Remote Handling within the Active Cells Facility at the European Spallation Source, Using Digital Reality Techniques

Emil Boman and Lukas Smisovsky

DEPARTMENT OF DESIGN SCIENCES
LUND UNIVERSITY
2016

MASTER THESIS

EUROPEAN
SPALLATION
SOURCE

Remote Handling within the Active Cells Facility at the European Spallation Source, Using Digital Reality Techniques

Emil Boman and Lukas Smisovsky

LUND
UNIVERSITY

Remote Handling within the Active Cells Facility at the European Spallation Source, Using Digital Reality Techniques

Published by

Department of Design Sciences
Lund University
P.O. Box 118, SE-221 00 Lund, Sweden

Subject: Interaction Design (MAMM01)

Supervisor: Joakim Eriksson, Lund University.
Co-supervisor: Magnus Göhran, European Spallation Source.
Examiner: Jonas Borell, Lund University.

Abstract

The aim of this thesis was to investigate the possibilities of using Digital Reality (Augmented Reality and Virtual Reality) techniques in the remote handling within the Active Cells Facility at the European Spallation Source. The remote handling within similar environments as the Active Cells Facility has normally been performed using radiation shielding windows. As the operations get more complex, and both Virtual Reality and Augmented Reality technologies get cheaper, more advanced, more robust, and easier to use, there is a growing interest in trying to apply these technologies for better control and monitoring within these environments. This thesis will try to answer the question of what requirements on hardware and software these kinds of solutions would have, and which designs would be most promising as these technologies get better.

Different ideas were explored by researching existing documentation and exploring existing solutions and products. Experiments on these ideas were conducted on different products that were commercially available at the time. Different solutions were tried using these products and were then evaluated using both informal and formal user tests. The final prototype was tested on 14 volunteers at the European Spallation Source.

The results from these tests indicated that the application of Digital Reality techniques to the remote handling within the Active Cells Facility could indeed prove to be very useful. It could improve the visualization of the operations inside, and increase the confidence among operators. The Digital Reality technologies are rapidly improving and the products could be powerful enough for this kind of application within a few years.

It is important to note that Digital Reality is not necessarily useful in and of itself. It is important to identify the tasks to be performed and the difficulties in performing these, but also the capability and limitations of the hardware at hand. Once these have been identified, it is easy to create an appropriate Digital Reality environment with complementing non-Digital Reality technology for efficiently performing the tasks.

Acknowledgements

We would like to thank Joakim Eriksson, our supervisor at LTH, who has been a constant advisor, teacher and friend to us throughout the progress of this thesis, and was always ready to help us whenever we seeked consultance. We would like to thank Magnus Göhran, our supervisor at ESS, for choosing us to work on this thesis, and for his encouragement and absolute faith in our work, skills and judgement. We would like to thank our coworkers at the Target Division, and all the people at ESS for providing us with valuable material and feedback. We would also like to thank Jarich Koning for his continuous support and valuable advice.

Finally, we would like to thank our families and friends for all the support and encouragement they provided us with throughout all our years of study and through the time this thesis took place. Without them, this accomplishment would certainly not have been possible to achieve. Thank you.

Terminology

Augmented Reality (AR)
: The augmentation of the real world using computer-generated auditory, olfactory, visual, haptic, and gustatory input.

Virtual Reality (VR)
: The complete replacement of the real world with a virtual one, overriding the senses to give an impression of being somewhere else.

Mixed Reality (MR)
: The merging of the real world with a virtual one.

Digital Reality (DR)
: Umbrella term for Augmented Reality (AR), Virtual Reality (VR), and Mixed Reality (MR).

Head Mounted Display (HMD)
: A display device, worn on the head or as part of a helmet.

Vertex
: A point in 3D space.

Polygon
: A plane made up of three or more vertices.

Contents

1 Introduction

1.1 European Spallation Source

The European Spallation Source (ESS) is a multi-disciplinary research facility currently under construction in Lund, Sweden. By its completion, it will be the most powerful neutron source in the world (see Figure 1.1). It is an international effort of at least 17 European countries that will act as partners in the construction and operation of ESS. The facility is expected to be fully operational by 2025 [1].

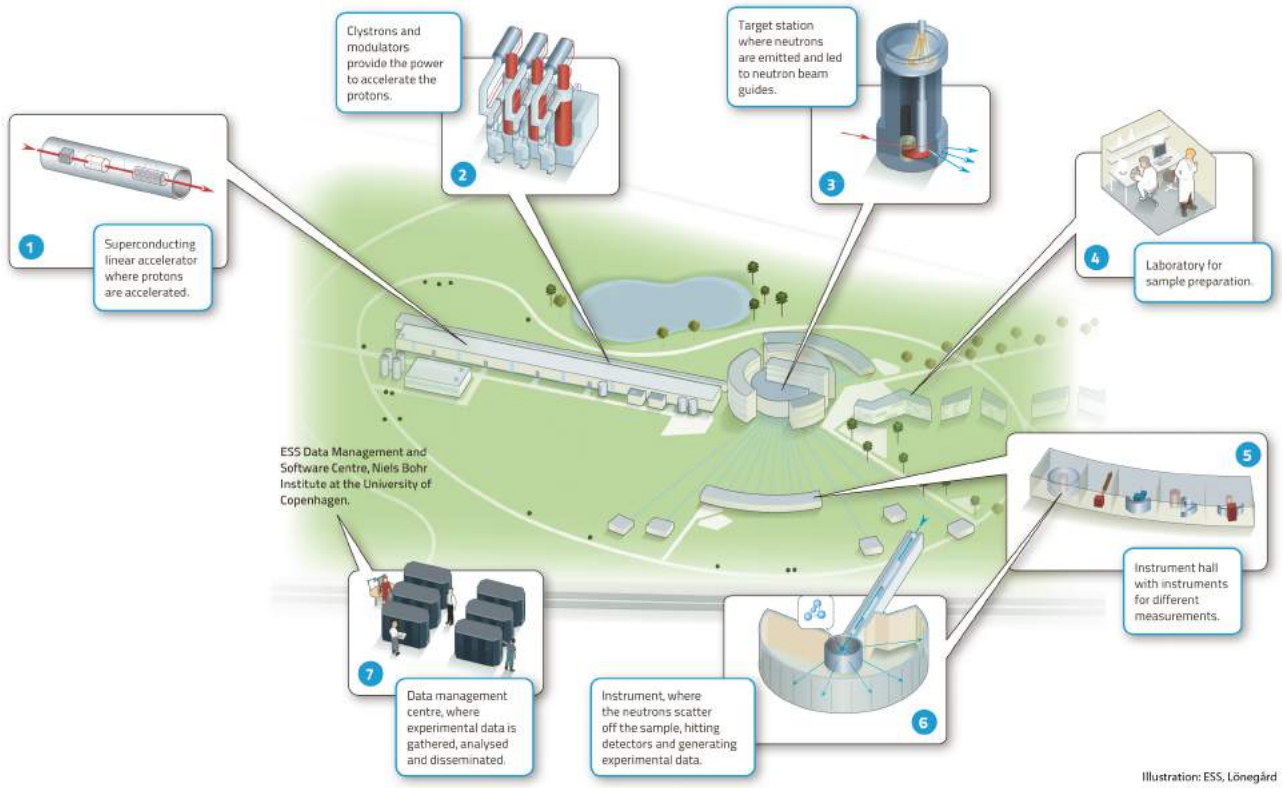


Figure 1.1: Brief description of the European Spallation Source

1.2 The Active Cells Facility

ESS consists of different departments and facilities which together form the components of what will be ultimately known as the European Spallation Source. One of these facilities is the Active Cells Facility that is designed and delivered within the Remote Handling Systems within the Target Division at ESS. During the design of the Active Cells Facility, the design group has faced a challenge concerning how the operations inside the Active Cells Facility will be performed [2]. This challenge is described in detail as follows.

The Active Cells Facility is a special type of a hot cell. A hot cell is an isolated chamber, which contains radioactive components and protects the exterior environment from harmful radioactivity (see Figure 1.2). Throughout many years, operations in hot cells have been performed under the surveillance by means of looking through a thick lead glass window. Usually, an operator stands outside of the hot cell and remotely performs operations inside it while supervising their operations through the window.

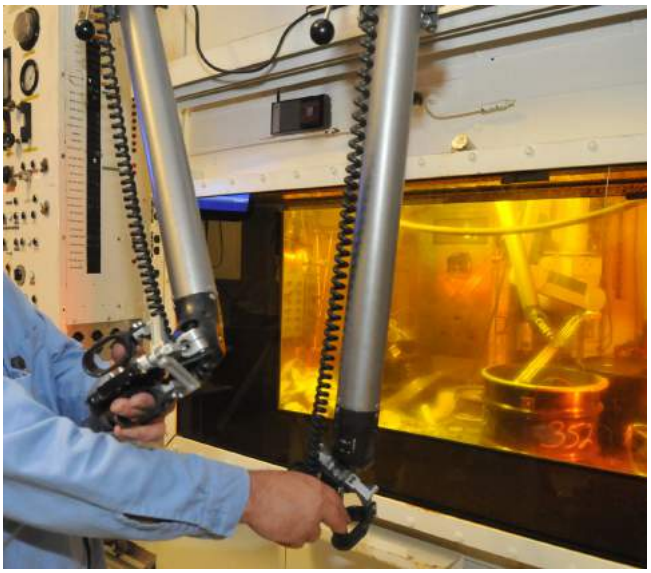


Figure 1.2: A typical small hot cell, with a lead glass window, and remote manipulation. The picture is for illustrative purposes and has nothing to do with ESS.

ESS aims to construct the Active Cells Facility with very thick and protective concrete walls, without any windows. The absence of windows is partly due to the fact that the Active Cells Facility has a big and advanced interior spaced geometry (as opposed to a classic small cube spaced interior) making it very difficult to provide a full insight into the whole space of the facility with just one or two windows. Also, thick lead glass windows for radioactive protection are very expensive, and a full surrounding of concrete walls provides a much more compact and safer protection against harmful radioactivity. Therefore, in order to be able to supervise the operations and processings of radioactive components inside the Active Cells Facility without any windows, the interior of the facility will be monitored by surveillance cameras (see Figure 1.3).

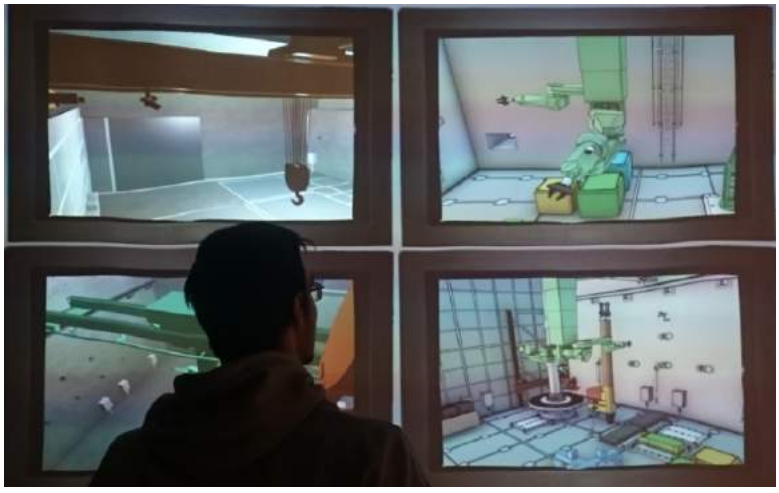


Figure 1.3: Camera surveillance in the Active Cells Facility. The picture is for illustrative purposes, it is not a real camera surveillance.

Operations inside the Active Cells Facility are remotely performed by an operator who is situated inside a control room. The Active Cells Facility is furnished with different equipment, e.g. manipulators, cranes, cameras, robots, etc. The operator remotely operates and manipulates with this equipment. The operations are quite complicated due to constraints caused by limited visibility through the surveillance cameras, as well as narrow spaces and complex operational tools and tasks. The real-time camera surveillance is an important feedback for the operator, but it has its limitations: from camera images it is difficult to perceive the correct locations of objects, their relative distance to each other, the depth in the scene, the low field of view, etc. To remedy for these constraints and improve the operability of the Active Cells Facility and enhance the visibility for the operator, Digital Reality (DR), i.e. Augmented Reality (AR) and Virtual Reality (VR), could be a viable and highly sophisticated solution.

1.3 Purpose and Goals

The main purpose of this master's thesis is to explore the range of potential uses of modern tools and technologies within VR and AR (both of which will be explained in the following chapter) for monitoring and operating the Active Cells Facility more naturally. It will present ideas, solutions, and examples, attempt to prove the technical feasibility of these ideas, and finally propose potential possibilities and solutions with the future technologies to come.

Goals

- Build a computer generated testing environment, which simulates the behavior, the remote control, the monitoring, and the most usual tasks performed inside the Active Cells Facility.
- Investigate different ideas on how to view and operate the Active Cells Facility with VR and AR.

- Develop a simple user interface prototype based on the most attractive ideas with the greatest potential.
- Test the interface prototype on certain users to investigate the feasibility of the prototype and the usage of VR and AR within the Active Cells Facility.
- Research on what requirements or implications DR sets on both hardware and software.

1.4 Overall Approach

In the design and development of the user interface prototype, a user centered design process is applied [3]. This means that the user interface prototype is designed with the end users in focus. All of the end users' wishes, needs and expectations are taken into account when designing the prototype, with the goal to produce a prototype which will satisfy the end users. In this thesis we divide the user centered design process in stages, and describe each stage in its own chapter throughout the thesis. The following chapters form together the whole design process: *Analysis of the Active Cells Facility Workflow*, *Conceptual Design*, *Virtual Control Room - First Iteration*, *Virtual Control Room - Second Iteration*, and *User Tests*. In *Analysis of the Active Cells Facility Workflow* we analyse and collect together all facts regarding the system the prototype is developed for, and which requirements the prototype needs to fulfill. In *Conceptual Design* we suggest a conceptual design of the prototype. In *Virtual Control Room - First Iteration* and *Virtual Control Room - Second Iteration* we implement the prototype. In *User Tests* we test and evaluate the prototype. The entire design process is performed iteratively.

1.5 Scope

We aim to create a prototype of a product - a proof of concept - in which we explore and demonstrate some possibilities and limitations using different VR and AR technologies. The objective is to use existing technology and hardware, not to construct our own. The proof of concept will demonstrate what is possible to achieve with AR and VR with this technology and how it might help in overviewing and operating the Active Cells Facility. Thus, the main focus is on the design and construction of the user interface prototype and not the hardware itself. If the user interface is satisfying, while the hardware is lacking in performance, then no great measures will be taken in improving the hardware. However, the hardware will be subject to evaluation of what requirements DR sets on it.

The user interface prototype is developed with a user centered design approach. As there were no actual users (the operators of the Active Cells Facility) at the time at ESS, we base our testing and evaluation of the prototype on documentation and interviews, and use well-known design principles to design as much of a user friendly interface as possible.

2 Technical Background

In this section we describe and explain the most vital and relevant technical areas behind this master's thesis.

2.1 Natural User Interface

A *natural user interface* is an interface that is based on natural, everyday human behavior and is therefore effectively invisible and does not have to be learned in order for it to be used [4]. One example of such an interface is a touchscreen: to press a button, one physically presses the button.

2.2 Augmented and Virtual Reality

Even though both *Augmented Reality* and *Virtual Reality* share many qualities, there are some nuances between the terms that are important to distinguish, so as not to create confusion.

Augmented Reality is the fusion of the real world with a virtual one [5], using computer-generated feedback (see Figure 2.1). The real world is analyzed using different input devices such as cameras and microphones, and then the virtual world is adapted to this data. The resulting model of this virtual world can then be projected back into the real world either visually, auditorily, or both, using devices such as monitors, head-mounted displays, speakers, and headphones.

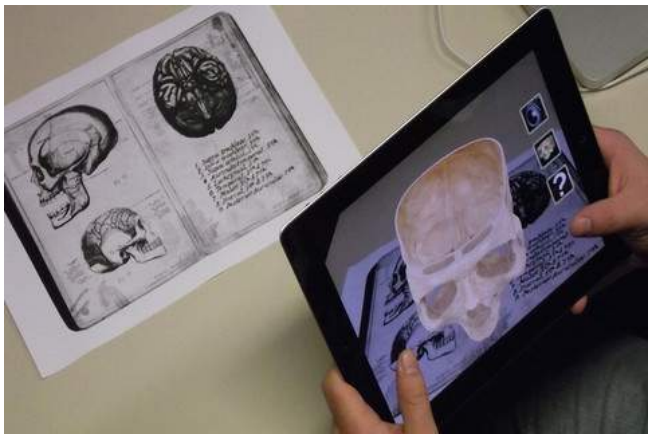


Figure 2.1: Augmented Reality in practice with a tablet and litterature in medicine.

Augmented Reality can be contrasted to *Virtual Reality*, in which a virtual world is experienced through the senses instead of introduced into the real world [5]. The result is that the user is fully immersed in a virtual setting and isolated from the real world (see Figure 2.2). The body movements are tracked, e.g. the position and rotation of the hands and the head. This is coupled with output devices that, instead of augmenting reality, overrides and replaces it with something completely virtual.

Figure 2.2: Virtual Reality in practice with an Oculus Rift and a game.

It should be pointed out that AR and VR should not be considered as two strictly different concepts. In fact, they should be considered related, as is discussed in *Augmented Reality: A class of displays on the reality-virtuality continuum* [6]. Although one can easily establish clear definitions about each concept and the differences between them for oneself, it can sometimes be hard to determine where one concept ends and where the other one starts. Sometimes the difference is established depending on the hardware, sometimes it is a question of philosophical understanding, and sometimes it is just a matter of opinion.

We would argue that both AR and VR could and should therefore be gathered under the umbrella term *Digital Reality*. This term would imply that one or more of the senses have been altered or heightened using digital representations of reality, alterations of reality, or a completely fictional one. The term was first coined by Adam Draper at Boost VC, as reported by Eric Johnson of Recode [7].

2.2.1 History

Both *Augmented Reality* and *Virtual Reality* are not new concepts and have actually been around for a while. Dreams of entering and experiencing other worlds have been around since the dawn of mankind, and ever since the first computers people's imaginations of digital worlds have had no boundaries. Back in 1955 and 1962 Morton Heilig designed and built, respectively, the *Sensorama* [8], which is famous for its aspirations within the realm of VR (see Figure 2.3). However, because computing power was so far from being enough for the kind of calculations that would be needed, the Sensorama was mechanical and non-interactive and could be described as being an immersive movie. It wasn't until recently that VR had gained enormous popularity outside of fiction, because of

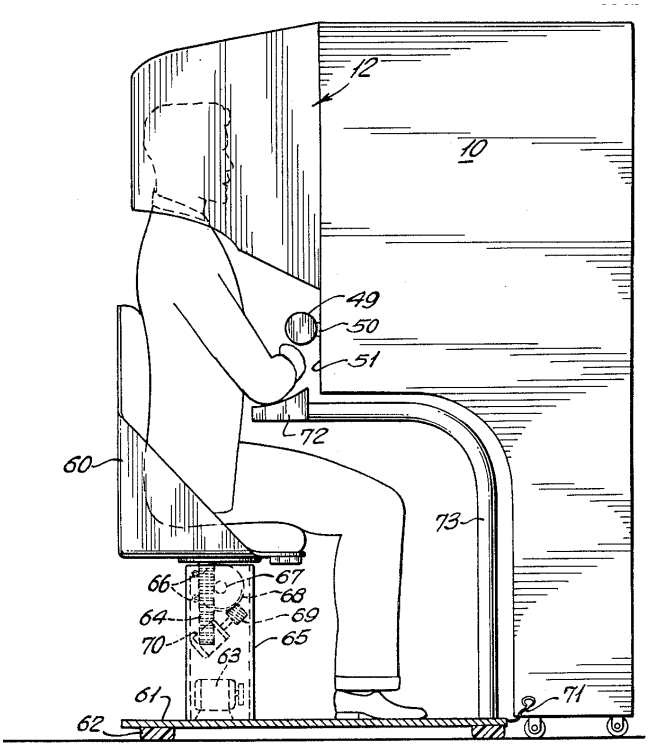


Figure 2.3: Sensorama

the increase in computational power, and the improvements that had been done in order to generate convincing virtual worlds.

2.2.2 Head-Mounted Displays

A *head-mounted display* (HMD) is a display device worn on the head. HMDs can be divided into two types: *closed-view* HMDs and *see-through* HMDs [9]. *Closed-view* HMDs do not allow the user to view the real world, as opposed to *see-through* HMDs that allow the user to view the real world by different means, as either *optical see-through* HMDs or *video see-through* HMDs. *Optical see-through* HMDs are HMDs with transparent visors or mirrors, and have the capability of both allowing the user to see the real world and reflecting projected images back to the eyes of the user. *Video see-through* HMDs are HMDs with a video display screen on the visor, and allow the user to see the real world through one or two cameras on the HMD.

In combination with the different HMD types mentioned above, HMDs can also be *monocular* or *binocular*. Monocular HMDs show only one image before the eyes, and work mostly just like a suspended monitor. A binocular HMD on the other hand projects two different images to each eye, and thus gives a sense of depth. This effect is called *binocular disparity* and is based on the simple fact that the image of an object seen from one point becomes slightly distorted when seen from a different

point. Our eyes are horizontally displaced and thus see two different images [10].

2.2.3 Virtual and Augmented Reality Products

HoloLens

An emerging technology in AR is Microsoft's HoloLens, which is an optical see-through head-mounted display and is expected to be released "in the Windows 10 timeframe" [11] (see Figure 2.4). The glasses are able to monitor position, acceleration, and rotation and track your viewport to the virtual world (see Figure 2.5). The virtual world is then projected back onto the visors, and you can experience AR. The glasses are reported to have "very low latency."

Figure 2.4: Microsoft's HoloLens

Figure 2.5: Official demonstration of the HoloLens [12]

Oculus Rift

Another emerging technology is the Oculus Rift, developed by Oculus VR. It is a binocular closed-view HMD. The user is completely visually immersed by the projected world and is thus unable to see the real world. The Oculus Rift which is explored in this thesis is the Development Kit 2 (see Figure 2.6) [13].

A consumer version is expected to be released first quarter of 2016 [14].

Figure 2.6: The Oculus Rift DK2 HMD.

2.2.4 Interaction

Being able to experience VR/AR is one thing, being able to interact with it is another. The user could be able to see, hear, smell, and sense the other world, but could still not be able to interact with it.

Keyboards, mice, and microphones are examples of input devices we use nearly every day, and joysticks are quite common as well. These are all very familiar devices, but only the microphone could be considered a true natural user interface, as direct interaction with the actual device is usually not necessary in order to use it. To make a convincing VR/AR environment, we need a natural user interface and the interaction methods should ideally come completely natural to the user.

LEAP Motion

One such device that enables the kind of interaction mentioned in the section above is the LEAP Motion (see Figure 2.7), developed by its eponym LEAP Motion, Inc. [15]. It monitors the location of the user's hands and their fingers, and thus makes it possible to interact with the virtual environment physically (e.g. push a virtual button or pull a virtual lever, see Figure 2.8). It works by outputting infrared signals and capturing images using infrared cameras to sense the position of the objects in front of the sensor. The technology still has some way to go before it could be reliably used for improving productivity, but it is suitable for prototyping.

Figure 2.7: The LEAP Motion Sensor.

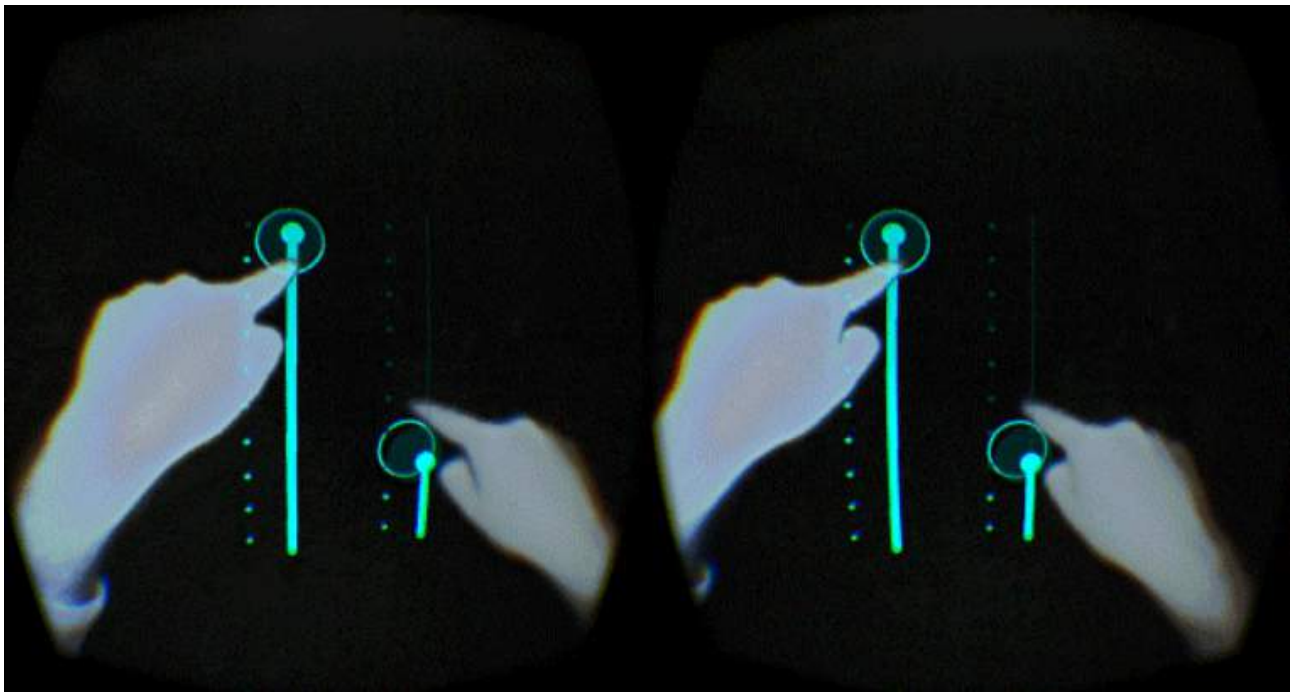


Figure 2.8: Manipulation of virtual sliders, as seen in one of the official examples released by LEAP Motion.

2.3 Computer-aided Design

Computer-aided design (CAD) is the use of computer software specialized in helping to create, to modify, to analyze, and to optimize a design, and model objects or environments. One example of such a software is AutoCAD [16]. CAD often uses vector graphics for precision. This means that it is possible to scale the blueprint until the desired size or level of detail is acquired. This is contrasted to rasterized graphics [17], where there is just one level of detail. This is illustrated in Figure 2.9.

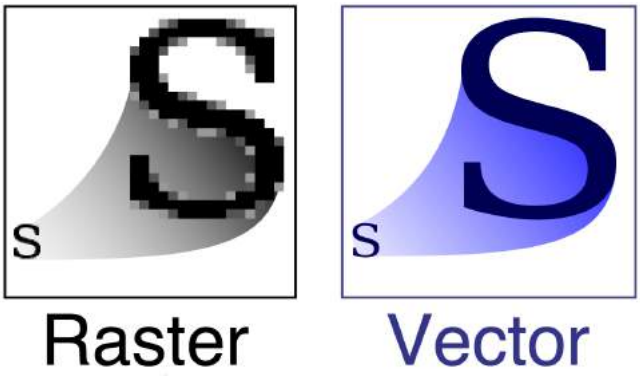


Figure 2.9: Comparison of raster-based vs. vector-based graphics.

CAD software is very often used in architectural design, as doing it by hand requires much more time and effort. As some aspects of detail have to be compromised to render something that's inherently three-dimensional on a two-dimensional plane, be it a monitor or a sheet of paper, using a three-dimensional virtual environment to visualize this could prove extremely useful and productive.

2.4 Game Engine

VR/AR applications visualized with computer graphics are usually created with game engines. A game engine is a software tool with the purpose of developing video games fast and easily [18]. Of course, a video game is usually related to entertainment media, but it can also be successfully employed for simulation/visualization purposes. With a game engine one can add physical properties to computer generated models (e.g. movement, gravity, colors, etc.), create environments with different appearances and events, and simulate elements which closely resemble real world elements (e.g. light, weather, and sounds) or completely imagined elements (e.g. liquid rainbow). The game engine of choice in this master's thesis is Unity (see Figure 2.10) [19].

Unity

Unity is a popular game engine which has the advantage of being portable to many platforms. The reason for choosing this tool in this thesis is because it is free of charge, and it is possible and easy to create VR/AR applications with it.

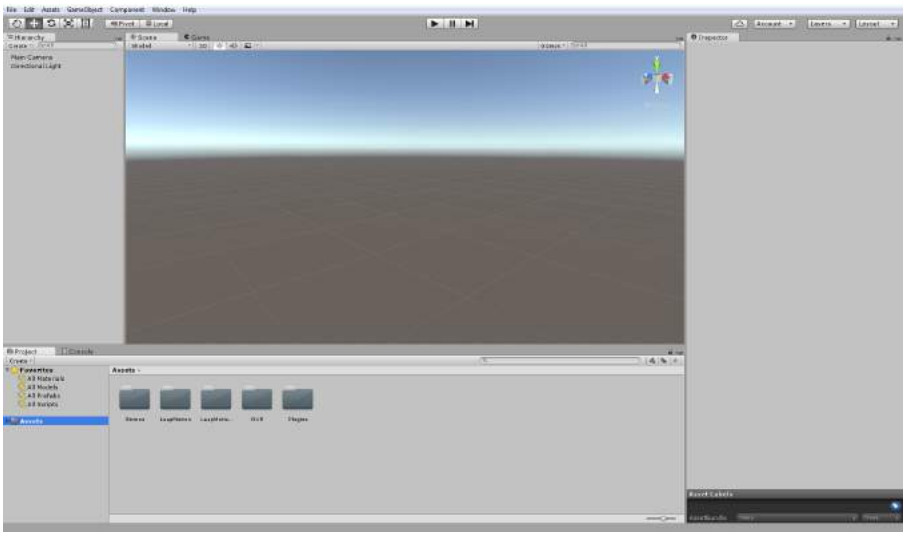


Figure 2.10: The Unity Interface.

2.5 Computer Vision

Computer vision is the use of different algorithms in analyzing images or an image stream from a camera. The goal is to identify and describe what is seen in the images, and where. This is important, because critical data could be hard to identify using the images alone.

Depth can be perceived using a variety of *depth cues*. These are typically classified into *binocular cues* (visual data from both eyes) and *monocular cues* (visual data from only one eye). When using only one camera, the only way to perceive depth is to use monocular cues. There are some monocular cues that are used by animals in their everyday life [20]: an object that is closer to the horizon is recognized as being further away than an object that is further from the horizon, and an object that is known to be of equal size to another is recognized as being at the same depth as the other if they are also perceived as being the same size.

These kinds of cues are not always available, and are not reliable enough to be used in a critical situation. The use of multiple cameras at different angles can solve this problem.

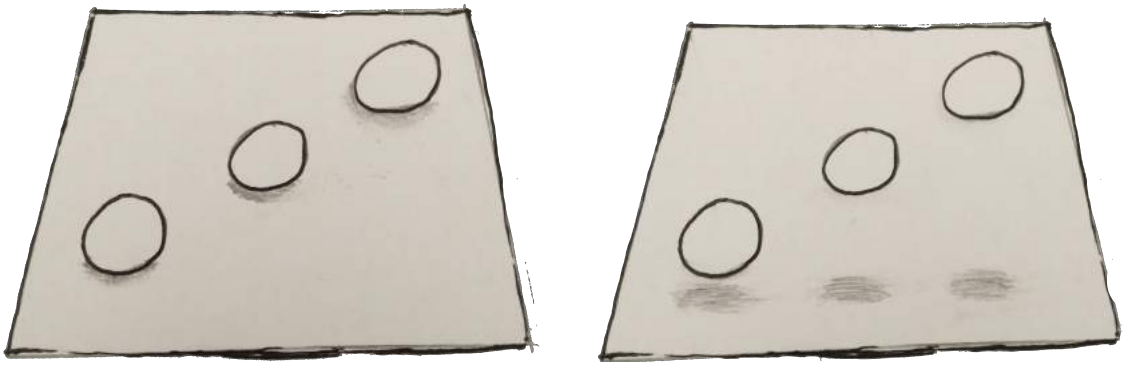


Figure 2.11: Without the shadows, the depth of the spheres is unknown, and the images would seem to be identical. Another camera from another angle would need to be added to determine the depth.

2.6 Shaders

A shader is a kind of graphical computer program used to do *shading*. It calculates the geometry, color, and appearance of an object, or the colors of the pixels within an image. It is often used to achieve a desired visual effect. It can also be used to perform parallelized mathematical operations [21].

3 Analysis of the Active Cells Facility Workflow

In this section, we will investigate the operations performed in the Active Cells Facility at ESS, and the technical specifications of the facility regarding its structure and the equipment it will contain. It should be pointed out that during the progress of this master's thesis, the Active Cells Facility had not been built yet. All technical specifications were fetched from the documentation provided by ESS, and extracted from interviews with the staff of the Active Cells design group. Until the final Active Cells Facility has been built, the technical specifications - and thus the operations as well - may be subject to change.

3.1 Approach

The purpose of the analysis was to collect vital information to compose a list of necessary system requirements. The aim was to overall find out more about the following:

- Active Cells Facility appearance and structure
- Scenarios in the facility, i.e. usual tasks to be performed
- Use cases and requirements
- Difficulties in performing the use cases with the traditional system
- User expectations from the system
- Modern and future technical features and solutions that would be useful or relevant for the system

The research was conducted mainly by means of reading provided documentation, performing interviews and meetings, and searching through articles, the Internet and literature.

3.1.1 Documentation

Using the provided project description and the documentation [22], we browsed through it to find answers to any of the points mentioned above. It was useful to browse through the documentation repeatedly, mostly for the purpose of discovering new key points that were not initially obvious, to identify core requirements that were really necessary for the system, and to decide which requirements were not as necessary as they initially seemed. The documentation mostly described the general necessity for a sophisticated visual feedback that would use the benefits of VR and AR, and a number of aspects to consider when implementing the system. However, the documentation did not say very

much about the general tasks performed inside the Active Cells Facility, nor how the facility would look. We needed to ask the engineers and the personnel responsible for further information.

3.1.2 Interviews

We had regular meetings with our employer where we presented our current ideas, asked questions and made suggestions. Our employer's answers were useful in the sense that it became more clear to us what we needed to focus on in order to implement a system that would be satisfactory.

To gain more information about the appearance, the structure, and the tasks inside the Active Cells Facility, we were advised to consult the people responsible for the graphical representation of the facility. They answered most of our questions regarding the structure and the general user tasks. They also provided us with pictures, links, computer graphic models, and short movie clips with graphical animations.

We had meetings with people that were experienced in designing interfaces that applied VR and AR. In these meetings we gained more information regarding the usefulness of Digital Reality (DR) in similar Human Machine Interfaces (HMI) and what we should keep in mind when designing the DR for the Active Cells Facility. These meetings helped us to understand how the operators operate with similar HMIs, how they define VR and AR respectively for themselves, and what kind of difficulties they are facing while operating different machines (e.g. manipulators and cranes).

To further enhance our understanding of the tasks performed inside the Active Cells Facility, identify the difficulties in visibility and operability in performing those tasks, and remedy these difficulties with suitable DR solutions, we should have preferably interviewed the end users, i.e. the users that would remotely operate the facility. The experience and knowledge of these users would have been useful to us, and could have narrowed down our research to topics of greatest priority, and help us to focus more on what we should think about while generating ideas for the system. Sadly, we had no access or contact with the end users, mostly because ESS was still lacking an operation organization but also since experience was hard to find. Thus, we had to use the information we gained from the documents, interviews, and meetings we had so far.

3.1.3 Scenarios

Scenarios are descriptions or stories of a user's deeds/tasks/routines in the situation or environment for which the user interface is being built [3]. A scenario can describe events or routines a user will usually or most probably partake in when using the interface. It can also be a realistic story which puts both the user and the interface in a certain context, describing how the user may behave and interact with the interface in that context. The benefit with scenarios is that it forces the designer to think about the use and the impression of the prototype design in that certain situation. It will make the designer suggest ideas and improvements to the design which will work in the scenario or help the user to complete the scenario.

3.1.4 Use Cases

Use cases describe all the specific actions which occur in the interaction between a user and a system [23]. This system might be for instance a car or an elevator, and a use case is the action or goal which the user wishes to achieve in this particular system, e.g. start the car engine or move up with the elevator. Whenever the user interacts with the system, a specific action is supposed to occure, preferably one which the user expects to happen. In the system/situation for which the user interface is being built, there is a specific number of possible use cases. The objective is that the user interface is supposed to cover all the use cases and be able to perform each and one of these. It is good to identify the use cases in a system to set the requirements for what the user interface is supposed to handle and what design choices should be made.

3.2 Technical Specifications

From the documentation as well as the videos and visual material we obtained from the engineers at the Active Cells design group, we summarized the following technical specification regarding the structure of the Active Cells Facility and the equipment contained inside it.

3.2.1 Structure

The Active Cells Facility is a big chamber which consists of two rooms (see Figure 3.1). The chamber is in a form of a cuboid. The walls, the ceiling and the floor of the chamber are each made of a very thick concrete block which is supposed to protect against any radioactivity. There are no windows anywhere in the Active Cells Facility. There is a hole in the ceiling above the second room in the picture. Through this hole, the Target Wheel will be lowered into the chamber.

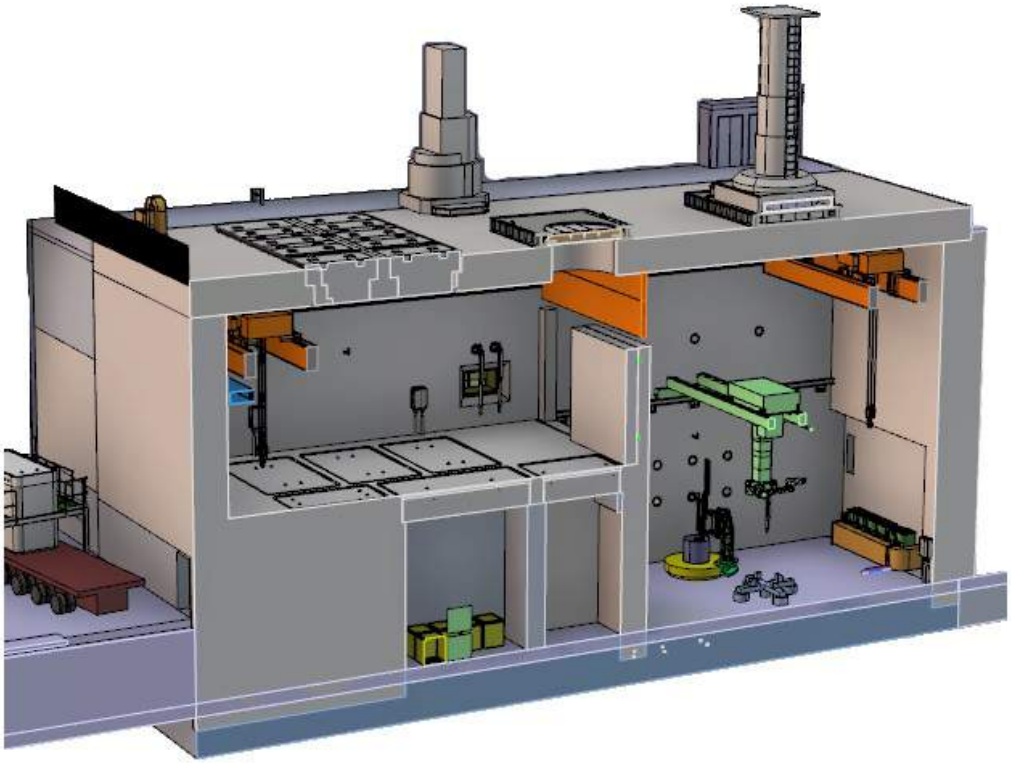


Figure 3.1: A CAD model of the Active Cells Facility, together with some of the basic equipment. The placement of the equipment in this picture is not definitive.

3.2.2 Cameras

The cameras that will be used in the Active Cells Facility need to be highly resistant to radiation, and need a reasonably high resolution and frame rate, in order to make any reliable computer vision possible. They also need to have a good range in panning and tilting, and a good optical zoom. They need to be easily controllable and fast to respond, and should be placed so as to facilitate as wide and relevant a view as possible.

3.2.3 Target Wheel

The target wheel consists of a thick disc placed onto a shaft (see Figure 3.2). The disc is filled with tungsten which is during operation exposed to high energy protons, resulting in an emission of neutrons which are then used for science. When the target wheel has reached its lifetime, it is inserted in the Active Cells Facility, fastened into place, and cut into pieces by a circular saw until it is disassembled to a state where it is possible to ship the pieces safely off site.

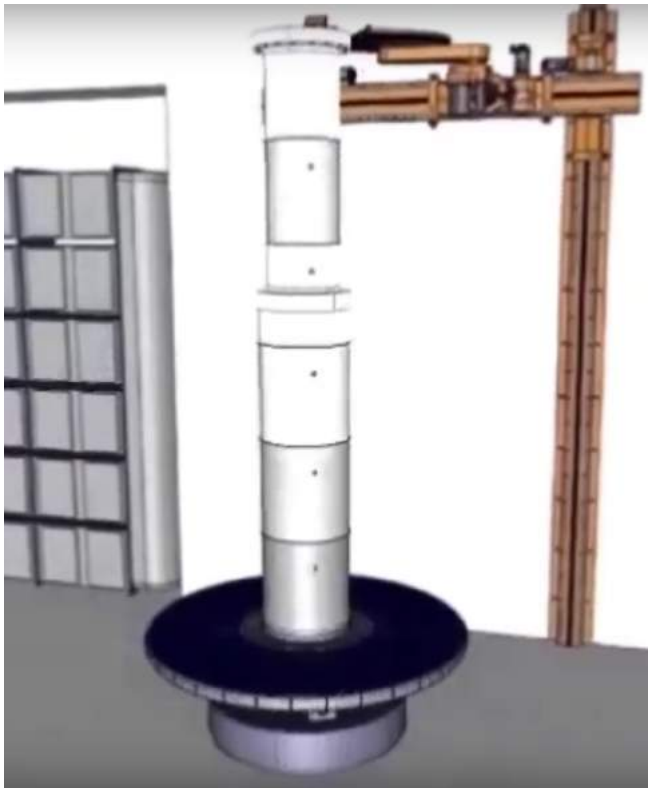


Figure 3.2: The Target Wheel in the Active Cells Facility.

3.2.4 Cranes

There will be two cranes in the Active Cells Facility. They are suspended alongside the walls on rails at a height of approximately 11 meters. The rails are approximately 30 meters long and span the entire length of the cell, while the cranes are approximately 2.5 meters wide (see Figure 3.3). The hook itself could be regarded as having a complete freedom of movement within at least one meter from the walls and down to the floor.

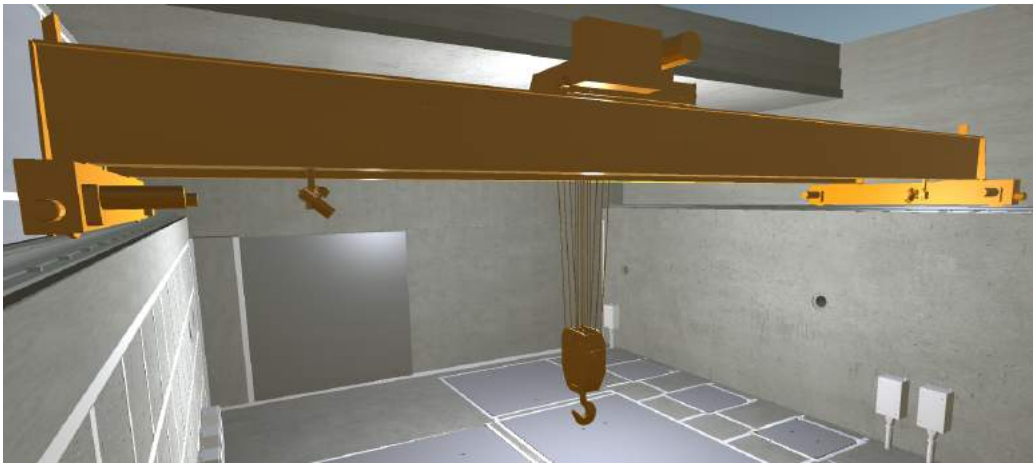


Figure 3.3: One crane out of two in the Active Cells Facility.

3.2.5 Manipulator

The manipulator is suspended on rails at a height of approximately 7 meters from the lower floor. The manipulator arms (see Figure 3.4) can reach any point within a certain radius. This, coupled with the fact that the elevator of the manipulator can adjust its height and rotate 360°around its axis, means that the manipulator arms can reach practically everywhere underneath the manipulator itself.

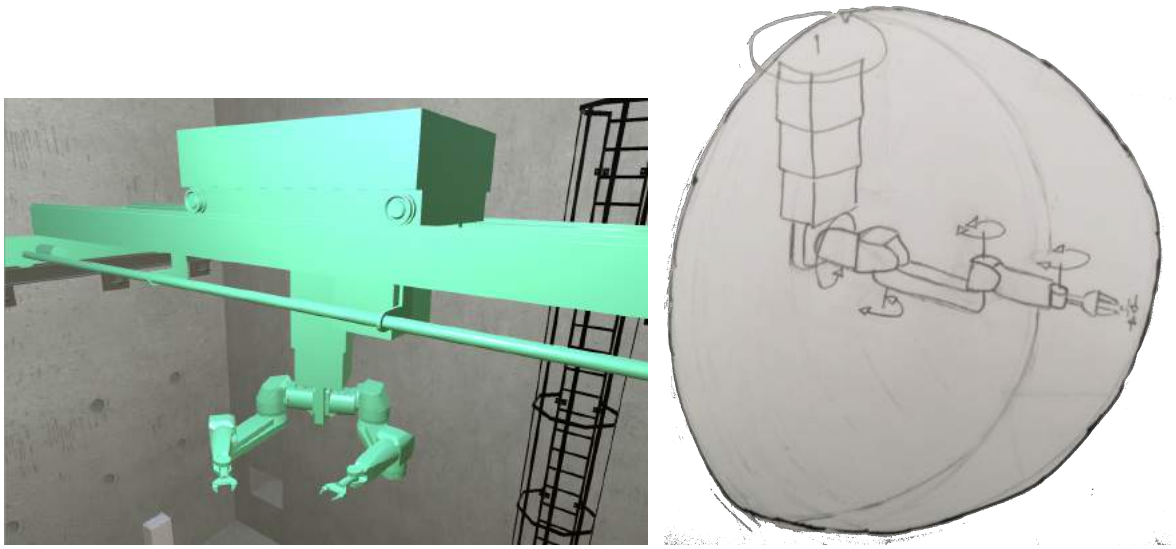


Figure 3.4: The manipulator, with the elevator and one of its arms with all its joints and their freedom of rotation.

3.2.6 Saw

The saw is mounted on the short side of the cell (see Figure 3.5), close to the target wheel to facilitate dismantling. It has freedom of movement along the wall and can be extended. It can also be rotated

freely.

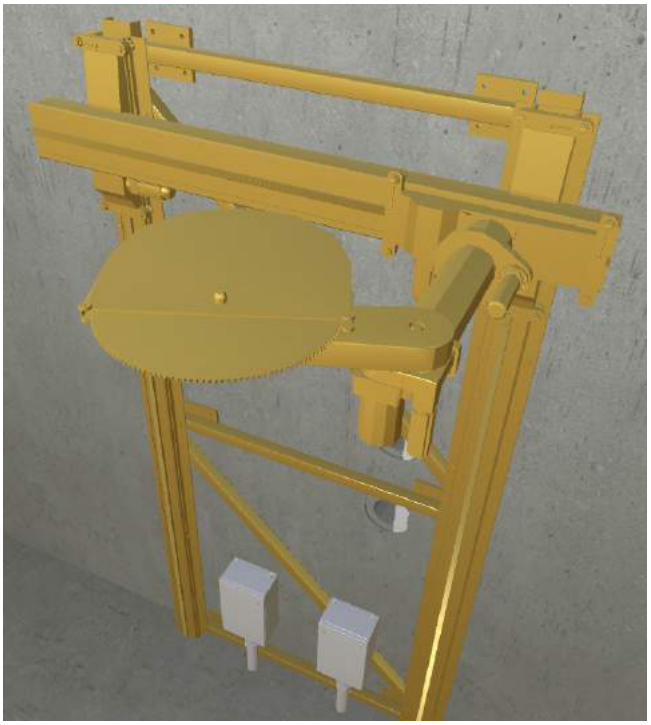


Figure 3.5: The saw in the Active Cells Facility.

3.3 Scenarios

After studying the documentation, as well as conducting meetings and interviews with our supervisor and the engineers at the Active Cells design group, two scenarios were summarized, illustrated with storyboards in Figure 3.6 and Figure 3.7 below.

Storyboards are images or illustrations in a sequence that describe a flow of events in time, i.e. a story. They are used widely in movie productions, animations, comics, demonstrations, etc. Their usefulness lies in the simple, fast, and inexpensive way to illustrate the general thought or idea of an action or event, without approaching more extreme measures in order to visualize the idea, e.g. building a full scale mock-up room. With storyboards, one can identify flaws in plans, switch or alter ideas, plan arrangements, and save time, cost, and effort in making crucial decisions before production begins [24].

3.3.1 Scenario 1: Deliver Target Wheel

1. The target wheel is lowered into the Active Cells Facility through a ceiling hatch.
2. The target wheel is put in place onto a fixture.
3. The manipulator is moved to the toolkit, where it grabs a screwdriver, and then is moved to the target wheel.
4. The manipulator clamps the target wheel to the fixture with the screwdriver.
5. The hoist that lowered the target wheel into the Active Cells Facility is disconnected from the target wheel and is pulled back up through the ceiling hatch.
6. The ceiling hatch gets closed. The target wheel is delivered.

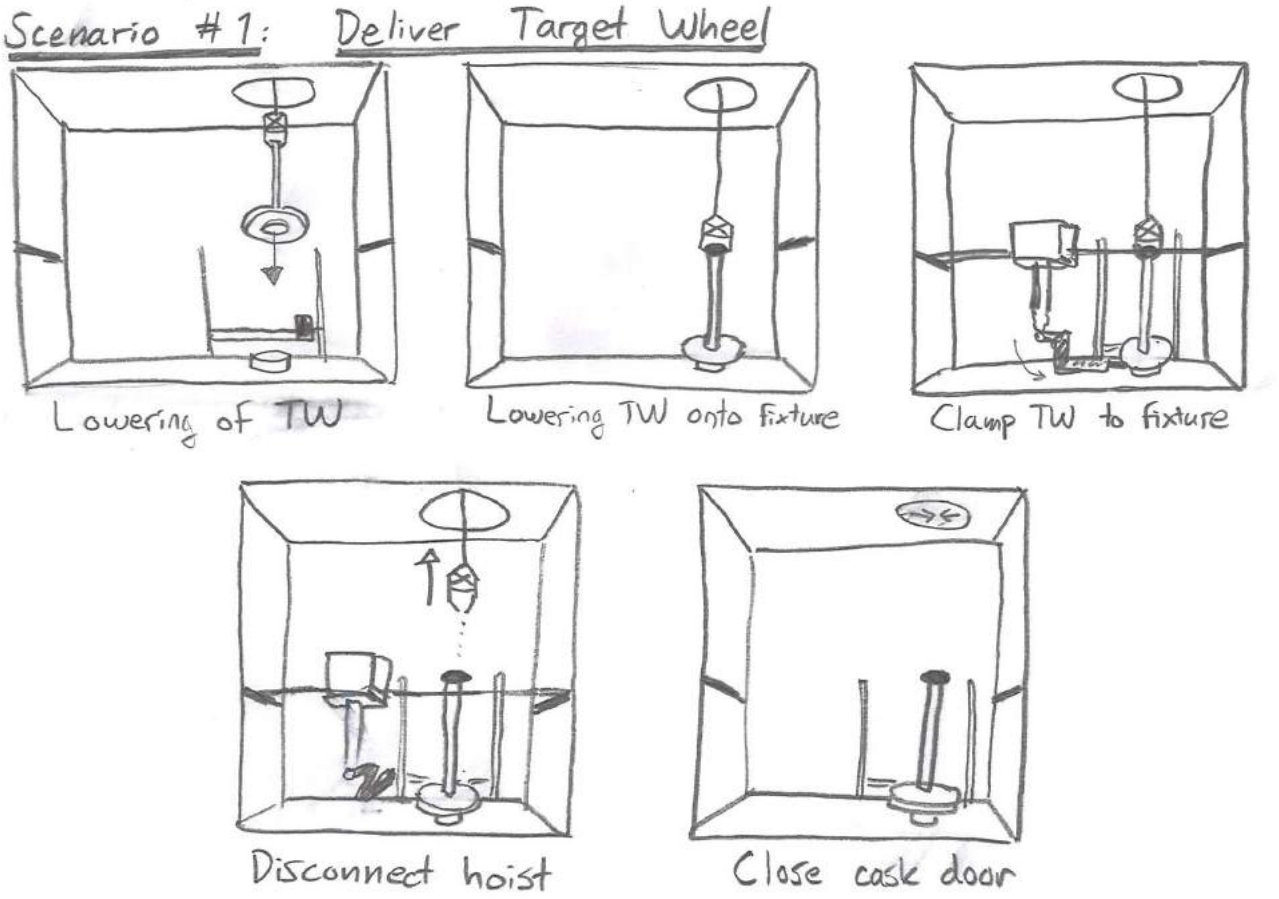


Figure 3.6: Scenario #1: Delivering the target wheel.

3.3.2 Scenario 2: Dismantle Target Wheel

1. The manipulator is moved to the top of the target wheel shaft, where it removes the cap with its hands.
2. The manipulator is moved to the toolkit where it fetches a small hook. Then the manipulator is moved to the pulley of the crane where it attaches the hook to the pulley.
3. The manipulator and the crane are moved to the top of the target wheel shaft. The pulley of the crane is lowered to the cap at the top of the target wheel and the manipulator hooks the pulley to the cap. Then the cap is removed by raising the pulley of the crane.
4. The manipulator and the crane are moved away to the toolkit where the manipulator removes the hook from the pulley.
5. The manipulator attaches a lifting adaptor to the pulley.
6. The crane is moved to the top of the target wheel shaft, the lifting adaptor is lowered onto the top and grabs it firmly.
7. The shaft below the lifting adaptor is cut with the saw. The shaft consists of an outer layer and an inner shaft. The saw cuts the outer layer first.
8. The cut piece of shaft is pulled up with the crane and is placed in a small container box.
9. The crane is moved back to the top of the target wheel shaft and the steps 6-8 are repeated with the inner shaft. This process is repeated until the entire shaft is cut and placed in boxes.
10. Finally, the disc on the target wheel is cut into smaller pieces with the saw.
11. The manipulator grabs the disc pieces and places them into the boxes.

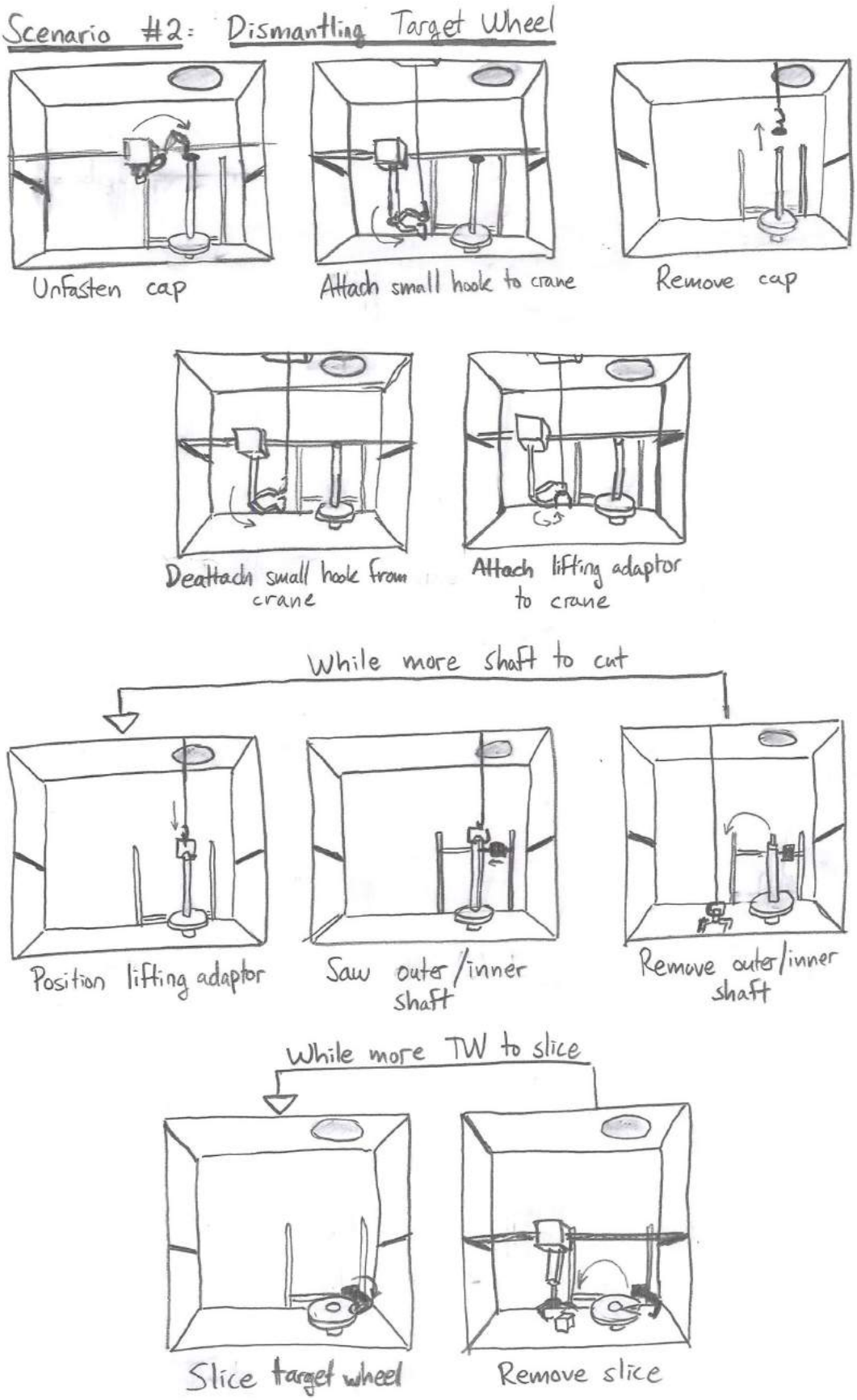


Figure 3.7: Scenario #2: Dismantling the target wheel.

Due to time constraints, we aimed to build and implement our prototype to be able to fulfill at least one of these two scenarios as efficiently as possible. We studied the scenarios to identify the most basic use cases. At ESS we interviewed one engineer who previously worked with VR and AR in hot cells. He provided us with useful and valuable advices, particularly the fact that the kind of AR that might turn out to be useful depends heavily on the type of tasks that are to be performed inside a hot cell, and that AR with minimal features and intuitive metaphors is preferred. We agreed that we should attempt to follow these advices as much as possible when developing the prototype. Therefore we carefully identified the use cases in the scenarios, which are listed below:

Use Cases

- Open/close the ceiling hatch
- Lower/raise hoist with/without something attached
- Move the manipulator/cranes
- Control the manipulator arms to grab a tool or put a tool back in place
- Control the manipulator arms to clamp the target wheel with a tool
- Control manipulator arms to unfasten cap
- Control the manipulator arms to grab a hooking device and connect/deattach it to the pulley of the crane
- Lower/raise the crane pulley to/from a specific point
- Use the saw to cut the shaft and the wheel
- Control the manipulator to grab a disc piece of the target wheel and place it into a box

We had at this point documented all the use cases we believed the system had to fulfill in order to be satisfactory. Once we defined the technical specification, the scenarios and the use cases as requirements, we could complement with our own judgment to create a conceptual design for our system.

4 Conceptual Design

In the previous chapter we identified all the necessary requirements the system had to fulfill. In this chapter we will instead focus on the design process we applied to come up with a design that would sucessfully cover most or all of these requirements.

4.1 Approach

The method of the design process was to use the information collected during the analysis phase to conduct experiments, evaluate the results, and then repeat the process iteratively until we obtained a satisfactory prototype system.

Once we had gained some information about the requirements and the necessary use cases, we divided the information into clear tasks and prioritized these. Before starting any implementation of software or hardware, we wrote down and sketched our visions and ideas of the tasks. Finally we debated the ideas until we reached a consensus on the best ones and implemented them. The implementation phase can be divided into the following parts: tasks and problems listing, idea generation, visualization, and implementation.

Tasks & Problems listing

With the information gained from our reseach, we needed to identify and list all individual tasks and problems associated with the operations performed in the Active Cells Facility. This was not a difficult task overall, but sometimes a task or problem could seem simple at first, and after further insight could turn out to be divisible into smaller subproblems. Usually, the list was short and it did not take a long time to complete. Once the listing was finished, we sorted the tasks and problems by priority and focused on the ones with greatest priority.

Idea generation

Once a list of tasks and problems was completed, it was time to generate ideas that might contribute to fulfilling the tasks in a more efficient manner and remedy for the difficulties and problems that would arise during the completion of the tasks. We aimed to keep the ideas familiar to the traditional system and the metaphors related to it. For the tasks and problems with the greatest priorities, we started to generate different ideas on how we could solve them. For this we used brainstorming.

Brainstorming consists of sitting together as a group and suggesting any kind of idea that comes up to anyone's mind. The idea may be good, bad, wise, crazy, or of any other kind of nature. The key rule is that no ideas are dismissed, all ideas are allowed. All ideas are recorded without judgment and once the group has no other idea to contribute with, the best ideas are chosen through voting. Alternatively, a new idea is composed from the other ideas. The main point with brainstorming, either for a group or for an individual, is to give room for inspiration and creativity. One idea can inspire another idea [3][25].

Metaphors are a kind of relatable interaction experiences. When a user is interacting with an interface attached to a software or a machine for the first time, they usually approach the interface with knowledge or experience gained from an earlier interaction in a familiar situation in life, e.g. *"Where is the Start button?"*. This situation might be, for instance, an interaction with a desk environment, putting groceries in a cart before buying them, ordering a pizza, following a map, aiming with a rifle, etc. Whenever designing a new interface or a system, it is appropriate to design an interaction which applies a metaphor the user might be familiar with. An interface with such a property might then encourage the user to navigate through the interface more easily, faster, with more familiarity, and without hesitations [3][24].

Visualization

The tasks, as well as the ideas generated during brainstorming, might have appeared good and clear at first glance, but up till that point, these components were presented only in textual form. Although it might have seemed there was little room for misinterpretation or different imaginations of the idea, there was always a risk it could happen. Each developer in the group might have had their own individual interpretation of the task, problem, or idea. Perhaps when presenting an idea in textual or verbal form to the employer or the user, they might not comprehend the entire idea as naturally as the developers, because they have not spent equal amount of time to think through the ideas and considered all the surrounding factors. It was therefore preferable to visually interpret the tasks and the ideas and sketch the interpretations down on paper. We mainly used storyboards or simple idea sketches for visualization. This approach is effective in the sense that new revelations about an idea might arise, either bad or good, and show the real potential of the idea before implementing it.

When we had visualized all of our ideas and tasks, we could finally choose which ideas to implement. The ideas were debated and evaluated using different criteria (such as usability, time to implement, cost to implement, potential, etc.) [25], and were finally chosen when a consensus had been reached. We could then implement them in Unity. We kept the implementation process simple, so we could perceive the basic potential of our ideas as fast as possible.

4.2 Experiments

From the brainstorming sessions we generated different ideas, based on early understandings of what the prototype was supposed to be able to cover. The ideas mostly touched upon the visual feedback of the prototype:

- Synthetic Viewing: representing real objects and operations from the real world with 3D graphics, and use these to create helpful AR.

- Click-To-Navigate Cameras: watch a camera from another camera and click on the visible camera to change perspective.
- Virtual Window: a monitor which simulates the property of looking through a normal window.
- VR where the Active Cells Facility is simulated, together with small monitors inside the VR, showing the real world from a camera's perspective.

We selected the most appealing of these ideas and conducted a number of experiments which could have the potential to evolve into the foundation stone of the final prototype which we would implement in the end. Here we present all the experiments.

4.2.1 Virtual Window

We started to think of a new way of interacting with the Active Cells Facility environment. Our first idea was a monitor (e.g. a big TV screen), which would create the illusion for the viewer of watching through a window. Using head tracking sensors, the monitor would project the visualized image in a different way depending on where the viewer's eyes were situated. This would make it possible for the user to look at an object from different sides. We considered this idea to be worth experimenting on since it could yield an interface which would exploit the metaphor of looking through a window in the wall. This metaphor could help the experienced end user operators familiarize themselves with the new interface by relating to the old interacting environment (i.e. remotely operating a hot cell by looking through a window) while offering new ways of interaction, e.g. AR on the TV screen, touch screen, the virtual window could be panned around anything in the Active Cells Facility, etc.

We started to study how this technique could be implemented and what was needed in order to achieve it. The basic technological necessities consisted of head tracking and projection. Head tracking is about tracking the position of the user's head with a device, e.g. an eye sensor, an infrared sensor, or a web camera. The real objective in this case was to track the position of the user's eyes, which are roughly at the same position as the center of the head. Once the position of the head would be obtained it would be sent into a simulation program in Unity which would apply the position of the head to a camera inside the simulation. The camera, perceiving the 3D scene in the simulation which the user sees on the TV screen, would mimic the movement of the user's head. Together with a correct projection of the 3D scene the user would be given the impression that the they are watching the 3D scene through a window or behind the TV screen. The projection makes sure that the view of the scene is projected in a correct way from the user's point of view, i.e. the head position, onto the camera view.

We experimented with the following devices for head tracking: The Tobii eye sensor [26], TrackIR [27], ordinary PC laptop integrated webcam, and Microsoft Kinect versions 1 and 2 [28]. We concluded that the Kinect version 2 was the most efficient and sophisticated head tracking tool, in the sense that it used pre-implemented computer algorithms which were fast enough for a real time feedback while being very accurate, and the camera in it had a large field of view. Before we could create a scene, we needed to make sure that the position coordinates of the head would be correctly transformed into the corresponding coordinates inside the simulation. Once we managed to do that, we created some scenes where we experimented with the projection. We implemented a solution for the projection which used the power of computer graphics shaders. The shader-based projection was very

satisfactory. Once everything was set up, we created different 3D scenes where our idea of a virtual window was well visualized.

Figure 4.1: The virtual window seen from the left and the right [29].

4.2.2 Active Cells Facility Simulation

We realized it would be a very helpful and supportive idea to design a simulation environment of the Active Cells Facility. So far, our experiments were based on mere assumptions of the difficulties that might surround the operability and the visibility inside the Active Cells Facility. In fact, we had no certain idea on how the facility environment looked like, how it felt like to maneuver inside it, and how it looked like from a camera perspective. Also, we had to test the feasibility of our ideas, specifically tailored for the Active Cells Facility. Therefore, we started to build up the Active Cells Facility simulation environment in Unity. We consulted the engineer who was responsible for the CAD model of the facility. Since the graphical representation of the facility was already finished as a CAD model, we were sure that it was fully possible to import the CAD model into Unity, and thus save a lot of time and work to create a new model.

There turned out to be difficulties importing the CAD model into Unity, since some parts of the Active Cells Facility model, as well as some equipment inside the model, contained a large amount of data (vertices) which Unity on our computer could not handle in real time. The CAD model was created as an .STP file, but unfortunately Unity can not import this type of file. However, Unity has extensive support for .FBX files (Autodesk's proprietary file format). Therefore we had to find a way of converting the 3D model into an .FBX file. Also we had to find a way to optimize the CAD model, i.e. to reduce the number of vertices in the model to be able to run it in Unity smoothly, while maintaining the quality of the CAD model. After some experiments, we found the following workflow to yield very positive results:

- Import the .STP file to Autodesk's AutoCAD.
- In AutoCAD, export it as a .3DS file.
- Import the .3DS file to Autodesk's 3D Studio Max.
- In 3D Studio Max, export it as an .FBX file.
- Import the .FBX file to Unity.

Once we had the Active Cells Facility model in Unity, we furnished it with all the basic equipment that would be inside the real facility (i.e. cranes, manipulator, target wheel, saw, doors, and cameras). We optimized the equipment models in the same way as the Active Cells Facility model. Once we had furnished the model, we implemented some physical properties to the Active Cells Facility environment: ability to control the cranes, the manipulator, the doors, the saw, and the cameras. For this, we consulted the engineer responsible for the CAD model, as well as our supervisor, who sent us some YouTube videos [30][31][32]. These videos described the hot cell in the JET facility in detail, more specifically the control and the monitoring inside their hot cell. The videos were very informative.

After we finished implementing the Active Cells Facility simulation, we had a decent graphical represention of the facility, with almost the same appearance as the planned Active Cells Facility, with the ability to control all the movable equipment and observe the interior of the facility model through the surveillance cameras.

Figure 4.2: The furnished and recolored simulation seen from one of the cameras.

4.2.3 Control System

In the simulation, it was possible to maneuver the cranes in planar directions, while lowering and raising the pulley on the cranes. The saw could be maneuvered in vertical directions, and controlled in every joint. The manipulator could be maneuvered in any direction in 3D space. However, its arms would also have to be controllable, and able to grip objects. To be able to do this efficiently we needed to apply *inverse kinematics*. In other words, we needed to make the position of the manipulator hands fully controllable, while the rest of the joints of the arms would adjust themselves according to the position of the hands. Furthermore, it was possible to switch between different cameras in the simulation. Each camera could be panned and tilted. And finally, there was a sliding door inside the simulation that could be opened and closed.

We showed the Active Cells Facility simulation to the engineers in the Active Cells Facility design group, and their reactions were positive. Once everybody had a firmer common mental grasp on how the Active Cells Facility currently looked like, how the view of the facility interior looked like from the perspective of camera surveillance, and how the instruments and the tools inside the facility were supposed to be controlled, we received some suggestions and feedback on how to control these devices. Some suggestions were joysticks (classic or Xbox/PlayStation joysticks), PLC control boards, haptic consumer products (Novint Falcon [33], Haption [34]), etc. We conducted some research on these, and we became quite interested in the haptic joysticks. A haptic joystick is a three-dimensional joystick that simulates the sense of touch by applying forced feedback. For instance, when a user uses a haptic joystick to touch a sphere in a virtual simulation the joystick will let the user freely navigate around the sphere, but will resist with force whenever the user touches the sphere. We considered this to be ideal as a means to control the manipulator. We had a meeting with our supervisor, and he suggested that we could use the Novint Falcon as an experimental prototype for the haptic joystick. With a haptic joystick like the Novint Falcon, we could attempt to apply it to the manipulator hands to get forced feedback in the joystick whenever one of the manipulator hands pushes against something.

Figure 4.3: The haptic consumer product Novint Falcon.

Before we started to experiment with the Novint Falcon, we decided to take a pause from programming and do some research with our simulation, create more storyboards, and generate more ideas for interaction, especially for visual feedback, and alternatives to operability. We studied the Active Cells Facility simulation, the film clips and pictures from the Active Cells Facility design group, as well as different videos on YouTube [30]. We identified some of the most usual difficulties regarding maneuverability and visibility inside the facility. One primary difficulty in maneuvering was the difficulty of perceiving the real 3D position of an object inside the facility model from a 2D image of a camera (horizontal and vertical position, depth, and relative distance to other objects). One idea was to add small information windows to the interface that would show the top view, the side view, and the front view of the room in a simplistic way, e.g. with just the colors black and white. This idea was not highly prioritized and was never fully realized.

4.2.4 Virtual Control Room

After a couple of weeks experimenting with the Virtual Window, the Active Cells Facility simulation, and the Novint Falcon, we decided to try one more idea. During the course of this master's thesis, Microsoft was advertising their new upcoming product: The HoloLens [11]. As described previously in *Technical Background*, HoloLens is an optical see-through HMD, which displays AR. Our idea was to project a miniature model of the Active Cells Facility in front of the user, for instance on the table, with an HMD similar to HoloLens. This would create an illusion of an Active Cells Facility hologram. With the HMD, the model could be viewed from different angles and allow the user to manipulate with it. Alternatively we could scale the model of the Active Cells Facility to a real life sized model and explore the inside of it.

The possibilities with the HoloLens were most appealing. Unfortunately, HoloLens was not yet available to the public, but we realized we could simulate this technology using a fully immersive closed-view HMD like the Oculus Rift to demonstrate our idea of the usefulness and usability of the HoloLens. With the Oculus Rift, we could create a virtual environment that represented a normal reality, for example an office, and project the miniature model of the Active Cells Facility onto a virtual table. With the help of the Leap Motion sensor, which could project graphical representations of the user's hands in a virtual environment, we could make it possible to manipulate the miniature Active Cells Facility model. We considered this an innovative and powerful tool for a more interactive and intuitive remote handling in the real Active Cells Facility.

We proceeded to realize this idea. We took an Oculus Rift DK2 and mounted a Leap Motion Sensor on the front visor (see Figure 4.4). Then we created a very simple miniature model of the Active Cells Facility, which we displayed in a virtual environment in front of the bearer of the HMD, who would be in a sitting position. The results of this simple experiment turned out to be very impressive. Furthermore, we discovered that the Leap Motion sensor recorded the real environment quite accurately with its infrared sensors. The recorded information of the real environment could be displayed in the virtual environment, making it appear that the Active Cells Facility model was rendered into the real world. In other words, the Oculus Rift and the Leap Motion sensor could be built together into a video see-through HMD to roughly simulate the HoloLens HMD. This combination also provided the ability to interact with the projected AR.

Figure 4.4: Oculus Rift DK2 and Leap Motion integration.

4.3 Resulting Design

After performing our experiments, we took the time to examine them, evaluate them, and compare them to each other. Then we came to a conclusion which experiments had the greatest potential to become our prototype. Once we came to that conclusion, we made some conceptual art on how the prototype should look like and how it should function.

We compared the ideas that comprehended visual feedback, namely *Virtual Window* and *Virtual Control Room*. The *Virtual Window* seemed to be an idea with promising potential uses, and if granted more time, we would have liked to explore and experiment with this idea even more. However, the *Virtual Control Room* contained many more advantages than the *Virtual Window*. The *Virtual Control Room* was easily implemented on the technical level and our prospects were that further development on the idea would not face many technical difficulties. The *Virtual Window* however needed some more technical implementations which we were not sure we could apply. For instance, if the *Virtual Window* was supposed to be efficient for visual feedback, we would have required a 3D stereoscopic monitor, which we did not have access to. At that time, the *Virtual Control Room* provided more extensive visual feedback than the *Virtual Window*, and we had all the hardware that was needed. Therefore, we decided to fully proceed with the *Virtual Control Room*.

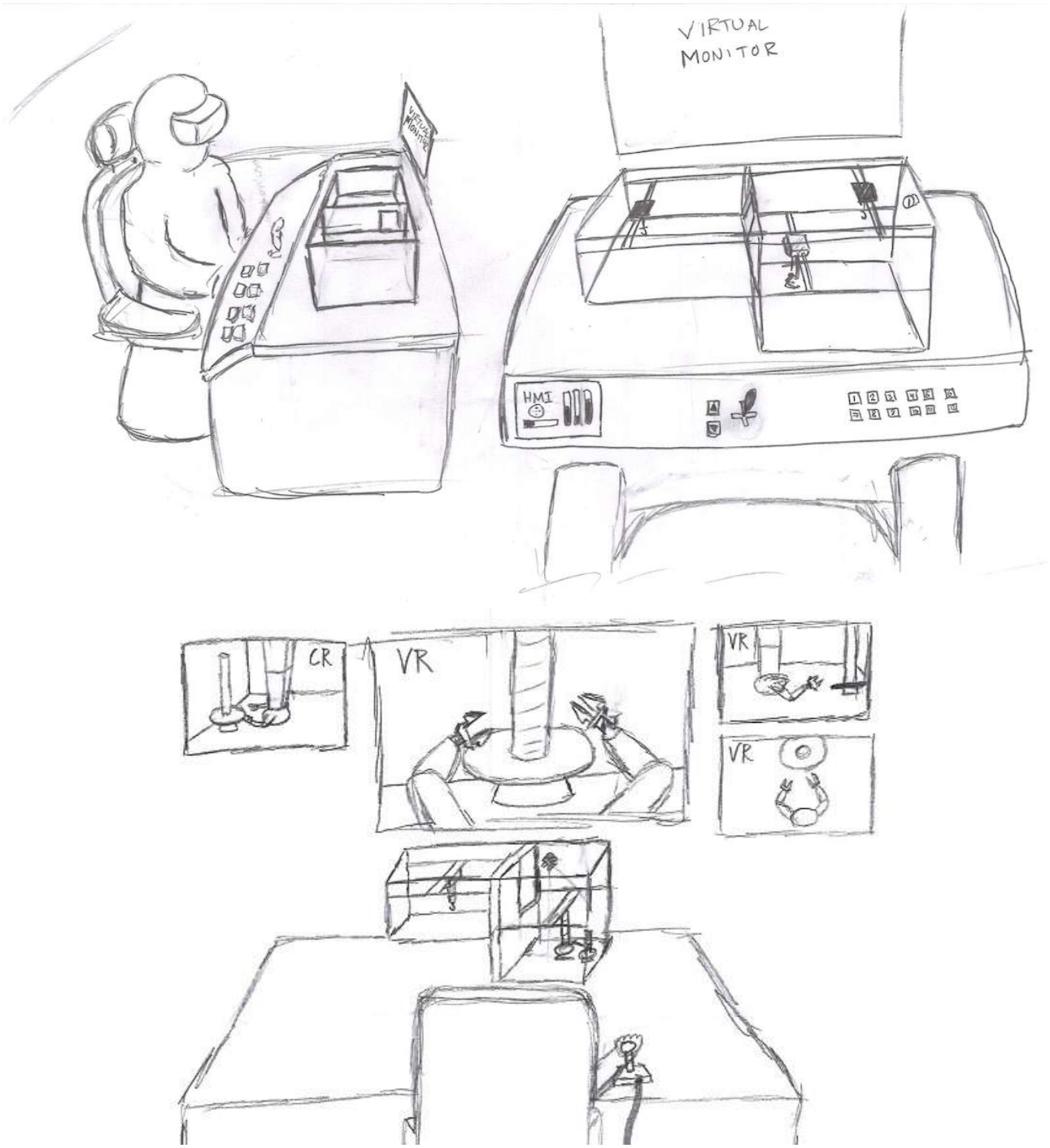


Figure 4.5: Conceptual art #1: The operator overviews a holographic representation of the Active Cells Facility on a table.

The conceptual design of the *Virtual Control Room* is visualized in Figure 4.5, Figure 4.6, and Figure 4.7 and can be summarized in the following steps:

1. The operator is seated in a chair in front of a table and mounts the HMD.
2. The HMD projects a miniature holographic representation of the Active Cells Facility onto the table, with AR.
3. Besides the Active Cells Facility hologram, virtual monitors are projected directly above the hologram. These monitors display the live-video feedback from the cameras inside the real Active Cells Facility.
4. The operator examines the hologram to gain a quick overview over the Active Cells Facility. The hologram shows basic information about the position of the equipment: the manipulator, the cranes, the cameras, etc.
5. To assure themselves about the situation inside the Active Cells Facility, the operator uses the virtual monitors to confirm that the information in the hologram agrees with the information displayed on the monitors.
6. The operator wishes to perform an action, according to the task list. For example, moving a crane from one side of the room to the other.
7. The operator moves the crane, either by using the holographic crane in the holographic Active Cells Facility, or with a dedicated hardware joystick.
8. While the operator moves the crane, the position of the crane is updated in the holographic Active Cells Facility in real-time. The operator watches the moving crane either in the hologram or in the virtual monitors.
9. Once the crane has been moved to its goal position, the operator performs a new action.
10. After finishing their tasks, the operator logs off from the interface, and unmounts the HMD.

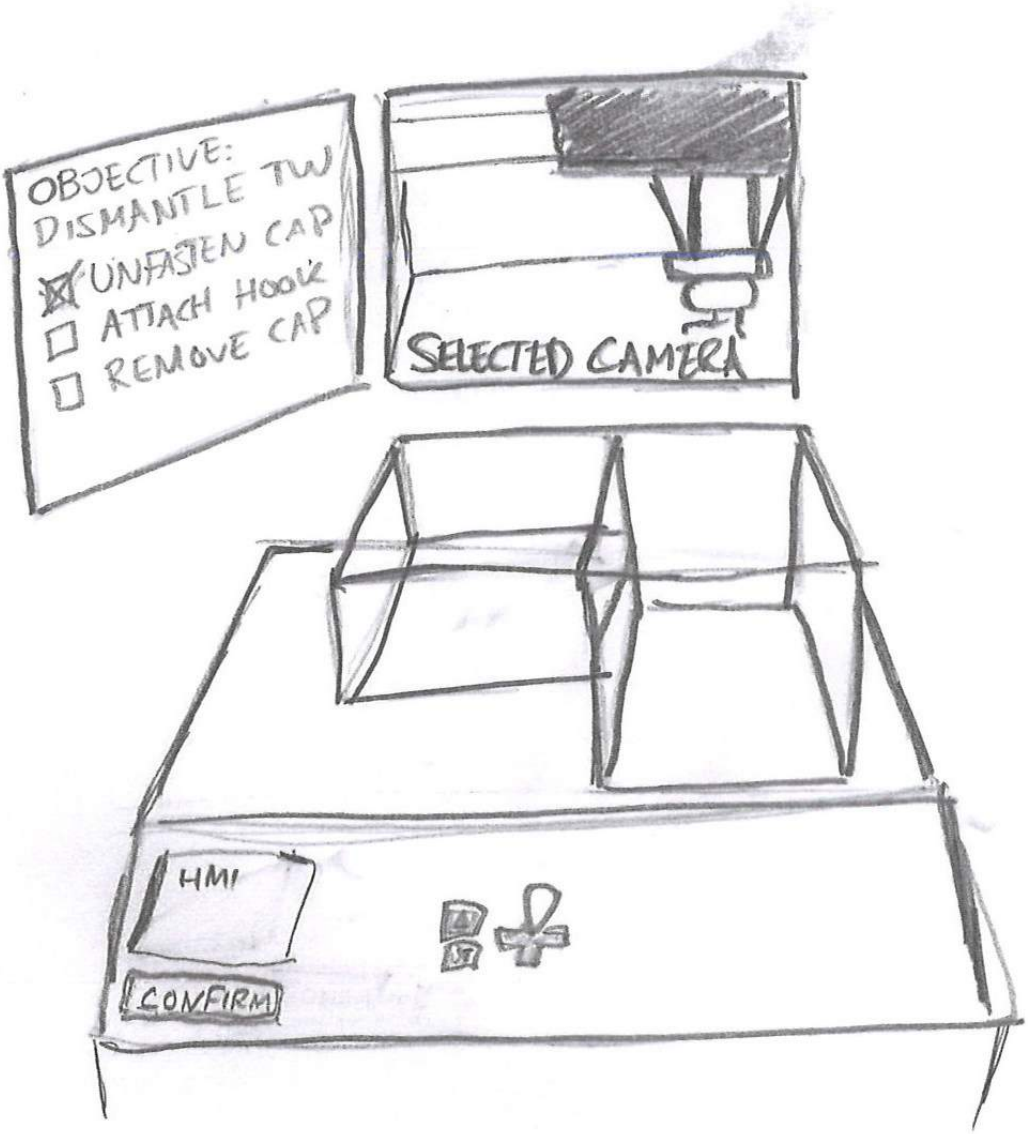


Figure 4.6: Conceptual art #2: Holographic monitors are hovering above the holographic model. The left monitor displays tasks to be completed, the center monitor displays the video feedback from one of the cameras inside the real Active Cells Facility.

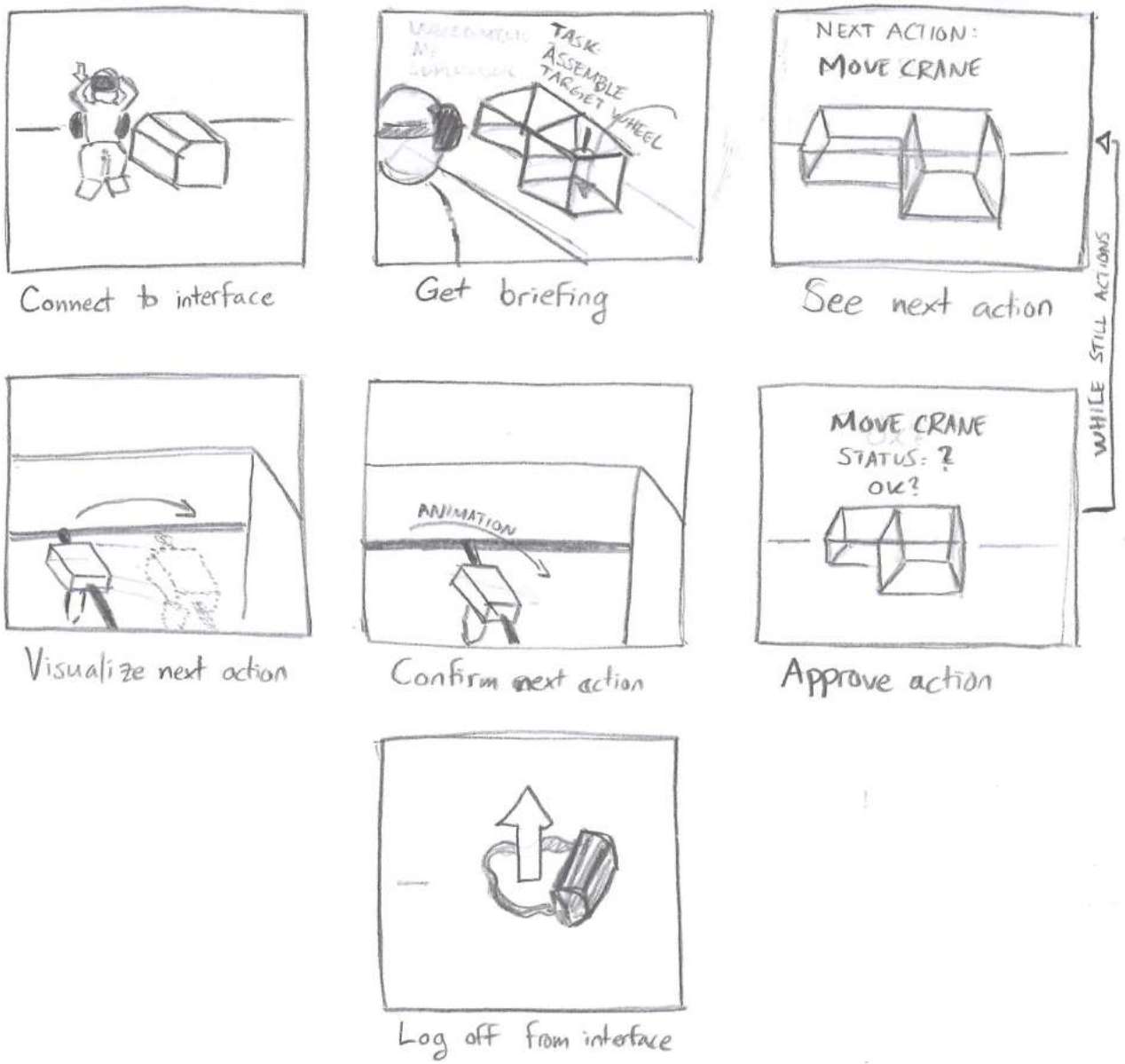


Figure 4.7: Conceptual art #3: Storyboard on how the operator works in the *Virtual Control Room*.

5 Virtual Control Room - First Iteration

After our initial research and experiments, we decided to proceed with the plan of implementing the prototype of the *Virtual Control Room*.

5.1 Approach

Once we finally had agreed on a conceptual design, we could begin implementing the functionality that was necessary. These functions were then subject to testing and evaluation. How this was done will be briefly discussed in this section.

5.1.1 Testing

In the testing phase there are two kinds of testing: function testing and user testing. The function testing, which is only performed internally by us developers in the master's thesis group, focuses on the technical aspect of the idea implementation. The user testing focuses on how the users react to and interact with the implementation of the idea.

Function testing

With brief implementations of our ideas, we were able to test them early on to attempt to save as much time as possible in case the idea would not turn out to be as efficient as it initially seemed, or in case there would be an unexpected complication. Sometimes, the performance could have proven to be slow, or additional information had to be incorporated into the implementation in order to realize the idea. If it would prove that the implementation is not satisfactory enough, we would either abolish the idea or find a work-around, then perform further testing. With a successful and smooth testing we proceeded to implement the idea to its full extent.

User testing

If we managed to achieve a satisfactory implementation of the idea from a technical perspective, we let a number of persons try the result. None of the persons was an actual end user, since we had no access to any of the kind. However, the test persons knew what the project was about, and thus we could expect to gain useful and relevant reviews, opinions, and criticism. The testing was performed either as an informal usability testing or with the Wizard of Oz technique [3].

Informal usability testing is an unplanned testing session with users which can be conducted at any place and at any time. The process is very simple: the user tests a certain prototype or

interface, while being casually observed by the supervisors. The user comments and answers questions regarding the prototype freely, i.e. they do not answer a questionnaire. This type of testing is efficient in the sense that it simulates a more likely situation in real life, i.e. when a user interacts with an interface in an uncontrolled atmosphere, while providing the developers with natural, instinctive, and authentic reactions and comments on the prototype. Without the necessity of planning, the test can be conducted fast and effortlessly.

Wizard of Oz is a way of studying a user's behaviour when they interact with an interface that seems to be implemented and finished to a technical and operational level, but is in fact only an empty shell without any real functionality. Instead, whenever the user interacts with the interface, another person intercepts the interaction and initiates functions that give feedback to the user. The user is then easily put under the impression that the interface is working, and reacts accordingly to it. This method is useful when the interaction part of the prototype is in focus, and implementation of all functionalities has to wait until the interaction part of the interface satisfies most of the end users' needs. While the user interacts with and navigates through the interface, their reactions and comments are discreetly documented [3].

5.1.2 Evaluation

After testing the different ideas and implementations of ours, we evaluated them according to the test results we gained from the testing phase. The evaluation was rather quick and informal. Our focus was whether the user found our idea attractive and if they understood the basic functionality behind it. If the feedback on the idea, obtained from either functional testing or user testing, turned out to be negative, we discarded the idea or changed it radically. If the feedback turned out to be positive, we kept the idea and improved it further if possible. All changes were scheduled to be tested in the next iteration of the design process.

5.2 Issues to Counter

The basic conceptual design gave us the impression that the model had more advantages on two aspects: visibility and operability. It certainly seemed that these aspects were much better than in a traditional camera-monitor surveillance. To prove this hypothesis we had to first identify the major difficulties in viewing and operating the Active Cells Facility with nothing but cameras and monitors. Once we identified these difficulties, we would design a prototype which would eliminate all of them and further improve visibility and operability.

Using our *Active Cells Facility Simulation* (described in section 4.2.2) we identified the most obvious difficulties in viewing and operating the Active Cells Facility from simple camera monitoring, which we list and briefly describe here.

Disorientation

With ordinary live-video feedback on monitors, the operator only sees parts of the interior of the Active Cells Facility and it can be difficult to piece together the information to a greater comprehension of the scale and space of the entire facility. For instance, it might not be clear that two different monitors actually display the same room, or the operator might not be able to easily tell which

wall is the western wall (relative to the cardinal directions) or eastern wall in a room from one monitor.

The same applies to operability. When only using monitors with live-video feedback, it can be difficult to predict which way a crane will move from a monitor's perspective when the operator commands the crane to go either left or right. For instance, if the operator watches a crane from a monitor and wishes to move the crane to the left from the monitoring camera's perspective, the crane might actually move to the right instead. This might happen, because the control signals in the hardware are implemented from only one point of view, and the implementation is completely unaware from which monitoring point of view the operator is observing.

Occlusion and Visual Limitations

The cameras are placed at different locations inside the Active Cells Facility so as to register as much of the facility interior as possible. Even though the cameras can be panned and zoomed, it might occur that certain objects will not be registered by the cameras due to the fact that these objects are occluded by other objects. Also, a single camera might not be able to cover all objects in its surroundings, forcing the operator to constantly switch between cameras to find the object of interest. For instance, situations might occur where the operator needs to keep track of two or more objects (crane, box, certain spot) but the camera field of view is unable to perceive all of them, i.e. some objects are lost from frame. In such a case, the operator has to proceed very slowly with their operations and pan the camera accordingly, or look at other monitors, which would contribute to disorientation.

Low 3D Perception

The cameras inside the Active Cells Facility are ordinary cameras, grabbing 2D frames. From a 2D frame the operator needs to analyze and extract information regarding the 3D space inside the facility and the real positions of the objects it contains. The operator should preferably be able to obtain this information quickly, for instance while moving an object with a crane. However, depth perception in a 2D frame is highly impaired, which could influence the operator's perception of an object's real position in 3D space as well as its relative distance to another object. Low perception of 3D space from a 2D frame is a common problem [3]. From a 2D image it is sometimes very hard to tell the exact location of an object relative to the 3D space, and it is even more difficult to predict the future location it is moving towards to. For instance, when the operator lowers an object with the crane, the object is hanging in the air above the floor. From a simple 2D image it is not easy to tell where exactly on the floor the object will eventually be lowered.

5.3 First Design

Once we identified the basic use cases in the Active Cells Facility, as well as the major difficulties with ordinary camera surveillance that had to be eliminated or diminished in the prototype, we started to design and implement the prototype of our conceptual design. We aimed to construct the prototype to a state where the user could use it to complete at least the first scenario (see Figure 3.6). As we were applying a top-down approach in designing the prototype, we decided to only implement the visual feedback of the prototype and simulate the first scenario by applying the Wizard of Oz method. In other words, all control of the different functionalitites in the Active Cells Facility (e.g. opening the ceiling hatch, moving the cranes, etc.) as well as physics would be simulated in the background.

When a user would interact with the prototype, the simulation would take care of all the control and physics, giving the user an impression that they are in control. This way, we could focus on building the prototype to make it visually informative instead of making it realistically controllable. The reason we chose this approach was because if the prototype would have a good visual feedback, i.e. giving the user a greater sense of control and overview over the Active Cells Facility with good VR/AR functions, then the control and operability of the facility would basically be easy to solve later. Therefore, we divided the design into two tasks: *Visual Interface Design* and *Scenario Scripting*. In the first task, design of the interface would be taken care of, and in the second task, simulation of the first scenario in the interface would be implemented.

5.3.1 Visual Interface Design

The design of the interface was commenced with the miniature model of the Active Cells Facility (see Figure 5.1). After having been furnished with the basic equipment (i.e. the cranes, the manipulator, the cameras, the saw, and the target wheel) the model looked as in Figure 5.2. This was just an ordinary model, which could be viewed (with the Oculus Rift) from different sides and angles (see Figure 5.3). Besides this, the model had no other implemented functionalitites. However, it already tackled the difficulties with camera surveillance, i.e. disorientation, visual limitation (occlusion), and low 3D perception. The model served as a graphical representation of the Active Cells Facility, and every controllable equipment inside was also represented by a graphical model, namely the cranes, the cameras, and the manipulator. The models of the equipment inside the Active Cells Facility model would mimic the positions and orientations of the equipment in the real facility, i.e. when the cranes or the manipulator would move in the real facility, the models of the eqipment in the facility model would move accordingly. This way, the user would have a full overview over the facility and the equipment inside, thus avoiding disorientation. If the user would, for instance, wish to move a crane, they would be able to follow the movement of the crane all the way to its destination with the help of the crane model in the miniature facility model.

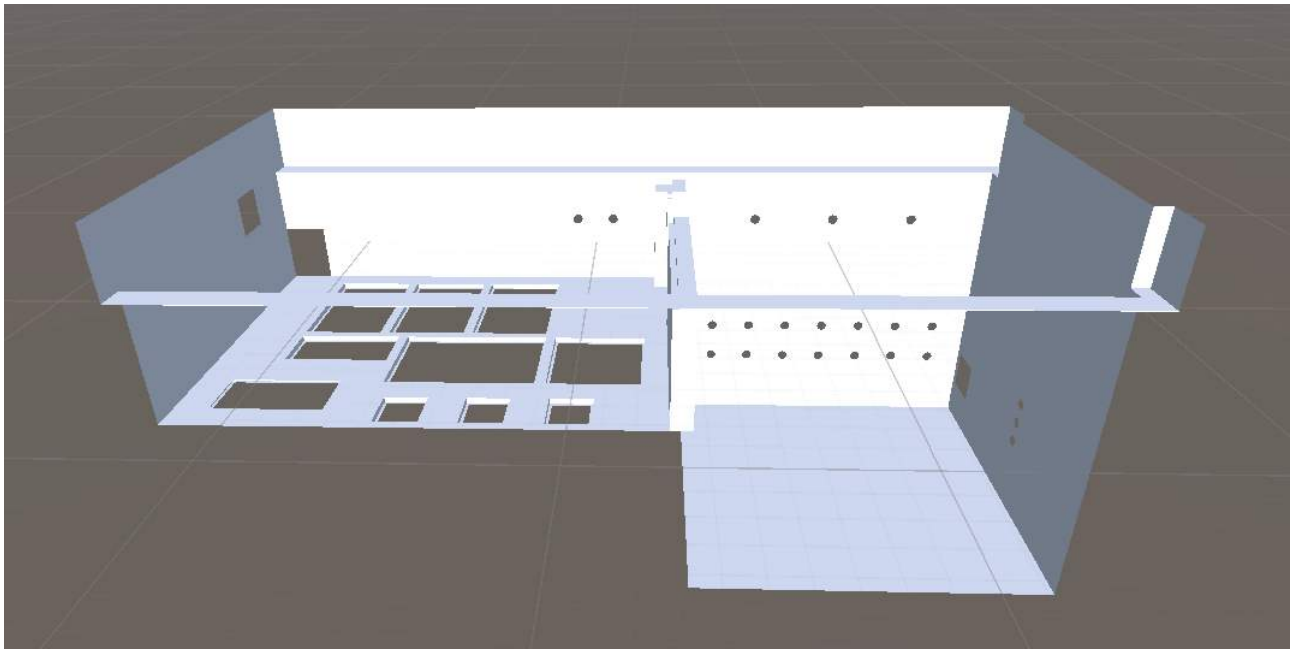


Figure 5.1: Virtual Control Room #1: First look of the design of the Virtual Control Room.

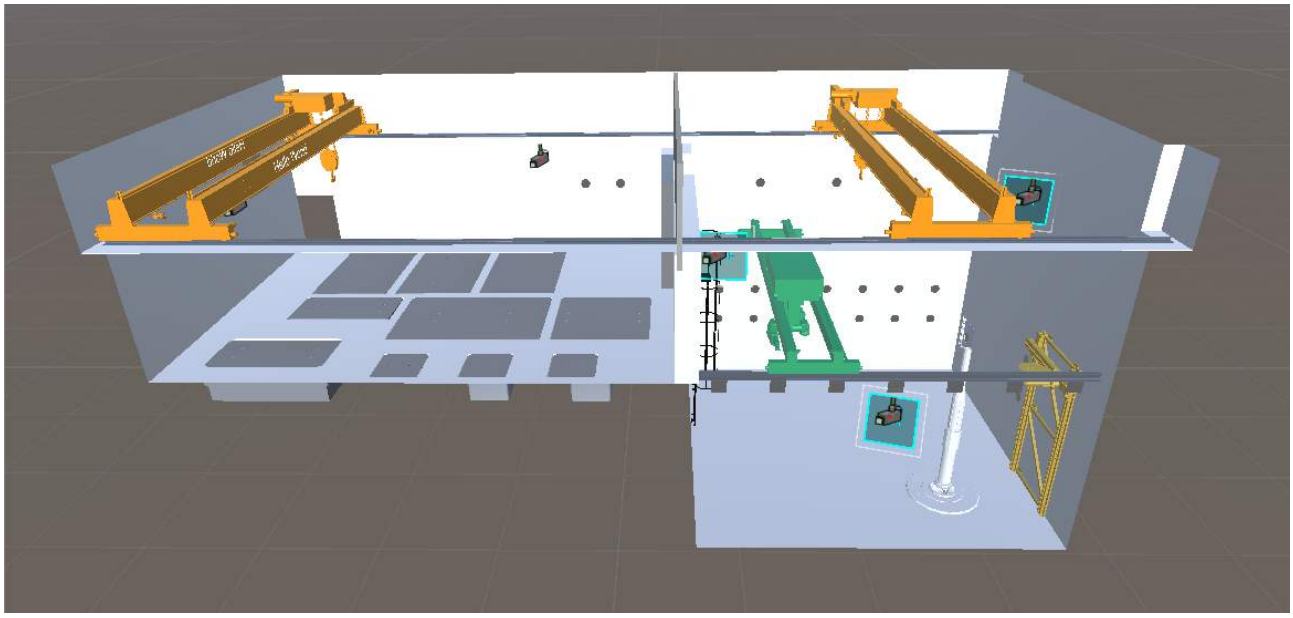


Figure 5.2: Virtual Control Room #2: The furnished Virtual Control Room.

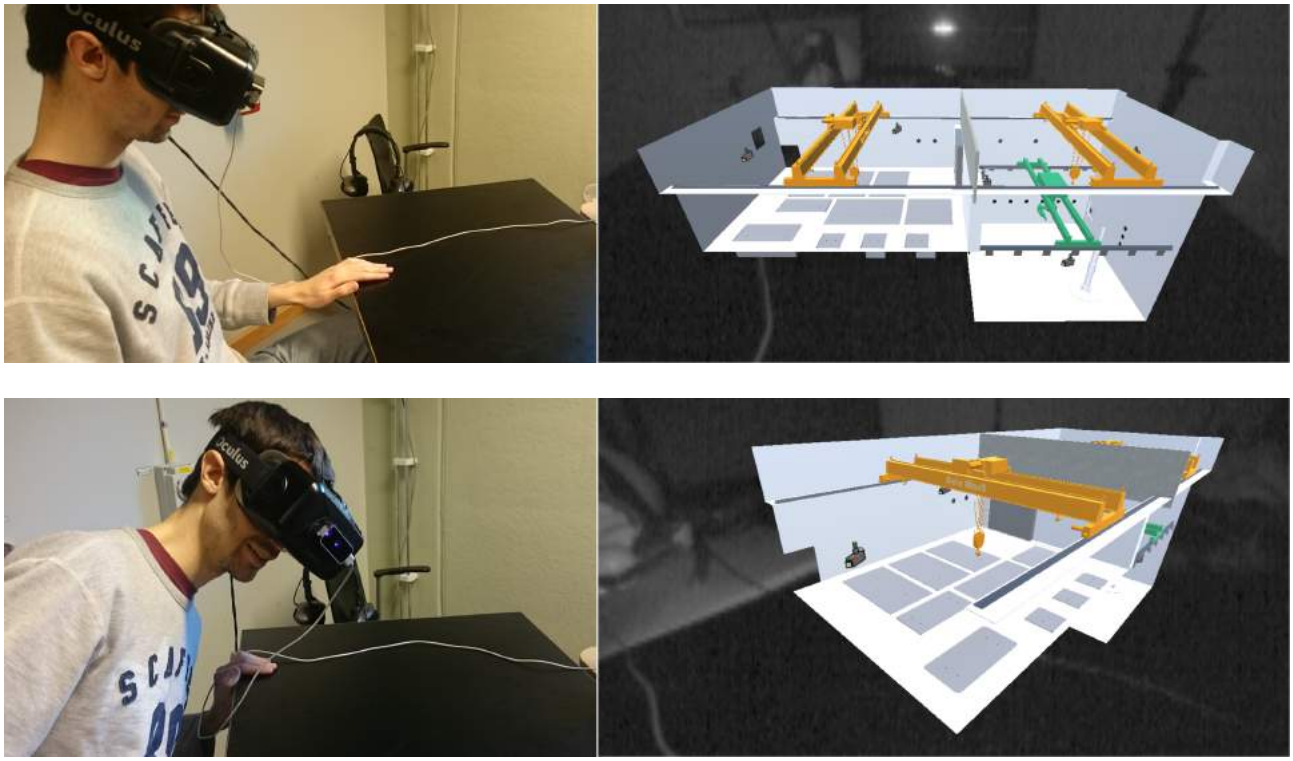


Figure 5.3: A user is watching the Active Cells Facility model with the Oculus Rift HMD from different angles.

If an object was behind another one, the user could move his head to change view over the Active Cells Facility model and look behind the occluding object model (see Figure 5.4), thus avoiding occlusion. Since the 3D model was displayed in a stereoscopic HMD, everything the user could see was displayed in 3D, and had an accurate depth perception. However, it must be emphasized that the goal of the miniature Active Cells Facility model was to serve as a simple graphical representation of the facility, with the basic equipment having their own representations in the facility model. Even though the miniature facility model was in scale with the real Active Cells Facility, and the positions and orientations of the basic equipment in the facility were accurately represented in the model, there might have been other objects inside the real Active Cells Facility that would not be displayed in the model.

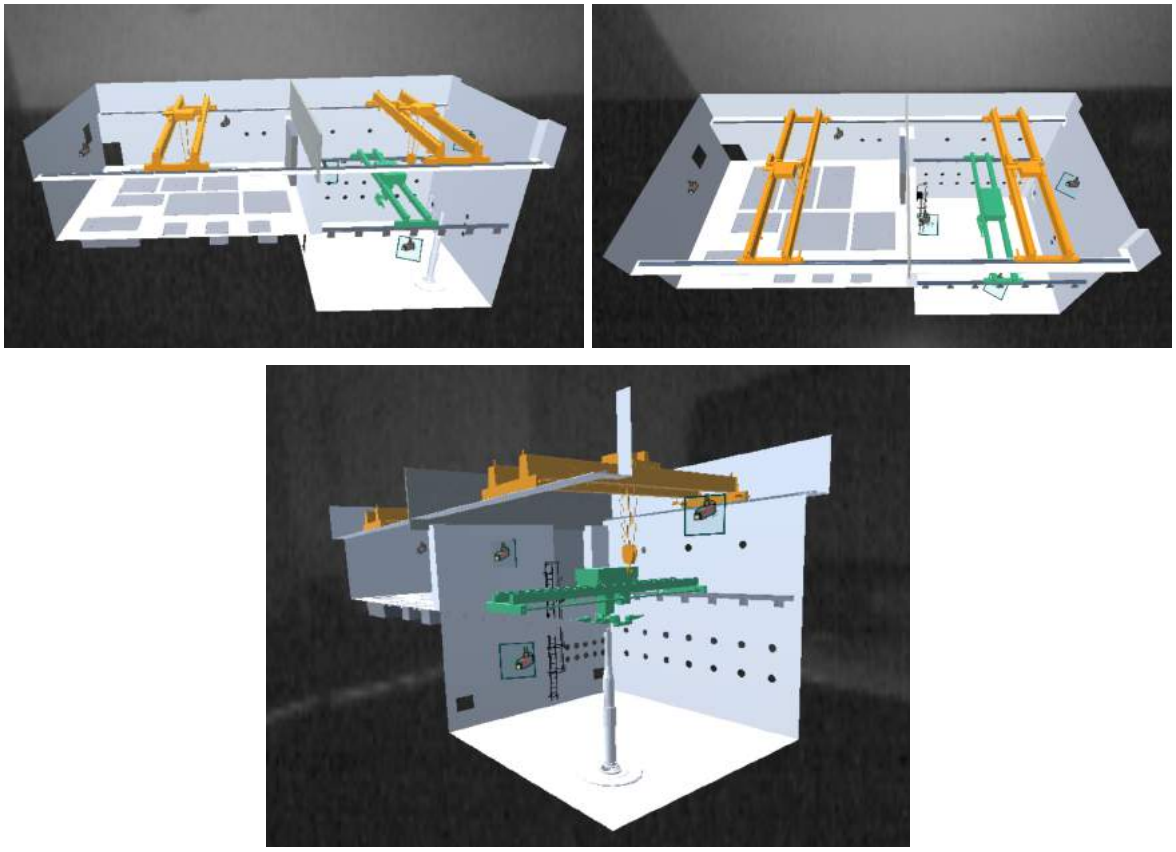


Figure 5.4: Virtual Control Room seen from different perspectives with the Oculus Rift HMD.

The decision to not display every object in the model was based on the visibility principle [35]. This design principle describes that only relevant information for a certain task should be visible to a user, while information which is not needed should not be visible so as not to cause distraction. The facility model would be overfilled with information if every object in the real Active Cells Facility would have a graphical representation in the model, and it consequently would not be easy to have a clear overview (see Figure 5.5). The only thing that was supposed to be controlled in the facility was the basic equipment (cranes, manipulator, cameras, saw, and target wheel), which therefore should have a graphical representation. Everything else in the Active Cells Facility could only be moved with the basic equipment itself. The idea however was that it should be possible to create a graphical representation of an object in the real Active Cells Facility into the model, if it would have proven to be convenient. For instance, if an object in the real Active Cells Facility would be moved by the manipulator or a crane, then the object should have a graphical representation. Everything else in the real facility that was not currently part of on an operation should not be visible in the facility model.

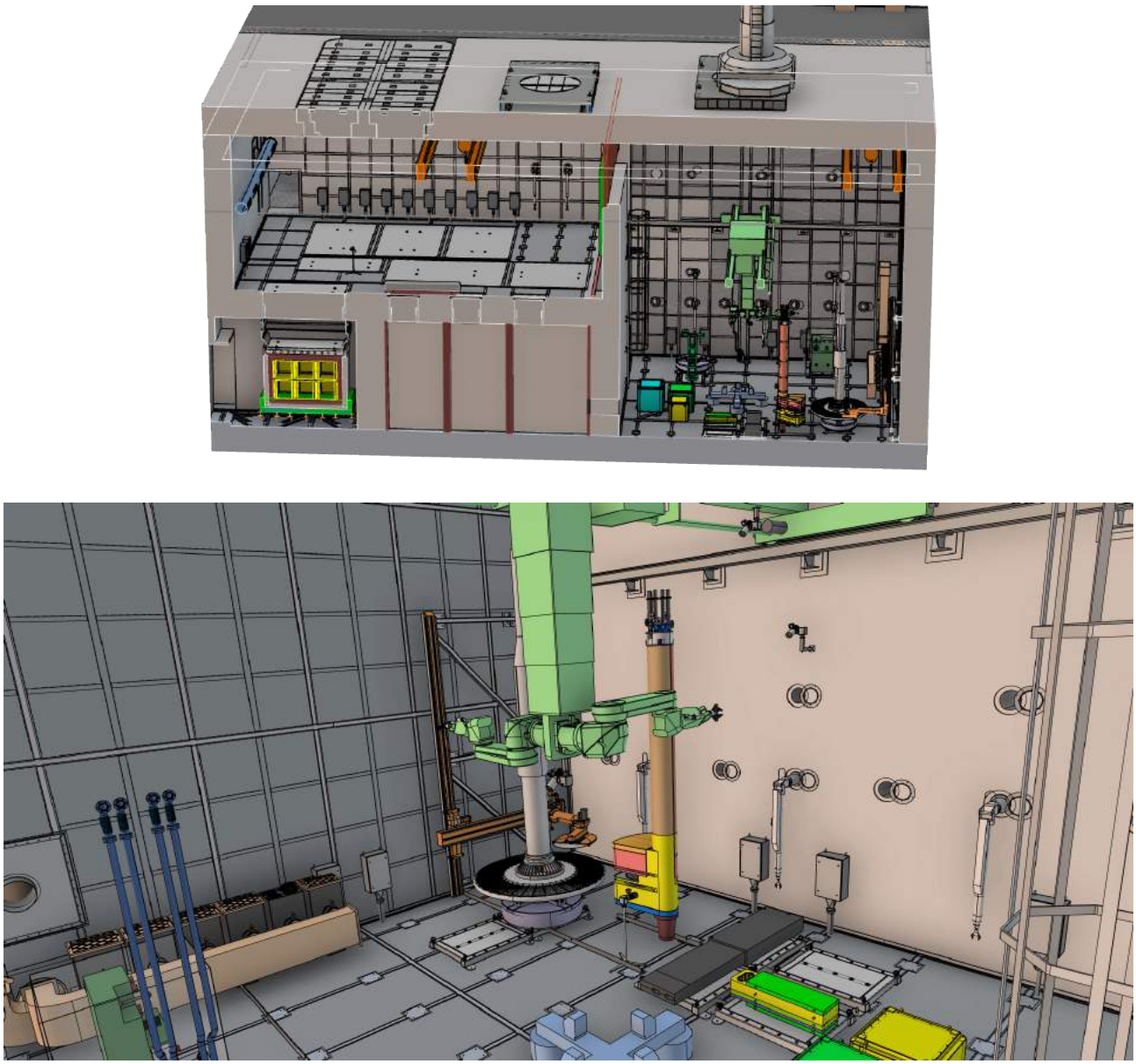


Figure 5.5: Pictorial illustrations on how the interior of the Active Cells Facility may probably look like.

No matter how accurate the facility model was, we were certain the model alone would not be safe enough for a complete overview. The operators would most surely not feel comfortable if they would just rely on the model while not be able to see inside the Active Cells Facility to confirm their operations. This assumption was based on the feedback principle in interaction design [36]. Feedback is about communicating the results of an action to the user, i.e. when the user performs an action, they should be informed about the progress of that action. In the case of overviewing the Active Cells Facility, the user should be given proper feedback on their operations in the facility in order to confirm that the operations have been done correctly and safely. Therefore, we believed that camera

surveillance would be necessary after all. We placed camera icons (see Figure 5.6) on specific locations in the Active Cells Facility model.

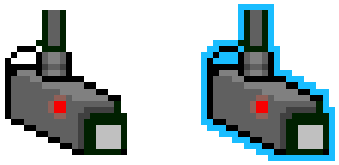


Figure 5.6: Ordinary camera icon and a highlighted camera icon

Then we added a single big monitor behind and above the model, resulting in the interface seen in Figure 5.7. When a user would select one of these camera icons with his hand, the live-video feed from this camera would be displayed on the monitor above the Active Cells Facility model, and the selected camera icon in the facility model would be highlighted. With the help of some Leap Motion widgets found on the Leap Motion website [37], we created virtual buttons which were interactable in the virtual environment. We placed one button over each camera icon in the Active Cells Facility model, and made the buttons invisible. Now it was possible to select a camera by simply touching the corresponding camera icon with the fingers (see Figure 5.8). The result turned out to be very promising, since the interaction with the Leap Motion hands was very smooth and precise.

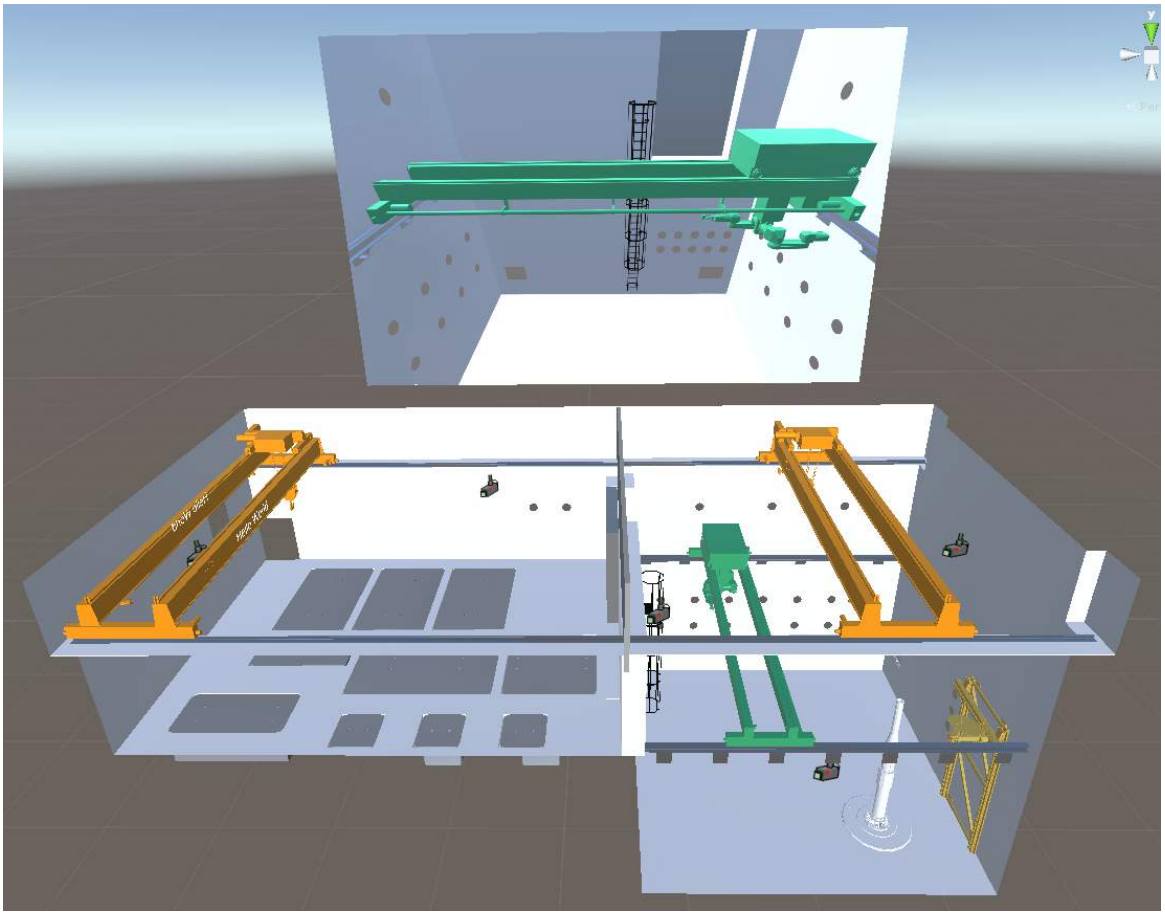


Figure 5.7: Virtual Control Room #3: Third look of the design of the Virtual Control Room.

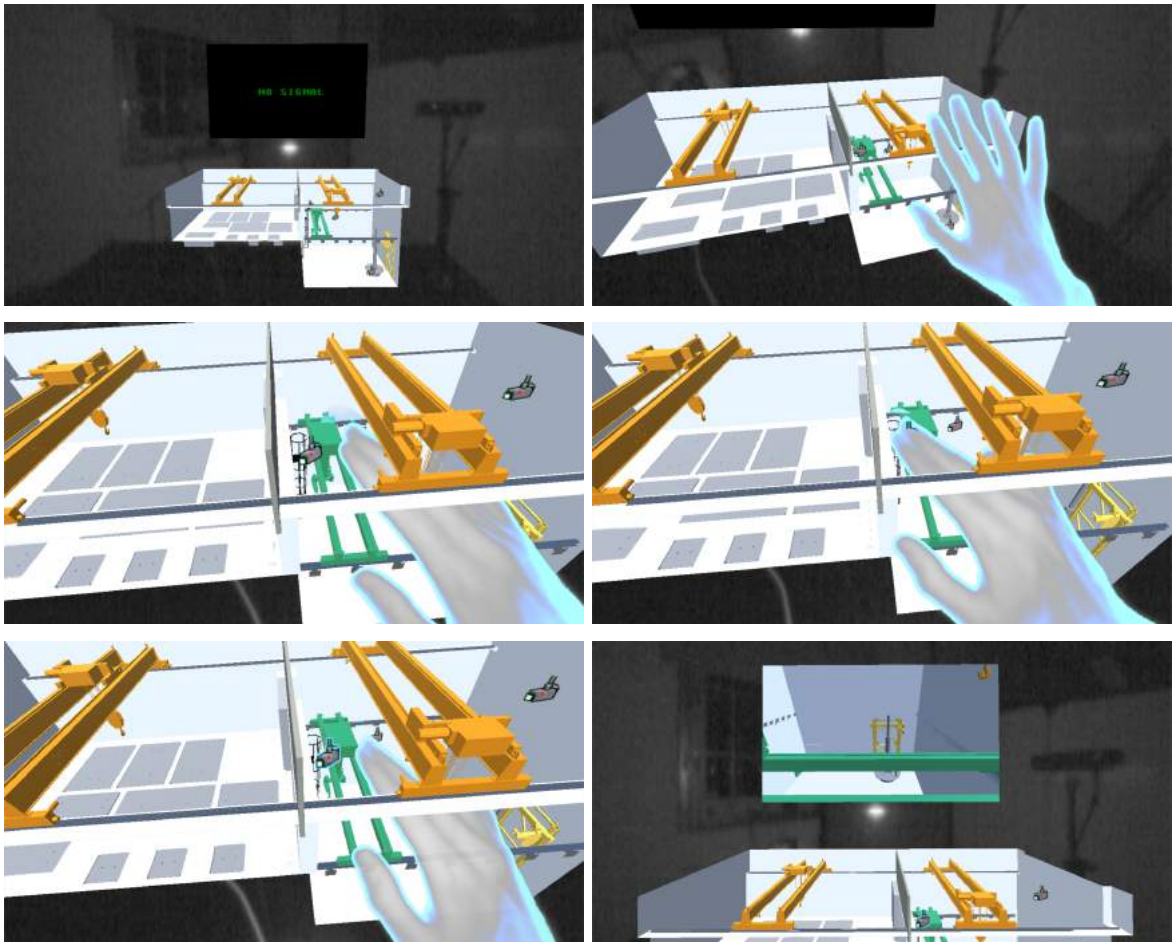


Figure 5.8: The user touches a camera icon and displays the camera view on the big monitor from the corresponding camera in the real Active Cells Facility.

This way, the overview of the real Active Cells Facility would consist of the facility model together with camera surveillance. No matter what operations the user would perform in the model, they could examine the surveillance monitor to see what is really happening inside the real Active Cells Facility. This collaboration between the camera placement in the Active Cells Facility model and the camera surveillance on the monitor would hopefully counteract disorientation with camera surveillance. By looking at the monitor, and then checking the placement of the currently selected camera in the Active Cells Facility model, the user would have a firmer grasp on which part of the real Active Cells Facility they were watching on the monitor.

When we added interaction into the model which involved touching the cameras, we thought we might try adding more possibilities of interaction. We were especially interested in the idea of making the equipment models in the Active Cells Facility model interactable by using the Leap Motion hands, for instance moving the crane model with the hand. This would make the model satisfy the concept of a *Natural User Interface* which was explained in the section *Technical Background*. We started to implement this idea of controlling the equipment with the Leap Motion hands. We put our focus on

one equipment first, the crane, from which we could later implement on the rest of the equipment if the implementation on the crane would prove to be satisfactory.

We started creating a small menu showing up above the crane after directly touching the crane with a Leap Motion hand. This menu would present different options and actions that could be performed with the crane, together with a small window containing basic or technical information about the crane. The basic actions inside this small menu would be a button that would enable the control of the crane. Once the user would be done with the crane they would press a button on the menu to close it. There was one slight problem with the small menu: since it was above the crane, it partly occluded the monitor on which the live camera feed was displayed. But since this problem was not overly serious, we continued implementing the controllability. The control of the crane was implemented using sliders (see Figure 5.9), which were part of the Leap Motion interactable widgets.

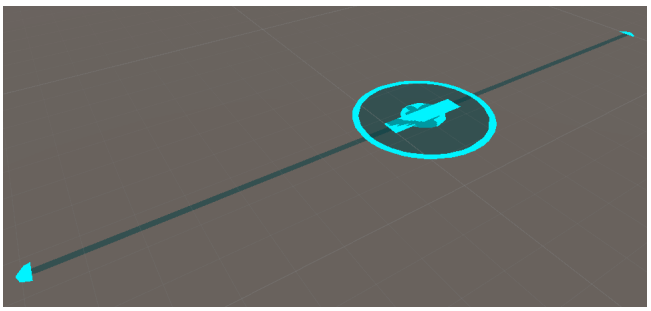


Figure 5.9: A slider widget from Leap Motion Widgets.

The crane model consisted of three different movable parts: the whole crane construction, the wagon on the crane, and the pulley of the crane. Together, these parts enabled the crane to move in all three orthogonal directions in 3D space. The sliders provided an interaction where the crane could be controlled in a natural manner with the fingers of the Leap Motion hands (see Figure 5.11). To make the interface less dense with the sliders, a button for each slider was added in the small menu. Only one button at a time could be toggled, allowing only one slider at a time to be present inside the model.

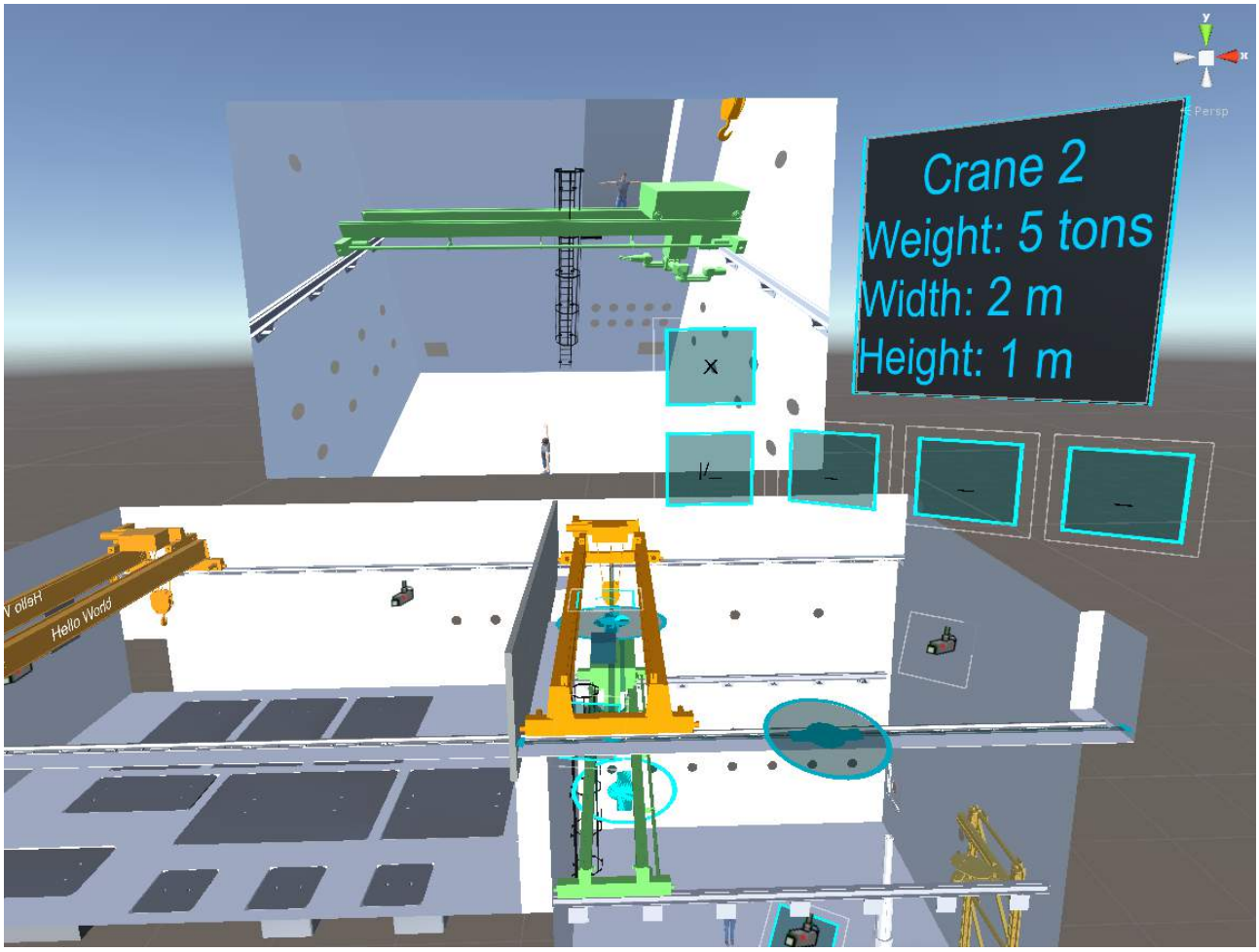


Figure 5.10: Virtual Control Room #4: Fourth look of the design of the Virtual Control Room.

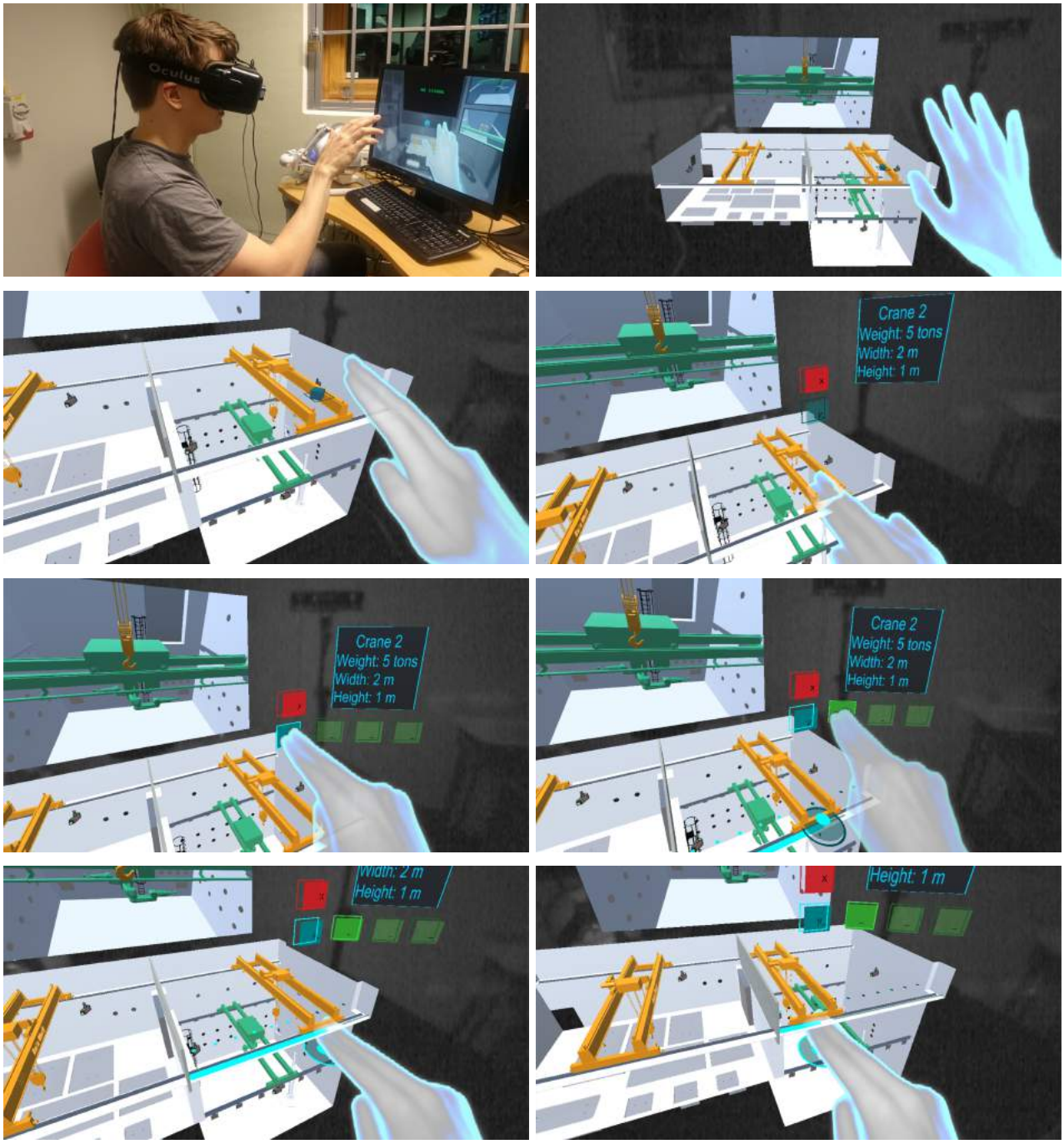


Figure 5.11: The user touches the crane model, which brings up a menu above it. The user selects the option which allows them to move the crane.

When we viewed different videos from ESS on hot cell operators who remotely operated manipulators in real hot cells, we noticed their use of multiple monitors to supervise the operations inside the hot cells. From the videos, we could tell that the operators usually preferred a glimpse from more than one monitor, mostly to confirm their operations from different perspectives. We thought that a

similar funcionality would be useful in our prototype. Therefore, we enhanced the monitoring system in our prototype, with one small monitor for each camera. We placed the monitors around the Active Cells Facility model. The monitors were placed conveniently: the monitors to the left of the model showed the interior of the left room in the model, and the monitors to the right showed the interior of the right room in the model. The final monitoring system consisted of small monitors, with one small monitor for each camera in the Active Cells Facility model, and one big main monitor, where one could display the live video feed from one of the small monitors (see Figure 5.12). If the user wished to display the content of a small monitor on the main monitor, they simply needed to touch the small monitor with their hand (see Figure 5.13). Besides the monitors, we also added an information panel to the interface, in front of the Active Cells Facility model. This information panel would inform the user about the task to be performed in the Active Cells Facility. The tasks on the information panel could be changed with two virtual buttons on either side of the information panel (see Figure 5.14).

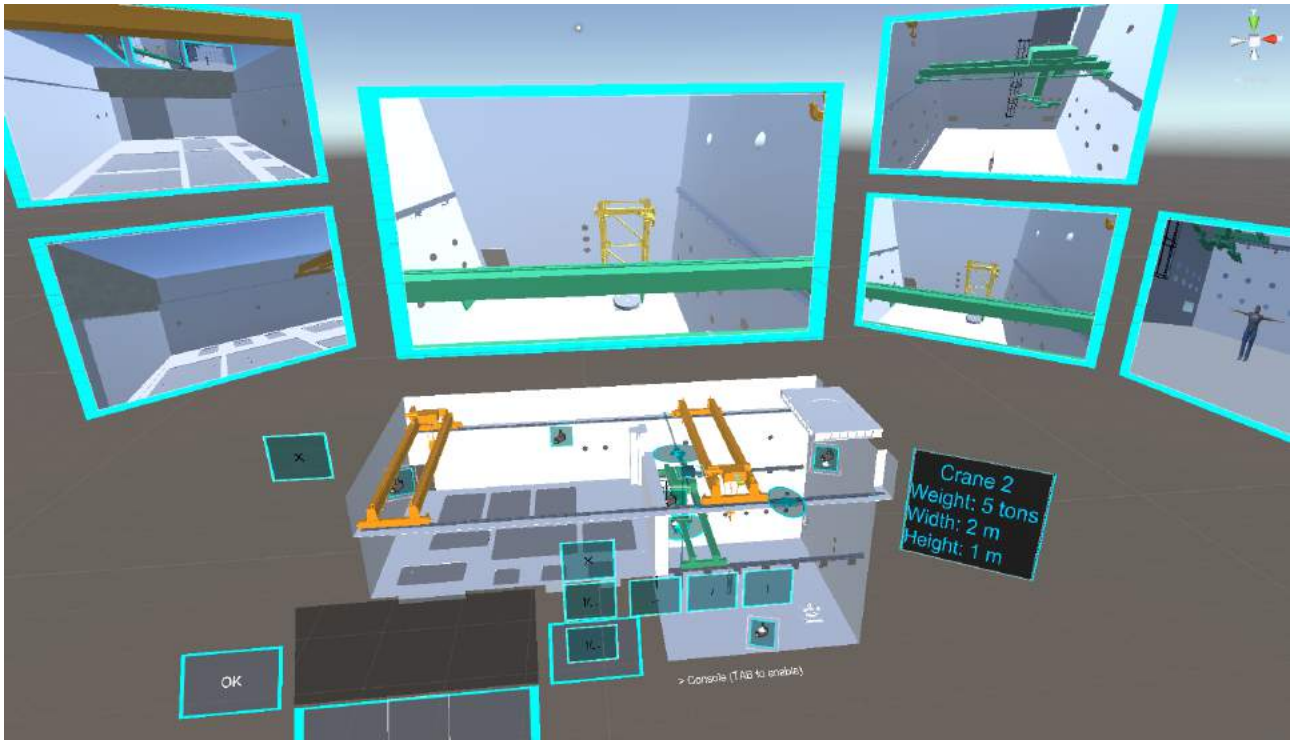


Figure 5.12: Virtual Control Room #5: Fifth look of the design of the Virtual Control Room.

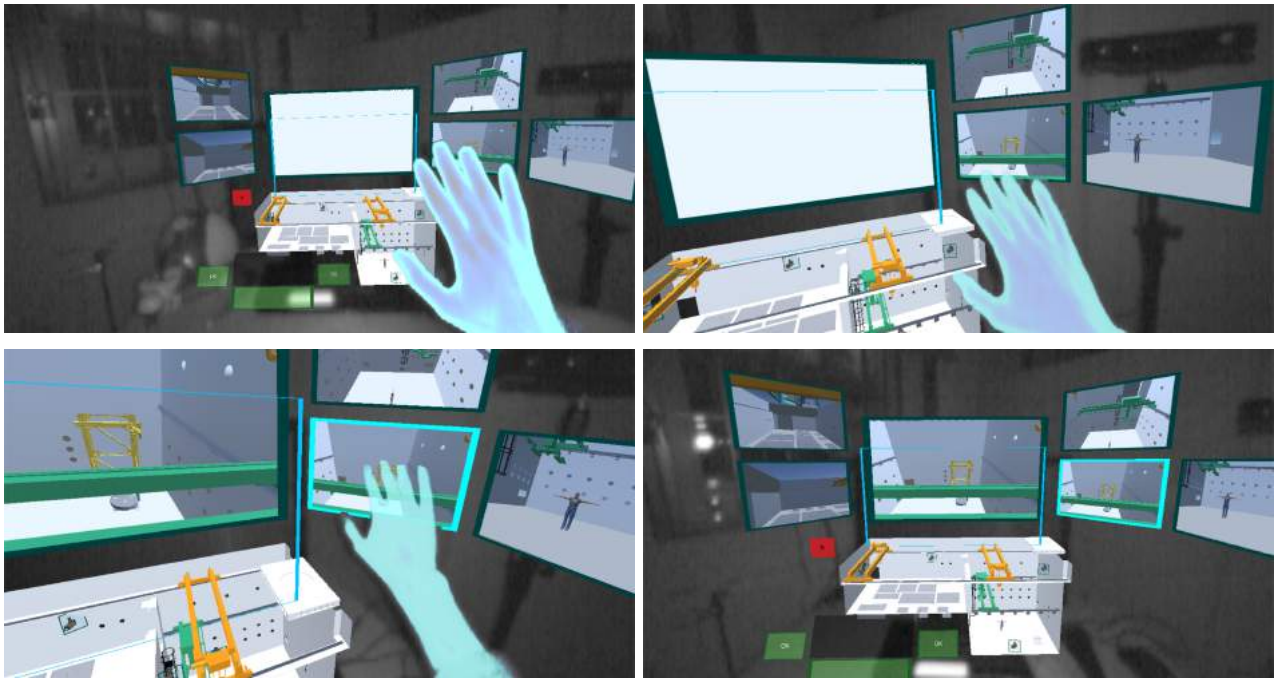


Figure 5.13: The user touches a small monitor, which leads to the displayed content in the small monitor to be displayed in the main monitor.

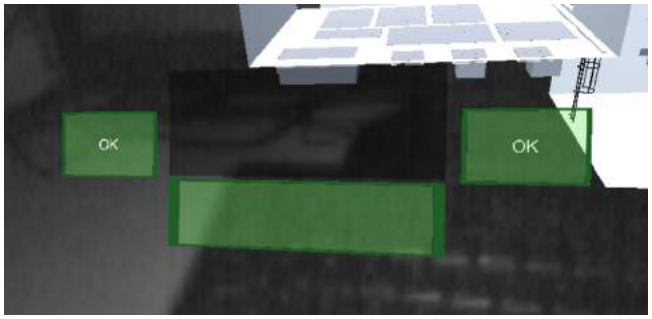


Figure 5.14: The information panel.

5.3.2 Scenario Scripting

In order to simulate a real-world scenario (as illustrated in the storyboards in *Conceptual Design*), we would need to either implement full functionality for the entire model and all of its components and modules, or make a completely scripted simulation.

The former alternative meant that we would need a good physics engine, properly modelled equipment and surroundings to accomodate to this physics engine, and a very accurate interaction system. This would prove to be infeasible within the time frame we had, so we decided to completely focus on the latter alternative, which we called *Scenario Scripting*.

The general idea behind Scenario Scripting was to make everything in the model (equipment, components, and virtual peripherals) modular and callable. In other words, everything would need to be independent from everything else, and have funcionality that was reachable from outside the module. This way, we would be able to simulate everything, much like the Wizard of Oz method described in *Conceptual Design*. We reasoned that the manual Wizard of Oz method was insufficient for our purposes, as the tasks were beginning to become very complex. After internal discussion, we made the decision to make it completely automated, hence the scripting part came to fruition.

The script would be loaded on launch, parsed, and then executed step-by-step. As all the modules were independent, we would be able to make any and all of their functionality conditional. For example, if we were to script a movement of the crane from one point to another, the crane could itself decide whether or not that was possible, whether or not it would block the rest of the script until it was completed, and the conditions of its completion. This proved to be extremely efficient when prototyping the storyboard, as any minor or major change would be instantaneously visible and evaluated within minutes.

The implementation itself was quite simple: External files were created and had lines of instructions that were loaded, parsed, and coupled with the referenced modules. When the simulation was started, one of these scripts was chosen and its instructions executed sequentially. The instructions were implemented by default to block the following instruction until it had completed. Any instruction could be specifically told not to block if applicable. Non-indented instructions were checkpoints and had to be confirmed by the user (by pressing the space bar) before the script could continue. When the last instruction had been executed, a new script could be started.

The following is a commented example of how the script could work:

```
// This needs to be confirmed by pressing the space bar
Open top door

    // Tell the simulation to wait 1 second before continuing
    SEQUENCER WAIT 1

    // Tell the ceiling door to open,
    // with a duration of 3 seconds,
    // without blocking (the exclamation mark is a non-blocker)
    TOP_DOOR !OPEN 3

// This needs to be confirmed by pressing the space bar
Lower target wheel

    // Tell the simulation to wait 1 second before continuing
    SEQUENCER WAIT 1

    // Tell the target wheel crane to lower the crane,
    // with a total duration of 6 seconds.
    TARGET_WHEEL_CRANE LOWER 6
```

For debugging purposes, a console was added which could be activated during the simulation using the TAB key, and where the script calls could be done manually.

The final script that was used during the testing phase can be seen in *Appendix* A.

5.4 Conclusions from the First Iteration

When we had a working prototype of the interface, we brought the PC which we used for development to ESS where we could demonstrate it to our colleagues and get valuable feedback. The feeling towards the general design was mostly positive. The most immediate comment we got was the lack of ability to zoom, as the model was sometimes too small to see what was happening inside of it.

Another thing we noticed was the difficulty of pressing the virtual buttons. Sometimes, when a user pressed a certain virtual button, e.g. a camera icon, the button would not react to the touch. This was mostly due to the LEAP Motion sensor not being as accurate as we wished. However, this could be slightly remedied by reducing the amount of infrared pollution in the room (closing curtains and shutting off lamps) and by moving the USB connection of the LEAP Motion device from a shared hub to its own dedicated hub. This improved the accuracy a little, but the general problem persisted. Besides this problem, it also sometimes ocurred that a user could not activate a specific virtual button without accidently activating another one, e.g. activating the crane interaction when actually aiming for the camera icon. To remedy for these issues, we would need to reduce the amount of unnecessary virtual interaction in the Active Cells Facility model, find alternatives where possible, enlarge the buttons as much as desirable, and bring them closer to the user.

We would also need to make the tasks clearer to the user, so as to avoid confusion. This meant that we needed to implement more auditory and visual feedback.

6 Virtual Control Room - Second Iteration

Using the feedback we received during the internal tests of the first iteration, we were able to outline a few necessary changes that were needed to be done. First of all, we needed to make the model easily navigable. In other words, we needed to make it possible to translate its position, as well as rotate and scale it freely. Second, we needed to make the tasks clearer and less ambiguous. Third, we had to either increase the size of the virtual interface and its buttons so as to facilitate smoother interaction, or simply reduce the number of buttons and remove parts of the interactive interface. We chose to do both.

6.1 Approach

The approach used for the second iteration was the same as the approach used for the first iteration, as described in section 5.1.

6.2 Second Design

6.2.1 Decluttering

We decided to get rid of most of the interactable interface inside the model, i.e. we removed all buttons and panels except for the clickable camera icons, as they were simple enough to function sufficiently. The information panel was removed from underneath the model as well as its virtual buttons (as illustrated in Figure 6.1). These buttons were instead moved to the physical keyboard (which has not been part of the interface until this iteration), as they were not an essential part of the interface and would be used just once in the beginning of the simulation. Instead, we added two information panels to the interface, one under the main monitor, and one above the main monitor. The upper information panel would inform the user about the task to be performed in the Active Cells Facility, and the lower information panel would give instructions to the user on how to accomplish the task.

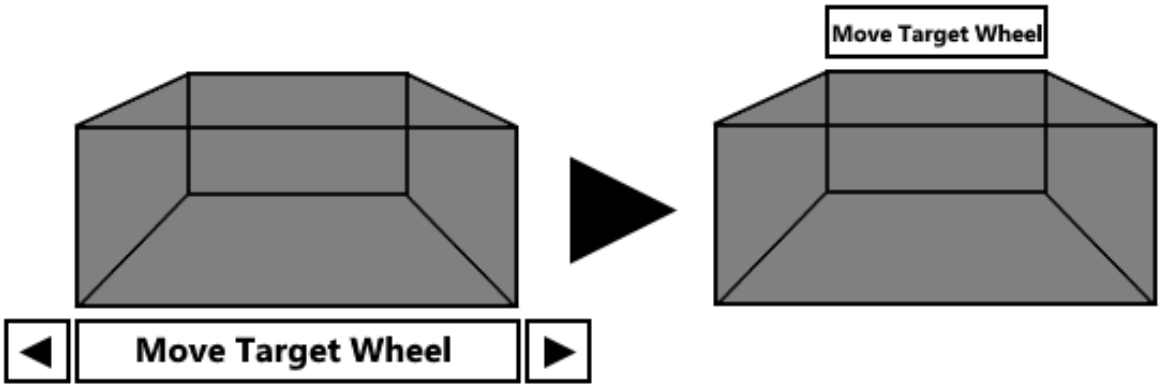


Figure 6.1: The information display was moved and its virtual buttons were removed.

6.2.2 Visual Interface Design

After the decluttering stage, the interface consisted of the Active Cells Facility model, the surveillance monitors, and the information panels (see Figure 6.2). The only possible interaction with the interface consisted of the following:

1. Touching one of the camera icons in the facility model to display the camera view on the main monitor from the corresponding camera in the real Active Cells Facility.

2. Touching one of the small virtual monitors to display its content on the main monitor.

Besides this, there was no other possible interaction with the interface. In the previous iteration it was possible to interact with the equipment, but after the testing phase we concluded that too many virtual buttons would clutter the interface. We also reasoned that the equipment models in the facility model should only read and mimic the positions of the real equipment in the real facility, and not the other way around. In other words, the equipment in the real facility should not be controlled by means of manipulating with a virtual equipment model in a virtual interface. Instead, we thought the equipment should be controlled with a joystick or any other mechanic/electronic device as is currently done today in the industry. This decision was based on the fact that the Leap Motion currently does not provide a level of precision high enough to be used in a professional tracking system [38]. Therefore, we decided to incorporate the Novint Falcon joystick (which was mentioned in subsection 4.2.3) into the interface to control most of the equipment, e.g. the cranes, the cameras, the manipulator, etc.

The reason we chose the Novint Falcon joystick, was because it has three degrees of freedom, i.e. it can be moved in all directions in 3D space, and it has haptic feedback. Although, we found it later too demanding to get the haptic feedback working in our prototype, so we gave up on that feature. The rest of the features however were still useful to our virtual interface since it was based in 3D space. We stationed the Novint Falcon to the left of the virtual interface, and connected it to the interface via a Unity/Novint Falcon API. Then we programmed the Novint Falcon to initially be able to control the following equipment: the cameras, the cranes, and the manipulator (see Figure 6.5, Figure 6.6,

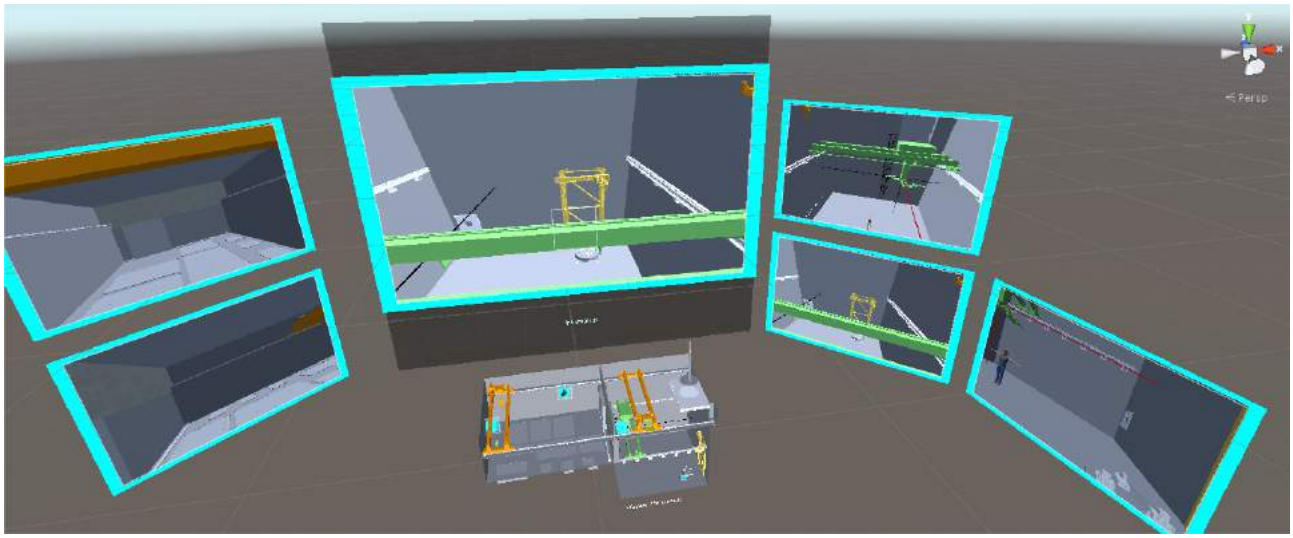


Figure 6.2: Virtual Control Room #6: Sixth look of the design of the Virtual Control Room.

and Figure 6.4 respectively). The Novint Falcon would control each equipment exclusively. Since we applied a physical joystick to control the equipment in the Active Cells Facility, we encountered one of the issues mentioned in subsection 5.2, namely disorientation. Based on which perspective the user was watching the facility model from, it was hard to predict in which direction an equipment would move when the user would manipulate it using the Novint Falcon. To counteract this issue we added a direction indicator above the cranes and the manipulator (see Figure 6.3). Whenever the user would wish to move an equipment, they would choose the direction with the Novint Falcon and check the arrow indicator to confirm if the equipment was going to move in the desired direction. Then the user would press a button on the Novint Falcon to set the equipment in motion in the chosen direction.

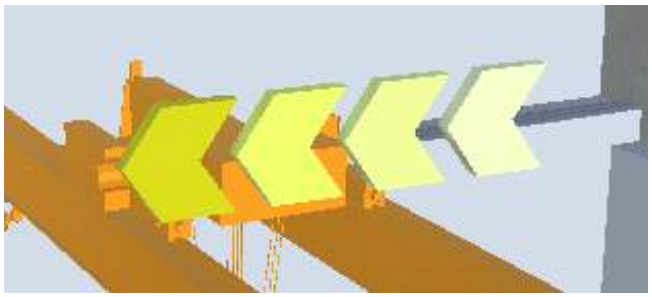


Figure 6.3: A direction indicator above one of the cranes.

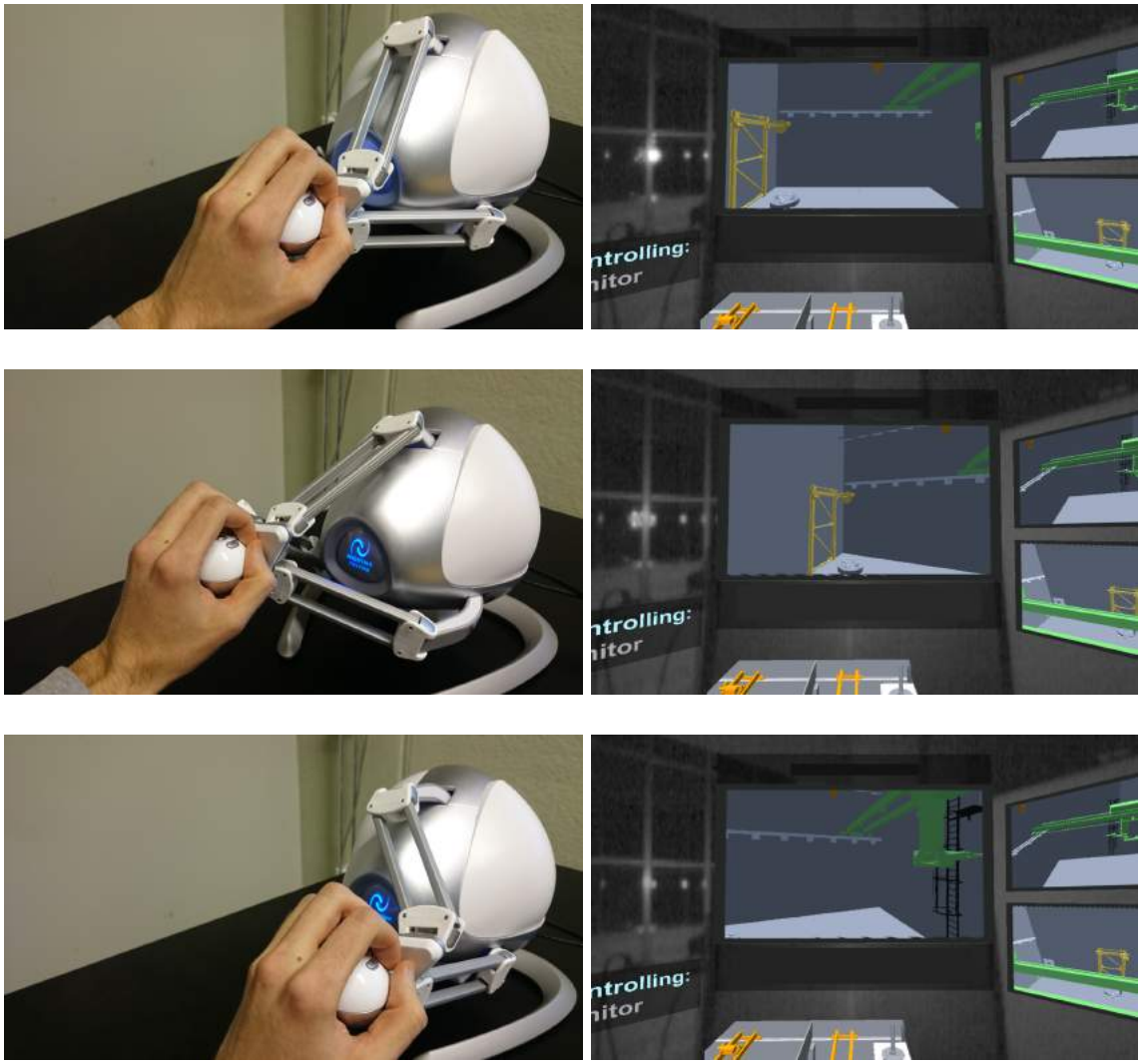


Figure 6.4: Camera control with the Novint Falcon.

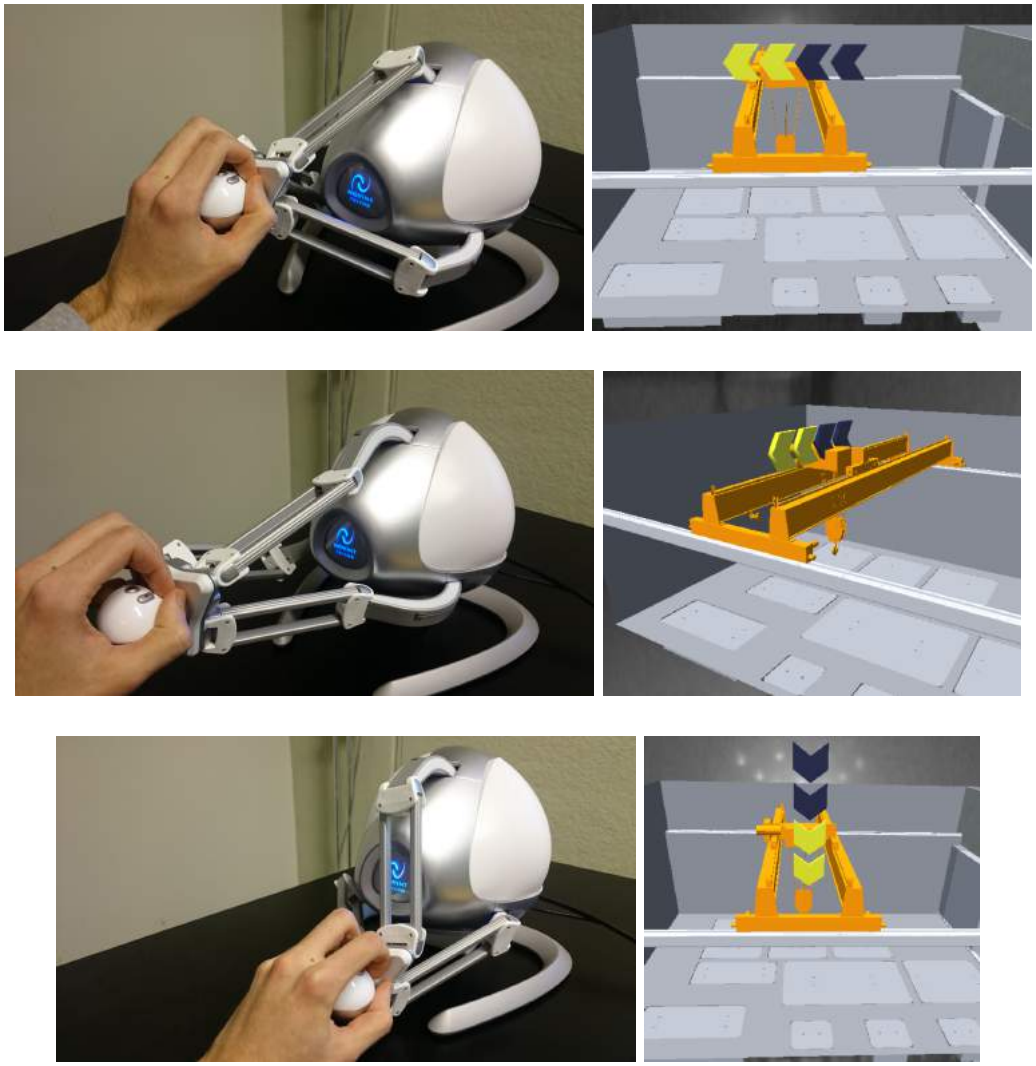


Figure 6.5: Crane/Manipulator control with the Novint Falcon.

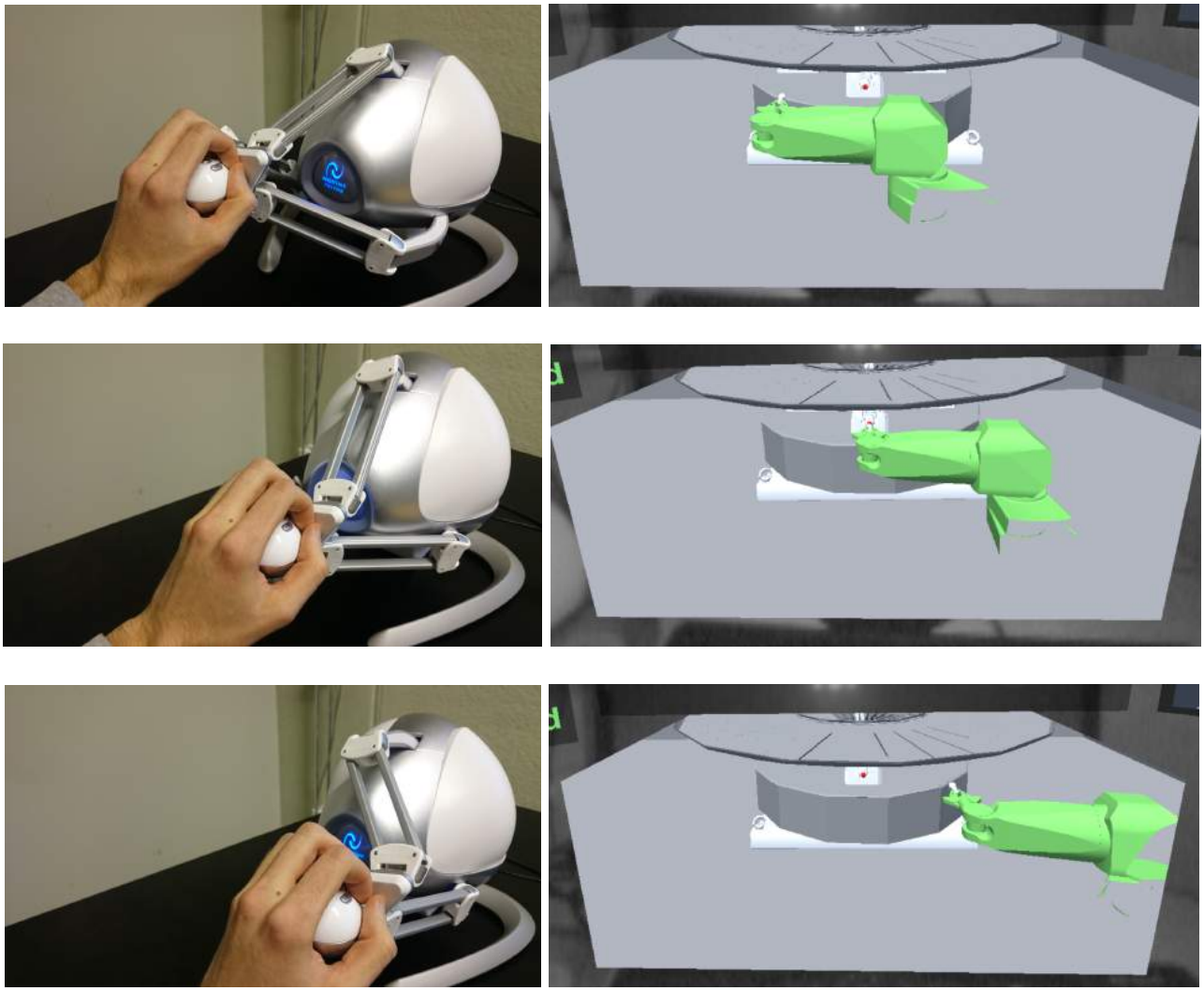


Figure 6.6: Manipulator hand control with the Novint Falcon.

As a consequence of stationing the Novint Falcon to the left of the virtual interface, we decided to remove the monitors on the left side of the interface and replace them with a small information panel right above the Novint Falcon. This panel would inform the user which equipment could be currently controlled with the Novint Falcon. Besides these additions, we also added a 3D reference to the manipulator, information panels, and sound effects. The 3D reference consisted of three red lines, orthogonal to each other, to the walls, and the floor around the manipulator, crossing each other in the middle of the manipulator (see Figure 6.7). The 3D reference was supposed to help the user see more exactly where the manipulator was situated inside the Active Cells Facility relative to the walls and to the floor. This reference could also be seen as an AR feature in the small monitors. As mentioned before, the two information panels were placed above and under the main monitor. The upper information panel informed the user which scenario was to be performed in the Active Cells Facility, and the lower information panel gave the user stepwise instructions on how to complete the scenario. Finally, we added sound effects to the virtual interface. The sound effects were added

because of the feedback principle [36], i.e. use sounds to confirm the operator's actions. The sound effects were given different characteristics so the user could be able to distinguish the actions according to the sounds. The sound effects are as follows:

- When moving the manipulator or the cranes, rolling mechanics can be heard.
- Sound effects when switching camera perspectives on the main monitor either via camera icons or the small monitors.
- Lowering and spinning of the target wheel.
- When moving to a certain target point a beeping sound can be heard. The frequency of the beeps increases, the closer the user gets to the target.

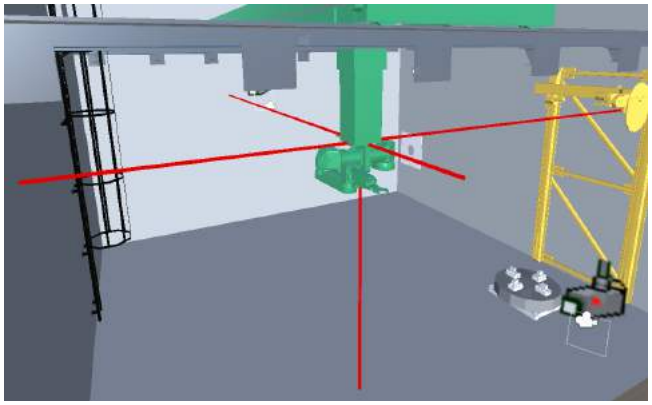


Figure 6.7: An AR 3D reference on the manipulator.

6.2.3 3D Window

To solve the problem of translating, rotating, and scaling the model, we had to make it non-instrusive and easy to use. Non-instrusive in the sense of not blocking the view of any vital or non-vital part of the interface or simply just being unnecessarily in the way. This would especially be a problem if we were to scale (i.e. zoom in and out of) the model.

We realized quite quickly that this had to mean that the model would need to be confined to the space it already occupied, with minimal additional space. After careful discussion and plenty of brainstorming sessions, we came up with an idea that would solve all of these problems. We took inspiration from the *Virtual Window* we experimented with during the initial brainstorming sessions (as described in *Conceptual Design*).

The general idea behind the 3D window is quite simple: There is a confined 3D space (in our case a rectangular box), in which the model is completely visible. Any part of the model that is outside of this space is either semi-transparent or completely invisible. In practice, this would mean that the box would work like the Virtual Window seen through a heptagonal window, or more figuratively a three-dimensional hole into another room.

To implement this, we created two shaders which we applied on the 3D window (the box containing the model) and the model, respectively. The shaders are presented in detail below.

3D Window Shader

The shader that was applied to the 3D window was very simple. The order in which it was rendered (its place in the queue) was set to *transparency*. A stencil mask was also applied, in which the shader always wrote 1. This stencil buffer could be read from and written to by other shaders [39].

The source code can be viewed in Appendix C.1.

Model Shader

The shader that was applied to the model itself (the structure, the components, and the equipment), however, was a little more complex.

First of all, a cutoff rate was defined. If the alpha value of a vertex was less than this value, it was assumed to be invisible and no further calculations were done [39]. This made for faster calculations, as de facto invisible vertices should not have to be rendered.

Second, a stencil mask was applied, which checked if the stencil buffer for the vertex was already set to 1. If this was not the case, the vertex that was being rendered was assumed to be outside of the 3D window and it was rendered either semi-transparent ($\alpha =]c, 1[$) or invisible ($\alpha = [0, c]$), where c is the cutoff rate as defined in the previous paragraph, and α is the transparency where 0 is completely invisible and 1 is completely visible [39].

If the stencil mask was already set to 1, the vertex that was being rendered was assumed to be either inside, in front of, or behind the 3D window. To check this, we needed to define the position of the camera and the position of the corners of the box. These positions could then be used to calculate the distance from the camera to the six planes of the box [40]. When we had calculated the distances to the closest plane ($d_{closest}$) and the plane furthest away ($d_{furthest}$), they were compared with the distance to the vertex (d_{vertex}) that was going to be rendered. Thus we knew the following:

- $d_{vertex} < d_{closest} \implies$ It is located in front of the box. It needs to be invisible and α is set to 0.
- $d_{closest} < d_{vertex} < d_{furthest} \implies$ It is located inside the box. Rendering is done as usual and α is set to 1.
- $d_{furthest} < d_{vertex} \implies$ It is located behind the box. If set to an invisible or semi-transparent state, vital information could be missed. To make the 3D window closer resemble a normal window, rendering is done as usual and alpha is set to 1.

The source code can be viewed in Appendix C.2.

Navigation

When the shaders were applied and the 3D window was parented to the model, navigation was now extremely simple. Everything that needed to be done to focus on a certain place within the model was to place and rotate the box where we wanted to focus and then scale the box to zoom.

The 3D window was made callable (as defined in *Scenario Scripting*), and panning, rotation, and zoom were now accessible through the script.

6.2.4 Inverse Kinematics

When controlling the manipulator, we usually want the hand to reach a target and then grab it. In a real-life situation, a remote master-slave manipulator would be used, and the position and rotation of the individual joints could be sent to the system. This was not possible for us, as we had no access to such a device, and a scaled mockup of the Active Cells Facility was not feasible.

This meant we had to somehow be able to simulate the manipulator and its movements. One way of doing this is by using *forward kinematics*, where we manually control the rotation of each joint.
Using this method, to move the manipulator hand (as seen in Figure 6.8) from point P to point T, we would first have to decrease the angle α and then increase the angle β slightly. This is difficult, as we basically would have to randomly try different angles until we're close enough, and even more difficult as we increase the number of joints.

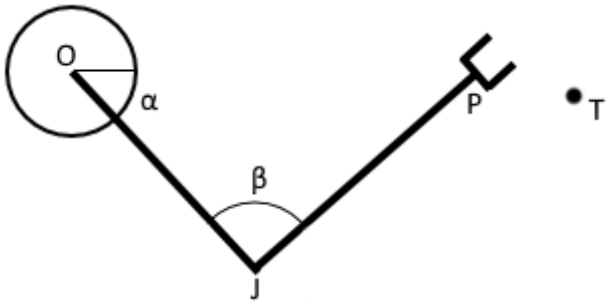


Figure 6.8: Two-dimensional arm with two joints.

The alternative to this would be using *inverse kinematics*, where instead of manually adjusting the rotation of each joint, we move P to T directly and then adjust every joint accordingly.

One problem that could arise when implementing this would be the possibility of multiple solutions. This could however be addressed by introducing constraints. One example of such a constraint would be limiting the angle β to the range $0°$ to $180°$.

Signed Projected Angle

We needed to define the mathemetical function *projAngle*(), which we would use later during the calculation of the angles of the limbs. The parameters we needed were *vector* (the vector to be projected), *normal* (the normal of the plane onto which *vector* will be projected), and *reference* (the vector to which the angle of *vector* would be calculated).

The function *projAngle*() was defined as follows:

```
projAngle(vector, normal, reference) {
    projected = vector - normal * dot(normal, vector);
    angle = 2 * atan(norm(x * norm(y) - norm(x) * y)
                   / norm(x * norm(y) + norm(x) * y));
    if (dot(projected, cross(normal, reference)) < 0)
        angle = -angle;
    return angle;
}
```

The formula for *angle* was based on the formula described in W. Kahan's paper *"How Futile are Mindless Assessments of Roundoff in Floating-Point Computation ?"* [41].

The Manipulator

The manipulator we had to simulate, the rotation of its joints and their corresponding positions can be seen in the schematic in Figure 6.9.

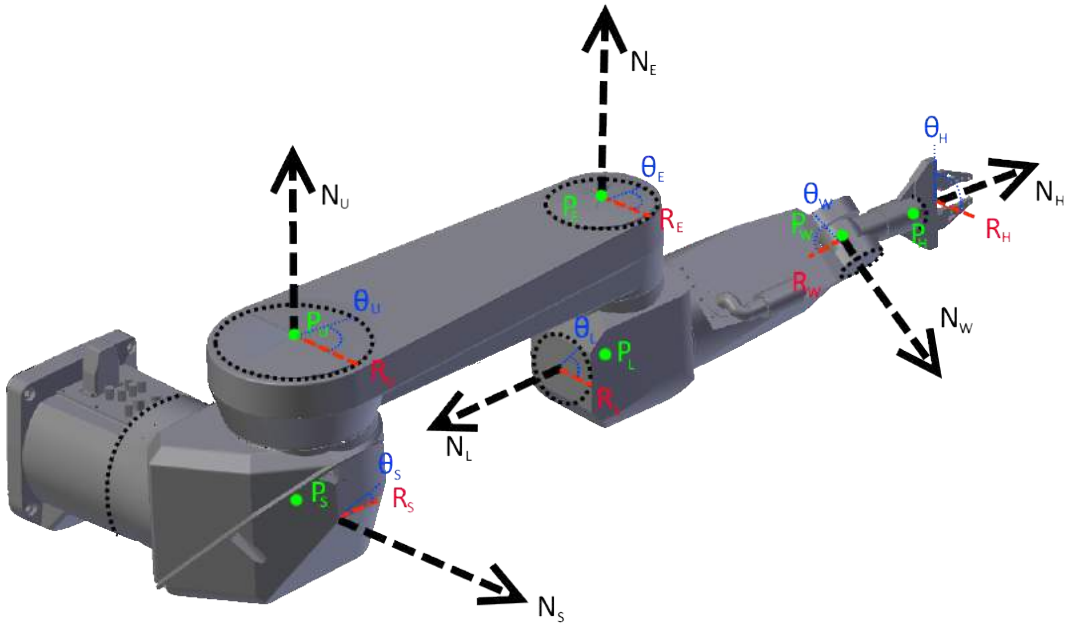


Figure 6.9: The manipulator arm and all its axes.

All of the joints could only be rotated along the dotted line (their axes). The directions of the rotational axes are represented by N. The rotation along these axes is represented by θ. The positions of the joints are represented by P. The twist joints θ_L and θ_H can be rotated 360°.

We had precalculated the following values:

L_U: The length of the upperarm

L_L: The length of the lowerarm

L_H: The length of the hand

The algorithm was done the following way each iteration, given target with position T_P and direction T_D:

- Calculate $P_W = P_T - L_H * D_T$.
- Calculate $\theta_L = projAngle(D_T, N_L, R_L)$. Set the rotation of the lowerarm to this value.
- Calculate $\theta_S = projAngle(P_W - P_S, N_S, R_S)$. Set the rotation of the shoulder to this value.
- Define X_T as the projection of P_W onto the plane defined by the normal N_U.
- Calculate the distance d to the target: $d = min(|P_U - X_T|, L_U + L_L)$
- If $d < |L_U - L_L| \implies$ IK is unsolvable. Stop.
  Else, continue.
- Calculate the angle α between the vector $P_E - P_U$ and the vector $X_T - P_U$.
- Calculate $\gamma = cos^{-1}(\frac{d^2+L_U^2-L_L^2}{2L_U d})$
- Calculate $\theta_U = \alpha + \gamma$. Set the rotation of the upperarm to this value.
- Align the elbow (θ_E) such that the underarm points at X_T.
- Align the wrist (θ_W) such that the hand points at P_T.
- Set the rotation of the hand (θ_H) to desired grip angle.

One problem that could arise, however, is the ambiguity of the direction of the elbow. This could partly be solved by changing the sign of γ depending on the desired direction of the elbow. If the target crossed the plane defined by the point P_S and the normal N_S, the sign of θ_U needed to be changed to preserve the direction of the elbow. Our algorithm is based on and inspired by the algorithms described in *Robotic Manipulation* by Richard M. Murray, Zexiang Li, and S. Shankar Sastry [42].

6.3 Companion Sphere

> *...she kinda reminds me of HAL.*
> - Test user

When we let people test the interface, we executed one of the scenario scripts we had prepared to simulate the operations of the Active Cells Facility. The problem, however, was that we still needed to tell the users what to do and when, as there was no feedback and no clear instructions given to the users. There was a small screen that showed the current task that needed to be done, but this screen was mostly ignored or misunderstood. There was also the problem of not being able to keep the user experience persistent.

The logical step was therefore to increase the amount of valuable feedback and expand on the instructions given to the users, and preferably automate it all so as to make it persistent and independent from our presence during testing.

We created a virtual avatar, with a very simple appearance: A small blue semi-transparent sphere, with the ability to speak instructions to the user from an audio recording. When speaking, it varied its radius in accordance with the sound waves generated from the audio recording. This proved to be very effective, as it made it seem very lifelike according to our testers, and we got many positive reactions to it. It can be seen in Figure 6.10.

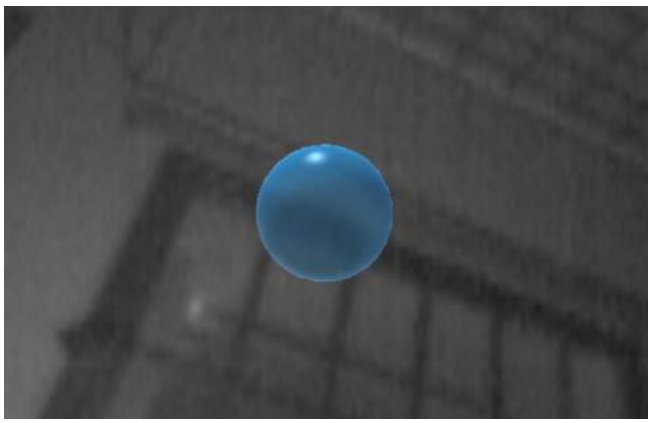


Figure 6.10: The semi-transparent companion sphere that guides the user through the simulation.

We scripted a dialog, which was then converted, with a Text-to-Speech application, to a synthetized voice audio recording. The dialog first introduced and tutored the user on how to use the interface, then guided the user through all the instructions, and also corrected the user if anything was done improperly. The audio recording of the dialog was then added to the companion sphere, which in turn was activated from the scenario script.

The final dialog can be read in *Appendix* D.

6.4 Conclusions from the Second Iteration

Before we added the companion sphere to the prototype, we did some informal tests with a couple of volunteers. The volunteers tested the prototype after we gave them brief instructions on how it worked, and with their feedback we made some final small changes to the interface, which mostly consisted of changing colors on some objects, and make the instructions on the information panels more clear. Otherwise, the volunteers were very positive about the prototype. We decided then to summon our supervisors from ESS and the university to a meeting where they would be able to evaluate the prototype and give us their own feedback. They told us that the prototype was satisfactory and that it was a good proof of concept. After this feedback, we decided to to put a stop to any further development of the prototype, and instead focus on designing and performing user tests. From our point of view, the design counteracted the issues mentioned in section 5.2 (i.e. disorientation, visual limitation, and low 3D perception) very effectively, it covered the use cases, and it was possible to complete the first scenario easily with it. We had to determine in the user test phase if the users agreed. By adding the companion sphere to the prototype, we created a demo which could help the test users to master and comprehend the interface with minimal assistance from us developers. This demo with the final prototype was used in the user test phase described in the next chapter. The final design can be seen in Figure 6.11.

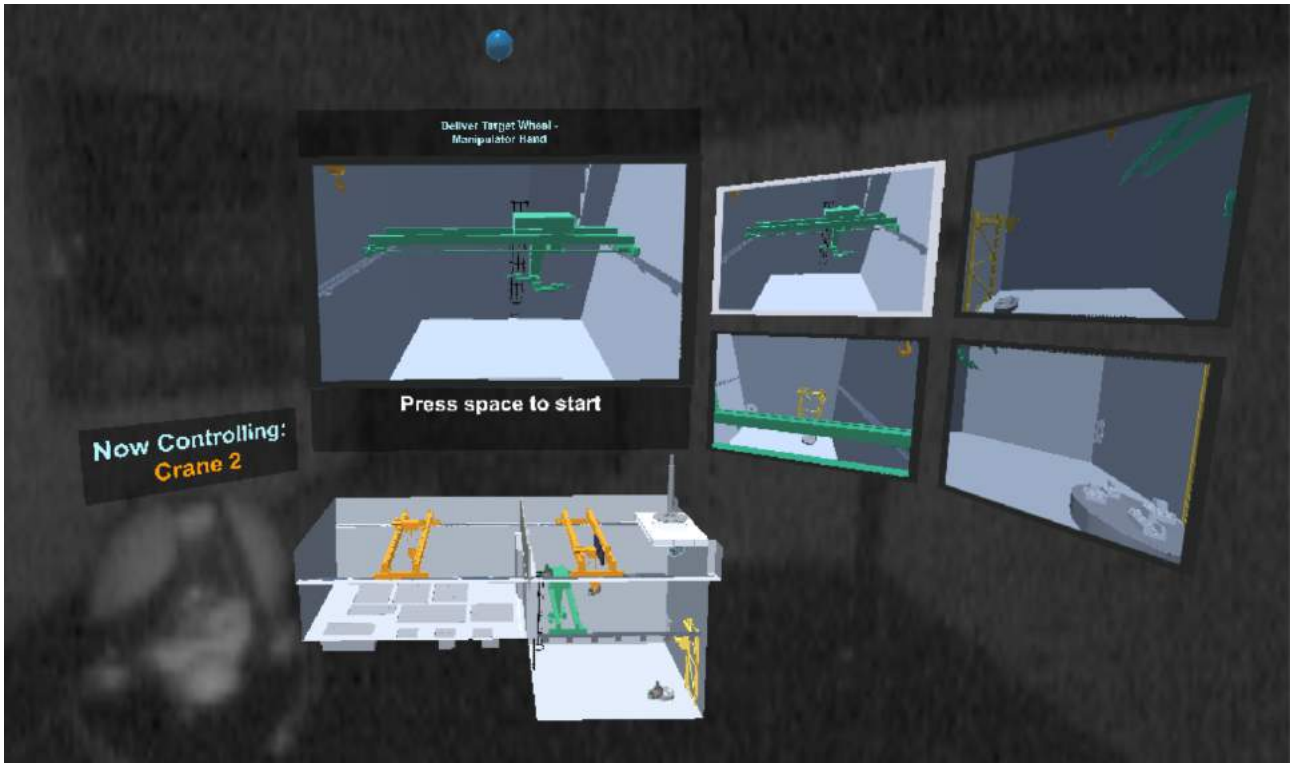


Figure 6.11: The final version of the Virtual Control Room.

7 User Tests

Once we had a fully functional demo, we aimed to perform tests on a certain user group to evaluate our final prototype from different aspects and answer the following questions:

- Are the VR/AR functionalities in the prototype adequate?
  For example, do the users find the Digital Reality interface helpful or not?
- Is the interface intuitive, i.e. are the users able to easily navigate through the interface and understand what they are supposed to do?
  For example, do the users learn quickly? Are the tasks easy to understand and natural to carry out?
- How is the VR experience perceived?
  For example, do the users feel comfortable in the virtual environment? Do they find the interface attractive? Do they find it stressful?

By letting the users complete the whole demo, we hoped to gain answers to all of these questions. We had to compose the tests to gain useful answers from the users, and not just yes/no answers. The goal with the user tests was to try to obtain positive as well as critical answers. Positive answers are useful to confirm that the idea behind the prototype is feasible and satisfactory, and which functionalities and properties of the prototype are worth keeping and further develop. Critical answers are useful to detect and fix flaws in the prototype, and improve functions where they seem to be failing or disappoint significantly, or completely remove them.

7.1 Method

The users in these tests were all voluntary participants, who had heard about our prototype either from our presentation or through our supervisor. There were 14 in total, and all were male. They had different backgrounds, but were all affiliated with ESS in some way.

The first approach on how to perform the user tests was to let each user complete the whole demo. During this entire session we would make notes on every oddity or problem that arose when a user was performing a certain task in the demo. These oddities could consist of, for instance, unclear or unintuitive instructions on what the user was supposed to do, failures to notice certain objectives, and unnatural interaction for the user. If the user had difficulties in progressing with the demo, we intervened and explained what needed to be done in order to continue the demo. Sometimes, technical

difficulties occurred, which we quickly remedied. The overall point with letting a user do the demo was to examine whether the prototype was intuitive enough for the users to complete the demo efficiently by themselves. Once a user completed the demo, we asked them some informal questions, e.g. how did they feel, did they like the interface, what troubles did they encounter while performing a task, and what were their thoughts when they tried to perform it. Then we let them answer voluntary questionnaires which would help us measure the satisfiability of our prototype.

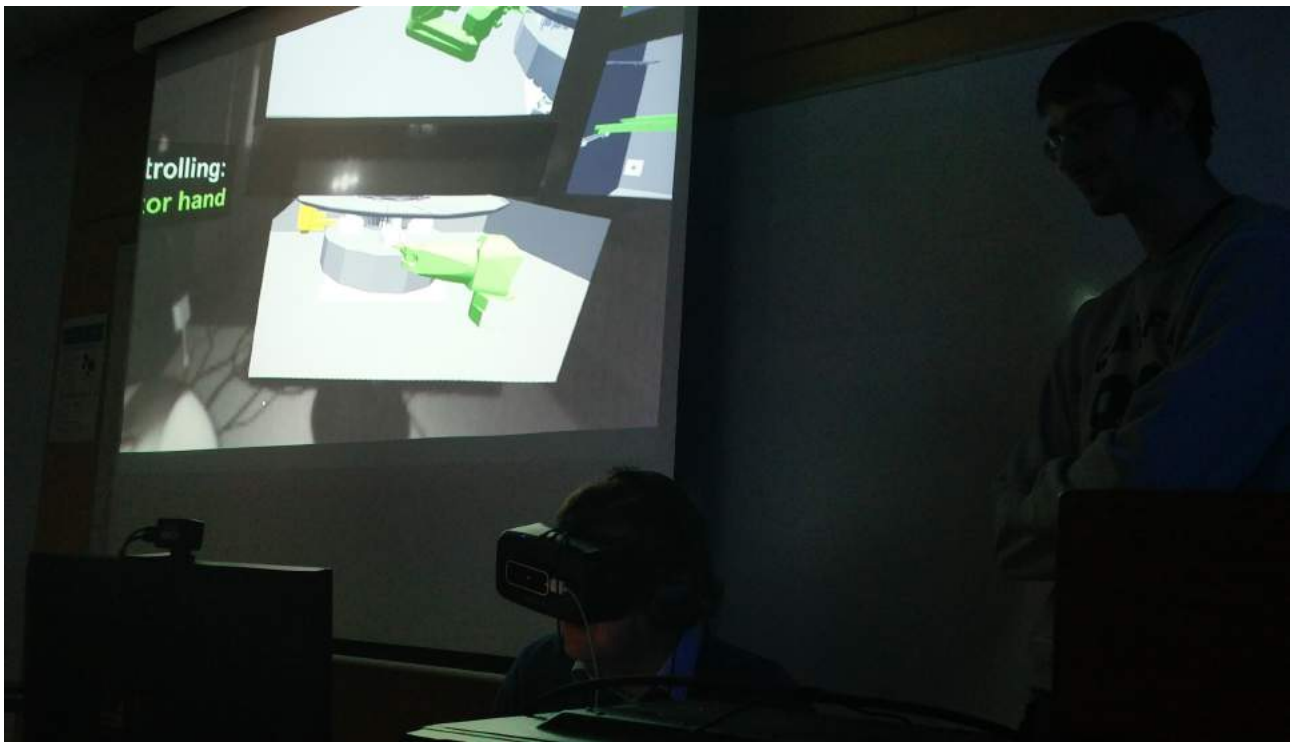


Figure 7.1: One of the participants trying to use the manipulator arm to fasten the bolts.

7.1.1 Questionnaires

We used two questionnaires that the users were asked to fill out at the end of the user test. These were accompanied by a form where the users were asked to fill in their age, sex, their previous experience with gaming, and their previous knowledge of the operations within the Active Cells Facility.

The first questionnaire was based on AttrakDiff [43], which is a tool for assessing usability and design developed by Marc Hassenzahl and Michael Burmester [44]. It gives a subjective score on how the user rates the pragmatic quality, the hedonic quality, and the attractiveness of the interface. The users were given ten word pairs and were asked to rate each word pair on a scale from 1 to 7. The word pairs in the AttrakDiff were modified to fit our purpose and task, and thus it should not be mistaken as the original questionnaire, but should instead be seen as a derivative of it. All the word pairs were randomly ordered, both row-wise and column-wise.

The second questionnaire was based on NASA-TLX [45], which is an assessment tool for perceived

workload developed by NASA. Its purpose was to find out how physically and mentally demanding the users found the interface.

The complete questionnaires can be found in Appendix B.

7.2 Results

The users' background data are shown in Figures 7.2, 7.3, and 7.4.
The results from AttrakDiff are shown in Figures 7.5 to 7.14.
The results from NASA-TLX are shown in Figures 7.15 to 7.20.

7.2.1 User Background Data

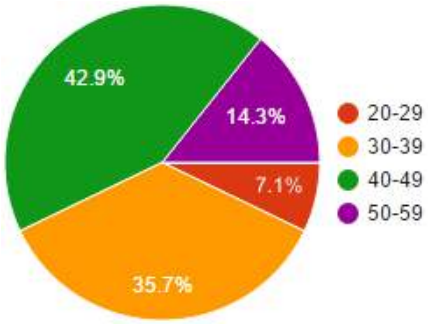


Figure 7.2: How old are you?

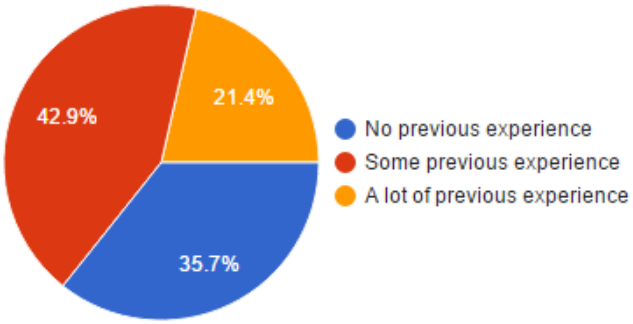


Figure 7.3: How familiar are you with gaming?

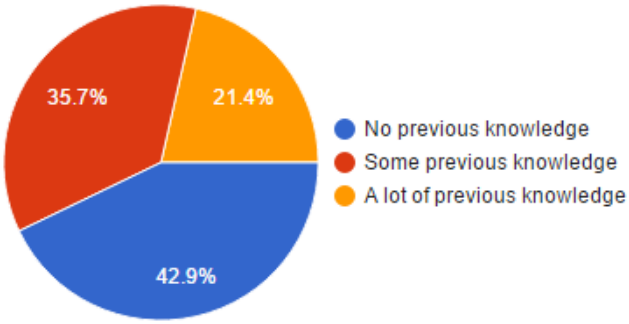


Figure 7.4: How familiar were you with the operations of the Active Cells Facility before this?

7.2.2 Results from AttrakDiff

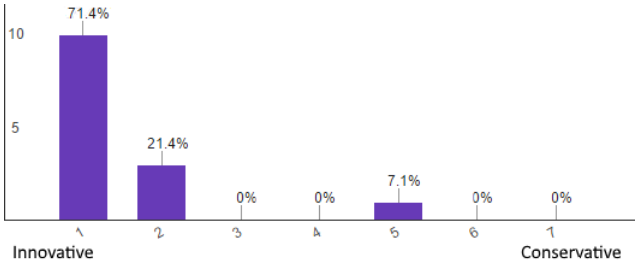


Figure 7.5: Innovative - Conservative

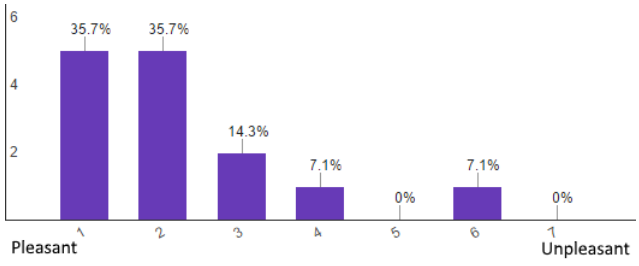


Figure 7.6: Pleasant - Unpleasant

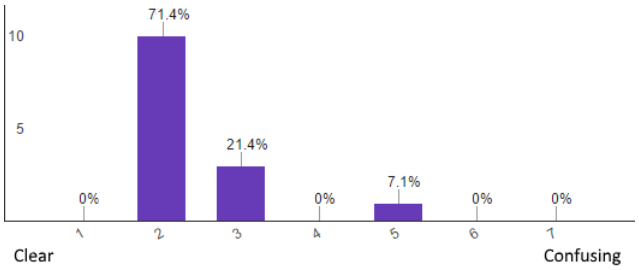


Figure 7.7: Clear - Confusing

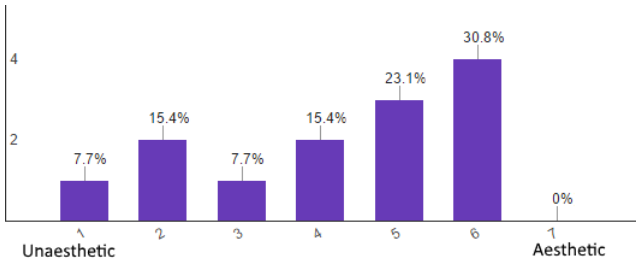


Figure 7.8: Unaesthetic - Aesthetic

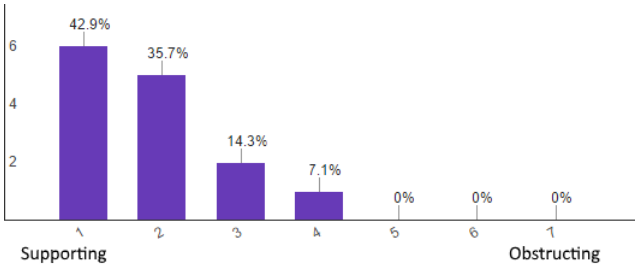


Figure 7.9: Supporting - Obstructing

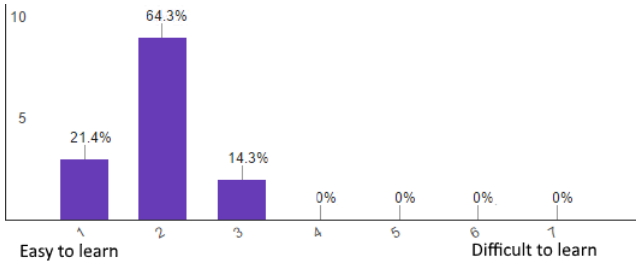


Figure 7.10: Easy to learn - Difficult to learn

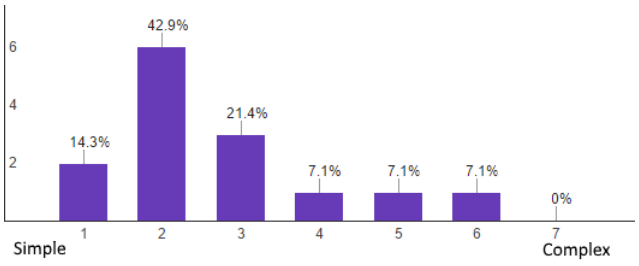


Figure 7.11: Simple - Complex

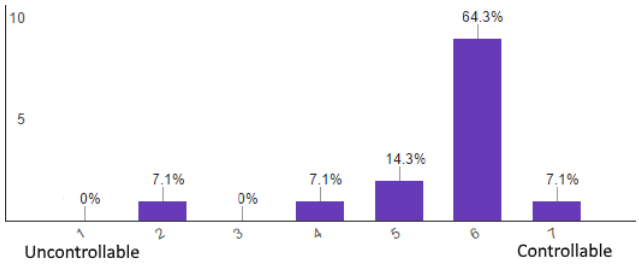


Figure 7.12: Uncontrollable - Controllable

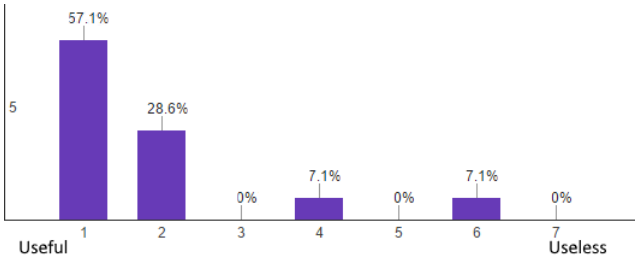


Figure 7.13: Useful - Useless

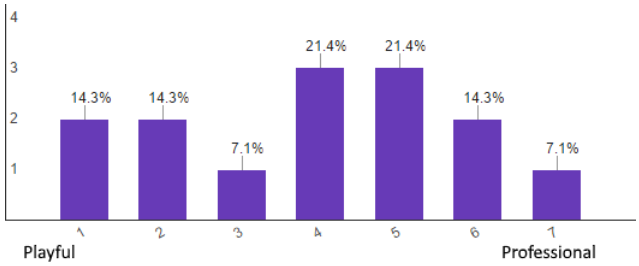


Figure 7.14: Playful - Professional

7.2.3 Results from NASA-TLX

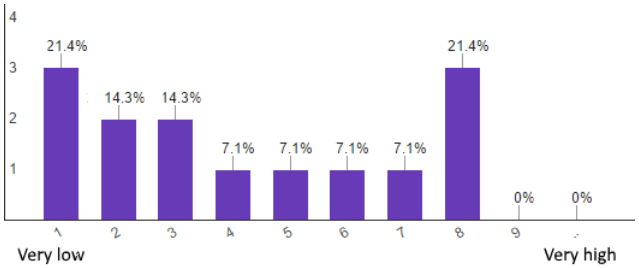


Figure 7.15: How mentally demanding was the task?

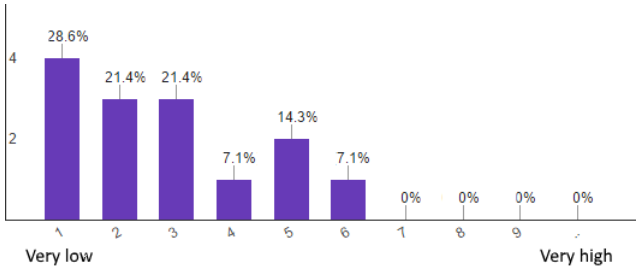


Figure 7.16: How physically demanding was the task?

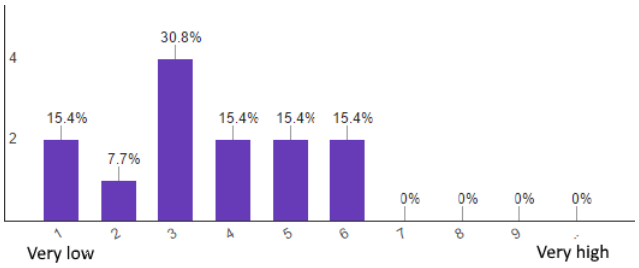


Figure 7.17: How hurried or rushed was the pace of the task?

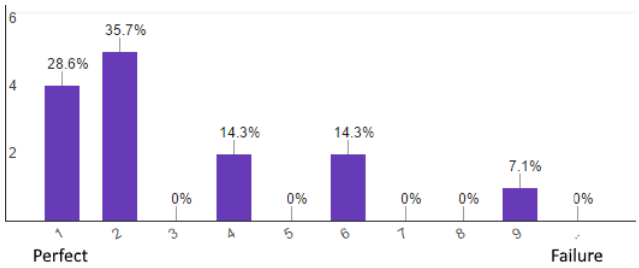


Figure 7.18: How successful were you in accomplishing what you were asked to do?

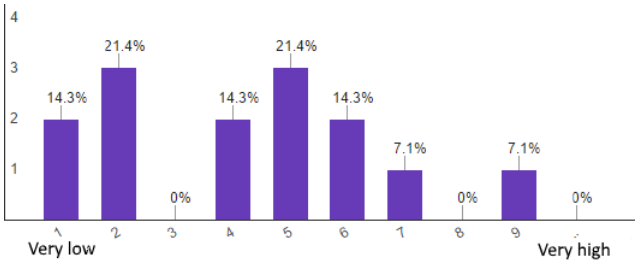


Figure 7.19: How hard did you have to work to accomplish your level of performance?

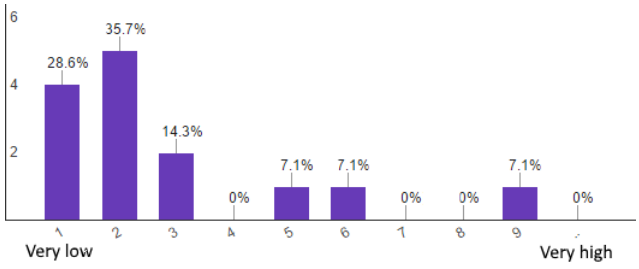


Figure 7.20: How insecure, discouraged, irritated, stressed, and annoyed were you?

7.3 Conclusions from User Testing

The test results were generally positive. Most test users found the prototype (in order of appearance) innovative, pleasant, clear, somewhat aesthetic, supporting, easy to learn, simple, controllable, and useful. We received mixed opinions on whether the prototype was playful or professional, with no general tendency either way (see Figure 7.14). This could be because of the gamelike tutorial and dialog.

Nothing decisive could be said about the mental demands of the tasks (see Figure 7.15), as both extremes (very low demand and very high demand, respectively) were equally represented. The amount of perceived effort (as seen in Figure 7.19), is somewhat evenly spread, but most seem to be in the middle. This is probably mostly because of the novelty of the technology, the nonexistent haptic feedback, and the inaccuracy of the LEAP Motion IR sensor. However, many users reported very low amount of stress, irritation, and anxiety (see Figure 7.20), and a very positive feeling towards the interface (see Figures 7.5 to 7.14). This could indicate that, while the technology was slightly lacking in performance, the interface itself was suitable for this kind of task. People were excited to try it out and some even said they felt inspired.

The subjects generally reported very low physical demands (see Figure 7.14), as was expected because of the nature of the task. They also reported being somewhat unhurried (see Figure 7.17), and all of them finished the test in the range of 10 to 15 minutes. From our own observations, those who reported having a lot of previous gaming experience, also seemed to experience less difficulty in completing the demo.

Many users had problems with properly navigating the manipulator arm when trying to fasten the bolts of the target wheel. However, all of them quickly learned to do this properly after only a few times. Many users said that the auditory feedback (the beeps getting stronger as the manipulator hand approached the target) was very useful and probably helped them a lot. Some said they found the Novint Falcon joystick difficult to use.

From these results, we would conclude that the participants found the proposed interface useful.

8 Conclusions & Discussion

8.1 Conclusions

We successfully built a testing environment, which simulated the remote control of the Active Cells Facility at ESS, explored different ideas of visualizing and operating the equipment, and iteratively developed a user friendly interface based on frequent user testing and evaluation. We finally created a formal user test inspired by established and standardized tests from both NASA and AttrakDiff, and tested the feasability of the prototype on users with this test. Combined with the research we had done and the challenges we had experienced, we could finally get an idea of the requirements that such an application of DR would set on both hardware and software within the scope of this thesis.

The results showed that the users found the DR environment useful and that the application of DR for the operations at the Active Cells Facility is indeed very promising. However, it is also important to remember that DR is not necessarily useful in and of itself. Traditional methods of interaction can sometimes prove to be more efficient and easier to use than those provided in DR, and there are also cases where DR might actually impede the user, as we discovered in this thesis. The kind of DR that will eventually prove to be useful, heavily depends on the tasks to be performed and the technology that is available at the time. This means that to really benefit from DR, it is important to identify the tasks to be performed and the difficulties in performing these, as well as the capability and limitations of the hardware at hand. Once these have been identified, it is easy to create an appropriate DR environment with complementing non-DR technology for efficiently performing the tasks.

8.2 Challenges

We had a few problems with the haptic control, as it had a very small volume in which it could be moved. There was also no real sense of minimum and maximum as its range was arbitrary, i.e. there was no intuitive way of telling if the control had been fully moved to either side just by looking at it. However, we reasoned that, because the actual real-life operations would be using a master-slave manipulator, this was a problem that would not be present in the actual remote control operations. Also, vertical and lateral movement of the cranes and other equpiment would preferrably be done by dedicated levers to each direction and task, in order to ensure precision and decrease confusion.

We also tested moving equipment using virtual levers and sliders, but this turned out to be very hard to use accurately. It was discovered that putting too much DR interaction at one place in the interface (high density of virtual buttons) can cause the user to accidentally interact with the interface and activate a function. We therefore believe that no vital operations of heavy equipment should rely

on input through the virtual interface.

When programming the shaders for the 3D window, we noticed times when the frame rate would drop and render the simulation useless. This was mostly due to the fact that we had to render transparent and semi-transparent objects, which meant that completely occluded objects were also rendered. This was remedied by reducing the number of complex structures or replacing them with more primitive ones. There is, for example, no reason to render a highly complex hatch with tens of thousands of polygons if it could just as easily be represented by a simple primitive with a few hundred polygons. Most times, the user would never be able to spot the difference, and in the cases where they would, it would not be especially important. The goal was to visualize the Active Cells Facility, not realistically representing it on all physical layers. Further reduction of the number of polygons was possible, but would be quite time-consuming, as this would ideally require remodelling the entire model.

Since the prototype was developed with a user centered design approach, the actual end-users should have been interviewed for their experiences in their line of work, as well as for their opinions on what the prototype should be able to achieve. As there were no end users present at the time when this thesis was conducted, all important or interesting information had to be extracted from available documentation. There is however one point to consider which our employer warned us about, namely the possibility that an interview with the end users would not have proven to be as rewarding as it would have initially seemed. The end users have a long time of experience of remotely operating machines by means of using a window or several monitors to supervise the operations. It is probable they would be reluctant to any major changes (adding DR) in the way of supervising the system, and thus would not provide many exact suggestions for improvements. In the end, it might have been necessary to extract the relevant information from the documentation anyway.

8.3 Discussion

Despite the challenges we had with the hardware, the DR environment in the prototype showed great promise. Based on our results, it can be said that there is great potential for DR in similar environments to that of the Active Cells Facility at ESS. Given more accurate and advanced DR technology combined with both access to the physical control room and the actual requirements defined by the end-user, a more robust and useful prototype may be developed in the future.

There are many advantages with DR in the Active Cells Facility, but the most outstanding one is the amount of overview and control DR provides over the supervised environment. The overview and control of the Active Cells Facility seemed to come more naturally with DR than through a computer screen or through a window. This is an advantage which might be greatly appreciated and applicable in other types of industries. On construction sites, machinery could be operated more safely [46]. In warehouses, goods could be stored and managed more efficiency [47]. In theaters or on movie sets, the lighting and furnishing of scenes could be setup more visibly for the director. Infrastructure of water, oil, and gas pipelines [48], and electricity [49] could be made more easily viewable. Robots and machinery on assembly lines [47], in factories [47], and power plants [49], or remotely handled space shuttles could be controlled more intuitively [50]. The operations of facilities on extremely distant places such as Mars could also greatly benefit from this kind of technology [50]. In architecture con-

struction of tall buildings, complicated facilities, or custom houses could be made more comprehensive and illustrative [51]. The possibilities are many.

The main advantage with DR in all these applications and other fields that include monitoring or remote handling, is that work efficiency and security could be considerably increased. When supervisors are observing a working field from a monitor, a window (in a wall, cabin, studio, etc.), or in real life, they usually only see parts of the whole area and complement the rest with their own judgement to get a full mental picture of the working field. With DR one would finally have a full overview of the entire area and be able to see what could only be imagined before. This way, decisions can be made faster and with greater confidence, while making sure that no security and safety measures are skipped.

8.4 Future Work

As we chose the Novint Falcon for increased freedom of movement in 3D space, a proper simulation of the physical environment of the Active Cells Facility and its equipment would definitely have been of great use. The simulation could have provided haptic feedback to the users, for example when the manipulator arm was able to move through the target wheel without the users realizing it. Another great area of interest would be better gesture recognition hardware, as most of the problems in the interface we encountered was due to low precision in the LEAP Motion sensor. In the future, when the precision of the tracking equipment has been improved, then it can be tested again if it is reliable enough for controlling more sensitive accessories. Till then, DR with interaction functions (at least the one provided by the Leap Motion) should not be used for controlling sensitive/heavy/dangerous equipment. Both the Atheer AiR [52] and the Magic Leap [37] (which is *not* related to Leap Motion, despite its name) are reported as having planned such a feature in their upcoming devices. These devices are striving to rival Microsoft's HoloLens and will be providing a comparable set of features [53].

Another thing that would have been useful to implement was the second storyboard (as described in *Conceptual Design*) and test it with the same users, and perhaps without the tutorial. Similarly, formal testing of the users with no tutorial at all could have been interesting. However, as we regularly already performed informal testing using no tutorial, we did not find it particularly important.

During the formal tests, we noticed that many users primarily used either the cameras or the model alone. Tests performed using only the cameras, only the model, and both the cameras and the models, respectively, could also have yielded very interesting results.

Making the *Virtual Control Room* more collaborative could prove to be extremely valuable and would be worth investigating further. To make it more collaborative, there would have to be some way of visualizing and interacting with it at the same time together with the other operators. This is not possible using multiple off-the-shelf Oculus Rifts, and probably will not be in the near future. Although the technical requirements themselves would actually be there in the first consumer version, Oculus CEO Luckey Palmer said they "are not promising this as a feature" [54]. It would be theoretically possible, however, to make the necessary changes ourselves. A common reference point would be needed, and using techniques such as that provided by Qualcomm's Vuforia [55] combined with

the fact that Oculus CV1 will be providing 360° freedom of movement, this could possibly prove to be quite feasible [14]. Other solutions are very likely to come up as DR products are getting more popular and the industry more established, as such a feature is in very high demand.

The VR landscape will be changing drastically the coming years, and as of right now, we believe that Oculus Rift [14], HTC Vive [56], Microsoft HoloLens [11], Meta [57], Magic Leap, and Atheer AiR are the prime focus of interest for this kind of application. The Oculus Rift, HTC Vive, and the Atheer AiR are expected to be released during 2016, while the HoloLens is expected to be released to developers first quarter of 2016 [58]. Magic Leap has not yet released any information about a release.

2016 has been dubbed "the year of Virtual Reality" by Fortune Magazine [59], The Guardian [60], among many others, and rightly so. What is lying ahead will be - to say the least - very exciting.

Bibliography

[1] **European Spallation Source** (2015) *"Open Call for Tender"*
https://europeanspallationsource.se/sites/default/files/oct-2015-180103001-001.pdf

[2] **Srdjan Vareskic** (2014) *"ICD-R: Active Cells - Integrated Control Systems"*
ESS-0020432

[3] **Dix A., Finlay J., Abowd G. D., & Beale R.** (2004) *"Human-Computer Interaction"*
Harlow: Pearson Education Limited

[4] **Wigdor D. & Wixon D.** (2010) *"Brave NUI World: Designing Natural User Interfaces for Touch and Gesture"*
Burlington: Morgan Kaufmann Publishers

[5] **Blade R. A. & Padgett M. L.** (2015) *"Virtual Environments: Standards and Terminology"*, in: **Hale K. S. & Stanney K. M.** (Eds.) *"Handbook of Virtual Environments: Design, Implementation, and Applications"*
Boca Raton: Taylor & Francis Group

[6] **Milgram P., Takemura H., Utsumi A., & Kishino F.** (1994) *"Augmented Reality: A class of displays on the reality-virtuality continuum"*
http://web.cs.wpi.edu/~gogo/hive/papers/Milgram_Takemura_SPIE_1994.pdf

[7] **Johnson E.** (2015) *"Choose Your Reality: Virtual, Augmented or Mixed"*, in: *Recode*
http://recode.net/2015/07/27/whats-the-difference-between-virtual\-augmented-and-mixed-reality/

[8] **Heilig M.** (1962) *US Patent #3,050,870*

[9] **Azuma R. T.** (1997) *"A Survey of Augmented Reality"*
http://www.cs.unc.edu/~azuma/ARpresence.pdf

[10] **Qian N.** (1997) *Binocular Disparity and the Perception of Depth*, in: *Neuron*
http://brahms.cpmc.columbia.edu/publications/stereo-review.pdf

[11] **Microsoft HoloLens** (accessed: 2016-03-09)
https://www.microsoft.com/microsoft-hololens/

[12] **IGN** (2015) *"Microsoft's HoloLens Is Actually Amazing"*
https://www.youtube.com/watch?v=lttveKXDaXg

[13] **Oculus Rift DK2** (accessed: 2016-04-19)
https://www.oculus.com/en-us/dk2/

[14] **Oculus Rift** (accessed: 2016-03-09)
https://www.oculus.com/

[15] **Leap Motion** (accessed: 2016-03-20)
http://www.leapmotion.com/

[16] **AutoCAD** (accessed: 2016-03-09)
http://www.autodesk.com/products/autocad/overview

[17] **Dawber D.** (2013) *"Learning Raphaël JS Vector Graphics"*
Birmingham: Packt Publishing

[18] **Gregory J.** (2015) *"Game Engine Architecture"*
Natick: A K Peters, Ltd.

[19] **Unity** (accessed: 2016-03-09)
https://unity3d.com/

[20] **Badcock D. R., Palmisano S., & May J. G.** (2015) *"Vision and Virtual Environments"*, in: **Hale K. S. & Stanney K. M.** (Eds.) *"Handbook of Virtual Environments: Design, Implementation, and Applications"*
Boca Raton: Taylor & Francis Group

[21] **Angel E.** (2010) *"Interactive Computer Graphics: A Top-Down Approach Using OpenGL"*
Boston: Pearson Education

[22] **European Spallation Source** (2014) *"Active Cells - Integrated Control Systems"*

[23] **Jacobson I.** (2011) *USE-CASE 2.0: The Guide to Succeeding with Use Cases*
https://www.ivarjacobson.com/sites/default/files/field_iji_file/article/use-case_2_0_jan11.pdf

[24] **Buxton B.** (2007) *"Sketching User Experiences"*
San Francisco: Morgan Kaufmann Publishers

[25] **Arvola M.** (2015) *"Interaktionsdesign och UX"*
Lund: Studentlitteratur AB

[26] **Tobii** (accessed: 2016-03-09)
http://www.tobii.com/

[27] **TrackIR** (accessed: 2016-03-09)
https://www.naturalpoint.com/trackir/

[28] **Microsoft Kinect** (accessed: 2016-03-09)
https://dev.windows.com/en-us/kinect

[29] **Boman E.** (2015) *"Virtual Window using Unity 5 and Kinect 2 - Dog"*
https://www.youtube.com/watch?v=Q4U9maVCOJg

[30] **DIFFER Remote Handling** (2013) *"Remote Handling Analysis of ITER components"*
https://www.youtube.com/watch?v=8NJmnnKAjWQ

(2013) *"Synthetic Viewing"*
https://www.youtube.com/watch?v=gBVH3fPDc7E

(2012) *"Welding Tool (physical mockup)"*
https://www.youtube.com/watch?v=xK3Tx3pOYWQ

(2012) *"Welding Tool (VR)"*
https://www.youtube.com/watch?v=GZNc_cSQ4no

[31] **EFDAJET** (2010) *"Remote Handling at JET"*
https://www.youtube.com/watch?v=bqwNptq6ZIM

(2012) *"JET_VTT_RH_SbS_EN.mp4"*
https://www.youtube.com/watch?v=YYj53Z5GbBw

(2010) *"Remote Handling at JET"*
https://www.youtube.com/watch?v=SPkgEubKRSk

[32] **ITER Organization** (2009) *"ITER - Maintain and upgrade: ITER's Remote Handling"*
https://www.youtube.com/watch?v=xfJQJI8jEnA

[33] **Novint Falcon** (accessed: 2016-03-09)
http://www.novint.com/index.php/novintfalcon

[34] **Haption** (accessed: 2016-03-09)
http://www.haption.com/site/index.php/fr/

[35] **Constantine L. & Lockwood L.** (1999) *"Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design"*
Boston: Pearson Education

[36] **Norman D.** (2013) *"The Design of Everyday Things"*, Revised and Expanded Edition
Basic Books
http://cc.droolcup.com/wp-content/uploads/2015/07/The-Design-of-Everyday-Things-Revised-and
pdf

[37] **Magic Leap** (accessed: 2016-03-09)
http://www.magicleap.com/

[38] **Guna J., Jakus G., Pogačnik M., Tomažič S., & Sodnik J.** (2014) *"An analysis of the precision and reliability of the leap motion sensor and its suitability for static and dynamic*

tracking."
http://www.ncbi.nlm.nih.gov/pubmed/24566635

[39] **Unity - Manual: Shader Reference** (accessed: 2016-04-21)
http://docs.unity3d.com/Manual/SL-Reference.html

[40] **Wolfram MathWorld: Point-Plane Distance** (accessed: 2016-04-21)
http://mathworld.wolfram.com/Point-PlaneDistance.html

[41] **Kahan W.** (2006) *"How Futile are Mindless Assessments of Roundoff in Floating-Point Computation ?"*, in: *Mindless*
https://www.cs.berkeley.edu/~wkahan/Mindless.pdf

[42] **Murray R. M., Li Z., & Sastry S. S.** (1994) *"Robotic Manipulation"*
Boca Raton: CRC Press

[43] **AttrakDiff** (accessed: 2016-03-09)
http://attrakdiff.de/

[44] **Hassenzahl M., Platz A., Burmester M., & Lehner K.** (2000) *"Hedonic and Ergonomic Quality Aspects Determine a Software's Appeal"*, in: **Turner T. & Szwillus G.** (Eds.) *Proceedings of the CHI 04 Conference on Human Factors in Computing Systems. Extended abstracts*
New York: ACM

[45] **NASA Ames Research Center** (1986) *"NASA Task Load Index (TLX) v. 1.0"*
http://humansystems.arc.nasa.gov/groups/TLX/downloads/TLX.pdf

[46] **Grayson W.** (2015) *"How Caterpillar is developing virtual and augmented reality to design and service heavy equipment"*, in: *Equipment World*
http://www.equipmentworld.com/how-caterpillar-is-developing-virtual-and-augmented-reality-t

[47] **McCutcheon R.** (2016) *"Manufacturers are putting virtual reality to work... and in surprising ways"*
http://usblogs.pwc.com/industrialinsights/2016/01/19/manufacturers-are-putting-virtual-real

[48] **Santos I. H. F., Soares L. P., Carvalho F, & Raposo A.** (2012) *"A Collaborative Virtual Reality Oil & Gas Workflow"*, in: *The International Journal of Virtual Reality*
http://www.ijvr.org/

[49] **Fortum** (2016) *"Fortum invites start-ups to hackathon"*
http://www.fortum.com/en/mediaroom/pages/fortum-invites-start-ups-to-hackathon.aspx

[50] **Reichhardt T.** (2015) *"Telepresence and Virtual Reality May Be the Keys to Mars Exploration"*
http://www.airspacemag.com/space/telepresence-space-mars-180957306/

[51] **Gaudiosi J.** (2015) *"How this 150-year-old company uses virtual reality"*
http://fortune.com/2015/08/25/mccarthy-construction-vr/

[52] **Atheer AiR** (accessed: 2016-03-09)
http://atheerair.com/

[53] **Strange A.** (2015) *"Magic Leap patent reveals what may be the next augmented reality system"*, in: *Mashable*
http://mashable.com/2015/07/23/magic-leap-patent/#ZbUk23bhWkqa

[54] **Mason W.** (2015) *"Oculus CV1 Positional Tracking 'Really Efficient,' Capable of Tracking Multiple Headsets"*, in: *UploadVR*
http://uploadvr.com/oculus-cv1-positional-camera-efficient/

[55] **Qualcomm Vuforia** (accessed: 2016-03-09)
https://www.qualcomm.com/products/vuforia/

[56] **HTC Vive** (accessed: 2016-03-09)
https://www.htcvive.com/

[57] **Meta** (accessed: 2016-04-27)
https://www.metavision.com/

[58] **Microsoft News Center** (2015) *"Microsoft redefines the laptop with Surface Book, ushers in new era of Windows 10 devices"*
http://news.microsoft.com/2015/10/06/microsoft-redefines-the-laptop-with-\
surface-book-ushers-in-new-era-of-windows-10-devices/

[59] **Morris C.** (2015) *"Is 2016 The Year of Virtual Reality?"*, in: *Fortune Magazine*
http://fortune.com/2015/12/04/2016-the-year-of-virtual-reality/

[60] **Hern A.** (2015) *"Will 2016 be the year virtual reality gaming takes off?"*, in: *The Guardian*
http://www.theguardian.com/technology/2015/dec/28/virtual-reality-gaming\
-takes-off-2016

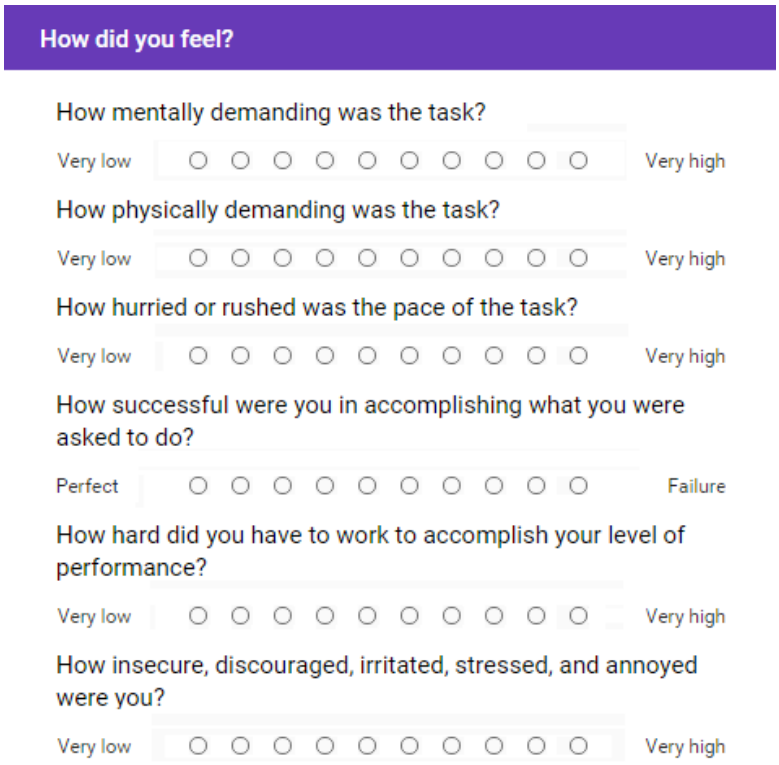
Appendix A Scenario Script

This is the final scenario script that was used during the user tests. It is based on the first scenario (*"Deliver Target Wheel"*) described in section 3.3.1. For more information on how scenario scripting works, please see section 5.3.2.

```
    TARGET_WHEEL RESET
    SPINNER RESET

Open top door
    SEQUENCER WAIT 1

    TOP_DOOR OPEN

Lower target wheel
    SEQUENCER WAIT 1
    TARGET_WHEEL_CRANE LOWER 6
    SPINNER ATTACH TARGET_WHEEL

Move manipulator to toolkit
    SEQUENCER WAIT 1

    // Fold
    MANIPULATOR !REACH_FOR RIGHT REST_RIGHT
    MANIPULATOR REACH_FOR LEFT REST_LEFT

    MANIPULATOR MOVE_TO TOOLKIT

Reach for screwdriver
    SEQUENCER WAIT 1

    MANIPULATOR REACH_FOR RIGHT SCREWDRIVER
    MANIPULATOR ATTACH RIGHT SCREWDRIVER
    MANIPULATOR GRIP RIGHT 2
    MANIPULATOR !MOVE 0 0.5 0
    MANIPULATOR REACH_FOR RIGHT REST_RIGHT

Move manipulator to target wheel
    SEQUENCER WAIT 1

    MANIPULATOR !ROTATE TORSO 90 4
    MANIPULATOR MOVE_TO TARGET_WHEEL_MOUNTING 4
    MANIPULATOR REACH_FOR RIGHT SCREWDRIVER_WAITING_POSITION

Fasten first bolt
    SEQUENCER WAIT 1

    MANIPULATOR REACH_FOR RIGHT SCREWDRIVER_TARGET_1
    MANIPULATOR ROTATE RIGHT HAND -720 3
    MANIPULATOR REACH_FOR RIGHT SCREWDRIVER_WAITING_POSITION

Spin target wheel
    SEQUENCER WAIT 1

    SPINNER SPIN 90 4

Fasten second bolt
    SEQUENCER WAIT 1

    MANIPULATOR REACH_FOR RIGHT SCREWDRIVER_TARGET_2
    MANIPULATOR ROTATE RIGHT HAND -720 3
    MANIPULATOR REACH_FOR RIGHT SCREWDRIVER_WAITING_POSITION

Spin target wheel
    SEQUENCER WAIT 1

    SPINNER SPIN 90 4

Fasten third bolt
    SEQUENCER WAIT 1

    MANIPULATOR REACH_FOR RIGHT SCREWDRIVER_TARGET_3
    MANIPULATOR ROTATE RIGHT HAND -720 3
    MANIPULATOR REACH_FOR RIGHT SCREWDRIVER_WAITING_POSITION

Spin target wheel
    SEQUENCER WAIT 1

    SPINNER SPIN 90 4

Fasten fourth bolt
    SEQUENCER WAIT 1

    MANIPULATOR REACH_FOR RIGHT SCREWDRIVER_TARGET_4
    MANIPULATOR ROTATE RIGHT HAND -720 3
    MANIPULATOR REACH_FOR RIGHT SCREWDRIVER_WAITING_POSITION

Move manipulator to toolkit
    SEQUENCER WAIT 1

    MANIPULATOR !MOVE_TO TOOLKIT 4
    MANIPULATOR ROTATE TORSO -90 4

Put screwdriver back
    SEQUENCER WAIT 1

    MANIPULATOR REACH_FOR RIGHT SCREWDRIVER_POSITION
    MANIPULATOR DEATTACH RIGHT
    MANIPULATOR GRIP RIGHT 0
    MANIPULATOR MOVE 0 0.5 0
    MANIPULATOR REACH_FOR RIGHT REST_RIGHT

Move manipulator away
    SEQUENCER WAIT 1

    MANIPULATOR MOVE_TO MANIPULATOR_RESTING_POSITION

Raise target wheel crane
    SEQUENCER WAIT 1

    TARGET_WHEEL_CRANE !RAISE 6
    SEQUENCER WAIT 3

Close top door
    SEQUENCER WAIT 1

    TOP_DOOR CLOSE
```

Appendix B Questionnaires

These are the questionnaires that were given to the users after the user tests. For more information, please see section 7.1.1 for how the questionnaires were made and section 7.2 for the results from the tests.



What did you think of this experience?

Innovative	○	○	○	○	○	○	○	Conservative
Pleasant	○	○	○	○	○	○	○	Unpleasant
Clear	○	○	○	○	○	○	○	Confusing
Unaesthetic	○	○	○	○	○	○	○	Aesthetic
Supporting	○	○	○	○	○	○	○	Obstructing
Easy to learn	○	○	○	○	○	○	○	Difficult to learn
Simple	○	○	○	○	○	○	○	Complex
Uncontrollable	○	○	○	○	○	○	○	Controllable
Useful	○	○	○	○	○	○	○	Useless
Playful	○	○	○	○	○	○	○	Professional

About you

- ○ Male
- ○ Female

How old are you?

- ○ -19
- ○ 20-29
- ○ 30-39
- ○ 40-49
- ○ 50-59
- ○ 60-

How familiar are you with gaming?

- ○ No previous experience
- ○ Some previous experience
- ○ A lot of previous experience

How familiar were you with the operations of the hot cell before this?

- ○ No previous knowledge
- ○ Some previous knowledge
- ○ A lot of previous knowledge

Appendix C Shaders

These are the shaders that were used to achieve the 3D window as described in section 6.2.3. They are written in Unity's proprietary shader language *ShaderLab* and HLSL, and are left here for future reference.

C.1 3D Window Shader

This is the shader that is applied to the viewport (i.e. the actual 3D window). It makes everything that uses the *Model Shader* and is not inside of or behind the 3D window invisible.

```
Shader "Window3DShader" {

    SubShader {

        Tags {
            "RenderType" = "Transparent"
            "Queue" = "Transparent"
        }

        LOD 200
        ColorMask 0
        ZWrite off

        Stencil {
            Ref 1
            Comp always
            Pass replace
        }

        CGPROGRAM

            #pragma surface surf Standard fullforwardshadows
            #pragma target 3.0

            sampler2D _MainTex;

            struct Input {
                float2 uv_MainTex;
            };

            void surf (Input IN, inout SurfaceOutputStandard o) {
            }

        ENDCG

    }

    FallBack "Diffuse"

}
```

C.2 Model Shader

This is the shader that is applied to the objects that should be restricted to being viewed through the 3D window. It makes everything that is not inside or behind objects that use the *3D Window Shader* invisible.

```
Shader "ModelShader" {

  Properties {
    _Color("Color", Color) = (1,1,1,1)
    _MainTex("Albedo (RGB)", 2D) = "white" {}
    _GhostAlpha ("Ghost alpha", Range(0,1)) = 0.05
    _FrontGhostAlpha ("Ghost alpha (Front)", Range(0,1)) = 0.05
    _Cutoff ("Alpha cutoff", Range(0,1)) = 0.5
  }

  SubShader {

    Tags {
      "Queue" = "AlphaTest"
      "IgnoreProjector" = "True"
      "RenderType" = "TransparentCutout"
    }

    // Render everything inside and behind the window

    Stencil {
      Ref 1
      Comp equal
      Pass keep
    }

    CGPROGRAM

      #pragma surface surf Lambert alphatest:_Cutoff
      #pragma target 3.0

      sampler2D _CameraDepthTexture;
      sampler2D _MainTex;
      fixed4 _Color;

      fixed4 _TopPoint;
      fixed4 _TopNormal;
      fixed4 _TopHorizontal;
      fixed4 _BottomPoint;

      fixed4 _FrontPoint;
      fixed4 _FrontNormal;
      fixed4 _FrontHorizontal;
      fixed4 _BackPoint;

      fixed4 _LeftPoint;
      fixed4 _LeftNormal;
      fixed4 _LeftHorizontal;
      fixed4 _RightPoint;

      float _Width;
      float _Height;
      float _Depth;

      int _Yggdrasil;

      // Return NaN if miss
      float distance(float3 worldPos, half3 planeNormal,
                     half3 planePoint, half3 planeHorizontal,
                     float maxHorizontal, float maxVertical) {
        float3 u = worldPos - _WorldSpaceCameraPos;
        float dot_p = dot(planeNormal.xyz, u);

        if (abs(dot_p) > 0.000001f) {
          float3 diff =
                 u * -dot(planeNormal.xyz,
                          _WorldSpaceCameraPos - planePoint.xyz)
                            / dot_p;
          float3 collisionPoint = _WorldSpaceCameraPos + diff;
          float horizontalDist = abs(dot(planeHorizontal,
                                         collisionPoint
                                           - planePoint));
          float verticalDist = abs(dot(cross(planeHorizontal,
                                             planeNormal),
                                       collisionPoint - planePoint));
          if (horizontalDist <= maxHorizontal
           && verticalDist <= maxVertical)
            return length(diff);
          else
            return sqrt(-1.0);
        } else {
          return sqrt(-1.0);
        }
      }

      struct Input {
        float2 uv_MainTex;
        float4 screenPos;
        float3 worldPos;
      };

      void surf(Input IN, inout SurfaceOutput o) {
        fixed4 color = tex2D(_MainTex, IN.uv_MainTex) * _Color;

        if (_Yggdrasil > 0) {
          float depth = length(IN.worldPos
                               - _WorldSpaceCameraPos);

          float frontDistance = distance(IN.worldPos,
                                         _FrontNormal.xyz,
                                         _FrontPoint.xyz,
                                         _FrontHorizontal.xyz,
                                         _Width / 2,
                                         _Height / 2);

          float backDistance = distance(IN.worldPos,
                                        -_FrontNormal.xyz,
                                        _BackPoint.xyz,
                                        _FrontHorizontal.xyz,
                                        _Width / 2,
                                        _Height / 2);

          float leftDistance = distance(IN.worldPos,
                                        _LeftNormal.xyz,
                                        _LeftPoint.xyz,
                                        _LeftHorizontal.xyz,
                                        _Depth / 2,
                                        _Height / 2);

          float rightDistance = distance(IN.worldPos,
                                         -_LeftNormal.xyz,
                                         _RightPoint.xyz,
                                         _LeftHorizontal.xyz,
                                         _Depth / 2,
                                         _Height / 2);

          float topDistance = distance(IN.worldPos,
                                       _TopNormal.xyz,
                                       _TopPoint.xyz,
                                       _TopHorizontal.xyz,
                                       _Width / 2,
                                       _Depth / 2);

          float bottomDistance = distance(IN.worldPos,
                                          -_TopNormal.xyz,
                                          _BottomPoint.xyz,
                                          _TopHorizontal.xyz,
                                          _Width / 2,
                                          _Depth / 2);

          float minDistance = min(min(min(frontDistance,
                                          backDistance),
                                      min(leftDistance,
                                          rightDistance)),
                                      min(topDistance,
                                          bottomDistance));

          float maxDistance = max(max(max(frontDistance,
                                          backDistance),
                                      max(leftDistance,
                                          rightDistance)),
                                      max(topDistance,
                                          bottomDistance));

          if (minDistance < depth
           && depth < maxDistance) {
            o.Alpha = 1;
          } else if (depth > maxDistance) {
            o.Alpha = 1;
          } else {
            o.Alpha = 0;
          }
        } else {
          o.Alpha = 1;
        }

        o.Albedo = color.rgb;
      }

    ENDCG
```

```
// Render everything in front of the window

Stencil {
  Ref 1
  Comp equal
  Pass keep
}

CGPROGRAM

  #pragma surface surf Lambert alpha
  #pragma target 3.0

  sampler2D _CameraDepthTexture;
  sampler2D _MainTex;
  fixed4 _Color;

  fixed4 _TopPoint;
  fixed4 _TopNormal;
  fixed4 _TopHorizontal;
  fixed4 _BottomPoint;

  fixed4 _FrontPoint;
  fixed4 _FrontNormal;
  fixed4 _FrontHorizontal;
  fixed4 _BackPoint;

  fixed4 _LeftPoint;
  fixed4 _LeftNormal;
  fixed4 _LeftHorizontal;
  fixed4 _RightPoint;

  float _Width;
  float _Height;
  float _Depth;

  float _GhostAlpha;
  float _FrontGhostAlpha;

  int _Yggdrasil;

  // Return NaN if miss
  float distance(float3 worldPos,
                 half3 planeNormal,
                 half3 planePoint,
                 half3 planeHorizontal,
                 float maxHorizontal,
                 float maxVertical) {
    float3 u = worldPos - _WorldSpaceCameraPos;
    float dot_p = dot(planeNormal.xyz, u);

    if (abs(dot_p) > 0.000001f) {
      float3 diff =
             u * -dot(planeNormal.xyz,
                      _WorldSpaceCameraPos -
                        planePoint.xyz) /
                          dot_p;
      float3 collisionPoint = _WorldSpaceCameraPos + diff;
      float horizontalDist = abs(dot(planeHorizontal,
                                     collisionPoint -
                                       planePoint));
      float verticalDist = abs(dot(cross(planeHorizontal,
                   planeNormal),
             collisionPoint -
               planePoint));
      if (horizontalDist <= maxHorizontal
        && verticalDist <= maxVertical)
        return length(diff);
      else
        return sqrt(-1.0);
    } else {
      return sqrt(-1.0);
    }
  }

  struct Input {
    float2 uv_MainTex;
    float4 screenPos;
    float3 worldPos;
  };

  void surf(Input IN, inout SurfaceOutput o) {
    fixed4 color = tex2D(_MainTex, IN.uv_MainTex) * _Color;

    if (_Yggdrasil > 0) {
      float depth = length(IN.worldPos -
                           _WorldSpaceCameraPos);

      float frontDistance = distance(IN.worldPos,
                                     _FrontNormal.xyz,
                                     _FrontPoint.xyz,
                                     _FrontHorizontal.xyz,
                                     _Width / 2,
                                     _Height / 2);

      float backDistance = distance(IN.worldPos,
                                    -_FrontNormal.xyz,
                                    _BackPoint.xyz,
                                    _FrontHorizontal.xyz,
                                    _Width / 2,
                                    _Height / 2);

      float leftDistance = distance(IN.worldPos,
                                    _LeftNormal.xyz,
                                    _LeftPoint.xyz,
                                    _LeftHorizontal.xyz,
                                    _Depth / 2,
                                    _Height / 2);

      float rightDistance = distance(IN.worldPos,
                                     -_LeftNormal.xyz,
                                     _RightPoint.xyz,
                                     _LeftHorizontal.xyz,
                                     _Depth / 2,
                                     _Height / 2);

      float topDistance = distance(IN.worldPos,
                                   _TopNormal.xyz,
                                   _TopPoint.xyz,
                                   _TopHorizontal.xyz,
                                   _Width / 2,
                                   _Depth / 2);

      float bottomDistance = distance(IN.worldPos,
                                      -_TopNormal.xyz,
                                      _BottomPoint.xyz,
                                      _TopHorizontal.xyz,
                                      _Width / 2,
                                      _Depth / 2);

      if (depth < min(min(min(frontDistance,
                              backDistance),
                          min(leftDistance,
                              rightDistance)),
                          min(topDistance,
                              bottomDistance))) {
        o.Alpha = _FrontGhostAlpha;
      }
    } else {
      o.Alpha = 1;
    }

    o.Albedo = color.rgb;
  }

ENDCG

Stencil {
  Ref 0
  Comp equal
  Pass keep
}

CGPROGRAM

  #pragma surface surf Lambert alpha
  #pragma target 3.0

  sampler2D _MainTex;
  fixed4 _Color;
  float _GhostAlpha;
  int _Yggdrasil;

  struct Input {
    float2 uv_MainTex;
  };

  void surf(Input IN, inout SurfaceOutput o) {
    fixed4 color = tex2D(_MainTex,
                         IN.uv_MainTex) *
                           _Color;
    if (_Yggdrasil > 0) {
      o.Alpha = _GhostAlpha;
    } else {
      o.Alpha = 1;
    }
    o.Albedo = color.rgb;
  }

ENDCG

  }

}
```

Appendix D The Companion Sphere Dialog

This is the scripted dialog which was later converted to a synthetized voice audio recording for the Companion Sphere.

Step 1
Welcome to the ESS Hot Cell Demo. We will now guide you through a basic tutorial on how to oversee and operate this magnificent facility. You will become an operator with exceptional skills in no time. Alright, let's start.

Right now, you are inside the control room. Here you will operate the Hot Cell chamber from a safe distance. You cannot enter the chamber itself, because the radiation inside is deadly dangerous. But don't worry, you are safe inside here.

Step 2
This control room, consists of three major parts: The screens in front of you, the 3D model under the screens, and the Falcon joystick next to the 3D model. We will now carefully explain the use of these parts. Let us start with the screens in front of you. Inside the Hot Cell, there are cameras. And these cameras show you what they see on these small screens. Take a small look on the screens for ten seconds.

Right. As you may see, the big screen shows nothing. Let us put something on it. Touch one of the small screens with your hand.

Great! Try again with another screen.

Outstanding. Whenever you want to put one small screen on the big screen, simply touch the small screen.

Step 3
Alright, now let's take a look at the 3D model. The 3D model is a simple graphical representation of the Hot Cell. It is something like a 3D map, where certain objects are moving, and you can even touch some of them. Take a look at it, for ten seconds.

Right. You can see the positions of the cameras in the 3D model. To display one of the cameras on the big screen. Touch one of the camera icons with your fingers. If nothing happens, put your hand in front of your face and spread your fingers until your hand shines blue.

Very good. Try it again, choose another camera icon.

Splendid. Whenever you wish to see the image of one specific camera, simply touch the camera icon.

Step 4
Finally, let's take a look at the Falcon joystick. With the joystick, you can manipulate different objects inside the Hot Cell: the cranes, the manipulator, and even the cameras. Above the joystick is the information panel. This panel shows what the joystick is currently controlling. Let us begin with the camera. Grip the joystick, and move around with it, however you want.

As you can see, the camera is following the movement of the joystick. The joystick has four buttons: left button, right button, front button, and a middle button.

To zoom out with the camera, press the left button. And to zoom in, press the right button.

Now, move the camera all the way to the left.

As you can see, the camera is constrained to the range of the joystick. If you wish to move the camera, further to the left or any direction, choose the direction with the joystick, and press the front button. Try it out.

Good. Now you know how to control a camera.

Step 5
Now let us try to control a crane. If you look at Crane 1, you will see a yellow arrow indicator hovering above the crane. This indicator shows in which direction the crane will move when you press the front button. Move the joystick to the right, and press the front button.

Now move the joystick inwards, and press the front button.

Now move the joystick down, and press the front button.

The crane will move in the same direction as the joystick once you press the front button. The indicator confirms the direction. You can control the second crane and the manipulator in this same manner.

Step 6
Alright. You have been properly introduced to the basics. Now you are ready for your first task.
A new target wheel needs to be put inside the Hot Cell, which will be used in producing neutrons.
This is the Target Wheel. You will deploy the Target Wheel, and put it on place with the manipulator.
Press the space bar to initialize the mission.

Don't worry, the model only focuses on the Target Wheel for you.

Look at the green flashing command line in front of you. Whenever the command line flashes green it is time for you to press the space bar. Press the space bar to execute the command.

You are doing fine so far. For safety precautions, we need to direct one camera up at the ceiling door to see if everything is fine. Choose the camera icon opposite to the ceiling, door.

Oops, wrong one.

Perfect. Now aim the camera up at the ceiling door. If everything looks good continue with the mission.

Step 7
Right. As you can see before you, the Target Wheel is now in place. But it still has to be properly fastened. We're going to need the manipulator for this. This is the manipulator. Look at the command line and execute the command.

Now, move the manipulator to the red point. One more thing. If you wish to adjust or control a camera, press the middle button on the joystick. Once you are finished with the camera, press the middle button again to return to previous control.

Spectacular. You are doing well. Continue executing the next command.

Ok, now it is time for you to properly fasten the Target Wheel with this tool.

You will now control the manipulator hand with the joystick. You will direct the hand to the red point, and fasten the bolt. Good luck!

You have successfully fastened the bolt. Now, move the manipulator hand back to the red point.
The Target Wheel will spin and present you the next bolt.

Very good, keep repeating the same process until all bolts are tight.

Step 8
Superb! You have properly fastened the Target Wheel. Continue executing the commands. Don't forget to look at the cameras to see if everything is alright.

Congratulations on completing your first task with success! You have been a good student. Feel free to explore and play with the control room furthermore.
Goodbye!