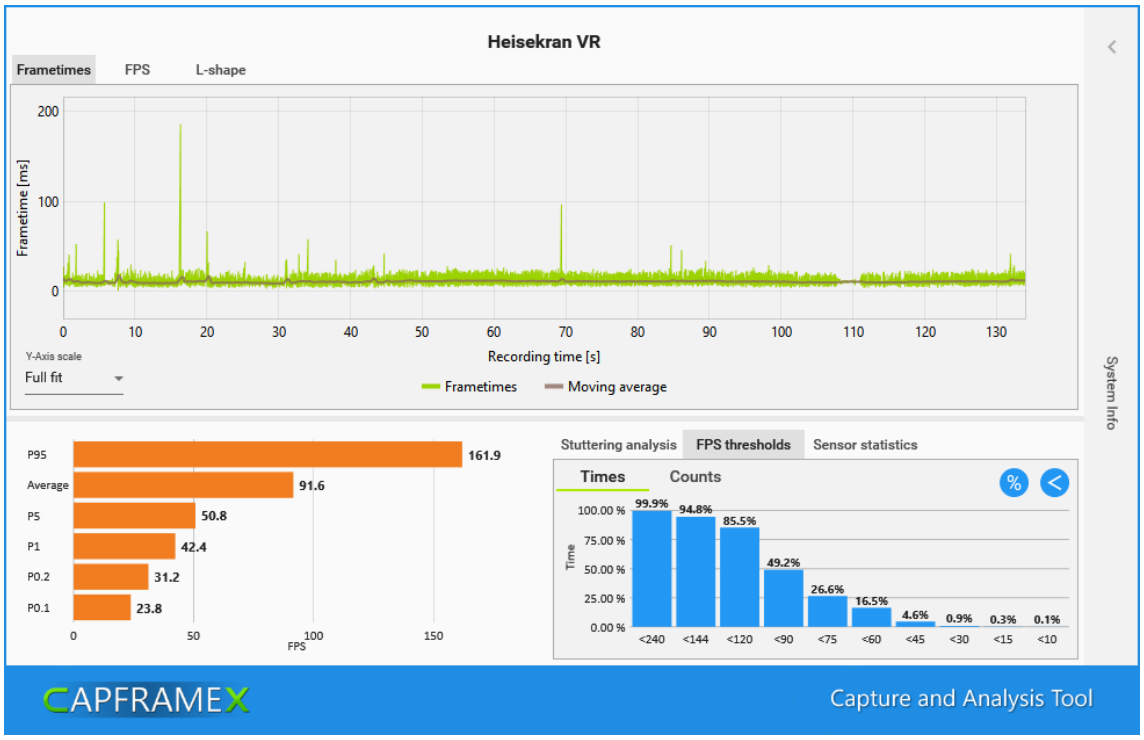


Figure 2: Benchmark on low-end laptop computer of the final desktop variant

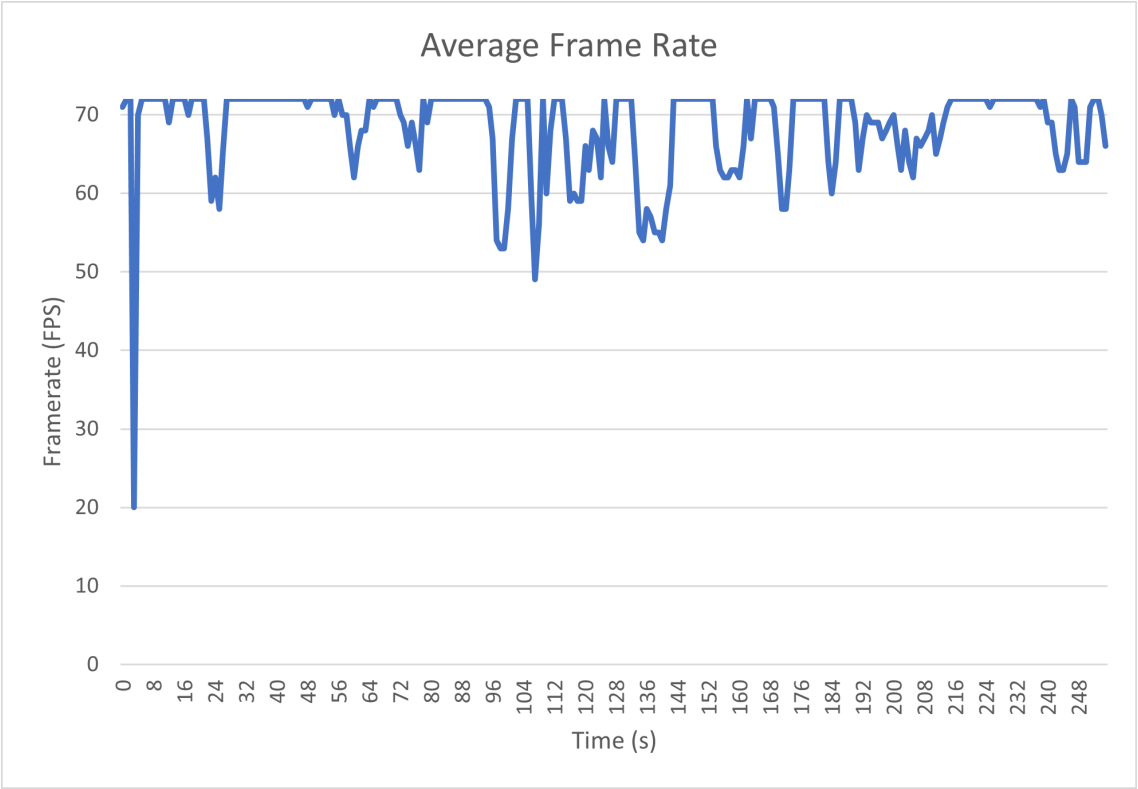


Figure 3: Benchmark showing average FPS on Oculus Quest 2 running the final standalone variant

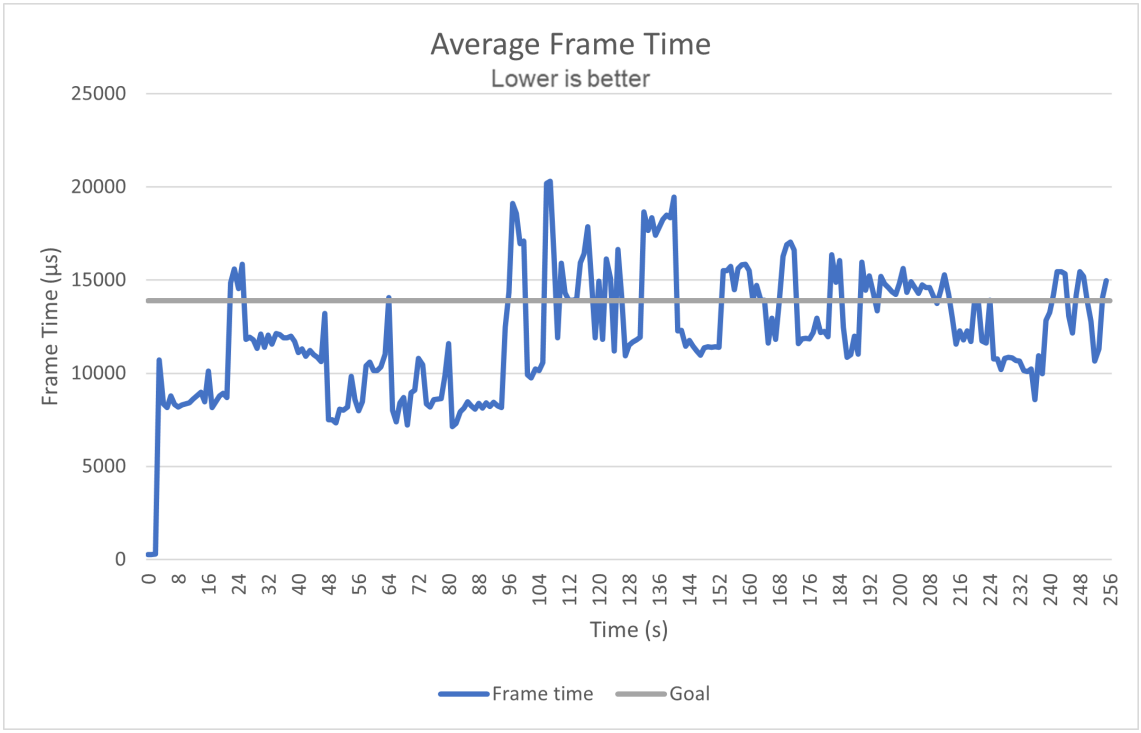


Figure 4: Benchmark showing frame times on Oculus Quest 2 running the final standalone variant. The grey bar shows 13,88ms of frame time, the goal that has to be met in order to render at the screen's 72hz refresh rate.

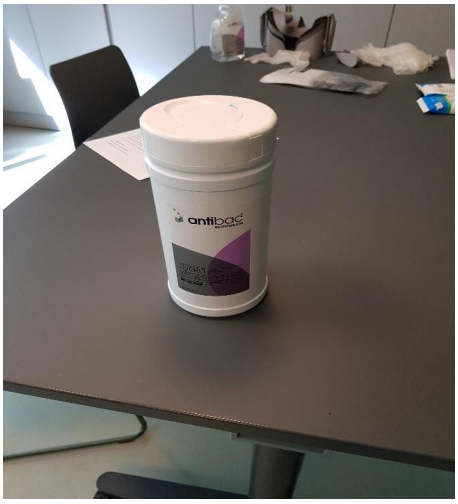
COVID procedure in the IMTEL lab

General lab procedures at NTNU: https://innsida.ntnu.no/wiki/-/wiki/English/Temporary+guidelines+for+necessary+work+in+laboratories+due+to+corona

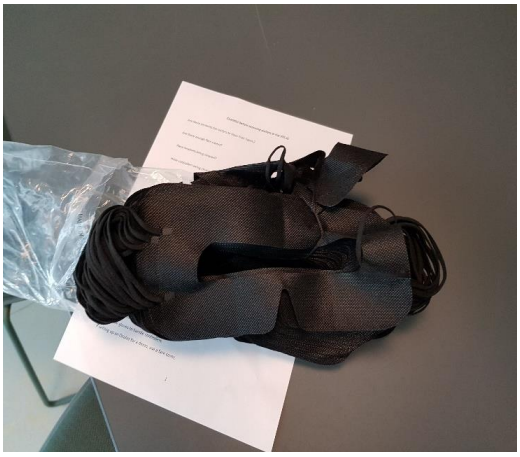
For project students and assistants:

Make sure to clean your hands before getting into the lab

Limit the number of things you touch. **Clean your workspace** (table, keyboard, mouse etc) before and after use. Try to limit it to what you need and clean anything after stop using it. There are wipes in the lab for cleaning. Consider using gloves to handle controllers. Avoid touching your face, use antibac on gloves/hands from time to time. **Register your stay (there are QR codes available). Maintain distance of 1 meter, if too many people at the lab, consider using Mars room (D113)**



Please wear a mask cover if you are going to use a headset. If you are a student assistant preparing a headset (for example, Oculus) you need to wear a cover.



Make sure to clean the headset cover before and after use. **ALWAYS** wipe the headset between users with sanitizer, both the part touching the face and the parts that might be touched by the user (on headsets with leather-like cover). **Not the lenses!** For headsets with textile cover (Oculus Go/Quest): wipe the parts touched by the users and **put them into the UVC machine.** If receiving visitors, **double-check** that the users wiped the controllers. If not, wipe them as well!

Tethered headsets in the stations need to be unplugged and put into the UVC machine after each user group.

Dispose of mask and used wipes on the dedicated bin.

Please do not throw anything else on this bin apart from wipes and used mask. Instruct visitors to throw their won masks and wipes in there.

Checklist before receiving visitors in the VRLab

Are there elements for visitors to clean their hands?

Are there enough face covers?

Have headsets being cleaned?

Have controllers being cleaned?

Is there someone who is going to tell visitors the norms for covid prevention in the lab?

Do you know the norms for covid prevention in the lab?

Procedure for visitors to the lab

Visitors to the VR lab will be reminded to follow covid-19 restrictions.

Visitors will be asked to provide their name and a way of contact to be informed in case an outbreak happens at the lab.

Visitors will be asked to wash their hands or clean hands with antibacterial before entering the lab

Visitors wishing to wear headsets are required to wear a disposable cover.

Visitors using controllers will be asked to use antibac wipes themselves and clean the controller before use, so they can be sure that it is clean.

Visitors will be asked to maintain a minimum distance from each other.

Informasjonsbrev og samtykkeskjema

Forespørsel om deltagelse: utvikling av virtuelle praksisplasser/virtuelle jobbsmaker med spill-baserte elementer i virtuell og utvidet virkelighet (VR/AR)

I dette prosjektet vil vi utvikle forskningsbaserte 'virtuelle praksisplasser' ved hjelp av innovative virtuell og utvidet virkelighet (VR/AR) teknologier med spill-elementer. Vi vil finne ut om disse teknologier og spill-basert læring kan benyttes som verktøy som motiverer og informerer brukere på vei mot arbeid. Gjennom simulering av en arbeidsplass får brukeren et innblikk i hva de driver med, og prøver selv: for eksempel hva det innebærer å drive med oppdrett. NTNU er behandlingsansvarlig institusjon for studien.

Basert på resultatene fra en kartleggingsrunde blant NAV-brukere og ansatte og input fra arbeidsgivere har vi utviklet VR app-prototyper med spill-elementer som representerer arbeidsplassene/bransjene, både med VR briller og uten. Disse er elektriker/vindmølleoperatør, vei/anleggsarbeider, bilmekaniker, blikkenslager, lagermedarbeider, kranfører og varianter av disse (f.eks med samhandlingsfunksjon), pluss en VR 'yrkeskatalog' der alle appene er samlet. Vi skal nå teste noen av disse appene. DIN tilbakemelding er veldig viktig for oss for å lage innovative spill-løsninger i VR/AR!

Hva innebærer det å delta?

Du vil først få prøve VR appene. I prosessen kan vi be deg 'tenke høyt' og kommentere det du gjør. Etter utprøvingen vil vi be om tilbakemeldinger i form av deltakelse i fokusgrupper, spørreskjemaer og evt. i intervjuer. De som ønsker å delta i videre runder, vil være med og evaluere mer avanserte spill-prototyper som vi utvikler.

Hvilke data samles inn?
Det samles inn lydopptak (under intervjuene og evt. under testingen), bilder og svar på spørreskjema. Det kan bli tatt bilder av deg og de andre deltakere under demoer for å dokumentere hvordan utstyret brukes. Dette vil være delvis anonymisert, da bildene vil for det meste bli tatt bakfra og ansiktene vil være skjult bak VR briller under mesteparten av utprøvingen. Diskusjonen i fokusgruppene/intervjuer og kommentarer under testingen skal tas opp med lydopptaker. Spørreskjema skal fylles på papir eler nettbrett/datamaskin. Disse dataene skal behandles konfidensielt. Det være aktuelt å foreta videoopptak med ansikt skjult bak VR/AR briller samt samle inn bruksdata fra selve spillene, f.eks. oppnådd poengsum, målt puls under spillsesjoner, tidsbruk, opptak av spillsesjoner osv.

Oppbevaring og bruk av data
Alle personopplysninger vil bli behandlet konfidensielt. Deler av opptakene vil bli transkribert (skrevet ned) og lagret elektronisk. De skriftlige dataene vil bli avidentifisert, slik at opplysningene ikke kan knyttes til enkeltpersoner. Alle data vil bli oppbevart i henhold til gjeldende regler for forsvarlig lagring av personopplysninger og kun personer knyttet til prosjektet vil ha tilgangs til disse. Alle data vil bli anonymisert ved prosjektslutt (31.12.2021), og det er kun anonyme data som kan bli gjort tilgjengelig etter prosjektets avslutning. F.eks. lyd og evt. videoopptak vil bli slettet når transkribering og analyse av dataene er avsluttet og senest ved prosjektets slutt bortsett fra utvalgt video- og fotomateriale der ansikter ikke er synlige. Disse og opptak fra innsiden av spillene vil kunne bli brukt for demonstrasjoner i forskningssammenheng på en slik måte at ingen informasjon vil være knyttet til enkeltpersoner. Avidentifiserte data kan bli brukt i vitenskapelige publikasjoner og i arbeid med å videreutvikle innovative løsninger for brukeroppfølgning.

Dine rettigheter
Så lenge du kan identifiseres i datamaterialet, har du rett til:

- innsyn i hvilke personopplysninger som er registrert om deg, og å få utlevert en kopi av opplysningene,
- å få rettet personopplysninger om deg,
- å få slettet personopplysninger om deg, og
- å sende klage til Datatilsynet om behandlingen av dine personopplysninger.

Hvor kan jeg finne ut mer?
Hvis du har spørsmål til studien, eller ønsker å benytte deg av dine rettigheter, ta kontakt med:

- Prof. Ekaterina Prasolova-Førland (NTNU), ekaterip@ntnu.no, tlf. 99440861.
- Vårt personvernombud: Thomas Helgesen, thomas.helgesen@ntnu.no, tlf 93079038.

Frivillig deltagelse
Deltagelse i dette forskningsprosjektet er frivillig og samtykke kan trekkes tilbake når som helst. NAV-ansatte får ikke tilgang til råmaterialet med personopplysninger, men kun anonymiserte data. Det vil ikke få noen innvirkning på NAV-brukernes forhold til NAV dersom de velger å takke nei til deltakelse. Prosjektet er meldt til Personvernombudet for forskning, Norsk samfunnsvitenskapelig datatjeneste AS.

Erklæring om samtykke

Jeg samtykker i at dataene fra studien kan lagres og brukes til forskning- og utviklingsformål slik det er beskrevet ovenfor

Navn_ Sted/dato _

Robert Eric Maikher

How to create accessible Virtual Reality (VR) experiences to aid young job seekers with career guidance

NTNU
Norwegian University of
Science and Technology