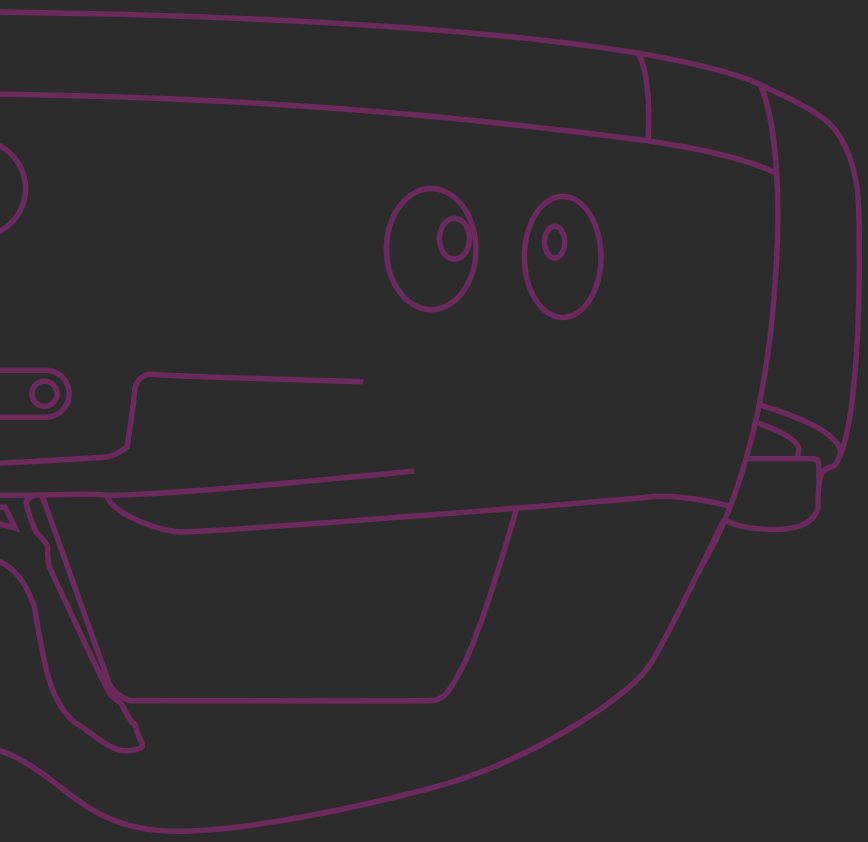


Mixed Reality Application and Integration with HoloLens© in a Manufacturing Environment

Joran van der Sluis

Bachelor Thesis
Industrial Design
March 2017

UNIVERSITY OF TWENTE.

Benchmark electronics

Colophon

Mixed Reality Application and Integration with HoloLens©
in a Manufacturing Environment

Author
Joran van der Sluis
s1368753

Bachelor Assignment Industrial Design
March 2017

Final exam
April 26, 2017

Bachelor coordinator
Dr. Ir. A. P. van den Beukel

Project coordinator
Dr. Ir. R. E. Wendrich

Second examinator
Dr. Ir. T. Vaneker

Coordinator Benchmark Electronics B.V.
C. Suurmeijer

Industrial Design
Faculty of Engineering Technology
University of Twente
Drienerlolaan 5
7522 NB Enschede

Benchmark Electronics B.V.
Lelyweg 10
7602 AE Almelo

This report was written in the context of the Bachelor Assignment of the study Industrial Design at University of Twente.

Preface

This report is written in the context of the Bachelor Assignment of Industrial Design at the University of Twente. This assignment came forth from a meeting at Benchmark. The initial idea was to create a technical project involving an application for manufacturing. During the course of the study, most information on manufacturing and production remains theoretical. Gaining work experience in the industry as an industrial designer could be a great addition.

During our meeting the subject of the Microsoft HoloLens© was discussed. The HoloLens© is a wearable computer that allows you to see holograms in your surroundings. As I have always been interested in emerging technologies, I was immediately interested. The decision was made to create the assignment around application and integration of this device in the manufacturing environment.

Benchmark gave me the experience to work within a large corporation. To some extend it has shown me how a large corporation operates and gave me some scope on the pros and cons of such an enterprise.

The project created a challenge, as it differs from other university projects or courses. During the project, I have acquired new skills and insights. It has been interesting to work with the HoloLens©. Thanks to this project I am now able to use the Unity game engine and gained experience in C# programming. The most important thing was learning to see technology as a tool instead of a solution.

Table of contents

1. Abstract

This report comprises possible solutions for problems observed and analysed in the manufacturing process of Benchmark Electronics Almelo. The focus is on application and integration of the Microsoft HoloLens© in the current and future manufacturing process. Two focus points are discussed, namely improvement of Work Process Instructions and data gathering for process improvements.

On a low-, mid-, and high-tech level, elaborations are made on how augmented holograms can be integrated into the process. This results in descriptions of workflow for several operator functions. Especially guidance during assembly and guidance during quality inspection seem to be promising directions for the use of holograms within the scope of the researched subjects.

Finally a proposal of a HoloLens© application for holographic guidance during assembly is described. Conclusions are drawn about the use of the HoloLens© in the manufacturing environment.

2. Introduction

We live in a world that is getting more and more digital. Interaction takes place mostly through digital interfaces on smartphones or computers. Our digital interaction has already evolved from a command line interface to a graphical user interface. Instead of telling computers exactly what to do, we have created digital metaphors in the form of pointers and icons. The next step will be the natural interface, where interaction with the digital world is intuitive and resembles the way we interact with the real world. Mixing reality and virtuality is a hot topic and more and more companies are emerging with new technologies allowing users to interact with the digital world in an intuitive way.

Benchmark Electronics Almelo (to be named Benchmark hereafter) is a company interested in emerging technologies that could benefit the workflow. Every quarter, Benchmark invests into a technology related project that could improve production. As some graduation projects in the field of augmented reality have been executed within Benchmark over the years, a certain experience had been acquired on the possibilities of augmented reality. When Microsoft released the HoloLens© (HL) near the end of 2016, this technology was chosen as project to research and investigate further. The HL is a wearable computer that allows users to emerge in augmented reality, where interaction is possible between the real and virtual world. Using holograms and natural interaction, one can interact with virtual objects in his real surroundings. As this is a relatively new topic, the question has arisen how the interaction could look within the scope of a company. Who will be interacting with augmented holograms and how will this look? How does this go about using a device that is present?

This Bachelor Assignment examines the technology of augmented reality using Microsoft HL as well as some processes within Benchmark to see if and how the two can match each other. The report starts off with additional context about the company and its products. Several tours through the production areas have given insight in the processes and workflow of the company. Interviews and talks have brought up some of the present problems Benchmark is facing. While keeping the possibilities of augmented reality using HL in mind, ideas and concepts were created. The HL is incorporated in each of the concepts in a different way. Eventually a proposal for a demo application to showcase on the HL is created. Using Unity and Visual Studio, the proposal was made into a holographic application for guidance during assembly.

3. Context

This chapter elaborates on the company Benchmark and the present Manufacturing Areas. It offers context and allows for better understanding of the later parts of the report.

3.1 Benchmark

Benchmark develops, produces, and assembles products containing electronic components for a business market. Worldwide 18 enterprises are present. Almelo is one of the four sites that houses engineering besides manufacturing, allowing for involvement in the whole process. A total of about 400 employees are working, divided over Engineering and Manufacturing. Walls and doors prevent the two from interacting with each other directly.

3.2 Products

Benchmark manufactures a lot of products that include electronics. Specifically, printed circuit boards (PCBs) are one of the main concerns. Using PCBs, products can be made on a small scale. PCBs are either shipped directly to clients or used in the creation of an assembly internally. PCBs are susceptible to defects when handled uncarefully, and due to the size, it is hard to determine whether a PCB is broken without testing it electrically. Manufacturing products with electronic components is complex, as there are lots of aspects that can result into defective products or products that will malfunction in the future. Benchmark wants to be a reliable partner to its clients, thus strives to prevent defects in outgoing products. Strict work instructions are used and all products are checked carefully on quality. Before shipping, every product is tested and evaluated.

3.3 Engineering Area

The Engineering Area is situated in a large room, where different sub departments are grouped together and surrounded by walls. Groups are either based on client or on function, such as design engineering or software. Engineering controls Manufacturing. Some of the main tasks of Engineering consist of:

- Creation of CAD-models
- Purchase of product parts
- Creation of Work Process Instructions (WPI)

Several different manufacturing areas are present in the building. Here, the different areas are described. Figure 1 also shows a visual representation of the different sorts of areas.

3.4 Clean Room

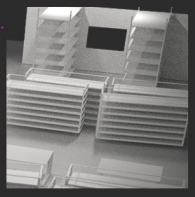
The Clean Room is used for assembly, quality inspection and testing of products from a specific product line. Most product parts come in from the warehouse, a restricted area for storage of larger and more expensive parts. Before entering the area, the parts are cleaned using pressurized air and alcoholic cloth. Once cleaned, the parts are handed to a Water Spider operator, who prepares orders of assembly kits. The parts needed for assembly of a product are combined with necessary labels, Kanban ('tech-consumables', small often used parts, like screws), and tools. All is numbered according to the assembly order. Carts are used for moving the orders to the assembly operators, who assemble products either on a table, or when a table is too small, on a large cart. Folders containing paper WPI are available that show how to assemble each product step by step. Finished products go through quality inspections, where an operator checks the product for visual and mechanical errors. After passing, a test engineer tests the product for defects in functionality by connecting it to a testing machine. If the product passes, it will be stored in stock, waiting to be sent out to the client or to be sent back as a prefab part into another assembly. In the Clean Room rules apply to clothing and air condition to prevent contaminations on products.

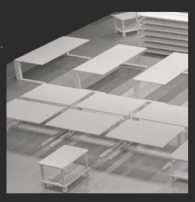
3.5 PCB Area

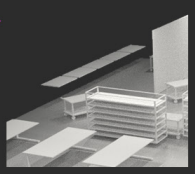
The PCB line is set in a long hallway and comprises a linear approach to the manufacturing process (see Figure 1). It starts on one side with loose components that are placed into machines. These machines create circuit boards. Operators control the mostly automated processes of the machines by switching them on or off and supervising. After creation, the PCBs are tested on functionality in special machines. If needed, other parts can be added manually further down the line. The machines creating the PCBs are expensive and high tech that are worked all day. A finished PCB is placed on a cart to be transported to an assembly area. As the PCBs are susceptible to damage, they are protected using special bags and trays.

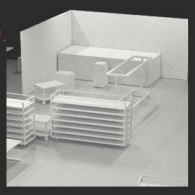
3.6 Box Build

Another assembly area, called Box Build, produces series of smaller products. Assembly and testing is done in a U-shape, where products go around a cabinet containing the parts needed. It is comparable to the workflow in the Clean Room, but on a smaller scale and following a linear path.

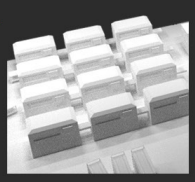
A. 
A water-spider operator collects the cleaned parts and kanban from the cabinets, and puts them onto a cart.

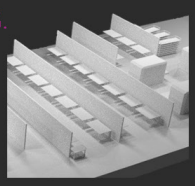
B. 
Assembly is performed on large and small tables throughout the area.

C. 
The far out corner is used by quality inspection. Rows of carts that need to be inspected are queued.

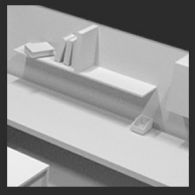
D. 
Testing is performed on large testing stations, where a product is connected via a set of cables.

E. 
A large machine cabinet provides roles of electronics that are put into the machines.

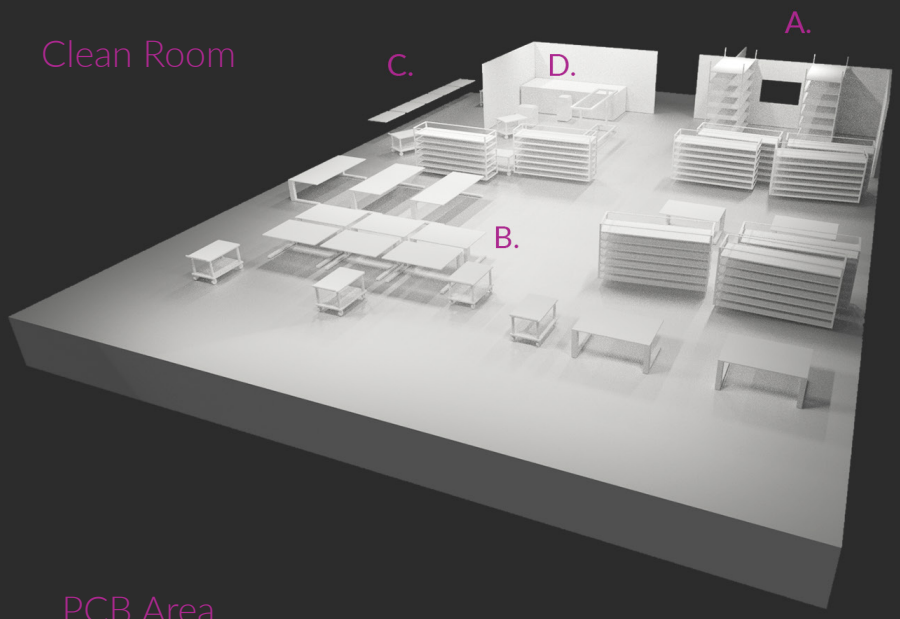
F. 
Large automated machines create PCBs.

G. 
Some of the PCBs are created or tested manually. Large sets of tables provide space.

H. 
Most of the parts that go into the assembly are present in the cabinet.

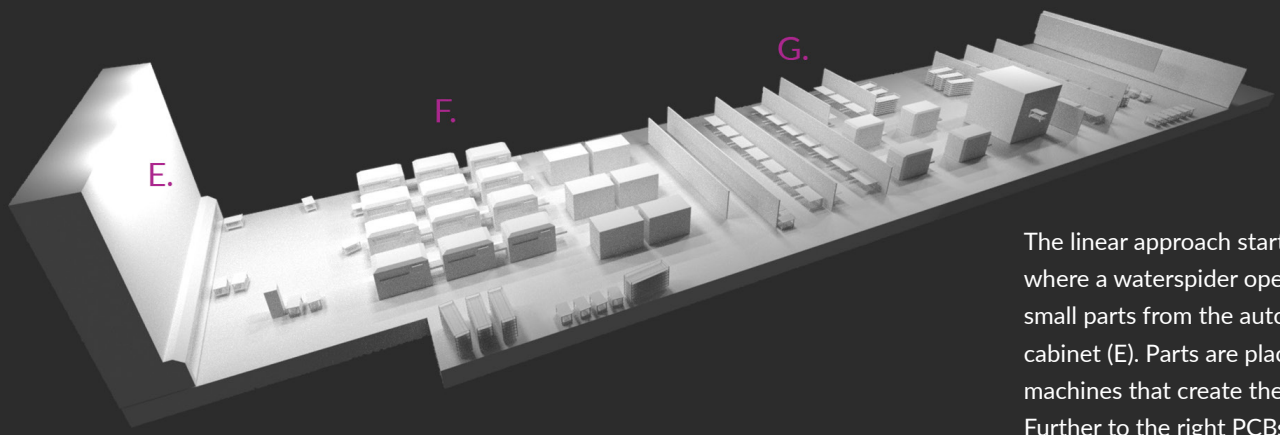
I. 
Products are created on the U-shaped table.

Clean Room



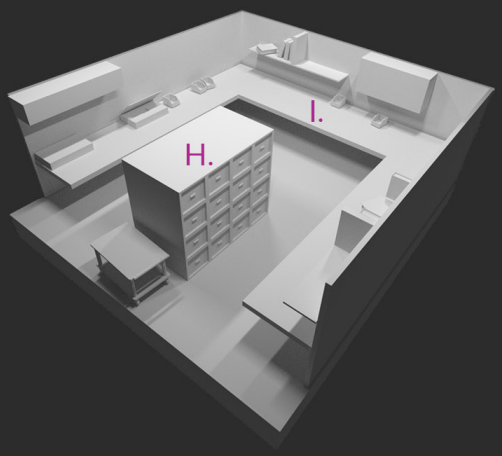
Product parts get in through the cleaning area, behind (A). From there the products go to the Water Spider operator, who prepares orders for assembly operators. He combines the large parts with small kanban, such as screws, and puts it all on a cart. Assembly operators take the cart to their workplace (B). Depending on the size of the product, either regular tables or movable table-carts are used. Finished assemblies are placed in line to be inspected on quality (C) and finally, before leaving the area, products are tested (D).

PCB Area

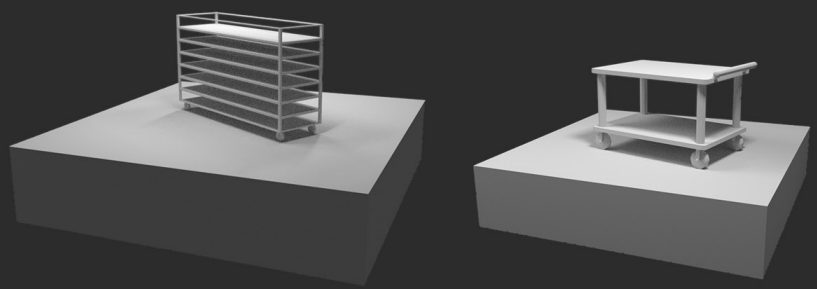


The linear approach starts on the left, where a waterspider operator collects small parts from the automated cabinet (E). Parts are placed into machines that create the PCBs (F). Further to the right PCBs are tested or manually assembled (G).

Box Build



Starting on the left, the creation of a product follows the U-shape (I) and a finished product, assembled and tested, can be taken somewhere else. The operators work around a cabinet (H) that contains most of the parts for that product.



The smaller and larger carts are used throughout all manufacturing areas. There is no standardized cart size, but most carts resemble one of the two.

Figure 1. Depictions of some of the important manufacturing areas at Benchmark Electronics Almelo.

4. Problem definition

The mandatory quality standard is not always met. Benchmark wants to reduce product loss rates. Several potential causes for product losses are described here.

4.1 ESD

Electrostatic Discharge (ESD) is one of the major problems when working with electrical parts and PCBs. ESD is current that builds up in persons or things. It can jump to anything close by once the voltage is high enough. People can only perceive ESD from about 3000V, hearing the sound of a spark. However, a discharge of 5V and up can already be devastating for an electronic product. Precautions have already been taken in the form of ESD-safe clothing or transportation bags, but these measurements do not completely prevent damage from ESD.

4.2 Filth

As strict agreements are made with clients about the number of dust or filth particles per surface area, products can be rejected for being too filthy.

4.3 WPI outdated

Products being manufactured get updated by Engineering over time and therefore a new version of the Work Process Instruction (WPI) is needed that incorporates the updates made to the product. The new instructions should replace the old ones in the manufacturing areas. This does not always happen, resulting in co-existing instruction versions. Because of this, products sometimes are based on the wrong instructions and therefore do not contain the update. This product relates to both assembly operators and quality inspection operators.

4.4 WPI unclear

WPI can be unclear to the operators. Lack of understanding or the quality of the instructions may be reasons for this unclarity. The instructions are printed on paper, sometimes only in black and white, while quality inspection needs colored prints to properly compare the product to the printed example.

4.5 Operators not following WPI

Some defects in faulty products can be caused by incorrect execution of assembly steps due to the operator disregarding the instructions. Possibly the operator does not understand the instructions and moves on to trial and error. Some of the operators choose to apply another order of steps to the assembly or inspection process, as they see more efficient ways of fulfilling their tasks. This problem is known to exist among assembly operators in Clean Room and Box Build.

4.6 PFS

Not all occurring errors during production are reported. Assembly operators have to fill in a list with questions after assembling a product. This list needs to be digitized into PFS, the system used to analyse the production processes and keep up statuses of products. With this list, it could be prevented that a product that is already defective will continue along the production process to eventually be rejected. A large part of the created assemblies will function as a prefab part in another assembly. If an assembly is already showing a certain defect, it can be rejected before being used in another product, that in turn will not pass quality inspection. However, assembly operators often do not fill in the question list or digitize the list.

4.7 Training/guiding

Training or guiding of new operators does not always happen. As new operators make a relatively larger number of mistakes, they should be guided through their functions by a more experienced operator. In practice, however, new operators are set to work straightaway, only to be checked on after a few hours.

5. Possible solutions

In the previous chapters, the environment and context were described, followed by an explanation of the present problems and challenges. In this chapter, some of the problems are looked at in more detail, and possibilities for solving the problems will be discussed.

5.1 Observations

During tours and interviews with employees, views from both Engineering and Manufacturing were heard. Engineering seemed to have a caring feeling for the products, as they are the ones that design them. They seemed to blame Manufacturing in some way for not putting in enough effort to prevent defects in products.

The operators on the other hand, seemed to care less about the created products, perhaps due to a larger emotional distance, since they work briefly on a product and see a lot of the same products. Operators were unhappy about the structure around WPI they were receiving, originating at Engineering.

Solving some of the problems surrounding the WPI could improve both workflow and happiness on the work floor and maybe bring the departments closer together.

5.2 Trends

Improving processes is hard for Benchmark as not all statuses of products and processes are documented well. Systems are present for keeping and interpreting data, but the lack of this data prevents them from doing so (Figure 2). This results in the impossibility of focusing on specific parts of the process where a relatively large number of defects originates. Looking at ways to improve the capturing of data could allow Benchmark to improve processes.

5.3 Tools

A possibility for solving some of the challenges present lies in the technology of mixed reality, where virtuality and reality are blended for an observer.
Milgram and Kishino (1994) describe reality and virtuality in terms of the properties objects. They state:
"Real objects are any objects that have an actual objective existence. Virtual objects are objects that exist in essence or effect, but not formally or actually" (p.1324).
Real objects are useful, as we can sense and use them. However, we have little control over them. Virtual objects on the other hand, we can control easily, but they are not as useful as real objects, as we cannot really sense them. They are a description of the properties of an object, that are not actually present in reality. Take for example a paprika.

A real paprika can be eaten or touched, but made into a blue paprika, as the initial properties of the paprika cannot be changed. It is more useful, as we can eat it and get energy from it. A virtual paprika can be given the color property blue, but we cannot eat it, as it does not actually exist. There is a high amount of control over the properties of the paprika.

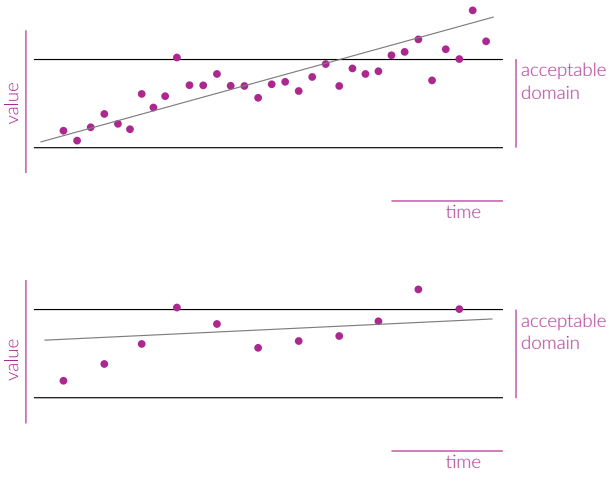


Figure 2. Example of how measured values in certain situations could give insight into a flaw in the manufacturing system somewhere. Above, a trend of values moving outside the acceptable domain is recognisable, while on the bottom, where only 1 in 3 values is shown, a trend is harder to recognise.

In mixed reality, both control and usefulness are present to some extent. Properties of virtuality are traded in for real properties. For example, a virtual paprika can be visible through a mixed reality medium, allowing someone to observe the paprika. It can be controlled in size and color, and it can be sensed. This arises possibilities, such as teaching someone about what color and size paprikas he should or should not eat by showing various paprikas, without having to go through the trouble of finding various shaped and colored paprikas.
In Appendix A, a research paper is appended on mixed reality and the various current concepts of it.

5.4 HoloLens©

The Microsoft HL is the first self-contained holographic computer. The device allows you to experience augmented reality through holograms you can interact with in the world around you (Microsoft, 2016a). The device is equipped with sensors that scan the real surrounding of the wearer (Microsoft, 2016c). In relation to the real environment, three-dimensional or flat holograms can be placed. Once a hologram is placed in the environment, the viewer can walk around it.
Appendix B describes the HL in detail. The figure on the next page summarizes the details of appendix B, based on information from Kipman (2016), Microsoft (2016d), and Microsoft (2016e) (see Figure 4).

Some of the already existing applications built around the HL, showcase the possibilities of using the device. For example, NASA has used satellite data to create an augmented environment of the surface of Mars (NASA Jet Propulsion Lab, 2016). Someone wearing the device can 'walk' over the surface of Mars. For NASA, this has opened new ways of interacting with and researching data.

The HL is not the first apparatus that allows users to view holograms or augmented reality, however the device is the first wearable augmented reality computer produced on large scale that places user experience first.

The HL seems a promising product with lots of possibilities. Possible solutions will be created for using the HL in the existing and future environment of Benchmark, while keeping in mind the current workflow and uses that apply.

5.5 Focus

As mentioned in the problem definition, a variety of problems has to do with the current way of instructing operators. In the next parts, possible technological solutions will be looked into that improve WPI. New solutions for improvements in work process instruction could also help in building up a system that allows for improvements through data analysis. On different levels of technology, solutions for integrating the HL will be looked into. The image below shows a graphical depiction of the focus area of the assignment (see Figure 3). In the next chapter, Ideation, preliminary visuals have been created that focus on the use of HL and augmented reality. From these ideas, 3 concepts have been created that focus on the improvement of WPI.

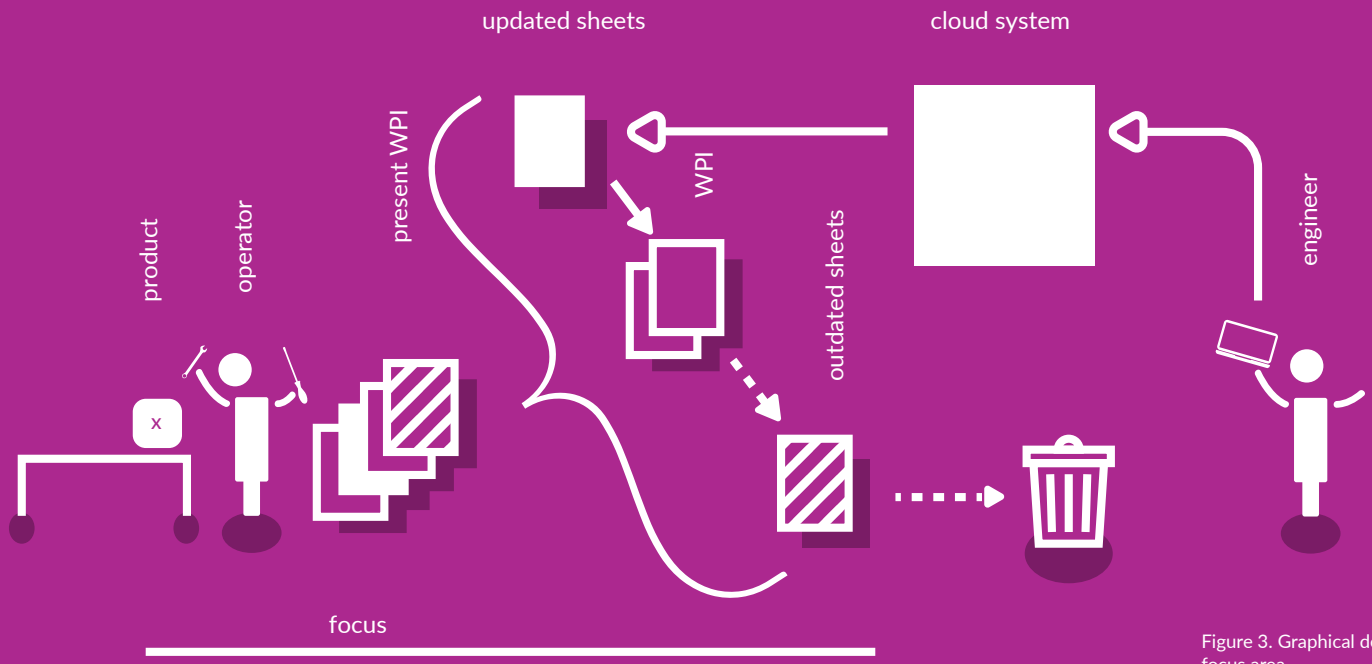
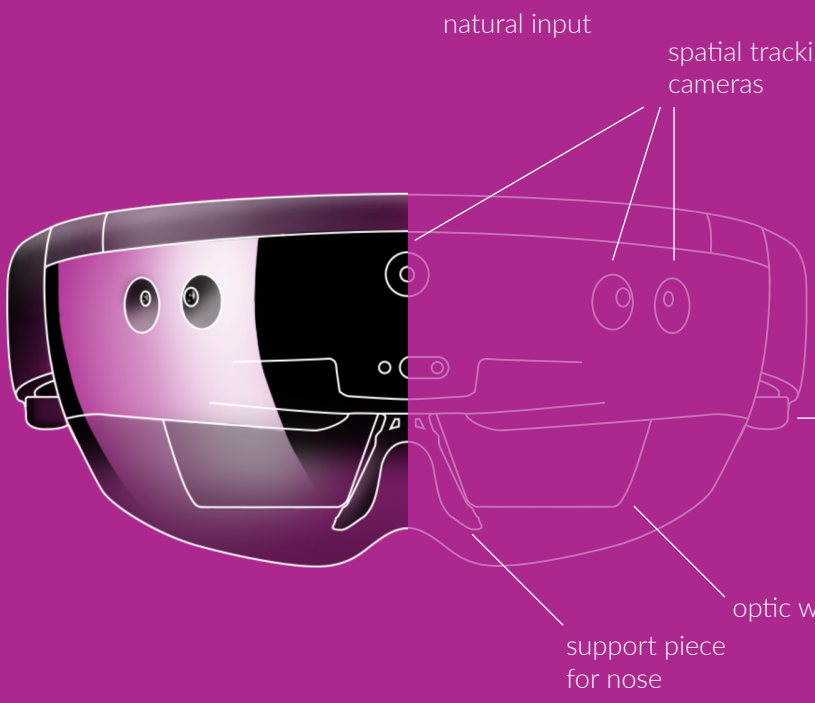


Figure 3. Graphical depiction of focus area

The device is able to recognise several gestures the user can make with his hand. Due to the fact that it is an optical see-through HMD, it is able to use gazing as an input method. Furthermore, the device can be voice controlled, through implementation of several microphones.

Spatial tracking allows for recognition of the environment. The device captures and creates a 3D model of the surrounding and uses this to calculate how the holograms should look to fit in the environment.



The device is a wearable Windows computer packed with several sensors and processors to understand environment and run holographic applications.

The device is worn as a pair of goggles. A band tightly strapped around the head of the user and a nose support piece ensure the device stays fixed on the head of the user.

Two optic wave lenses placed in front of the eyes of the user display stereoscopic holograms. Due to the intensity of the light emitted, the holograms blend in with the surrounding.

Figure 4. depiction of the front view of the HL with some explanations

6. Ideation

Certain steps are holographically displayed on a specific location of the product. For example, checking if 6 screws are present, is done by showing a 3D object of a screw, together with an object that indicates where to look (A1). When an operator gazes at that object (A2), the more specific locations are shown by adding a holographic visual to it, a bounding box (A3). Once an operator has checked a feature, a popup comes up, immediately asking if that step went well (A4). Instead of having to write it down, the operator can gaze at the right answer or just speak to the HL (see Figure 5).

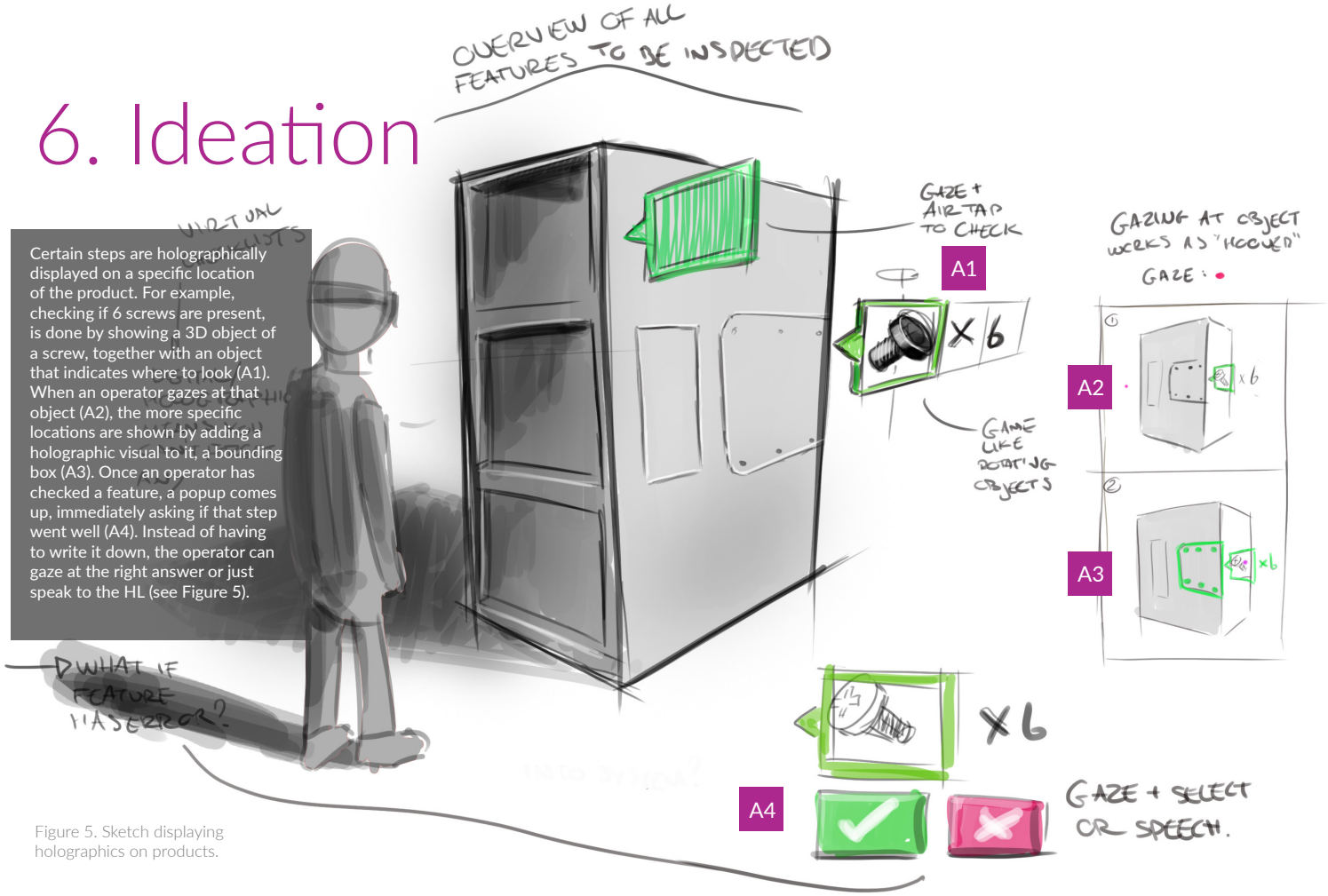
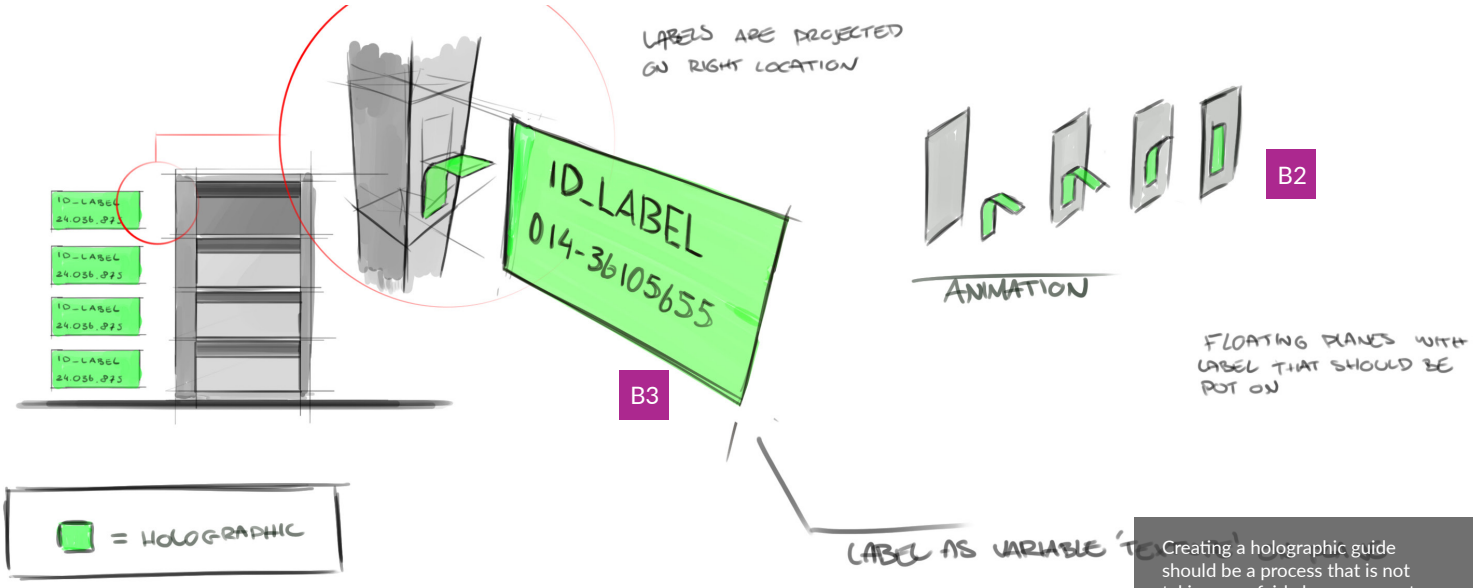


Figure 5. Sketch displaying holographics on products.



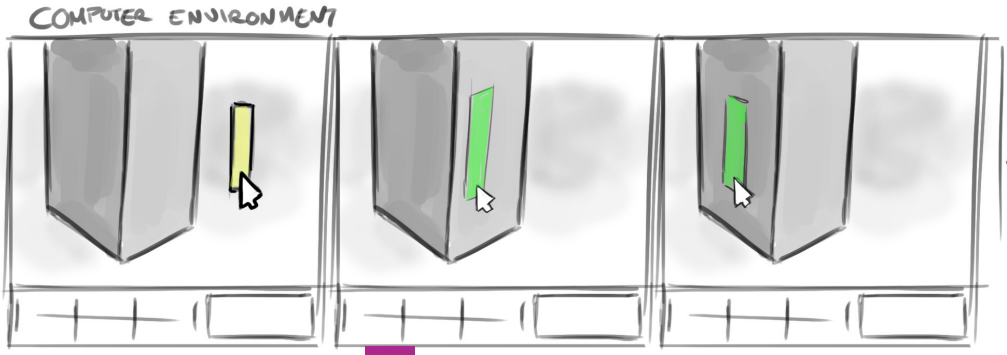
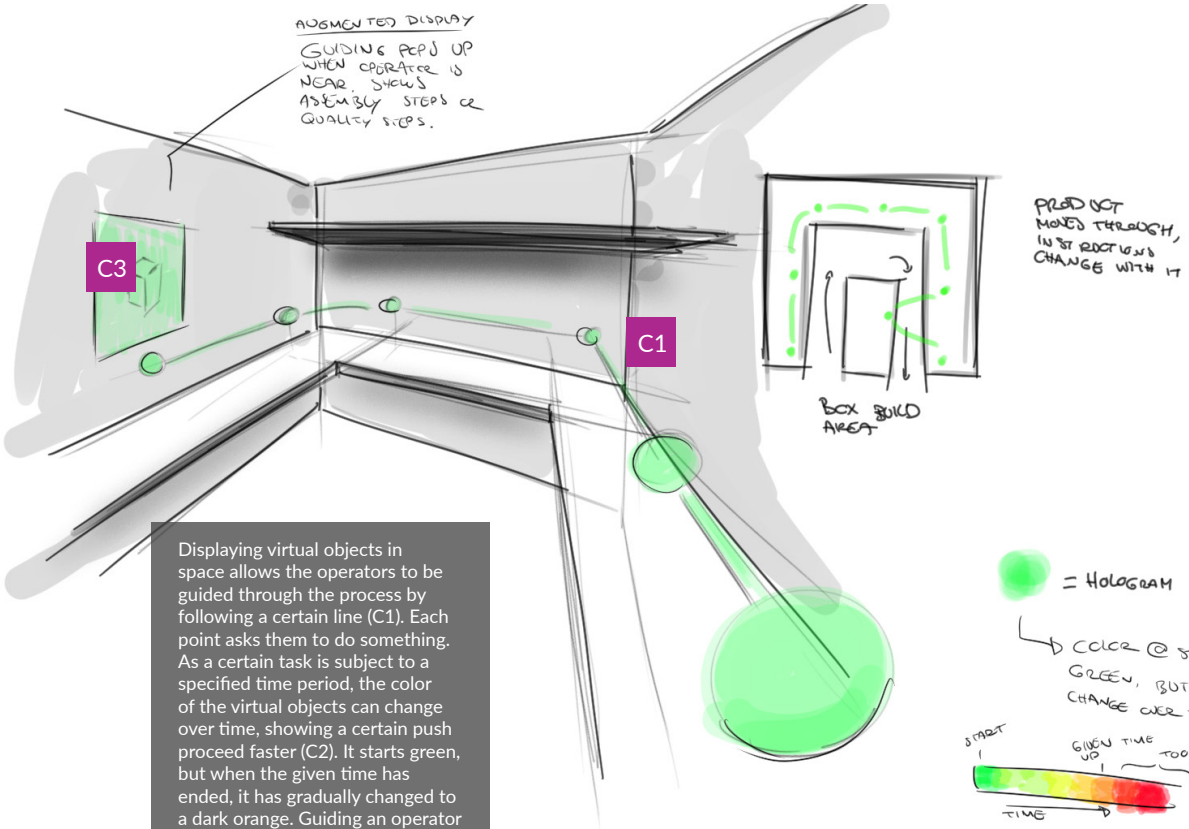


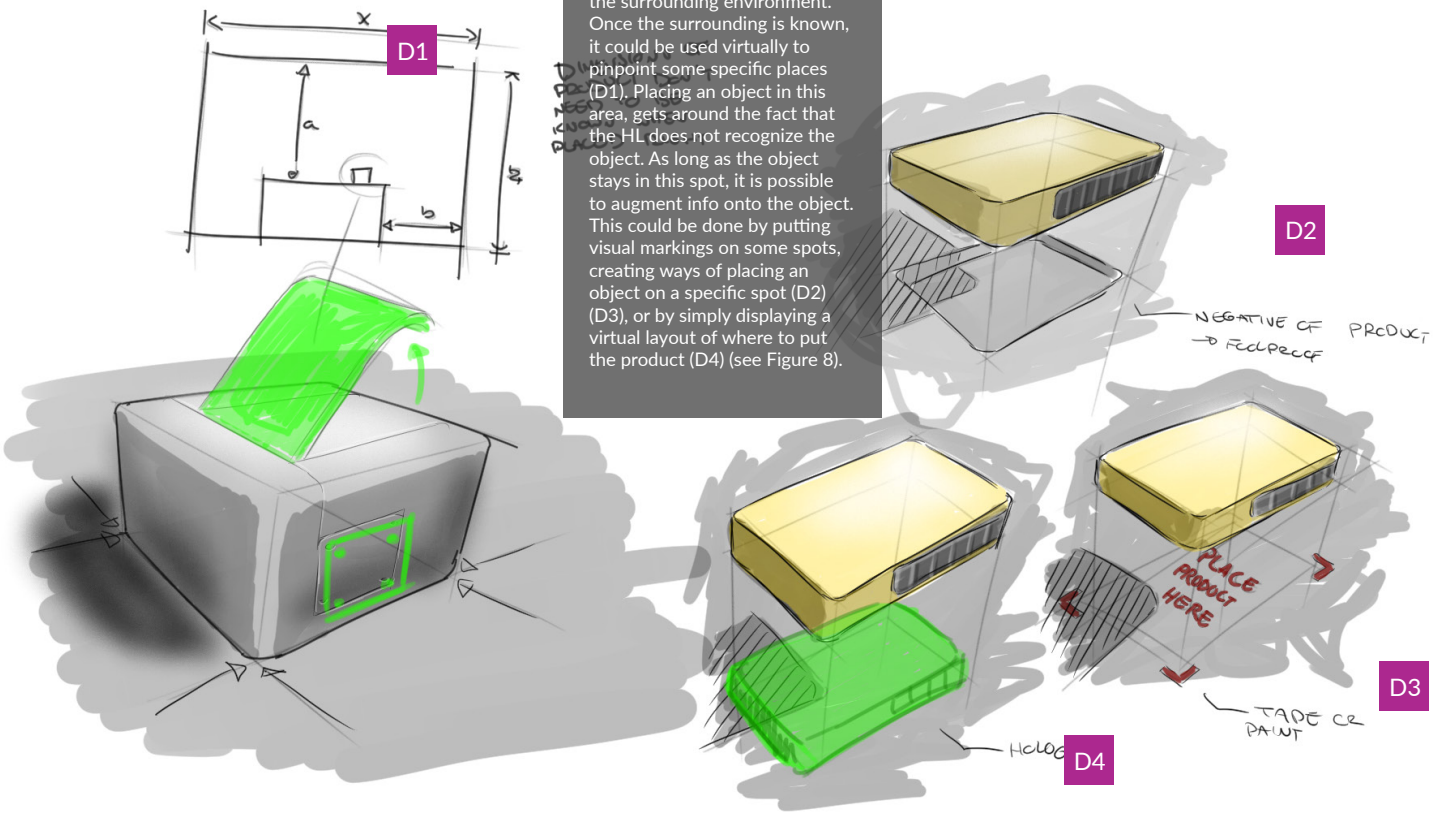
Figure 6. Sketch displaying ideas on creation of holographic guides.

Creating a holographic guide should be a process that is not taking up a fairly larger amount of time creating a written one does. Therefore, a library with some standardized animations should be built. In a holographic guiding building-application on a computer, one drags and drops features to a certain point on the 3D model (B1). The program calculates the perpendiculars and parallels to that part of the object and uses this in the guide (B2). Some content, like labels, are used as variables, that change depending on the serial number of the product the operator is working on (B3) (see Figure 6).



Displaying virtual objects in space allows the operators to be guided through the process by following a certain line (C1). Each point asks them to do something. As a certain task is subject to a specified time period, the color of the virtual objects can change over time, showing a certain push proceed faster (C2). It starts green, but when the given time has ended, it has gradually changed to a dark orange. Guiding an operator can be done through floating displays, showing instructions, either as text or as 3D images (C3) (see Figure 7).

Figure 7. Sketch displaying ideas on holographic guidance through a room.



HL does not recognize specific objects, but does recognize the surrounding environment. Once the surrounding is known, it could be used virtually to pinpoint some specific places (D1). Placing an object in this area, gets around the fact that the HL does not recognize the object. As long as the object stays in this spot, it is possible to augment info onto the object. This could be done by putting visual markings on some spots, creating ways of placing an object on a specific spot (D2) (D3), or by simply displaying a virtual layout of where to put the product (D4) (see Figure 8).

Figure 8. Sketch displaying ideas on combining virtual cues with real objects.

The way inspecting and building products is structured now, allows for multiple errors or defects. Due to the amount of steps and repetition, the chance on an error is increased. Restructuring the way information is gathered and input can result in skipping a few steps and also lowering the chance of an error. For example, it might not be necessary to first read the task on one paper sheet, than fill in another paper sheet with the results, to finally digitise that result sheet manually. When for example using a 'digital paper sheet', reading and filling in can be done from the same place, without the need to again digitise it (see Figure 9).

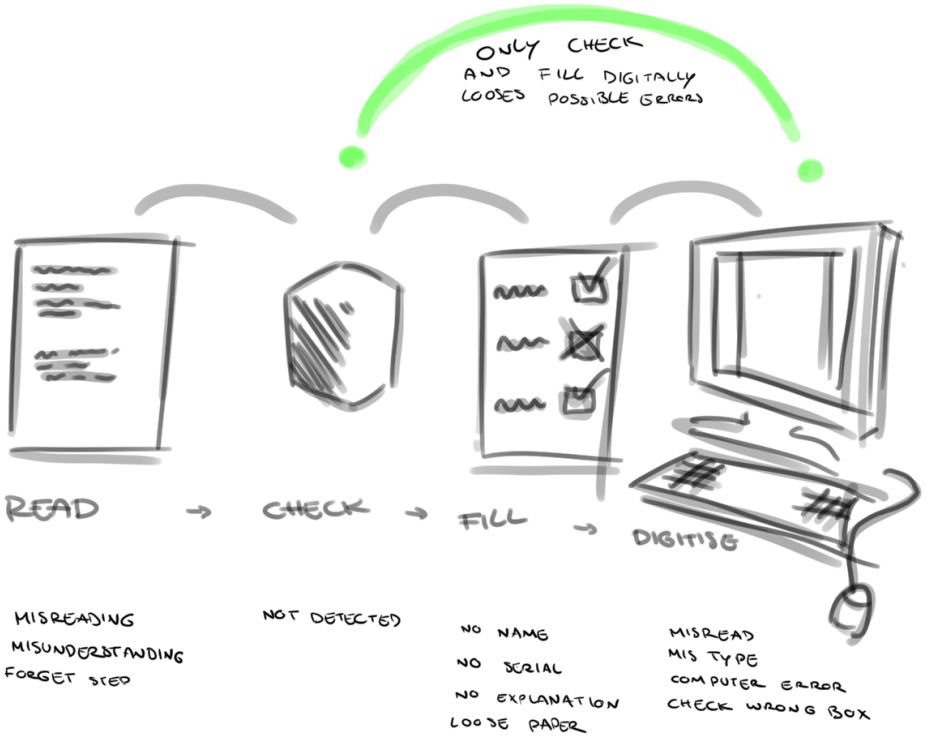


Figure 9. Sketch displaying ideas on how system interaction can be improved.

Using holographic models of products being built at Benchmark, a better insight and feeling for the product can be created. On one hand, an operator can learn how the product works and is assembled beforehand. This might lower defects, as he has seen his job holographically already. On the other hand, viewing or reviewing models can be used during creation of a product in the engineering side of the building. An engineer can already see the virtual product in real size (E1) and even collaborate on it with someone else (E2), without having to build the actual product. It can create a better sense of understanding of space, as the engineers can see the product in their actual environment (see Figure 10).

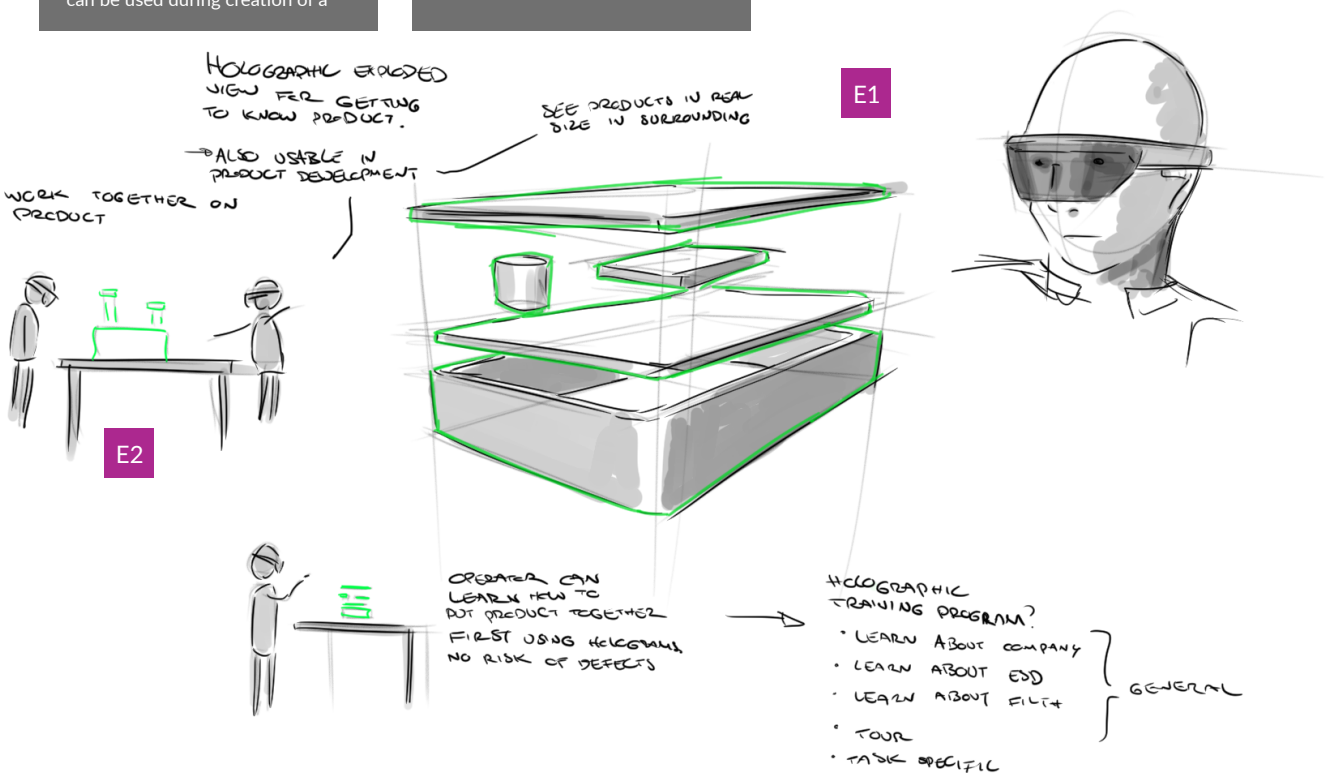


Figure 10. Sketch displaying ideas on (re)viewing holographic models.

Using holograms, it is possible to guide someone through the building very easily. Displaying all sorts of visual cues show someone how he should walk. This can be done in multiple ways, showing some direction arrows (F1), pushing one in the right direction, showing a virtual wire to follow (F2), or close off other routes virtually (F3). Using this way of guiding, it is possible to get products or goods from one place to the other fairly easy. It can also be used to show someone new around (see Figure 11).

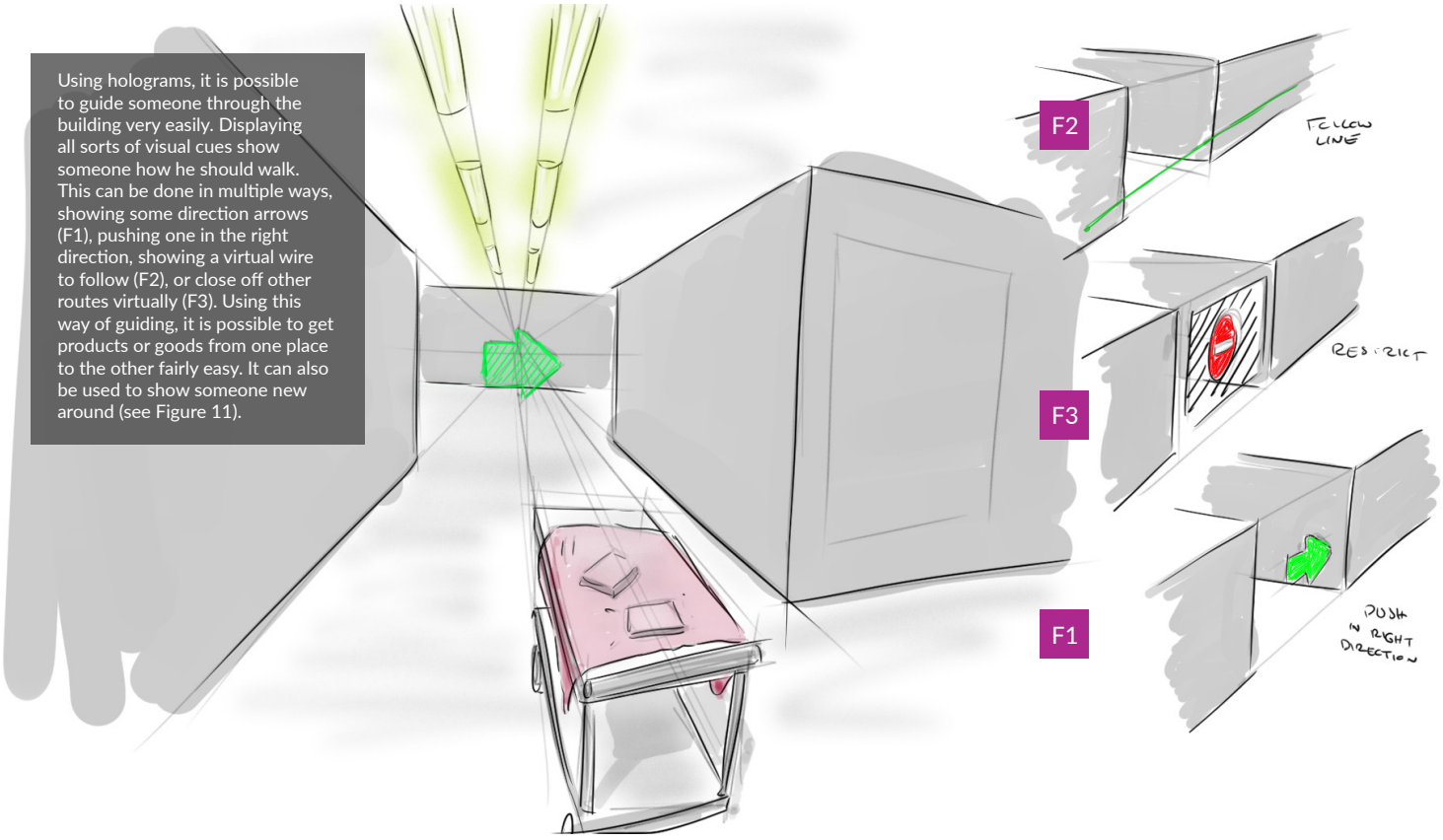
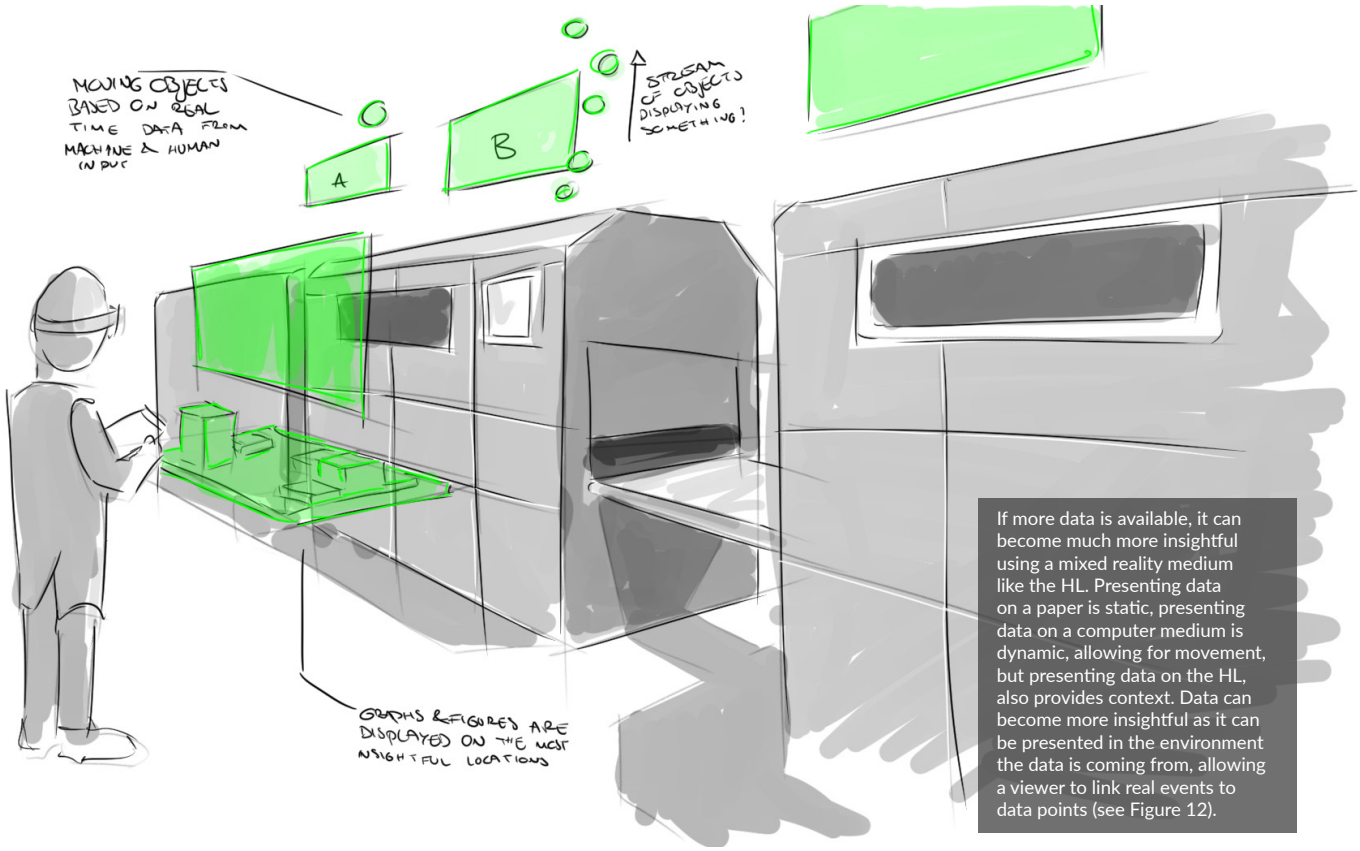


Figure 11. Sketch displaying ideas on the use of holographic cues for guidance through a building.



If more data is available, it can become much more insightful using a mixed reality medium like the HL. Presenting data on a paper is static, presenting data on a computer medium is dynamic, allowing for movement, but presenting data on the HL, also provides context. Data can become more insightful as it can be presented in the environment the data is coming from, allowing a viewer to link real events to data points (see Figure 12).

Figure 12. Sketch displaying ideas on displaying holographic data.

7. Conceptual phase

7.1 Low tech concept

This concept focusses on the current situation of the Manufacturing Area and how the current problems could be solved on a short-term notice. The main goal was to design a workflow without disrupting the current structure.

7.1.1 Workflow

A system is used that comprises all WPI and sends them to the right employees. Benchmark already uses inhouse databases to store all WPI. These can be used for storage. Tablets are connected to the cloud-based system and have data going between them (see Figure 13).

Small and light tablets offer most convenience in handling. In terms of interface, they can resemble the WPI already present. Using tablets skips a few of the overkill steps present when using paper and offers at least a foolproof way of getting process results by forcing an operator to fill in a certain value before being able to continue:

- operator reads step on tablet, possibly accompanied by an explaining image
- operator fulfills step on product
- operator fills in result
- button to continue becomes interactive, allowing operator to continue

An application for using the WPI on the tablets would be needed for this.

The tablets offer a set of advantages (see Figure 14 and 15):

- direct results from operators are stored in the cloud
- current work process instruction is always available
- colored and up to date pictures are available in instructions

7.1.2 HoloLens©

The HL is used as an added-value object. Especially as a marketing tool it can create happier clients. Projects can be taken digitally to the client and shown on location. Updates on products in the making can be shown in a whole new way.

Showing this technology to clients could be impressive to them. As most people have never experienced convincing augmented reality, they will probably be overwhelmed by

A. Cloud based data storage system, which is already present at Benchmark is used for storing data from Manufacturing and Engineering. It contains up to date versions of all process instructions for all products.

B. System is able to visualise graphs using data from manufacturing processes. As operators are 'forced' to gather data digitally, the possibility of losing data is minimized.

C. Engineers look at gathered data to see where most defects are originating. Insight is created in current inventory flow. This data is used to improve processes.

D. Processes, models, and instructions are editted and updated based on findings.

E. Updated models and WPIs are stored in the cloud database system as new versions.

F. Data and WPI are gathered from the cloud, and are always up to date. The operator only gets to see the most recent version.

G. Depending on their function, users get a certain interface that shows them the right instructions.

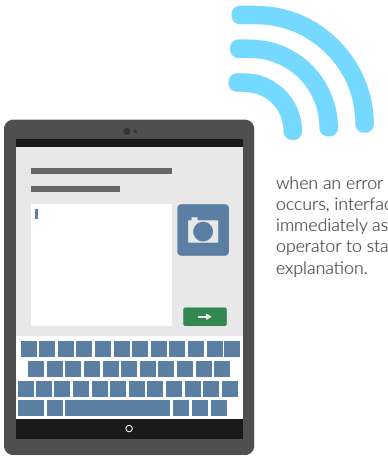
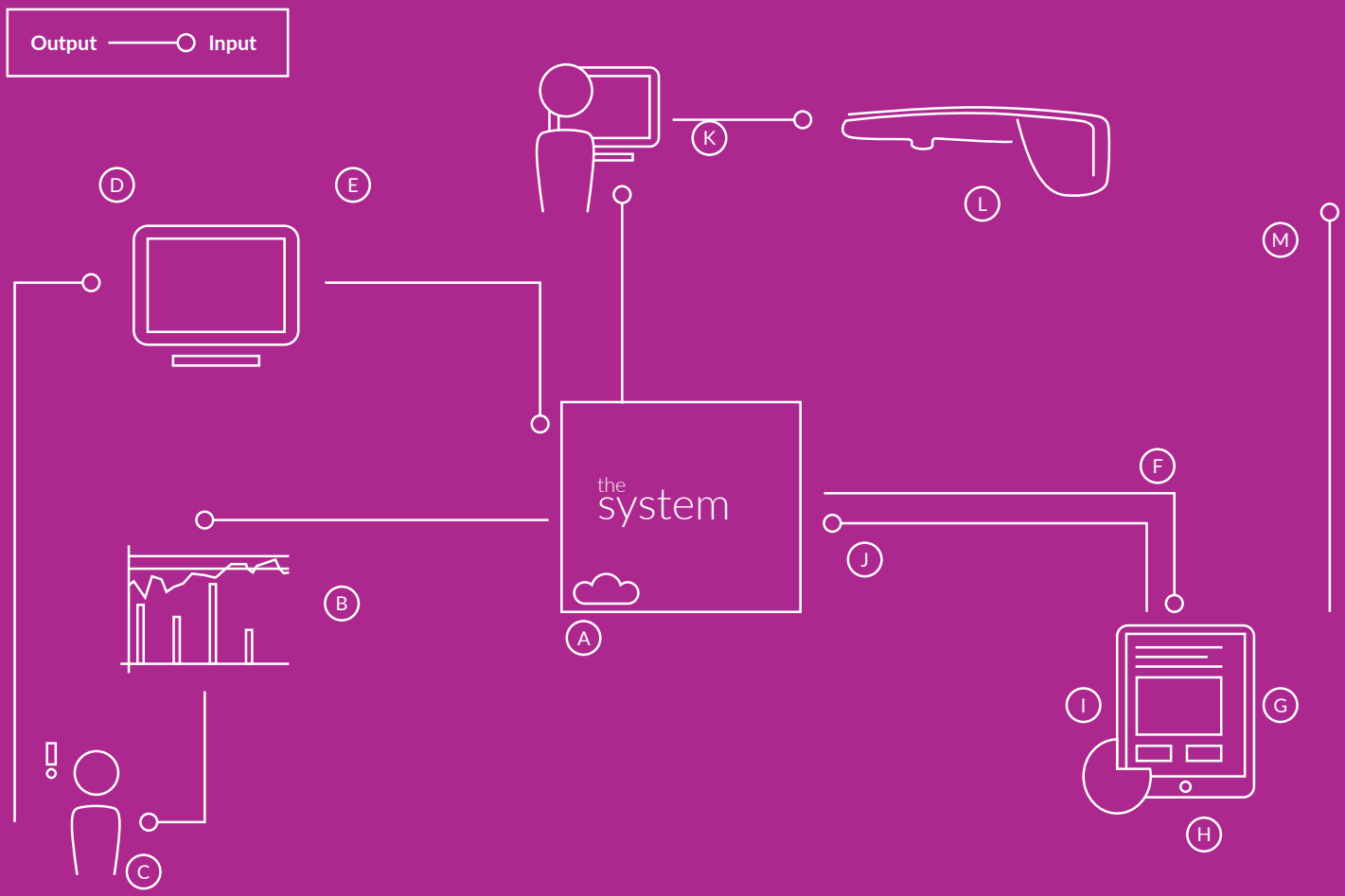


Figure 14. Interface changes dynamically to ask what problems occured.



H. Operators that work with paper versions of WPI get a small mobile tablet.

J. Data from processes are gathered from user input and sent to the cloud system. It is stored to be analysed.

L. Applications are deployed on HL and possibly combined with real objects. The HL can be taken anywhere to show to (potential) clients.

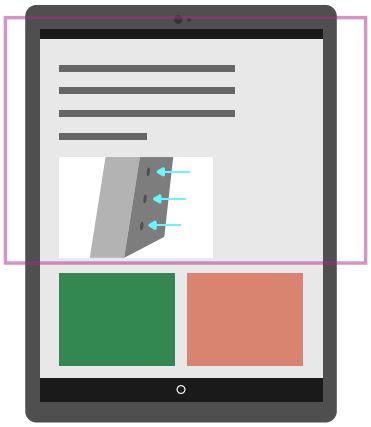
I. The interactive nature of the interface ensures that findings of the operators are saved.

K. Using the stored models in the database, holographic experiences or applications are built to show to potential clients or as final products.

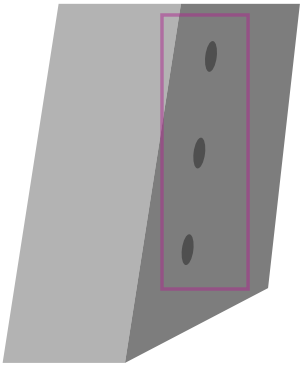
M. Final products can be tested or showcased in a holographic environment to review them as if they were already integrated into their final environment.

Figure 13. Schematic overview of the workflow.

read task and possibly see image



check product and fulfill step



fill in result on tablet to continue

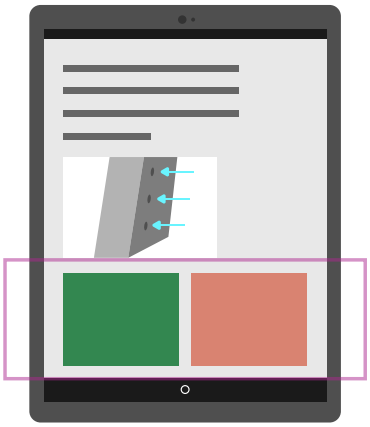


Figure 15. Depictions of how tablet interface could look. Left: operator reads task and checks image. Middle: operator checks or assembles product. Right: operator pushes pass or fail button, allowing him to continue.

the way holograms appear in their surroundings. Using the HL can help Benchmark improve its image of an innovative enterprise.

Showing an update of a product, does require someone to create a model that is usable on the HL. Due to some of the limitations of the HL, most CAD-models need to be converted before being able to deploy it on the device. For example, this includes the reduction of vertices.
The gap between the real and virtual world can be minimized by taking small real parts that can be held in place of a virtual part when reviewing or showing a product to a client (see Figure 17). This creates a feeling of scale and existence of the product, as an observer is seemingly physically touching a part of the hologram. Rapid prototyping techniques, like 3D printing, offer a great solution here. It is cheap and relatively fast and quality is not of great importance. Benchmark already possesses several 3D printers, allowing this to be a relatively easy outcome. Testing or showing finished products is also possible by taking a product and simulating an augmented environment around it.

This concept solves the problems with work instructions by creating a digital workflow rather than an analog one. This way, the paper workflow that creates a large amount of the current problems is excluded and data loss is prevented. Due to the fact that the workflow is very similar to the old one, problems switching will be avoided. WPI are still used in a similar way, so the only thing operators need to learn is how to operate the tablet and the application. Investing in tablets is a relatively cheap option. Problems such as battery life during the day, deterioration of the quality of the tablets, or breaking of the tablets may still occur. A period of testing with (a few) operators can be a suitable way of finding out whether this structure fits the company.

Using the HL as a tool to add value for clients is a way to make clients more appreciative of the work Benchmark performs. The extra time needed for creating holographic presentations of products is a tradeoff for both a more impressive as a more insightful way of presenting projects to clients. The HL is also used for showing virtual tours through the building. In doing so, the innovative nature of the company can be shown to potential clients or partners (Figure 16).

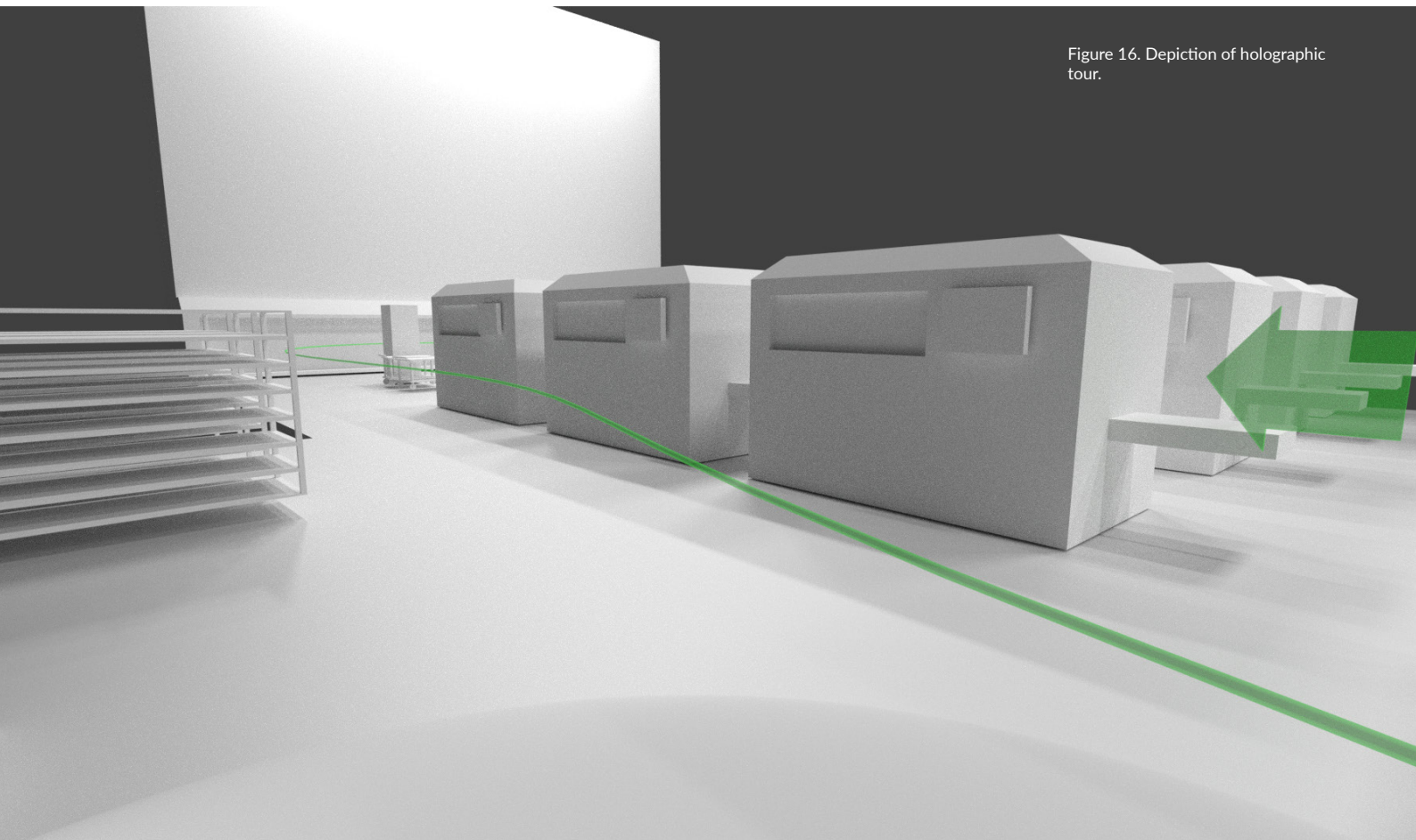


Figure 16. Depiction of holographic tour.

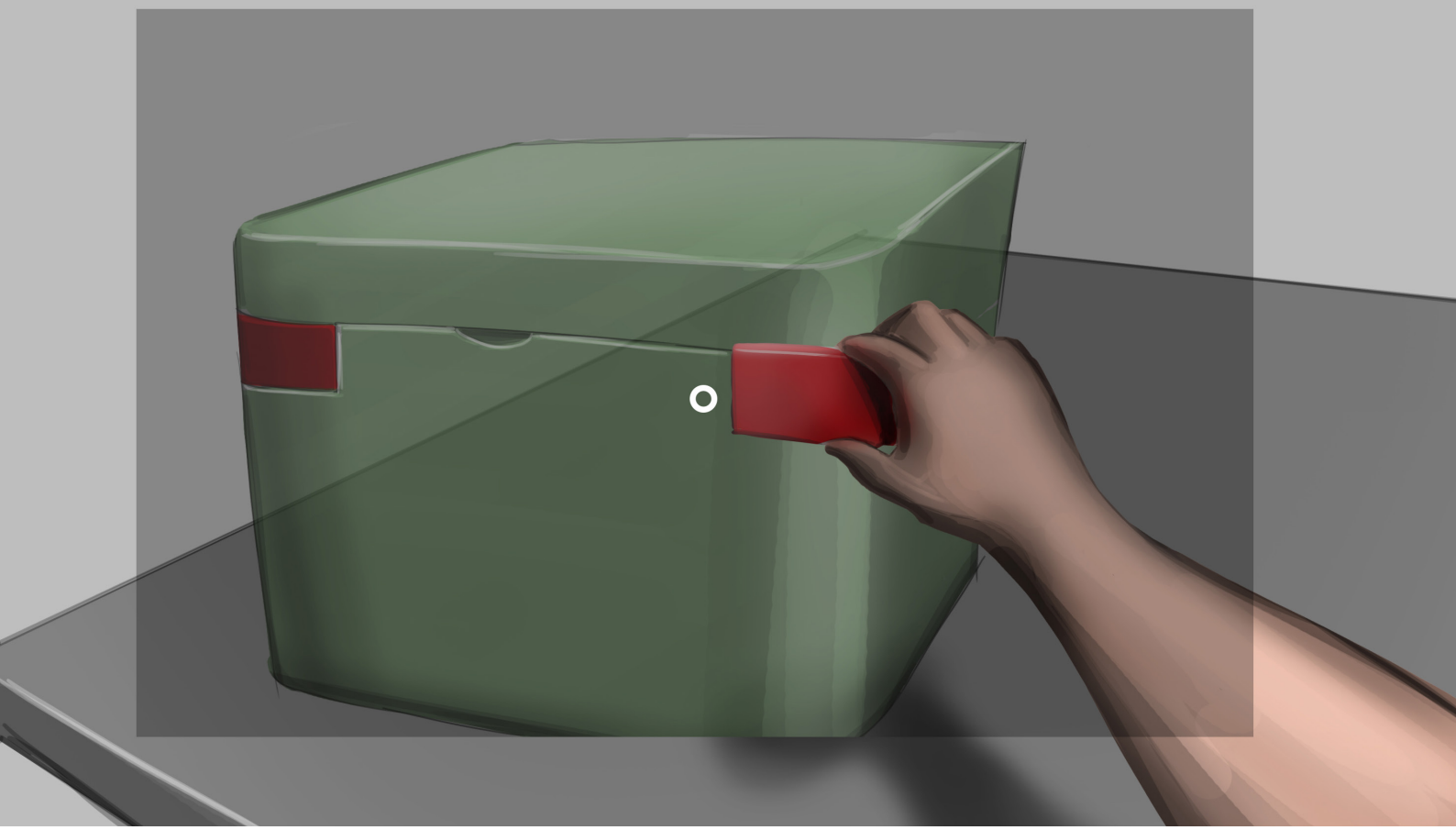


Figure 17. Depiction of field of vision when viewing holographic models combined with small parts.

7. Conceptual phase
7.2 Mid tech concept

A smart digital system is used for data analysis. The HL is used as a development tool and offers insight in data. Tryouts in Manufacturing show where the HL can best be integrated on a short-term notice. The HL has a supportive value. (see Figure 18).

7.2.1 Workflow

A digital system is used to aid and structure the processes and gain feedback about the production line. The system is central in the workflow. In the Manufacturing Area, a shift is made from analog, to digital. This allows for better tracking of data and offers operators the current work process information.

There are three aspects where the HL will be offering support to the company, as a development tool, as a data visualisation tool, and as a virtual WPI projection tool.

7.2.2 Development Tool

In the development area, the HL is used as a review tool. CAD models can be viewed holographically, to get an idea how they look. As most CAD files cannot be imported directly onto the HL, it is important to have a tool that converts the created CAD-files to a HL ready file format. This will keep up the workflow, as developers do not have to convert the files manually.

It is important that the file converter should do the following (see Figure 19):

- turn the CAD model into a surface-model file format for HL (.fbx .obj)
- reduce the amount of vertices
- remove unimportant/small parts, such as screws
- upload the converted file to the HL or to a shared database the HL has access to

Scenario development

Mark, a product engineer at Benchmark, is having trouble finding the right layout for a set of buttons on a product he is developing in his CAD program. He cannot quite grasp the right layout that is both easy to use and looks good on the product. He decides to view his model holographically, to get a better understanding of the product in a real environment, on true size. He saves his file and opens the conversion program. From his file explorer, he drags the file into the program. Once he clicks 'send' the program starts to convert the file to a format ready for the HL. In the meantime, Mark grabs one of the available HLs from a cabinet and puts it on. The program has finished converting the file and has sent it to the HL. Mark sees his model floating in front of him, waiting to be placed in his surroundings. He taps his finger and gazes at a table. With a tap, he places the object. He can now see the button layout he was thinking about on the product in front of him. The virtual product has given him conformation that this layout will both be useful and easy on the eye. He now puts down the HL to start working on his next task.

A. Cloud based data system, used for storing data from Manufacturing and Engineering. It converts data to use on HL and keeps graphs in environment up to date.

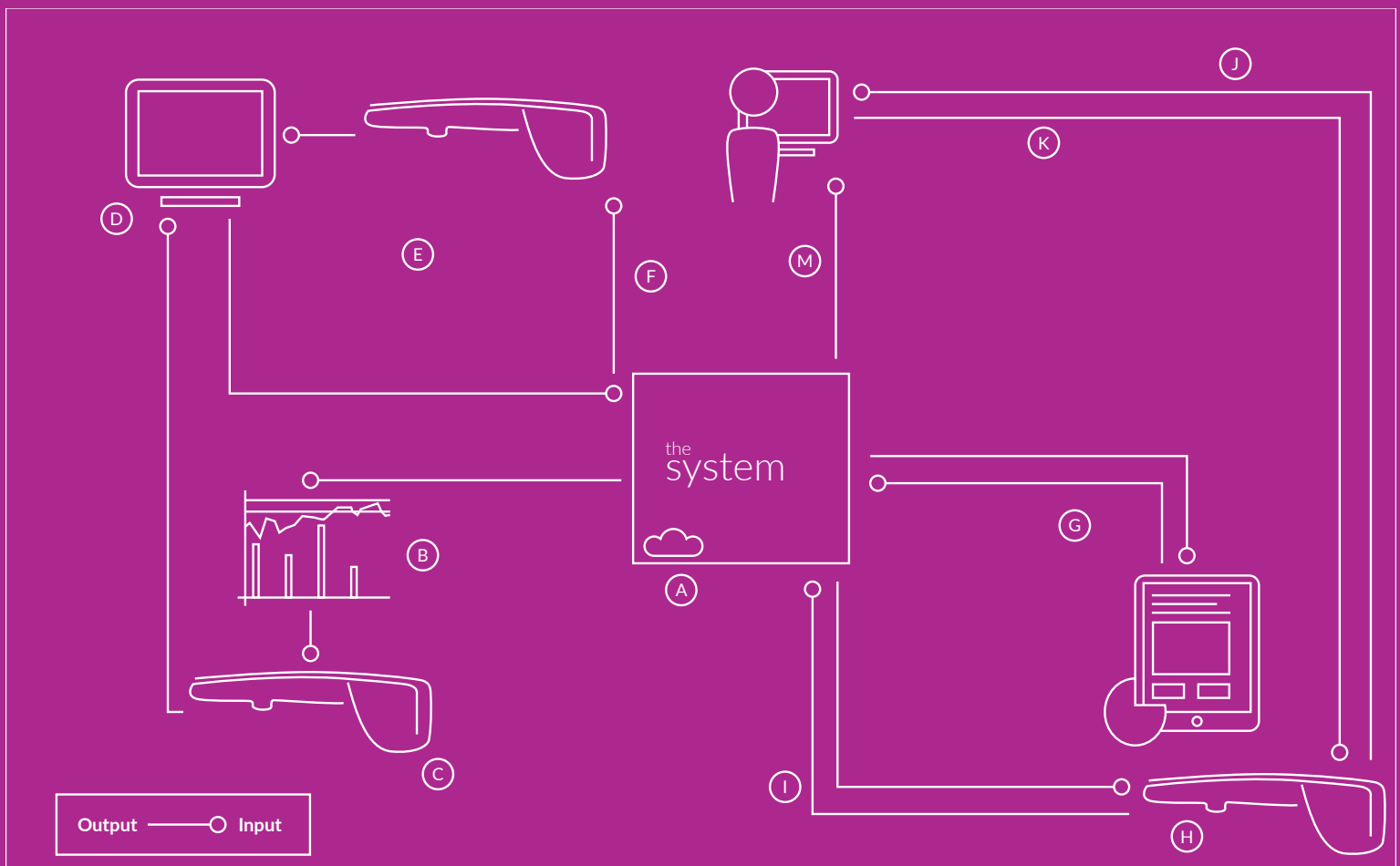
B. Data from Manufacturing is used to create insightful real-time graphs build into an application usable for HL. It connects data with real environment.

C. 3D visualisations are projected in the production area, to get better insight in the production processes and where improvements could be made. One is looking at data and area it has to do with simultaneously.

D. The gathered insight is used to edit products, instructions, or even processes.

E. Engineering works with HLs in development of products. While creating a model, an engineer can review it holographically. Created CAD models are editted for use on HL by the system automatically, to keep up the workflow. The 3D view of the created models functions as input for the further creation of the model or the process line.

F. System shares a database with HL for distribution and deployment of models.



G. An all-digital system ensures up to date information and allows to gather data to analyse.

I. Due to internet connection, instructions are up to date and data is gathered continuously.

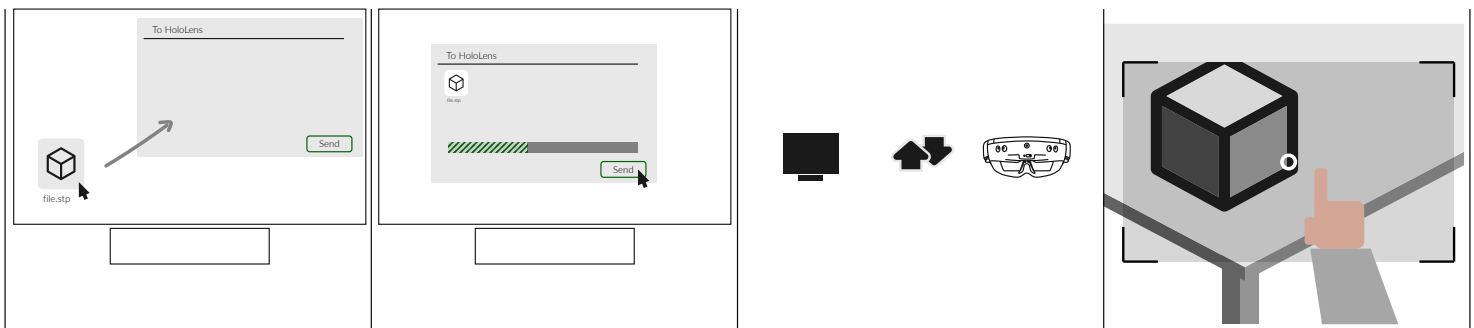
K. The engineer is able to see the view of the operator and can draw visual cues to help the operator get to understand the step.

H. Try outs with the HL in Manufacturing introduce workers to augmented reality solutions and offer some advantages like a handsfree workflow.

J. A connection can be established from the operator who is working on a product to an engineer. The engineer knows more about the product and can offer help.

M. The engineer can get information about the product the operator is working on from the database.

Figure 18. Schematic overview of the workflow



User places file in application on computer.

System converts CAD file to HoloLens format.

System uploads file to HoloLens or to database available to HoloLens.

User can (re)view model holographically.

Figure 19. Automatic file conversion

7.2.3 Data Visualisation

A shift to digital, makes it possible to continuously gather data from all processes. The output of all processes is combined by the system to create data visualisations. Visualisations can be updated in real time. Using the HL, the gathered data can be made insightful, by creating 3D graphs that, due to being displayed in a digital medium, can change over time. Another way the data can be clarified, is by placing graphs and visualisations into specific places in the working area at Benchmark.

Instead of looking at a paper stuck to a wall somewhere, someone wearing a HL could walk to a machine to see specific data about or related to it.

This would especially suit the PCB area. This room houses machines that are large enough to be picked up by the HL sensors. Since the machines are stationary, visuals can be displayed above or next to them. Displaying a graph that is linked to data from a specific machine or process, directly allows the viewer to get a feeling of how the machine is performing (see Figure 20).

Benchmark is already collecting and displaying data on boards throughout the Manufacturing Environment. As the HL is implemented, these could become interactively integrated within the work environment

7.2.4 WPI Projection

Integration of augmented reality in the Manufacturing Environment is a large step to achieve. It is possible, to get used to augmented reality in a familiar way by displaying the known WPI holographically. Floating panels are used to display the instructions. The panels can get placed anywhere in the environment. Using the HL to display the instruction guide, allows the operators to use a hands-free approach (see Figure 21).

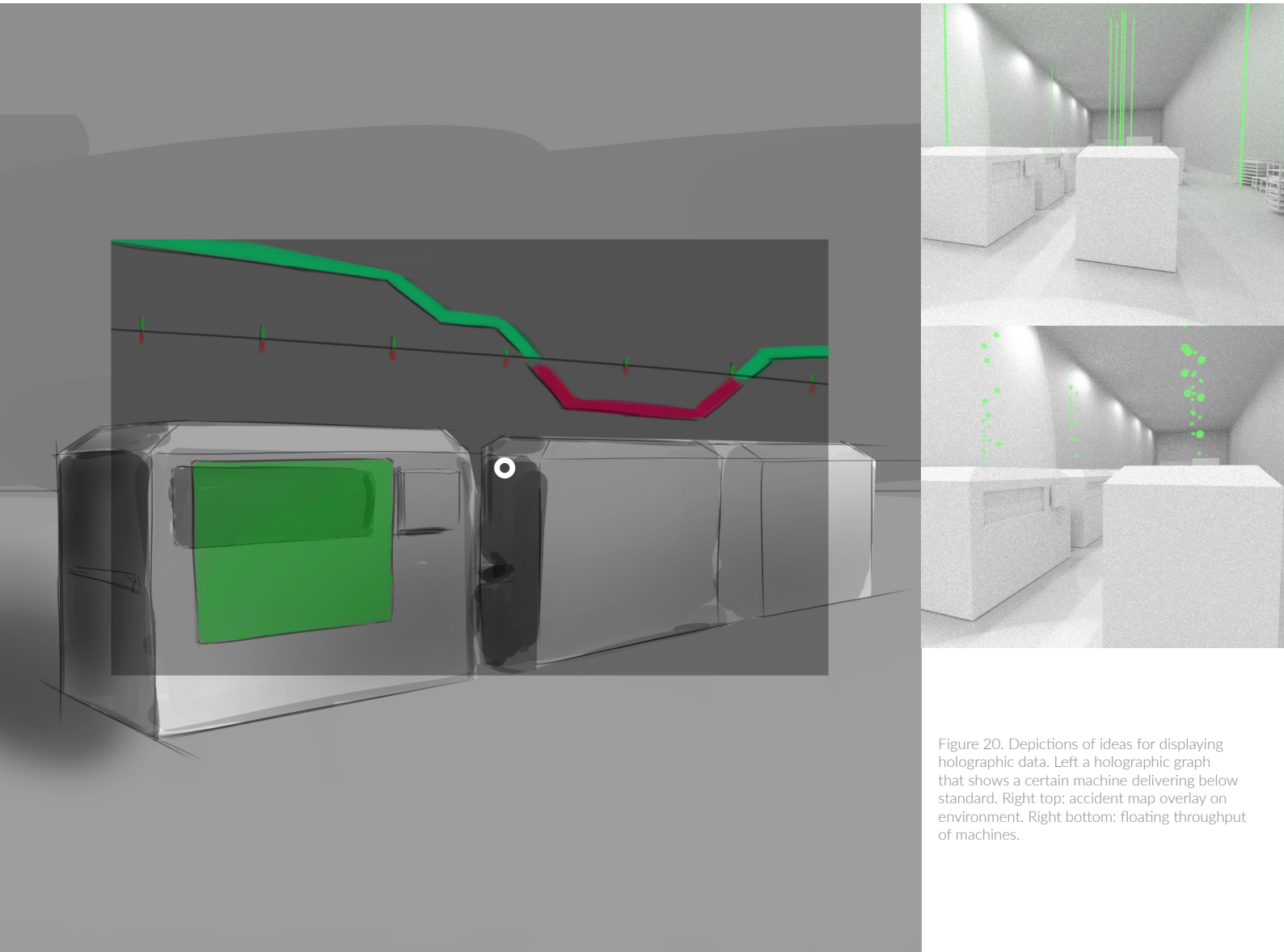


Figure 20. Depictions of ideas for displaying holographic data. Left a holographic graph that shows a certain machine delivering below standard. Right top: accident map overlay on environment. Right bottom: floating throughput of machines.

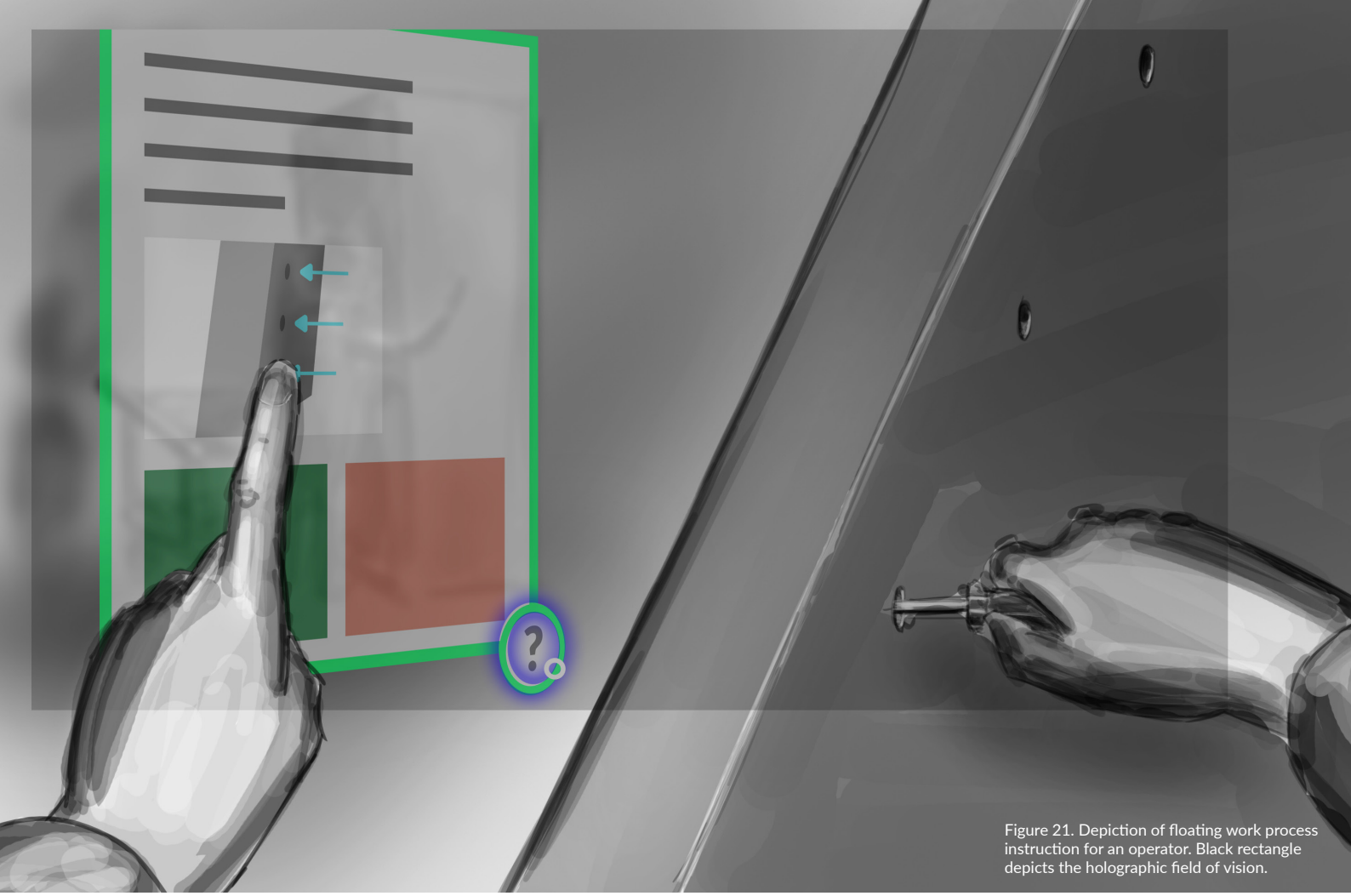


Figure 21. Depiction of floating work process instruction for an operator. Black rectangle depicts the holographic field of vision.

The application handling the display of the instructions holographically, also includes a shortcut to skype. Whenever the operator has a question about a step, the skype application can be launched. Using Skype on the HL allows the operators to call someone from Engineering. The engineer is able to see the field of view of the operator. Besides explaining by words, the engineer can also draw visual cues in the operator's view. The skype application built for HL already allows this.

One of the advantages of this system is that Engineering gets more approachable for operators. Instead of phoning or going and getting an engineer to explain, a simple skype call will suffice. The hands-free workflow is possible for all functions that would be using a tablet or paper sheet for instructions, such as assembly and quality inspection.

Scenario WPI projection

Operator Michael is working on a product assembly. He looks at the holographically projected instructions in front of him. As the instructions are floating in real space, he always has his hands free to work on the assembly. At a certain moment, he does not understand what the next step involves him to do. Instead of walking to the phone and trying to explain the problem he encountered or walking all the way to the department where the engineers are located to get someone to help him, he looks at the bottom of his virtual instructions and taps the help icon. A skype connection is started with the right engineer directly, as the system knows which engineer knows how to assemble the product right. The engineer gets a pop up on his computer, which he opens. Besides seeing the explanation on the step the operator needs to fulfill, he can also see a livestream of the field of vision of the operator. He guides the operator through this step of the process by telling him what to do. Both are happy it all worked out. The system registers that there was a call for help at a certain step in the process, and saves this for data analysis.

7. Conceptual phase
7.3 High tech concept

This concept is the most advanced in terms of technology. In this concept, it was attempted to take all the possibilities of the technology and combine it into a solution for the Manufacturing Area. The current situation and needs were considered, and a set of solutions established.

7.3.1 Workflow

The concept comprises a dedicated smart system, integrated in all workflow, that controls HLs employees are wearing (see Figure 22). The system controls all processes and leads all employees to tasks that need to be fulfilled. Using machine learning algorithms, the system will create an efficient workflow, that gets faster over time. All operators wear HLs and can be positioned in several functions. The system knows where manpower is needed and can guide operators there. Holographic guidance makes functions easy to understand. This matches the lean management philosophy Benchmark is trying to integrate in the production processes, where the goal is to reduce waste and wait times.

To create a truly leading system, all inventory is traced, both physically as well as digitally. Digital tracking goes through serial numbers and received updates from operators and machines. The physical tracking system uses cameras that are placed in the ceiling throughout the Manufacturing Areas. The carts that are used to move products and parts from one place to another can be followed. As a product is placed on a specific cart each time, it is enough to track the carts through space and not the products themselves. The system can integrate the tracking of these carts into the holographics of the HL. As carts might not get recognized by the HL due to their size and constant movement, the camera system can still push the coordinates of the carts, and so, the places where holograms need to be displayed. To do this, the dimensions of the carts should be known by the system, as well as their position in space. One way to accomplish this, could be to place easily trackable markers on the corners of each cart. The tracking system only tracks 4 points for each cart this way (see Figure 23). Besides the tracking of carts, the system also knows the location of the present operators, as they are all wearing HLs, which scan the environment and know their own position in space.

A. Smart integrated system controls all workflow by efficiency calculations

B. Data that is gathered by the system is internally processed and used to gain insight in where the process can be improved. Machine learning allows the system to gradually become more efficient.

C. Results from internal analysis are also presented to engineers for improvements to production lines and products, things that are not efficiency related to the system.

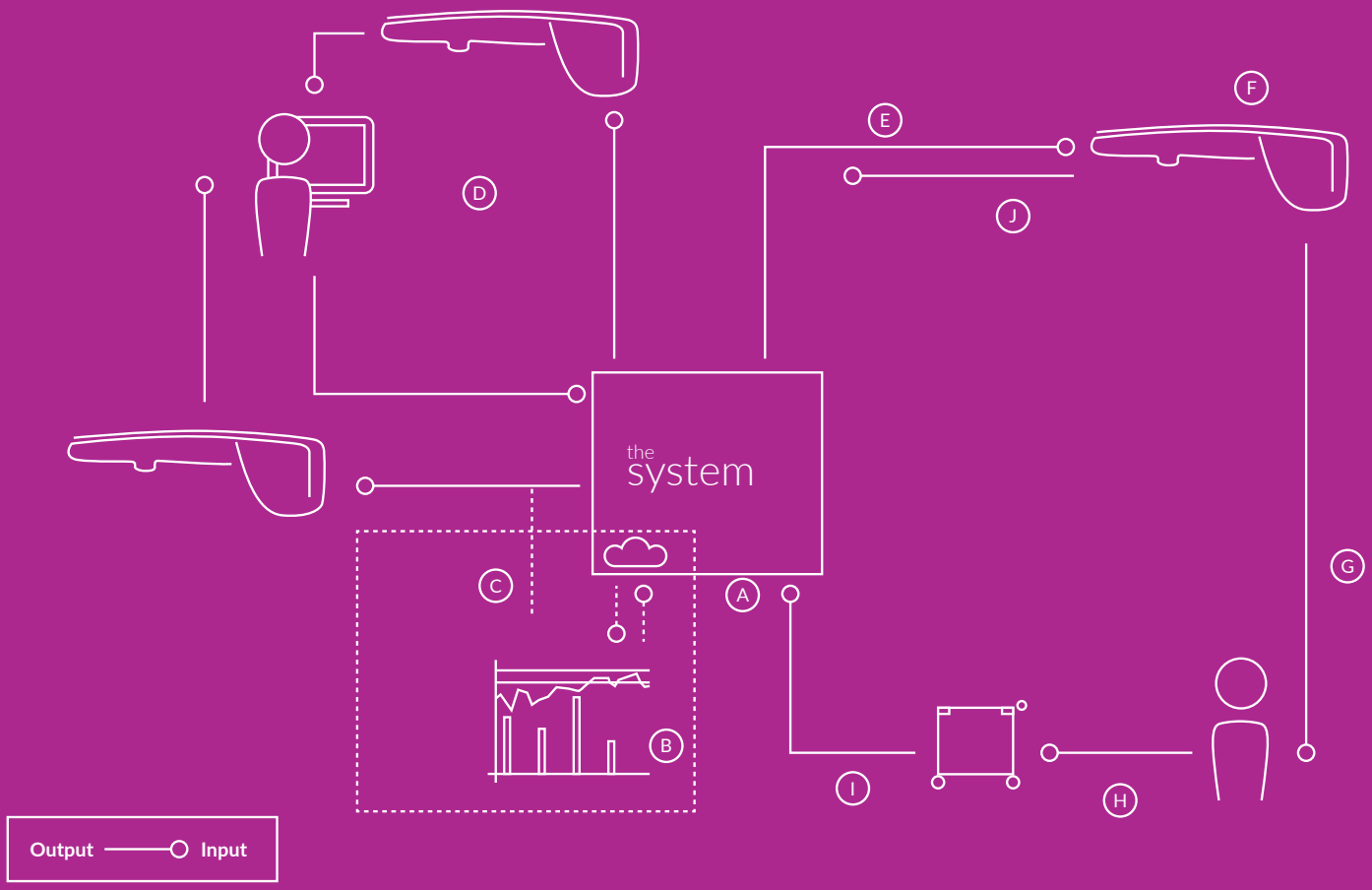
D. Using HLs, engineers create models and production lines. The HL is used as a (re)view tool.

E. System communicates guiding tasks to HL based on location, time, and processes.

F. Based on tracked processes, HLs are sent to a certain task. Tasks and function decide what an operator sees holographically.

G. HL guides operators through their task by giving visual overlays on the real world.

Holographic guidance is suitable for at least the following types of operators: assembly, quality inspection, and Water Spider. They can be found in the Clean Room and Box Build Area.



H. Operator moves parts and products through manufacturing areas on carts.

I. Through camera tracking, location of all carts is known to system.

J. HL reports back to system location and direction operator is facing and the status of the task it is occupied with.

Figure 22. Schematic overview of the workflow

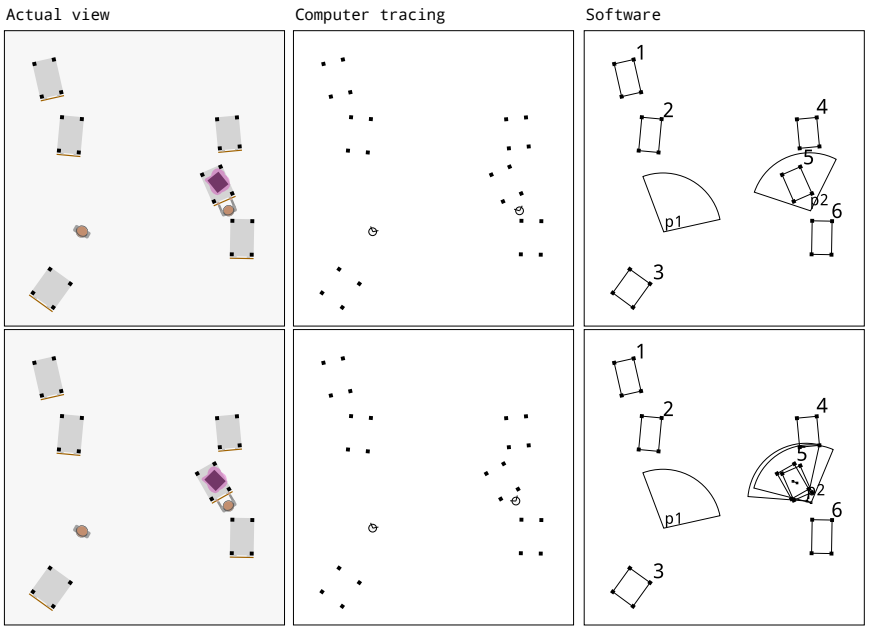


Figure 23. Visualisation of the camera tracking. Top: state 1. Below: state 2, after a small movement.

7.3.2 Holographic Guidance for Assembly

Assembly operators are guided through the process of building the product holographically. Instead of written text and pictures, guidance is accomplished by the use of holograms. A reference hologram, that shows the step the operator has to take is placed somewhere around the operator, while he is performing the assembly tasks. The operator can look at the hologram, that shows a virtual version of the product, and repeat the steps on the real product. The operator does not have to read instructions, and it is easier to see what the exact step is, as the operator can even walk or look around the product in 3D. He has his hands free for the task (see Figure 24).

Scenario assembly guiding

Jim starts his shift as an assembly operator. He goes through the procedure of putting on the right clothing and his HL. He logs on, and the system greets him. In a floating window, it shows him what he is going to be doing today as his first task, assembling a certain product. The HL asks him if he wants to start and once he does, a virtual path appears showing how to get to a certain cart containing the goods he needs for assembling a product. He collects the cart and takes it to an assembly spot. He selects the option to start assembling. The first step is to pin a holographic reference plane somewhere in the environment. This plane is used to show the steps he will have to perform, accompanied by audio guides. On the plane, the same assembly he has to create is being built holographically. He now easily can go through assembling the product. If the visual guide is not enough to create the assemblies, he can also open a large floating screen with written instructions.

7.3.3 Holographic Guidance for Quality Inspection

Quality inspection is done by displaying holographic cues onto the product. The use of floating bounding boxes is used to guide the operator his attention directly to a certain spot on the product. Holographic objects are used to indicate that a certain function has to be performed. For example, checking if all screws are present and screwed in tightly, displays a floating screw. Checking surface quality displays a small floating smooth or rough plane. Using voice commands and gesture recognition, the operator can go through the steps. (see Figure 25).

Scenario quality inspection guidance

Max is working in quality inspection. As he arrives at his job, he puts on his coat, his gloves and hairnet. He goes into the

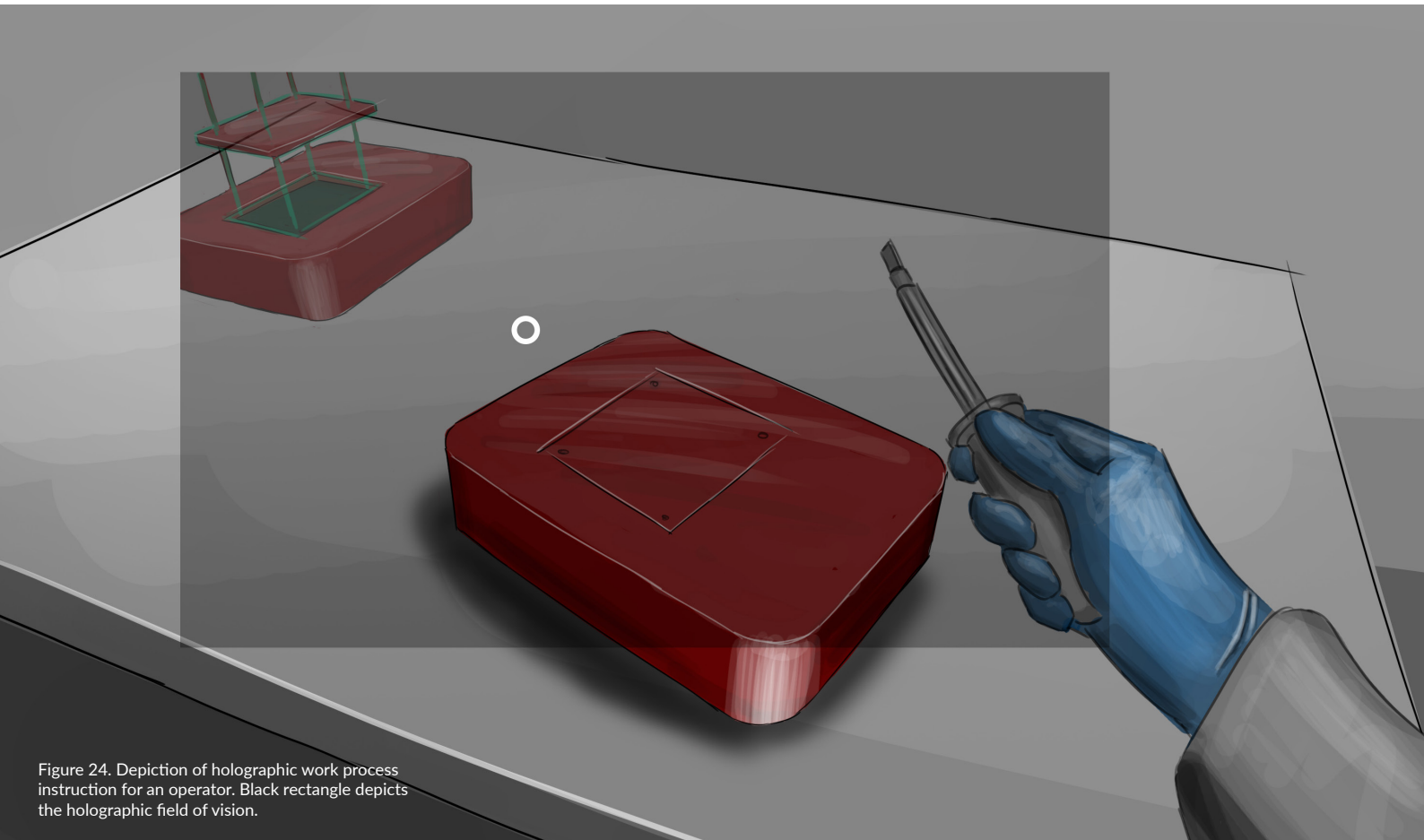


Figure 24. Depiction of holographic work process instruction for an operator. Black rectangle depicts the holographic field of vision.

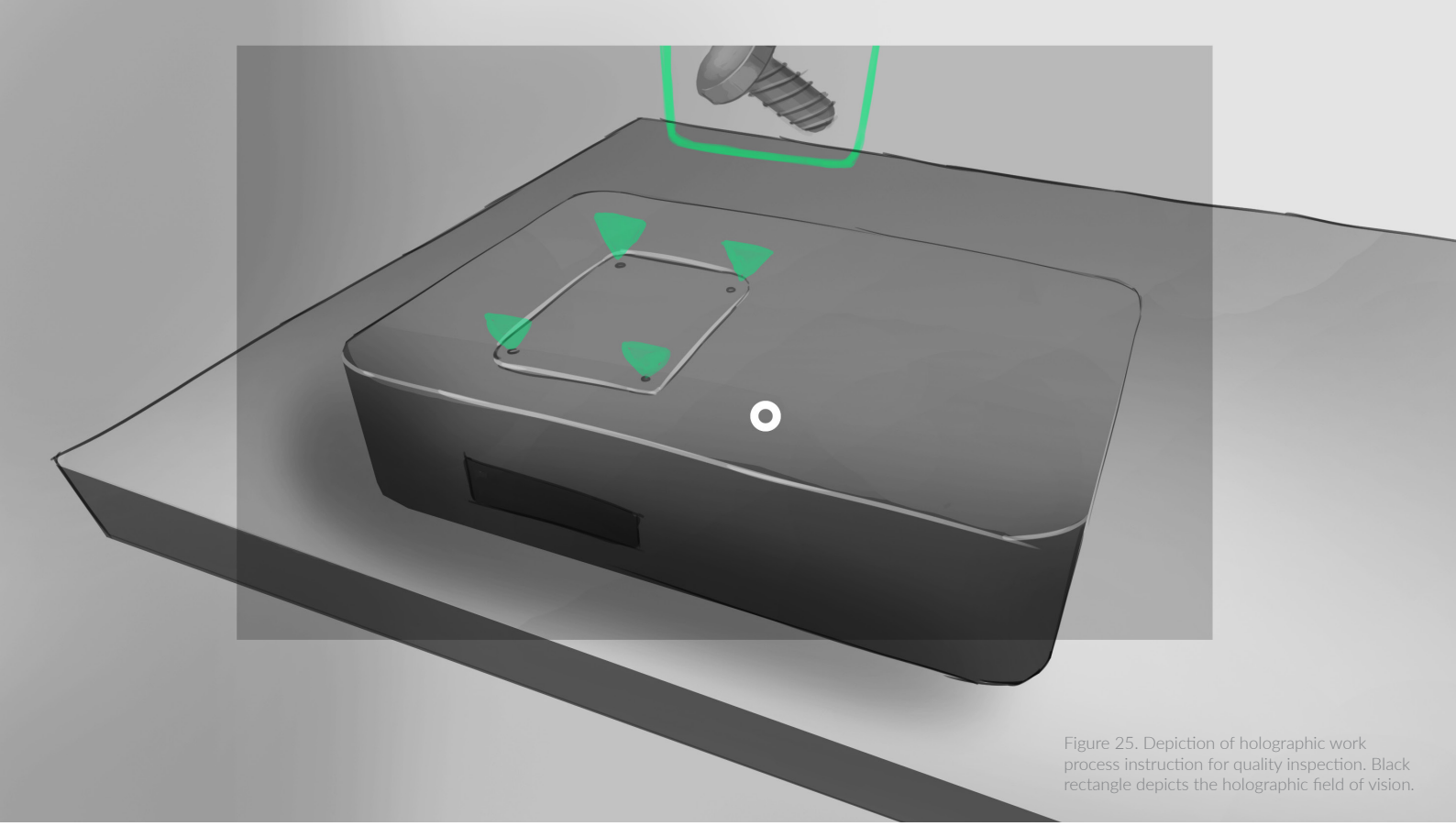


Figure 25. Depiction of holographic work process instruction for quality inspection. Black rectangle depicts the holographic field of vision.

Clean Room, where among other things, quality inspection is performed on assembled products. He walks towards his working space, a line of tables. On his table, he opens up a small drawer, where his HL has been charging overnight. He puts on the device and opens the start menu. Max looks at the wall next to him and pins the instruction menu onto the wall. He sees he has to get the cart with product number [GUVB-005-GLISE-7793]. When looking over at the row of carts, he sees one cart with a marker above it, that must be his. He goes and gets this cart from the row, and brings it to his work space. Looking at the product, he now sees holographic cues of what he has to do. The first task asks him to check the upper surface for scratches. He does not see any, so he says “clean”. The next task asks him to check the product for 6 screws. On the places the screws should be present, yellow boxes appear, while a screw is rotating above the product. While gazing at a one of the yellow boxes, he says “check” as the screw is present. The box turns green, and Max goes on. Once he has performed all steps for the inspection, he sees visual cues to bring the cart with the product to a row of other carts that have been checked.

7.3.4 Holographic Guidance for Water Spider

The Water Spider collects parts and prepares orders for assembly. Using the HL, it becomes easier to collect all parts, as visual cues are displayed on where to pick up the parts, and where to put them. As the storage of the small parts is fixed on a specific location, it is possible to augment visual cues from the specific places where the parts can be found. A specific list is available for each order that has to be picked, and therefore virtual locations can be displayed per order.

Scenario Water Spider guidance

Michael enters the room and puts on his HL. He logs in and starts the appropriate app. A large virtual screen appears. On it, he can see a list of orders that need to be prepared. Every order contains a set of small parts that need to be included with some other parts on a cart. These parts create the full assembly. The list is ordered, so that priorities are on top. He can select multiple orders at once. When he looks at the cabinets where all small parts are located, he can see multiple floating colored balls, showing him which parts he has to pick up. All floating balls of the same color correspond to one order. The colored balls give a cue for the location of a pickup. Once he gets close to one, a small floating message appears, showing the product and amount he has to collect. Once he has grabbed all parts for that ball he can close it by saying ‘check’, or by double airtapping it. All small parts are collected in small boxes and placed on a cart that has an overlay in the same color of the floating balls. Once finished, Michael is visually instructed to place the cart in a certain location in the area.

7.3.5 Engineering

In engineering, some changes need to be made in order to make this concept possible. As WPI have become holographic guides, there is a need for someone who creates holographic guides, based on the available CAD files. A special program is needed to create these guides. The engineer loads a model in the program. Using in-program controls such as dragging objects, he creates a guide. The program will, eventually, render a holographic application guide from the input of the engineer and store this in the cloud. For quality inspection, the engineer clicks some part of the model that needs to be inspected and adds a certain inspection variable to it. A variety of animations and prefab models are included in the program, so the creation of a guide is kept simple. Using the directions of the faces selected, the program knows how to display the holograms. A bounding box will for example be parallel to the selected face, while an arrow will be perpendicular (see Figure 26).

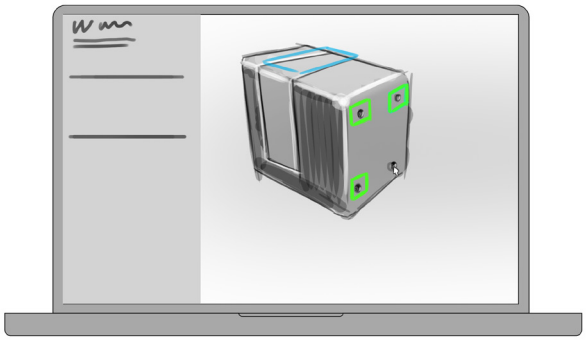


Figure 26. Depiction of creating a holographic quality inspection guide.

The system tracks all objects and persons. Instead of tracking data and outputting this visually in graphs to show what could be improved, the system itself creates the most efficient workflow. The system can estimate how much time is needed for a certain operator to fulfill a certain task and use this to control the workflow. If one operator is almost finished in quality inspection, the system can already start to think out what his next task could be. As there might not be another product to be inspected soon, the system could guide him to another function. As the HL will offer a managing approach to the operators, operators can be put into different roles. Tasks get easier to follow and operators are no longer bound to learning curves that force them to a specific function. The system, calculating where more or less power is needed, can now distribute workforce.

8. Executive phase

During the concept phase, several possible solutions were created for the integration of holograms. In this phase, an elaboration is made on one of the aspects of the concepts, namely holographic guidance for assembly.

The goal of this phase was chosen as a prototype application, working on the HL. Benchmark sees the benefit of having a prototype or demo application in-house, to make people enthusiastic about the HL and show and explore the possibilities of the device. Having an application to show and use, makes the whole concept of using the HL vivid. It is more approachable when compared to a report laying around. For me it is an interesting challenge, that forces me to learn how to work with the device and how to create an application for it. Creating a holographic guiding application seemed challenging, yet achievable, in the given time span.

8.1 Design application

Creating an application that guides the user through assembling a product, starts with a product. A product used in the assembly line in the clean room, was obtained. Together with the WPI this was used as the base for developing a holographic assembly guide (see Figure 27).

The application is based on the holographic guidance for assembly, also described in the High Tech Concept. It should show the user a holographic version of the product he is assembling. By repeating how the holograms move, he can assemble the real product the right way. As the HL does not have any buttons, natural input will be present in two ways. To proceed to the next step, the user can air-tap, which the HL will recognize as a gesture to proceed. The other way is by speech input. The user can say something like “proceed”, “next”, or “next step” to proceed to the next step, or ‘back’, “previous”, or “previous step” to go back to the previous step. A holographic model of the product to be assembled is placed in the environment of the user, for example on the table the assembly is created on (see Figure 28).

8.2 Method

In order to create the application, several steps needed to be taken. In Appendix C a general method can be found for building applications for the HL, based on the descriptions of Vroegop (2016). Here, the specifics creating this application are explained.

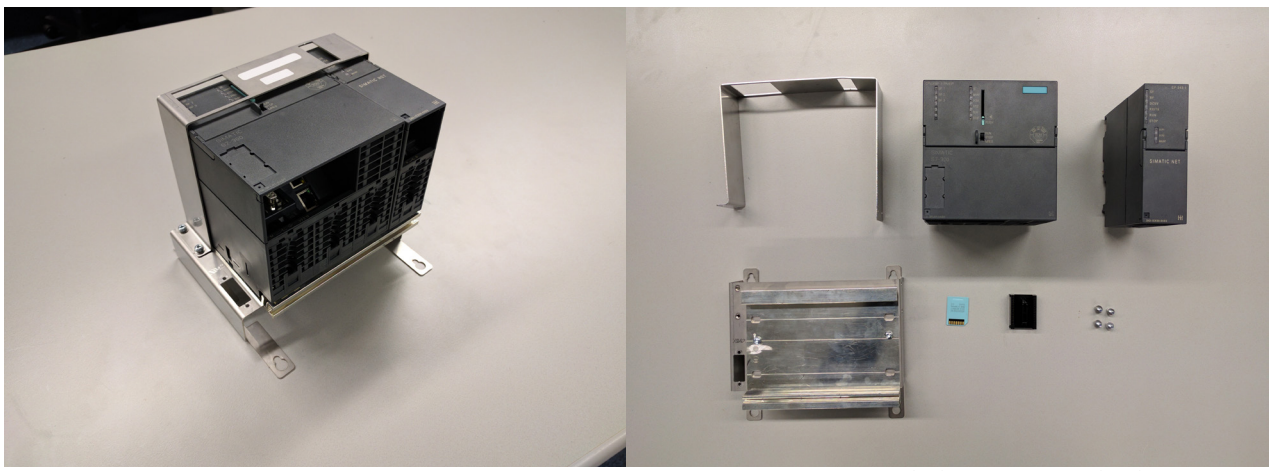


Figure 27. Product assembled (left) and disassembled (right).

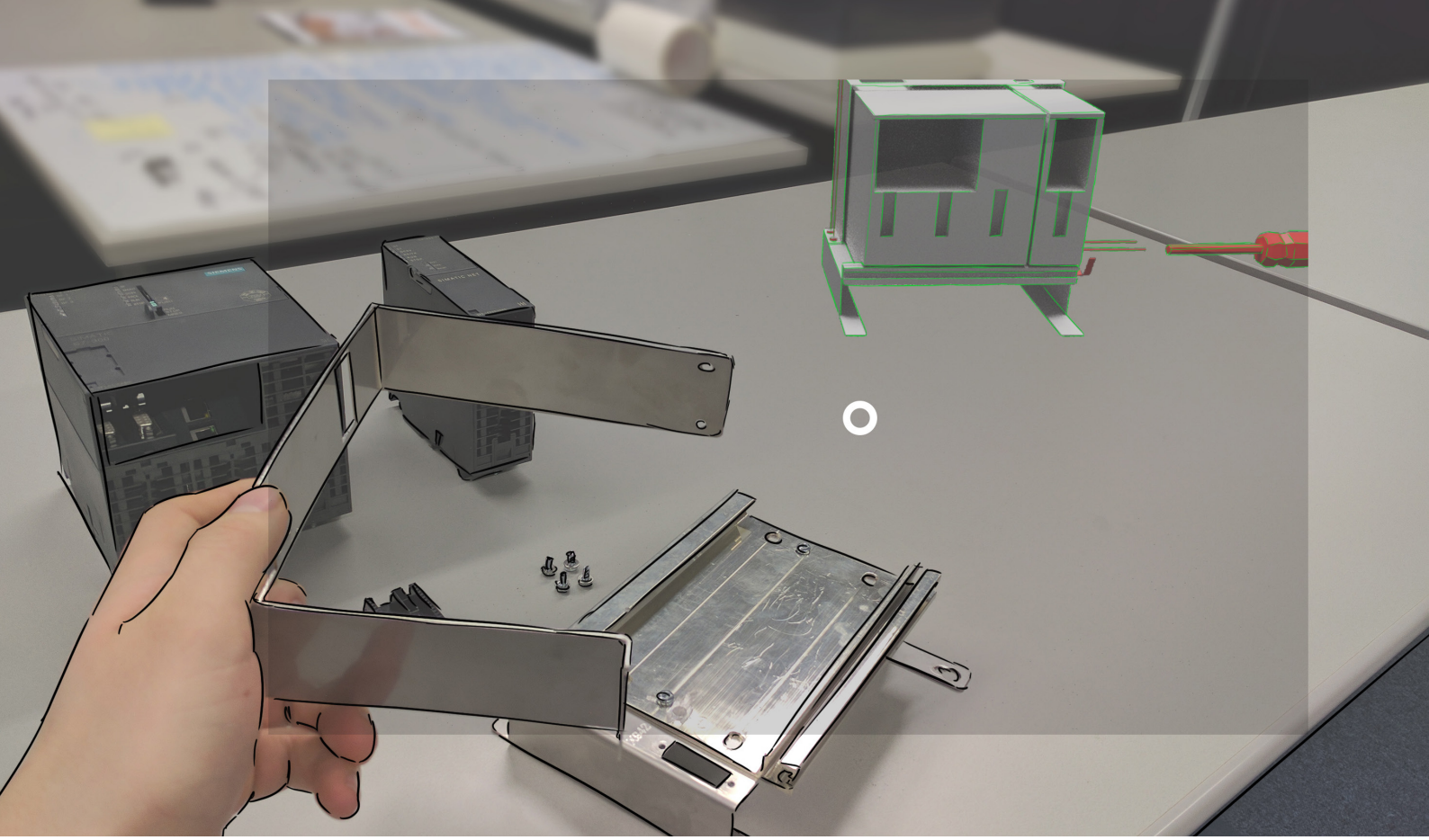


Figure 28. Depiction of how working with the application should look.

8.2.1 Model

The available CAD-files used for manufacturing, cannot be used on the HL. The HL uses surface models, existing of flat surfaces. Either the CAD-file of the product could be converted into a surface model, or a new model could be created from scratch. The choice was made to quickly recreate a new surface model resembling the original one. For testing purposes, it comes in handy to have control over the model, the number of vertices, and the subparts it consists of. The model was recreated in Blender. In Figure 30, a rendering of this model can be seen, where (A) depicts an exploded view, (B) the assembled product, and (C) the disassembled product (see Figure 30).
Note that the model, as it is created now, is a simplified version of the real product. When comparing Figure 27 and Figure 30, it can be noticed that the original product has a lot more detail in it. The limited number of vertices that HL can use, make a simplified model preferable, but one should always consider to what extend the details of a model can be left out. The more a model is simplified, the harder it might be to recognize the parts. An outcome to preserve both recognizability and a limited number of vertices, can be to apply a texture resembling the real product onto the virtual one. For this relatively simple product it did not seem necessary.

8.2.2 Instructions

The instructions show what steps need to be taken and what is needed in order to create the assembly. The instructions for assembly of the product were carefully read. Since certain parts of the product were missing or already taken care of (labeling the product), not the whole instructions were used. For a demo application, this however is not a problem and these steps were left out of the holographic application.

The instructions were followed a few times, and once clear, a set of drawings was made, based on how the application would be going to look (see Figure 29).
The goal for the application was to create a holographic guide that offers a good explanation of how to assemble the product. In step 1, 4, and 6, a screwdriver is used to screw or unscrew. To let the user know he should use a screwdriver, it should also be incorporated in the application and shown to the user. The screwdriver should also rotate in the right direction, clockwise or counterclockwise, to give the user extra feedback on what exact action he should perform.

1. Unscrew 4 screws with a TORX TX20 screwdriver and remove the top bracket.

2. Connect the 2 modules using a couple piece.

3. Place the coupled unit on the rails of the bottom bracket by hooking it behind the outer edge.

4. Screw in the 4 already present screws in the coupled unit using a flathead screwdriver 1.0 x 5.0.

5. Place the Memory Card in the slot on top of the coupled unit.

6. Screw in the top bracket using the 4 screws from step 1 using the TORX TX20 screwdriver.

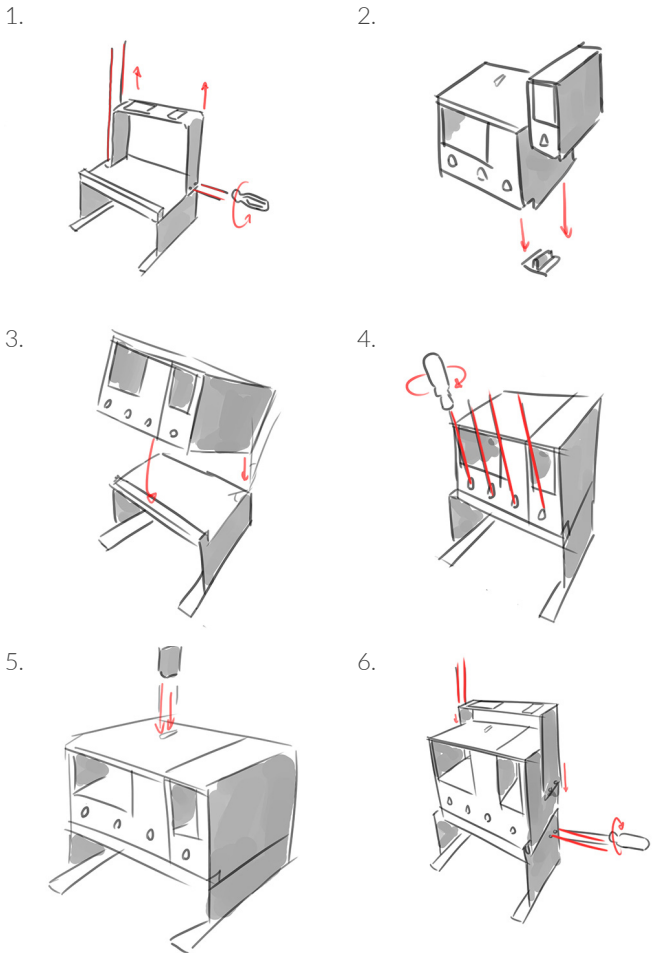
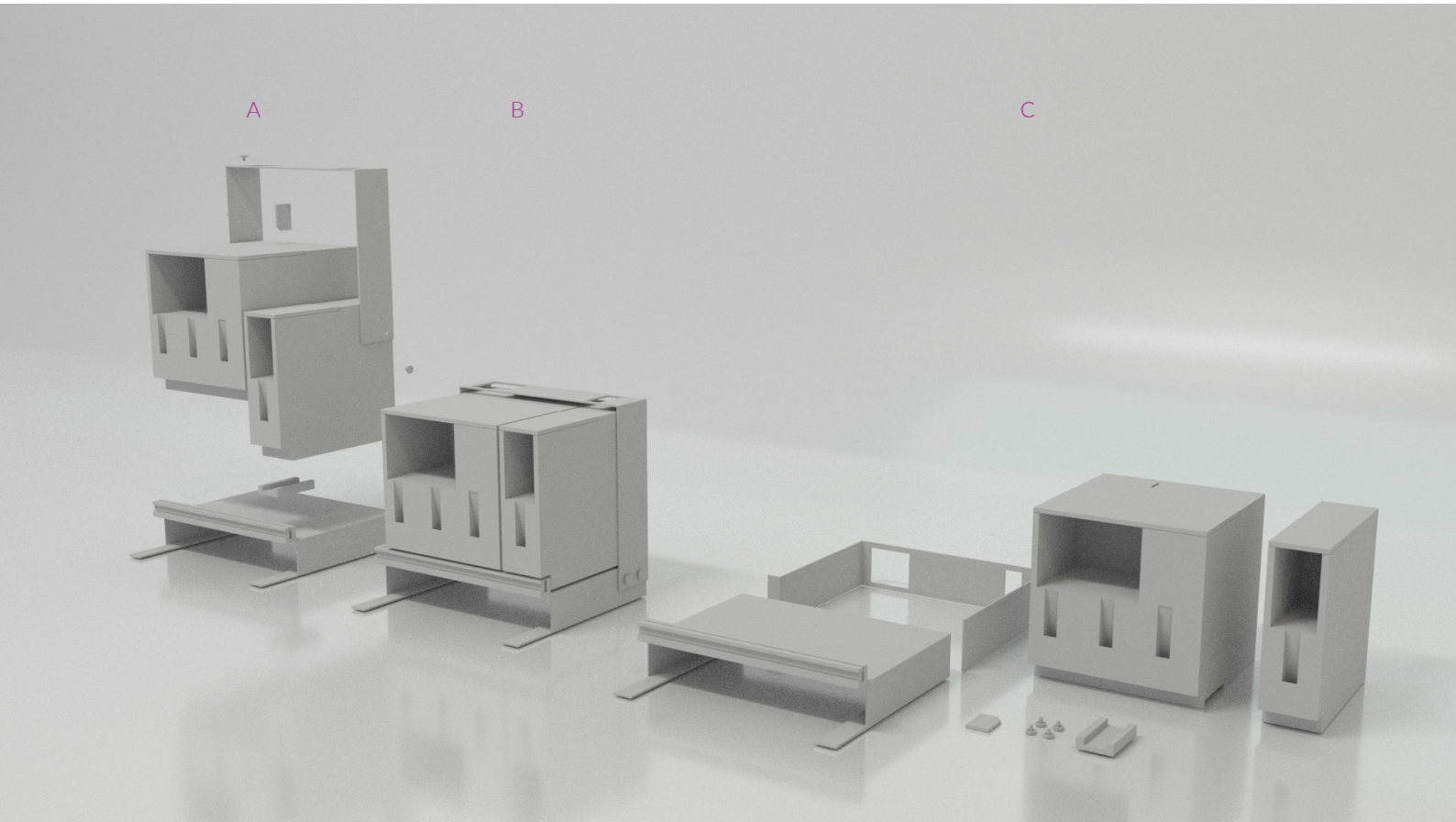


Figure 29. The steps to be created and animated in the application.

Figure 30. Renderings of the model.



The next step was to translate these rough steps into more approachable sub steps, that eventually can be programmed. As an example, step 1 is broken down into sub steps here. The full steps read
Unscrew 4 screws with a TORX TX20 screwdriver and remove the top bracket.
In sub steps this would mean the following:
1.1 Screwdriver moves from initial position to screw, indicating to put the screwdriver on the screw.
1.2 Screwdriver rotates for a few seconds in the right direction (counterclockwise).
1.3 Screwdriver moves back to its initial position, indicating that the screwdriver needs to be pulled out of the screw.
1.4 The bracket needs to move upwards, showing that it has to be removed.
1.5 Everything needs to go to its starting position to repeat the animation.

Note that a few things are not present in these sub steps; the choice was made to have the screwdriver only move to one screw (for the sake of only having to program one movement). The other screws are indicated with red wires. Removal of the screws is not animated, as removing the screws is a logical result of unscrewing.

The other steps were broken down likewise, where every sub step is translated into objects that visually move or rotate.

8.2.3 Unity

Building an application for the HL is done in Unity (Unity, 2016). Models can be loaded in and by writing scripts and attaching these to so called GameObjects (3D objects in Unity), the objects can be moved around. In Unity, the project needs to be set up as a holographic one, specifically for the HL. Practically this means that the application will try to run as fast as possible, ditching quality when performance is needed. It is important to keep a high frame rate in the application as users will eventually be looking at and moving through the application very close to their eyes. Low latency makes it easier on the eyes and can prevent motion sickness (Hettinger, 1992).

Unity C# scripts standardly have two methods, the Start() and the Update() method. Start() is used to initialize the application and runs only once, when the class is called for the first time (usually when starting the application). Update() is called every frame the application is running. Writing a function that for example handles the animations for step 3, cannot be called in either of these methods right away, as this would mean that it is either performed once, at the start of the application, or that step 3 is started every frame the application is running.
In other words, just using these functions is not possible for the way this animation is used. A framework was set up that handles the animations.

A variable integer StepNumber was created to handle animations. This integer can be edited by user input (speech or air-tap), and is fed into the update method. In the update method, the StepNumber is checked every frame and accordingly another method is called. In other words, as long as the StepNumer is 1, the function StepOne() will be performed, and the animations scripted for that step will be played. As the user changes StepNumber to 2, the update method will now call another function that handles the animations of second step.

Each step contains a set of sub steps that need to be carried out. The method for each step contains a SubStepNumber, that is used to create an animation in steps. The sub steps for step 1 have already been described, and the application handles them in the same order. By creating a sub step number, the objects can be moved independently and in a specific order (see Figure 31).

Before developing the application using the actual model, tryouts were made using simple cubes that moved from point to point, rotated, and disappeared, to check if the written code was actually working. Most of the testing was done in the integrated test area in Unity. Instead of human input (speech and airtap) mouse clicks were used. Once the cubes were moving according to how they should, the application was changed to work with the actual model and HL input.

Microsoft has created a set of HL specific scripts that take care of actions like spatial perception, and gesture recognition. These scripts are available to developers and have been used in this application.

For the full code see Appendix D.

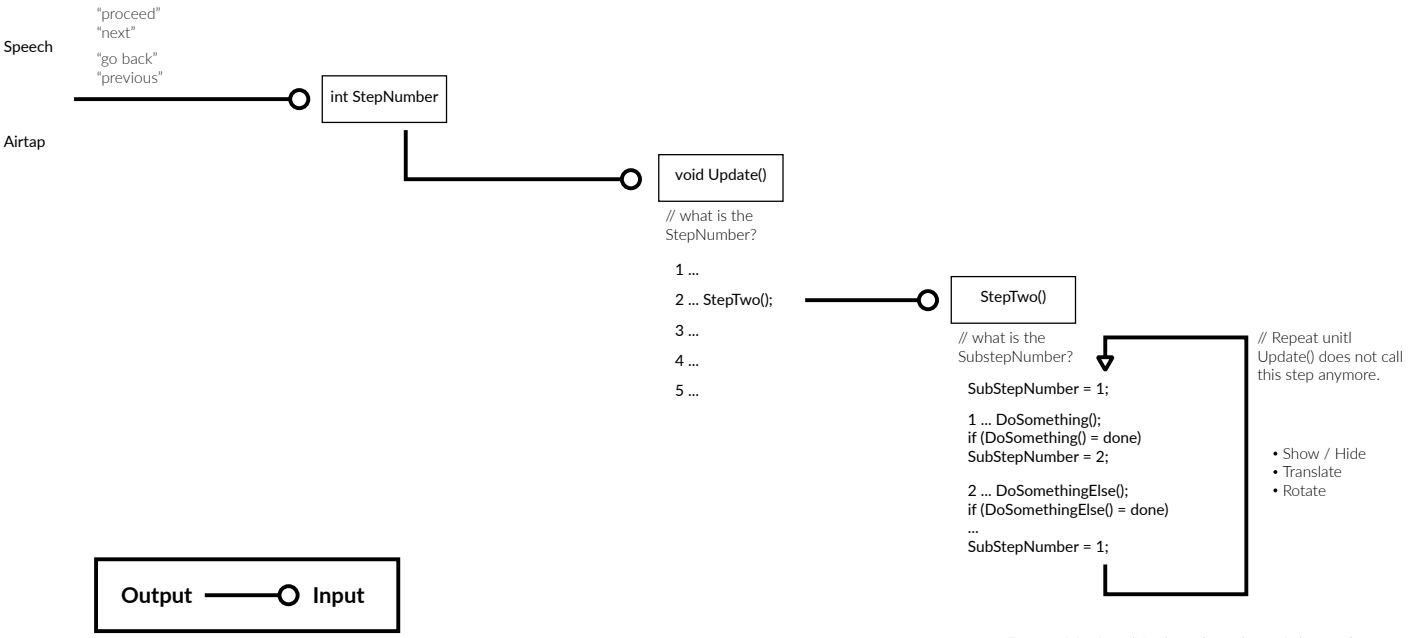


Figure 31. Graphical explanation of the code.

8.2.4 Deploying

A HL emulator running on a computer was used to test the application. The emulator runs a simulation of the HL and uses button presses for gazing input and air-taps. The computer microphone is used for speech recognition. After many modifications, the application was working properly. Screenshots of the actual application running on the emulator have been appended (see Figure 32).

Testing the application on the HL was done by sideloading the application over a USB connection to the device.

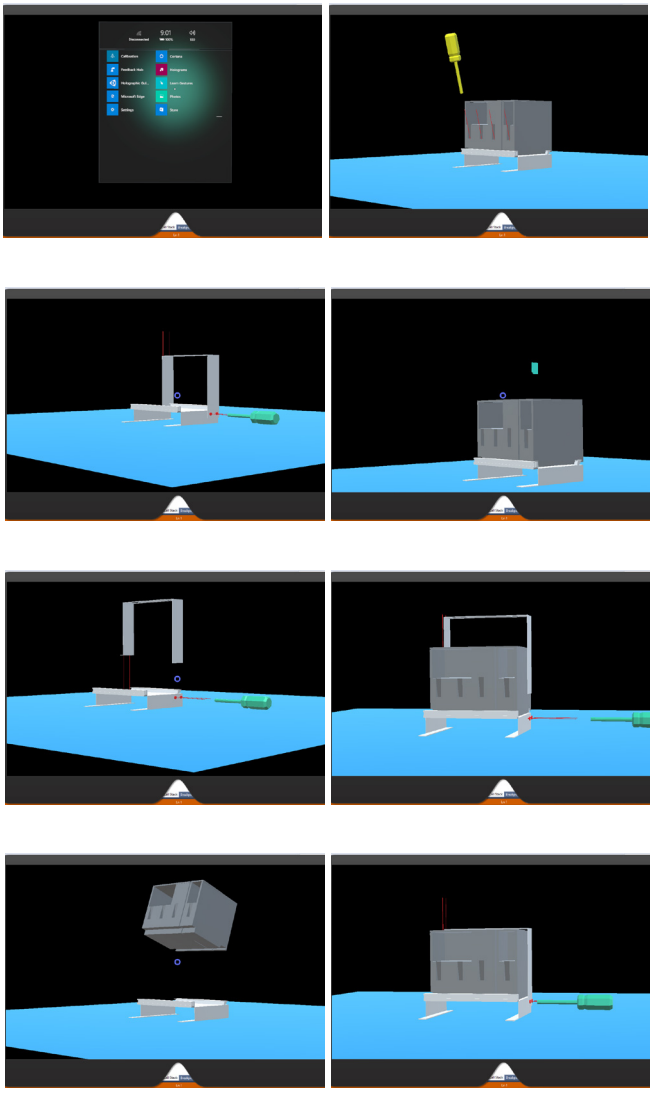


Figure 32. Screenshots of the application running on the HL emulator.

Initially deploying the application on the device did not work. An unknown error came forth while deploying the application. It turned out that the path before the build solution, the folder containing all the needed files to deploy the application, may only contain 23 characters and no spaces (Windows Holographic Developer Forum, 2016). In order to read the files, the maximum length of a path is 60, but most of it is already used in deeper folders required for running the application.

for example:

C:\Users\COMPANION\Documents\University of Twente\Bachelor Assignment\Holographic Guiding Prototype\Build

Contains both more than 23 characters and spaces, and gives an error when deploying. Placing the build solution in the root of the C directory solves this problem:

C:\HOLO\Build

After solving the error, the application was running fine. Some tests have shown that holographic guiding is an easy way of approaching the assembly process (see Figure 33). Feedback was used for some minor improvements, such as creating colored screwdrivers, based on the color of the actual screwdrivers used.

Figure 33. Someone succesfully assembling a product using the holographic guide.

9. Recommendations & future

Work needs to be done in order to create a workflow suitable for using the HL within the company. Specific programs need to be written that make the job of creating holographic guides easier. On the subject of CAD development, it is also recommended to find a way to make reviewing models on the HL easier. Now, a model needs to be converted into the right file format and uploaded to the HL manually. For some cases, such as showing a product to a client a few times over the course of a project, this is an alright approach. Integrating augmented reality into the company, should not limit the workflow by having employees constantly fiddle with the right models, file formats, or device limitations. These things should be automated.

As shown, a step in holographic assembly can be broken down into sub steps of three types, showing/hiding, moving, and rotating. It should be possible to create a user-friendly program that makes it easy to apply animations to steps. The model can be loaded in, and parts can be selected and given one of the possible animations.

Technology is constantly changing and improving. It is wise to keep up with the latest applications and possibilities of the HL. Apps can be improved by editing them with newer software and functions. This version of the HL has had criticism on some points. It might be the case that Microsoft is working on a second version of the HL already, that will have all sorts of improvements. It is recommended that Benchmark keeps an eye out for announcements of Microsoft revolving the HL.

Introducing a new technology into a work environment where specific rules and structure apply is a large operation. I recommend Benchmark to think about how much they are willing to invest in this technology. Based on that decide on where in the company they are going to integrate augmented reality technology.

It is important to keep in mind that not everyone is comfortable working with augmented reality or a device worn around the head. Some people might experience headaches from either having to cope with holograms and constantly shining bright light in their eyes or the device being too tight around the head. It is recommended to keep exploring ways to have normal work conditions and augmented reality co-exist in the Manufacturing Environment.

10. Conclusions

Researching the HL has allowed to get an insight in the possibilities and limits of the device. The HL is a new product, an emerging technology. The HL offers imposing techniques and results. However, compromises are present that do make the HL somewhat unpractical in some situations. At the time of writing, the product is still promoted as a developer product, not yet as a finished consumer product. The HL is an expensive tool, and therefore it should always be a consideration if the product is good enough now to invest in. Benchmark should really think about what their motive is to start using the HL. If the main goal would be to make more money, the HL might be an uncertain investment. The question arises if investing in the integration of HLs is a viable investment, as a second, improved version of the product could come out next year. If the goal is to innovate within the working environment, investing in the HL offers a look and step into the future and could offer an advantage of already being acquainted with applied augmented reality solutions. It could offer a long-term advantage of being ready when new devices find their way to the market and prove their usability. Integrating new emerging technologies is an expensive and risky investment, especially on a larger scale (for example when giving the whole Manufacturing crew a HL). The level of investment Benchmark is willing to make, also determines what possibilities are present for augmented reality.

Looking at the created concepts, we can conclude that the HL can be used in several aspects of the industry, in different ways, with different advantages to them. The way of integrating depends on to what extend a shift is made to using HLs. Due to the fact that augmented reality is a new technology for the manufacturing environment of Benchmark, it could be wise to integrate solutions step by step, for example following the concepts from low to high, letting the company structure and employees get used to having HLs around to work with.

Investing in the technology alone is not enough. As noticed during this assignment, there is more to it than just buying a HL. It costs time and research to get to understand the device. There is no extensive application collection yet to choose some usable applications from to use. The HL is a developers device and so, a developer is needed to create applications for it. Designing an augmented reality application also requires some thinking in 3D in terms of animation and interaction.

Creating an application for the HL is done through Unity, which has proven not to be the most straightforward method. Streamlining the workflow would be a necessity if more of these guiding applications will be made. Ideally a program is created where, using a graphical user interface, the guide is created. This also goes for other ways of integrating the HL in an industrial environment. The workflow is disrupted quite a bit when someone has to convert a model or write a whole application before being able to use the HL.

Emerging technologies can be so new, that while developing, new features come out. Over the course of this bachelor assignment, integration of the HL went from a unity SDK add-on to fully integrated into the software. A new tracking system for augmented reality has started collaborating with Microsoft, called Vuforia. Some of the initial ideas included tracking objects, however, the HL would only be able to track surroundings. With the collaboration between Microsoft and Vuforia, tracking software for specific objects is now available to the HL.

At the moment, the HL asks for too much investment in development to be a usable solution in the company. Creating an application demo has shown what is involved in making something work on the device. Benchmark has its core business somewhere else, and therefore it is improbable that deep integration will ever be built by Benchmark itself. However, this report has also shown that the device can be used in a variety of other situations where less development might be needed.

11. References

1. Azuma, R. T. (1997). A survey of augmented reality. *Presence: Teleoperators and virtual environments*, 6(4), 355-385.
2. Benchmark (n.d.) Benchmark. Retrieved from: http://www.bench.com/
3. Bostrom, N. (2003). Are we living in a computer simulation?. *The Philosophical Quarterly*, 53(211), 243-255.
4. CPI (n.d.) The Windowless Fuselage. Retrieved from: http://www.uk-cpi.com/windowless-fuselage/#.WDQf4ubhBhH
5. Dubois, E., & Nigay, L. (2000, April). Augmented reality: which augmentation for which reality?. In *Proceedings of DARE 2000 on Designing augmented reality environments* (pp. 165-166). ACM.
6. Erickson, S. (19 November 2015) Microsoft HoloLens and Volvo Cars explore the future of car buying. Retrieved from: https://blogs.windows.com/devices/2015/11/19/microsoft-HoloLens-and-volvo-cars-explore-the-future-of-car-buying/#riTOOUlBgzwpm5Gg.97
7. Falk, J., Redström, J., & Björk, S. (1999, September). Amplifying reality. In *International Symposium on Handheld and Ubiquitous Computing* (pp. 274-279). Springer Berlin Heidelberg.
8. Falk, J., & Björk, S. (1999, May). The BubbleBadge: a wearable public display. In *CHI'99 extended abstracts on Human factors in computing systems* (pp. 318-319). ACM.
9. Ford, P. (2 October 2009) Diminished Reality. Retrieved from http://paulford.com/diminished-reality/#more-5
10. Furht, B. (Ed.). (2011). *Handbook of augmented reality*. Springer Science & Business Media.
11. Glasgow Multimodal Interaction Group (2014). Augmented Virtuality Prototypes. [Youtube video]. Retrieved from: https://www.youtube.com/watch?v=fEiyzJDFiJI
12. Google (15 November 2016) Now your photos look better than ever – even those dusty old prints. Retrieved from: https://blog.google/products/photos/now-your-photos-look-better-ever-even-those-dusty-old-prints/
13. Hettinger, L. J., & Riccio, G. E. (1992). Visually induced motion sickness in virtual environments. Presence: *Teleoperators & Virtual Environments*, 1(3), 306-310.
14. I3D Past Projects (25 March 2016). holoportation: virtual 3D teleportation in real-time (Microsoft Research). Retrieved from: https://www.youtube.com/watch?v=7d59O6cfaM0
15. Iwatani, Y. (1998). Love: Japanese Style. in Wired. Retrieved from: http://archive.wired.com/culture/lifestyle/news/1998/06/12899
16. Kipman, A. (February 2016). A futuristic vision of the age of holograms [Tedx presentation]. Received from: https://www.ted.com/talks/alex_kipman_the_dawn_of_the_age_of_holograms
17. Mann, S., & Nnlf, S. M. (1994). Mediated reality.
18. Microsoft (retrieved on 28 November 2016) Developing for the Microsoft HoloLens. Retrieved from https://www.microsoft.com/microsoft-HoloLens/en-us/developers
19. Microsoft (retrieved on 28 November 2016) Holoportation - Microsoft Research. Retrieved from: https://www.microsoft.com/en-us/research/project/holoportation-3/
20. Microsoft (2016). Microsoft HoloLens | Hardware. Retrieved from: https://www.microsoft.com/microsoft-HoloLens/en-us/hardware
21. Microsoft (2016) Gestures. Retrieved from: https://developer.microsoft.com/en-us/windows/holographic/gestures
22. Microsoft (2016). Gaze. Retrieved on 6 December 2016 from: https://developer.microsoft.com/en-us/windows/holographic/gaze
23. Microsoft (2016). Introducing the Microsoft HoloLens Development Edition. Retrieved on December 4 2016 from: https://www.microsoft.com/microsoft-HoloLens/en-us/development-edition
24. Microsoft (2016). Order options | HoloLens. (n.d.) Retrieved from: https://www.microsoft.com/microsoft-HoloLens/en-us/order-now
25. Microsoft HoloLens (19 November 2015). Microsoft HoloLens: Partner Spotlight with Volvo Cars. Retrieved from https://www.youtube.com/watch?v=DilzwF90vec
26. Microsoft HoloLens (29 February 2016). Microsoft HoloLens: Gesture Input. Retrieved from: https://www.youtube.com/watch?v=kwn9Lh0E_vU

27. Microsoft Research (23 November 2016) Mobile Holoportation. Retrieved from: https://www.youtube.com/watch?v=nTkFO2xNkIk&feature=youtu.be
28. Milgram, P., & Kishino, F. (1994). A taxonomy of mixed reality visual displays. *IEICE TRANSACTIONS on Information and Systems*, 77(12), 1321-1329.
29. Milgram, P., & Colquhoun, H. (1999). A taxonomy of real and virtual world display integration. *Mixed reality: Merging real and virtual worlds*, *1*, 1-26.
30. Moravec, H. (1999). Simulation, consciousness, existence. *Intercommunication 28:*98-112 (1999)
31. NASA Jet Propulsion Lab (30 March 2016). Mixed-Reality Tech Brings Mars to Earth. Retrieved from https://www.youtube.com/watch?list=PLTiv_XWHnOZpzQKYC6nLf6M9AuBbng_O8&v=wPOCcG33mJQ
32. Ranasinghe, R.A.N. (2012). Digitally stimulating the sensation of taste through electrical and thermal stimulation. In: *ISWC '12 Proceedings of the 2012 16th Annual International Symposium on Wearable Computers*. DOI 10.1109/ISWC.2012.2
33. Robertson, A. (1 March 2016). Microsoft's HoloLens is new, improved, and still has big problems. Retrieved from http://www.theverge.com/2015/5/1/8527645/microsoft-HoloLens-build-2015-augmented-reality-headset
34. Schnabel, M. A., Wang, X., Seichter, H., & Kvan, T. (2007). From virtuality to reality and back. *Proceedings of the International Association of Societies of Design Research*, *1*, 15.
35. Unity (5 July 2016). Unite Europ 2016 - Real World HoloLens Mixed Reality Development with Unity Be Part of the Future. Retrieved from: https://www.youtube.com/watch?v=JbsmKVSSLM8
36. The Virtualization Studio - Research. Retrieved from http://www.cs.cmu.edu/~virtualized-reality/page_Research.html
37. Vroegop, D. (2016) Cimsolutions presentation about HoloLens.
38. Warren, T. (18 June 2016). Microsoft: don't expect HoloLens' field of view to get much better. Retrieved from: http://www.theverge.com/2015/6/18/8809323/microsoft-HoloLens-field-of-view-kudo-tsunoda
39. Windows Holographic Developer Forum (Edited August 2016) Cannot build and deploy to HoloLens anymore, SerializationWeaver input line too long. Retrieved from: https://forums.hololens.com/discussion/1824/cannot-build-and-deploy-to-hololens-anymore-serializationweaver-input-line-too-long

Appendix A
Research on mixed reality

Introduction

Between reality and virtuality there is a whole range of ways of visualising objects. It can be confusing when a certain term should be used and what a certain term actually means. Therefore, a clear taxonomy is needed for the description of everything from reality to virtuality.

For starters, a clear description is needed for what is real and what is virtual. Categorizing these terms can be difficult, since the observations we make of both worlds, can resemble each other. Using computers we are able to create physically accurate renderings of all sorts of objects, resulting in an unclear distinction of observations of what is real and what is virtual. Something that is real and something that seems real, can be totally different objects, since a represented object can be a virtual one. A picture of an object taken with a camera, represents a real object. A picture of an object made with a computer rendering technique, also resembles a real object, however only the first one is actually representing the real object. Both observations of said picture, will however look alike and it can be hard, or either impossible, for the viewer to tell which one is the actual real object.

The distinction between virtuality and reality is therefore not arising from the possible observations of an object, but rather from the set of properties it has. Milgram and Kishino (1994, p.1324) describe the distinction between the two sorts as follows;

"**Real objects are any objects that have an actual objective existence.**
Virtual objects are objects that exist in essence or effect, but not formally or actually."

Furthermore, they describe a taxonomy for observing real and virtual objects as follows:
Direct viewing is a direct observation of an object without the interference of a certain medium, with the exception of transparent mediums that can not alter the real properties of the object (seeing an object through air or glass).
Indirect viewing is the observation of an object with interference of a certain medium. Since virtual objects by definition do not actually exist in reality, they can only be viewed indirectly, by simulating them, making them appear through a certain medium. Indirect viewing of real objects can be done by sampling them, using some sort of apparatus e.g. a camera.

Finally, a distinction is made between real and virtual images as follows:

Real images have a certain luminosity at the location they appear to be[1]. **Virtual images** on the contrary, do not have luminosity at the location they appear to be in. Doing so, they are able to make a distinction between an photograph of an object (real image) and a stereoscopic image from two photographs, resulting in an 3D image. This 3D image is situated in the real world environment, and therefore the luminosity is incorrect with its surrounding (virtual image).

Between real and virtual, there is a continuum that mixes parts of both reality with virtuality.
Milgram and Kishino (1994) use the Virtuality Continuum to describe the spectrum of mixed reality (see Figure A1).

To get a grip on this broad term of mixed reality, research was carried out. Besides the terms Milgram and Kishino set, other forms of mixed reality were found, that are researched.
Based on the found results, a new continuum was made, including more in-depth forms of mixed reality. In the continuum short descriptions of each form are adopted and some example applications are adapted (see Figure A2).

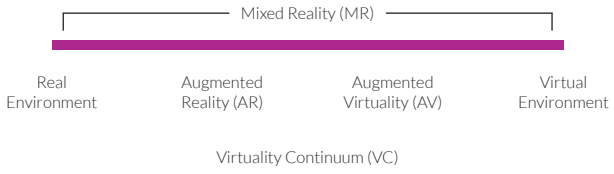


Figure A1. Milgram and Kishino's Virtuality Continuum.

1 Milgram and Kishino do not note the theoretical possibility that a certain real environment can be scanned, and the luminosity from said surrounding could be projected on a simulated virtual object, resulting in the definition of a real image. This would however still be a virtual image.

Reality Virtuality continuum

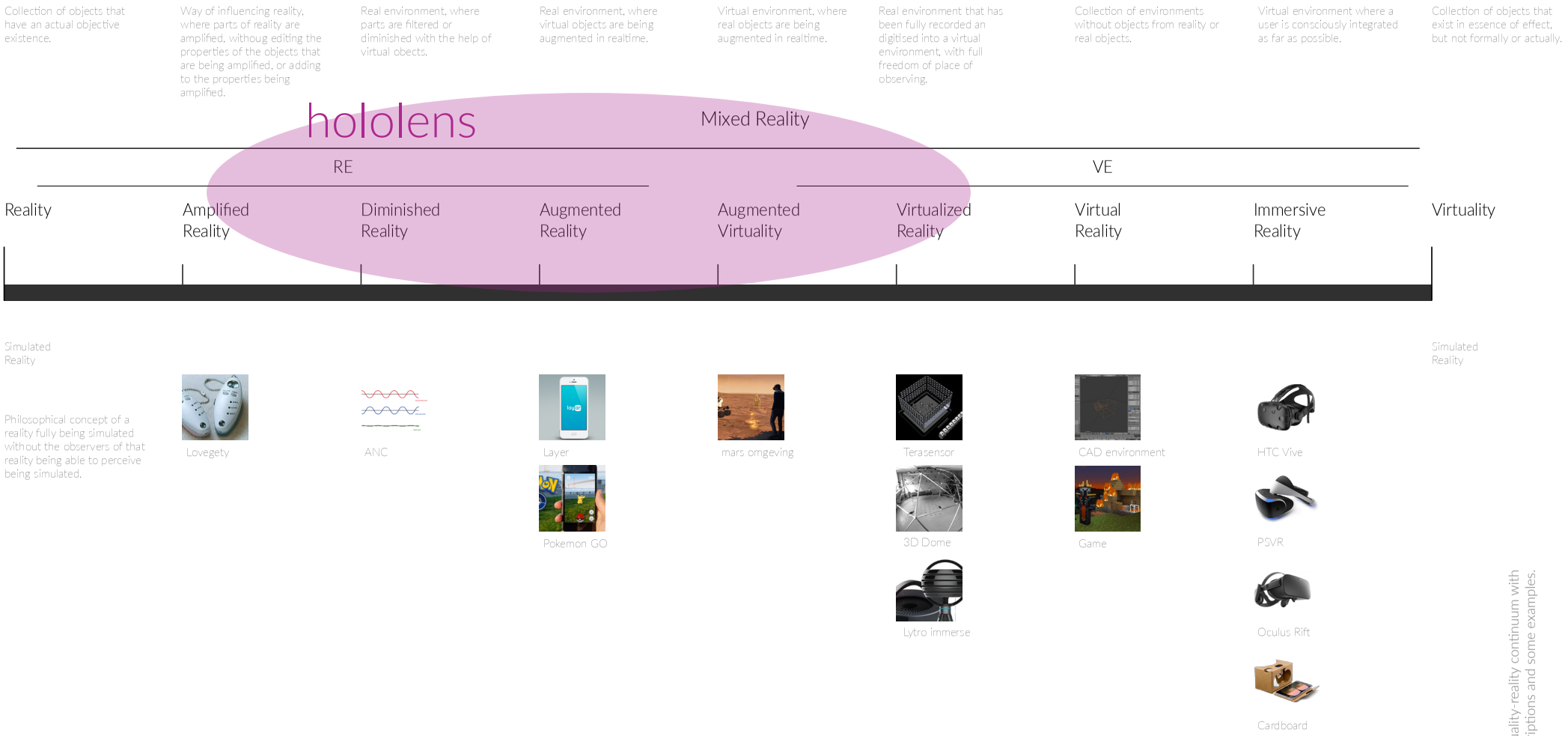


Figure A2. Virtuality-reality continuum with descriptions and some examples.

Applications

Reality

Milgram and Kishino (1994, p.1324) determine real objects with the following;

“Real objects are any objects that have an actual objective existence.”

Since this determination follows from certain set of properties of an object that have to be known by the viewer, it can be difficult to explain reality.

Observations in reality are made through the senses. Over time we have learned how our senses work and since we know how they process input, we can generate our own input.

Sight is formed by photons that hit objects and reflect into the eye. In the eye cells generate impulses based on the incoming light and the brain develops an image. But there is more to it. We can see in three dimensions and estimate distances between objects. There are several ways we unconsciously use to perceive our surroundings.

1. The difference between the left and right eye. When looking at a certain object, the distance between the left and right eye result in a slightly different view of the object.
2. Recognition of objects and their relative size. Objects that are close to the viewer, appear larger. We use this illusion to compare sizes of objects we know with the size we remember them to be, to be able to estimate how far away they are. Arrangement of the objects also helps with this (close objects appear in front).
3. Focus and defocus of the eye. Some parts of the field of vision appear blurry, when the viewer focuses on something closer or further away. This helps estimate if things are at about the same distance from the viewer.
4. Relative movement of objects close and far. When moving the point of view, object that are close to the viewer, will appear to move relatively more than objects that are far away. As the point of view is always a bit shaky (the brain does cover up for this), unconsciously the observations of relative movement help a lot to estimate depth.

The combination of these (subconscious) observations, enables us to have a great understanding of the world around us. As we rely heavily on these simple principles for our vision, it is easy to create our own input 3D input, for example using stereoscopic displays (see Figure A3).

Hearing is based on sound waves, vibrations in a gaseous or liquid medium, that are catched by the inner ear. In the inner ear, vibrations are turned into impulses that the brain processes to hear sounds. Determining the source of sound is possible by comparing when either left and right ear hear the same sound. By creating sound emitting devices that have a small difference in when they play the same sound for left and right ear, we can fool the ears in hearing sound from a certain direction (see Figure A4).

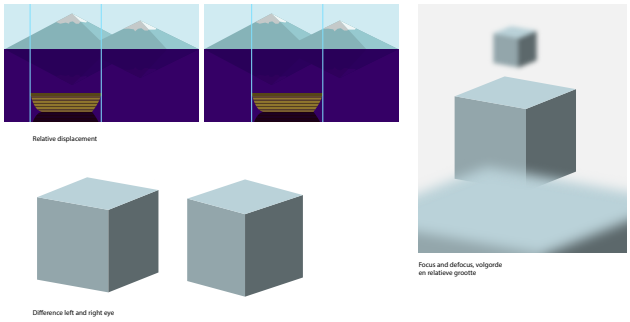


Figure A3. Depiction of how to estimate depth.

Taste and smell work in a similar way. These senses create impulses when a certain chemical binds with receptors. We cannot really fake smell (yet), but according to Ranasinghe (2012) taste can be created using electrical surges on the tongue, combined with heat dissipating foil.

Touch is sensed by deformations of (parts of) the body by a certain pressure external to that part of the body (see Figure A5). Deformation of the skin and hairs on the skin send impulses to the brain that conclude something is being touched on that specific spot. We cannot really fool ourselves to being touched, as simulating touch would actually done by forcing pressure on the body, however we can kind of simulate the reason for touch. Haptic feedback is an example, used in touch keyboards on smartphones. The vibrations when pressing a button, resemble the touch of pressing real buttons.

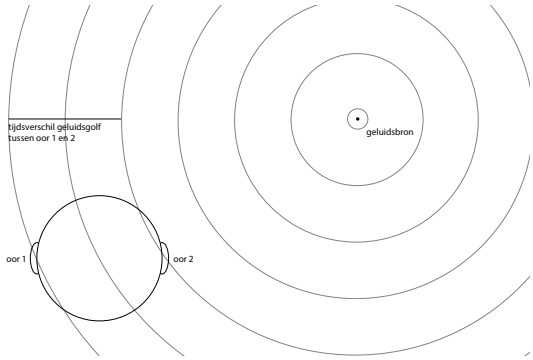


Figure A4: Depiction of how to estimate the source of a sound.

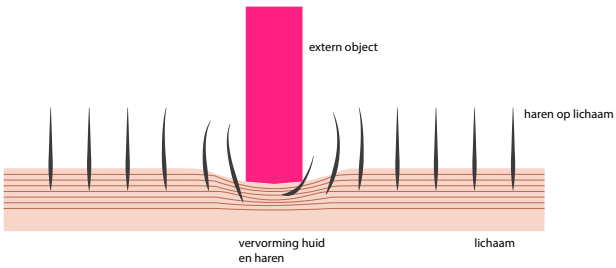


Figure A5: Depiction of how to sense.

Virtuality

According to Milgram and Kishino (1994), Virtual objects are objects that exist in essence or effect, but not formally or actually.

Since virtual objects do actually not exist, a certain medium is needed to allow a viewer to see a virtual object. In essence, a virtual object exists out of a set of virtual properties in a virtual environment. These virtual properties describe the object, but since they are just virtual properties, they can not really be observed. Simulating these properties allows a virtual object to - indirectly - be observed (see Figure A6).

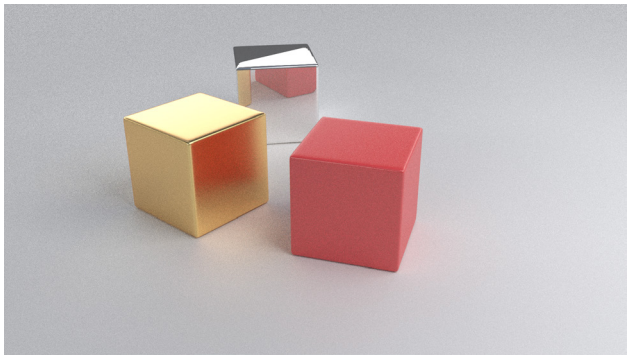


Figure 6: Rendering of 3 cubes with different properties.

The virtual environment comprises a global coordinate system, specifying the relative locations of all objects. Each object has its own specific properties; a local coordinate system, encompassing length, height, and depth of the object, reflective behaviour, and color.
This virtual environment can be simulated and rendered to make it possible to observe it.
An observation of a virtual object is for example possible through a simulation of an image.

As stated, a direct observation of total virtuality is not possible. Every observation a human can make, always goes via the senses, indicating some form of reality. All observations made through the senses, are based on physical properties of reality. To observe, for example, a virtual cube, first a calculation must be made for all of its properties, which have to be rendered. The render, in turn, has to be made visible through another medium, such as a display. This is always an indirect way of viewing the object.

Theoretically, an observation in virtuality is possible, but all senses and mediums should be excluded and the brain should have a direct connection to the virtual world to directly process virtual impulses.

Simulated Reality

When looking at the amount of realism we can already accomplish using simulated objects on, for example, displays, it could well be possible that at a certain moment in time, the distinction between real and virtual is not noticeable anymore. Combine this with a possible way to do direct observations in the virtual world and it might be impossible to know whether the viewer is actually in the virtual or the real world (Bostrom, 2003). If a person, for what reason so ever, forgets that he is actually in the virtual world, he may start believing that that is his reality. There is no reason anymore for the viewer to believe in a world other than his virtual one, and there is no reason to believe that this virtual world is not the real world. Simulating an entire truth like that, is called Simulated Reality.
Lots of books and movies incorporate this idea of simulated reality. In such movies the brain is often connected to some sort of computer system that creates impulses. One of the most famous movies to show this, is the Matrix. In the Matrix all of humanity is put in small spaces, suspended in some liquid. Their brains are connected to machines using pins in their head and back. The machine simulates another reality, where all of humanity lives consciously.
According to several philosophers the concept of simulated

reality could be rather plausible. Bostrom (2003) was one of the first to explain this, but has since gotten support from several other philosophers and scientists. The explanation is as follows;

Looking at the past, humans have been growing technologically. As we can see, technological growth has not yet stopped, and one can expect with reasonable certainty that humans will keep growing technologically, to reach a certain point where computer simulations are powerful enough to simulate large parts of or even whole realities.

Now, at least one of the following is very probable to be true;

1. Humans will never reach a point where these kind of simulation can be made. Technological growth will have to cease at a certain point to prevent us from creating these simulations. A reason for this could be the scenario where all technology is destroyed, or where humanity is exterminated. Another reason could be the physical impossibility of technological growth at a certain point.
2. Humans will sometime be able to create a large amount of simulations of realities, but due to reasons they do not want to. Reasons could be the cost of computing power, ethical reasons, or just a lack of interest in simulating a certain reality.
3. Humans will sometime be able to create large amount of simulations of realities, and will do so. Following on that, a whole simulated reality, will in turn be able to create simulations of realities, creating an infinite amount of simulated realities. Given that there is an infinite amount of simulated realities, it is very improbable that our reality is the first one to create a simulation and therefore, we are probably living inside a simulation of reality (see Figure A7).

However, as stated before, one would not be able to determine whether he is in a simulation or in reality. This makes it a speculative discussion, as we will probably never know what reality is.

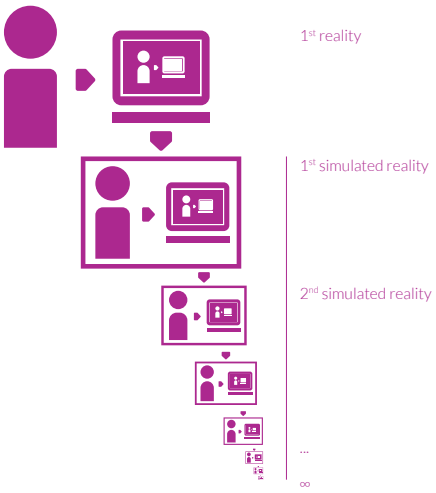


Figure A7: Graphic depiction of simulating realities

Mixed Reality

Between the two extremes of the spectrum, there is the collection of observable reality adjustments, that mixes real and virtual objects, call Mixed Reality (MR). In MR there is always some form of real objects and some form of virtual objects. A certain medium links both worlds to each other. Such mediums can lead the observations, as happens in immersive reality or offer more supportive observation adaptations, as happens in amplified reality. In between, there are several gradations in how reality and virtuality are mixed and what the total output is.

MR uses tools to influence the senses, fooling them in sensory observations. People have a strong sense of empathy and can easily be okay with something looking or sounding kinda-real. Have a look at children, who see all sorts of scary things at night. Something resembling the slightest form of a human figure (a chair with clothes on it), makes them assume there actually is a real person sitting in their room. This empathetic power is the key to mixed reality applications. Looking at something that kinda resembles anything recognisable can often be enough to have a viewer believe that he is looking at a thing instead of for example a group of pixels. The peculiar thing is, that even if the viewer knows he is looking at a group of pixels, he is still believing in what the group of pixels represents and can be emotionally, though irrationally, moved.

Practically, convincing observations of virtual objects in MR arise from the combination of several senses. For locating a certain object, sound can help to put the eyes in the right direction, so combining sound with sight is a great way to find an object in space.

Amplified Reality

Amplified Reality amplifies certain observable properties of real physical objects in reality using computers. According to Falk, Redström & Björk (1999), one of the most important distinguishable aspects of amplified reality, is the fact that the result should be publicly observable (in contrast to, augmented reality (see augmented reality), which is more user-centered). Furthermore, only initially principally

observable properties of objects are amplified and no virtual objects will be created that would initially not be observable.

Examples of amplified reality are projects like the Lovegety, a product that publicly emits a certain desire of a person to participate in a certain activity (Iwatini, 1998). Note that the initial state of the person already incorporated the desire to participate in an activity and the product just emitted this publicly to other owners of the product. Another example is the BubbleBadge, a wearable computer with display that shows certain characteristics of the wearer to people looking at the wearer (Falk & Björk, 1997).

Mediated Reality

Mediated Reality is officially the umbrella for observations made in reality, that are somehow mediated using virtual objects and computers.

There are two main ways of mediating reality:

1. Adding objects to the observations of reality, which is also and commonly known as Augmented Reality
2. Removing objects from the observations of reality, which is also known as Diminished Reality

Of course an application using both augmented and diminished reality is also possible. Both will be described separately in the following chapters.

An important aspect of mediated reality, is that it mediates the observations of reality (Mann, 1994). As the viewer observes, his observations are mediated in realtime.

Diminished Reality

In Diminished Reality, certain parts of the real world are filtered and reduced or removed for the observer. Certain parts of his observation are less available, while his focus remains with his real surrounding. A reason for this can be to protect private information in some way.

Another application can be diminishing some interfering factors from the surrounding. A user can focus on an object, but due to external factors, he is less able to do so. Filtering out (some of) the interfering external factors can be a way to gain more focus for the user.

Active noise cancelling headphones use this principle. The headphones actively monitor the noise coming from the surrounding and create a soundwave opposite in phase to the monitored noise. As both soundwaves are exactly each other's opposite, they cancel out, and the noise is significantly reduced. The result is that the user could use a lower volume on his headphones or have less distraction from his surrounding (see Figure A8).

Another example is a technique used to filter visual imperfections. Old photographs often are printed on very glossy paper. All sorts of reflective areas from the environment are visible when looking at the photo, especially light reflections are constructing the view. Using a technique that takes several samples of the photograph, the visual imperfections can be arranged on a different part of the photograph each sample (Google, 2016) The samples that do not show the imperfections can digitally be combined into one perfect picture (see Figure A9).

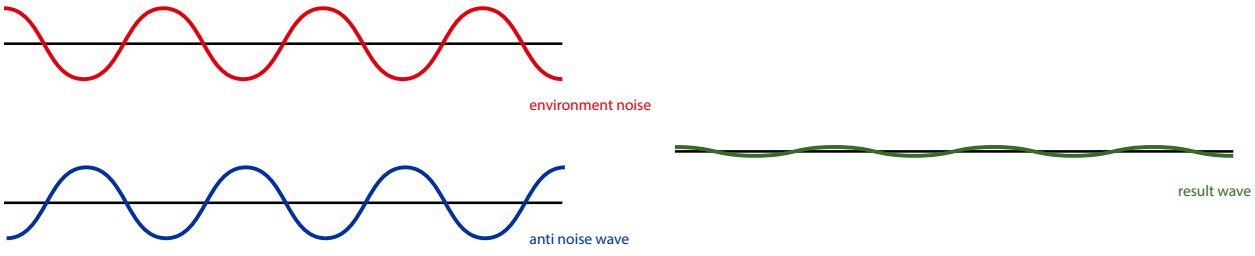


Figure A8: Example of the working of active noise reduction.

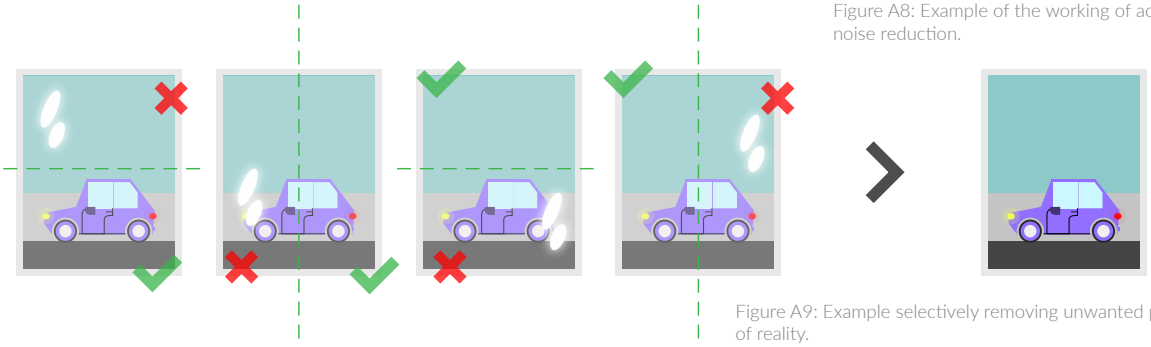


Figure A9: Example selectively removing unwanted parts of reality.

While this technique is actually not used in realtime, it could easily be extrapolated as a diminished reality application. Likewise application exist for diminishing repetitive patterns, as wire fences, as long as enough samples are made.

Basically there are two ways of diminished reality applications. The first one relies on pattern recognition and prediction of the pattern. This is the case with noise cancelling, as the headphones should predict how the noise will be filtered at the same moment it sensors it. This future-driven way of diminishing is less reliable since the technology should predict what it should fill in. Especially abrupt changes in the surrounding are hard to deal with.

The second one constantly samples parts of the surrounding and combines those to diminish a certain object. This is the case with the visual imperfections filtering technique. The technique relies on samples taken in the past. This past-driven works more reliable since the technology does not have to 'create' something new. This technique, however, is only usable if it is clear what object to diminish.

Augmented Reality

Augmented Reality (AR) comprises looking directly or indirectly in realtime at the fysical, real world, to which virtual information has been added into the context of the environment. These additive factors can, for example, give extra information about the environment or simulate extra objects. This way, a user can be assisted in the tasks he is doing.

According to Azuma (1997), AR systems have the following characteristics:

1. Combine real and virtual
2. Interactive in realtime
3. Registered in 3D[1]

Adding, or augmenting, objects to reality is a fairly broad term, and therefore, a broad variety in technological levels is present within AR. Carmigniani and Fuhrt make an extensive description of the AR field (Fuhrt et al, 2011). Four sorts of devices are used to create an AR experience: display, input, tracking, and computing.

Displays are used to view the environment and to augment objects onto. Displays can either be video see-through, displaying a camera feed onto a display, or optical see-through, using transparent material to project on. Different types of displays are usable for AR. Head-mounted displays (HMD) are strapped to the viewer's head and show information directly in front of the eyes of the user. Handheld displays are small displays, such as smartphones, that are capable of using video see-through to augment objects.Spatial displays are displays that do not need to be held by the viewer.

Since AR is interactive in realtime and registered in 3D, the environment has to be tracked in some way, to augment registered objects. Basically there are two ways of tracking in relation to the surrounding environment, defined and undefined. If the environment to augment in is defined, the system is looking for defined features of the environment that are recognisable. How specific the features are defined depends on the application. Very specifically defined features look for an exact object in the environment (see Figure A10). Here we can see that the exact image is needed in order for the system to know that it should augment something. Less specifically defined constraints are present in for example Snapchat, where the application augments certain virtual features on a human face. The face is recognised based on some abstracted features, namely the presence of some contrast where mouth and eyes should be (see Figure A11).

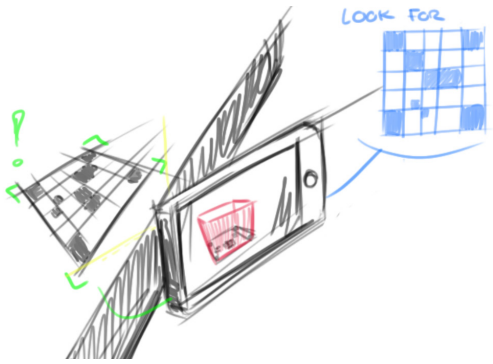


Figure A10. Defined environment, specifically: layer.

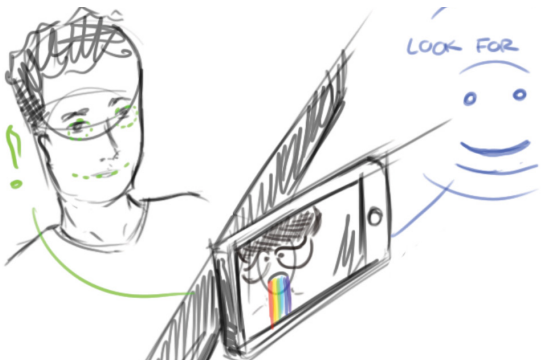


Figure A11. Defined environment, unspecifically.

1 With the third statement, Azuma excludes all systems that make use of 2D overlays, such as Google Glass. One could argue whether such systems should be included or excluded from the term AR.

As the systems already knows some defined constraints of the objects it is looking for, it can augment based on the aspects it finds. In example 1, the system knows how the image should look in 2D, and using recognition and distortion of the object in 3D, it is able to create a 3D space.

Augmented virtuality

Augmented Virtuality (AV) comprises adding elements from the real world to a virtual environment. Milgram and Colquhoun Jr. describe augmented virtuality in detail and add some examples (1999). In AV, a virtual environment is enhanced using real world objects, or real images.

AV can, for example be found in a lot of video games, where textures, real images of certain materials, are used to specify a certain object. Instead of modelling all properties of the object, only the dimensions are modelled. Using a texture, can save both a lot of time and computing power, as the system does not have to calculate how that specific material behaves.

Another example can be found in immersive reality (see immersive reality for explanation), where due to the covering of the eyes, the real world cannot be observed anymore. Glasgow Multimodal Interaction Group (2014) has made an application, where the real objects are augmented in the virtual environment to help the user, for example, locate his keyboard to then use it as an input device.

Virtual Reality

Virtual reality is the umbrella for all environments consisting fully out of virtual objects.

Virtual Reality is often used for all sorts of HMDs that place the viewer inside some sort virtual environment (VE). Although these HMDs indeed place the user inside a VE, which is a form of VR, the specific term used for these goggles, is immersive reality (see immersive reality). VR is a broader term, comprising all sorts of fully virtual environments.
Virtual reality can alsof be used in terms of, for example a game or a CAD-environment, where all objects are indeed virtual.

Virtualized Reality

Virtualized Reality is the term for fully virtual environments that arise from a virtualized real environment using scanning and processing techniques.
Virtualizing a real environment can be done using cameras that scan and digitise the environment. Processing the data allows for the creation of a virtual environment. In the virtualized reality, the viewer can take on any point of view.

Events in virtualized reality have to be recorded from every angle. As the viewer in a virtualized environment is able to take on any point of view, this should be the same when an event is happening. A visual event should therefore be captured with multiple cameras, that are able to record the event from every angle and postprocess it in 3D.

All events in virtualized reality appear from real events, happening in realtime or that have already happened and have been recorded and stored.

Practically, creating virtualized reality applications can be a rather difficult process. Due to the large amount of control needed over the real environment, it is hard to record events. The Robotics Institute has been working on a large virtualized reality sensor. Realization of this sensor should be accomplished using about 1000 cameras and 200 microphones to collect data within the sensor. Once finished, the sensor should produce about 5.7 terabyte of data per second (The Virtualization Studion, n.d.).

Immersive Reality

The goal of immersive reality is to create an experience that integrates the viewer as much as possible into the virtual environment, so that he will be most conscious in the VE, or that the experience created, will be as real as possible.

To create immersive experiences goggles are often used that create a stereoscopic set of images in front of the user. Accompanied by sound, this is often enough to get the viewer immersed into the virtual environment. Tracking the rotation and place of the head and translating this into the position of the camera in the virtual space, enlarges the belief in the reality of the experience.

The most common technique for creating immersive reality uses small display screens that are placed in front of the viewer's eyes. Using lenses, the image is made visible at this short distance from the eyes. Although development of

immersive reality sets has improved over the past years, the experience still has some disadvantages.

Immersive reality sets on the high-end of the spectrum, can deliver a convincing experience of the VE, especially when the user's movement in 3D is being tracked. Besides rotation of the head, some sets also track the position of the viewer in space within a certain area. Sight is one of the most important senses, and therefore, immersive reality is a convincing and overwhelming experience.
However, due to the fact that immersive reality is such an overwhelming experience, the flaws in quality can start to stand out. The amount of pixels might be large, but by magnifying them close to the eyes, they can become easily visible.
When rotation and movement of the user is being tracked, this amplifies the experience. However, movement has to be understood by the software and processed into visible movements for the viewer. As the viewer is fully immersed into a virtual environment, every movement that has a form of latency can cause nausea. The impulses from the vestibular and the eyes do not correspond to each other. The maximum latency between movement of the viewer and the moment the display shows this movement, should be shorter than the viewer can notice.

Three examples of immersive reality in low-end, medium-end, and high-end that are momentarily available for purchase:

Google Cardboard uses the fact that almost every person has a personal smartphone, capable of generating 3D environments and tracking rotation. A cardboard viewer with two plastic lenses is used to place the smartphone in. Samsung Gear VR makes use of the same principle of putting a smartphone in a viewer, but their viewer has build in straps to wear the pair of goggles and some extra sensor for tracking rotation of the user.
On the high-end spectrum, we see a collection of immersive reality sets that are head-mounted, have built-in sensors and even use extra cameras to track rotation as well as position in space. These headsets, like HTC's Vive use a dedicated computer, which has to be attached to the set via cables at all times. This prevents taking the set with you wherever you go, but it allows for better immersive experiences due to computing power.

Appendix B
Research on HoloLens©

Introduction
A promising product in mixed reality is the Microsoft HL. This pair of goggles can project all sorts of holograms into the surrounding of the viewer. Holograms can be any virtual 3D or 2D object and due to the fact that the HL scans the environment continuously, holograms can be mapped to specific places in the environment, or even move around the real environment (Microsoft, 2016f).

Developer Microsoft already has a lot of ideas about applications for the HL, and shows potential in a variety of markets with demos or commercials. However due to the fact that at the time of writing this thesis the product is fairly new to the market, not a lot of real applications have been developed yet. Microsoft has gotten a lot of positive attention for what the HMD can do at the moment and this has woken interested parties.

Benchmark is one of the interested parties that likes to incorporate the HL into her processes. Based on previously conducted research by Benchmark itself as well as other students, the decision was made to achieve two HLs to explore the possibilities within the company.

In this chapter the focus is on the HL. The functions of the product will be described as well as experiencing the product. To get a grip on the potential of the product, some applications of large developers will be looked into. Furthermore, a description is given on how to develop applications for this product, and the constraint of the product are stated. It is important to know both possibilities as well as limits when creating solutions incorporating such emerging technologies.

Function
First a description is given on how the device functions more from an experience point of view, second a more technical description is given on how the product functions.

Experience
The HL, which kinda resembles a pair of futuristic skiing-goggles, uses sensors to scan the real surrounding of the wearer. In the real environment holograms can be placed, that either consist out of 3D shapes, or are flat planes flying in the air or pinned against a wall. Once a hologram is placed in the environment, the viewer can walk towards it, walk away from it, walk around it, all whilst the hologram stays in the same place. Looking away will, of course, make the hologram disappear, as it is out of view. However, when looking back, the hologram will still be visible on the same spot. As the HL is a computer on its own, there is no need for cables during the use of the product. The fully mobile HL will render all objects on its own in front of the eyes of the user.

Practically using the HL goes a bit like this:
First the user has to put on the HL. The apparatus needs to be put on the head of the user, so that the eyes are covered by the lenses. A scalable headband is used to put the product in the right place and adjusting it, insures that the HL will not move during usage.

After putting on the HL, the user basically just sees the real environment he is in. Before holograms can be seen, they need to be called and placed somewhere, or an application that ensures this has to be started. Interacting with the HL mostly goes about using 3 varieties of natural interaction, gaze, gestures, and voice.
Gazing in terms of natural user interfaces (NUI) comprises following the gaze of the user. In the middle of user's field of view, a small white dot is placed, which acts a bit like a computer cursor. Looking at certain objects can be called gazing. While gazing, selection can be made in one of two way; using a specific gesture for selecting something, or through voice commands (Microsoft, 2016d).
Gestures are specified hand movements that are recognised by the HL. At the moment there are two gestures supported by the HL, the Airtap, and the Bloom (Microsoft, 2016d). Airtapping can be done by stretching your arm in front of you and pretending to tap someone in front of you on the shoulder. The HL recognises this as a select-gesture. Varieties on the single-tap, can be tapping and holding,

tapping and dragging, or double tapping. Support for these varieties differs per application. The bloom action can be evoked by putting all fingers of a hand together, facing upwards and than spreading them out. This action resembles what happens when the start-button is pressed on a Windows-computer. Here, a flying start menu will appear in front of the user (see Figure B1).

The final way of interacting naturally with the HL, is through voice commands. Voice commands for several actions are programmed into the HL, such as “select”. If recognised, the HL will act on these commands.
Once a certain application is started, holograms can be visible in the physical space. Looking through the lenses, shows objects that seem to be present in reality. The holograms are very bright, and appear to be on a specific point in space.

Technical

The HL can be divided into several parts working together. Microsoft explains the HL as an optical see-through HMD using holographic lenses, consisting out of waveguides, to project stereoscopic images on (Microsoft, 2016c). According to them, the key to creating a great holographic experience is to use holograms that are lightpoint rich, i.e. have high holographic density. Lightpoints can be described as small collections of light, that together build up the holograms.

Holograms are placed in the real environment, with regards to the context. You cannot see a hologram through a table, for example. Scanning the environment is established using multiple cameras scanning the surrounding. About 5 times per second images are taken from the HL cameras and processed in a specially developed Holo Processor Unit (HPU) into a 3D map of the surrounding. This is stored and constantly combined with new data to create a better understanding of the environment the user is in. This is called spatial recognition. Spatial recognition is limited to a few scans per second due to two main reasons. Firstly, scanning and understanding the environment in 3D requires a lot of processing power, which drains the battery faster. Secondly, it is not needed to constantly map the environment, as it probably will not change noticably over the course of a few seconds. Furthermore, limiting the spatial recognition speed, allows for ignoring fast changes in the environment, such as someone walking by, that are actually not part of the environment.
The spatial maps of each environment the HL has been

Figure B1. Gestures. Left: Airtap, Right: Bloom. Retrieved from Microsoft Windows Dev Center.

in, are stored in the memory of the HL and linked to the hotspot used for wireless connectivity on that place. This way, returning to a certain environment, allows for quickly reloading the already made maps of that environment. Even the placed holograms and their position in space are stored.

As stated in the part on experiencing the HL, mostly forms of natural user interfaces are used. This is a good thing, a handsfree device like the HL should not be immobilised by the use of input devices that are not portable e.g. keyboards, mouses etc. The advantage of using natural input, is that the user does not have to learn representations for interacting with the interface, such as cursor is your focus. Fully leaving the graphical user interface (GUI) behind has not happened yet. When using the HL, a lot of structural menus are used to navigate through. These menus are, just as in a GUI, 2D screens with buttons for settings. The menu can be pinned somewhere to a wall. It is unclear why they chose for this approach. It might have to do something with the convenience of not having to create new ways of navigating through the menus. Still, unfortunate, as MR opens the door to new and creative ways of losing 2D representations of menus and cursors.

On the side of the HL, two small speakers have been placed, directed at the wearer’s ears. Spatial sound, creating sound that seems to be coming from somewhere in the environment, is used to help create a feeling of reality to some holograms.

All input and output are processed in the HL itself. A specially designed chip, the holographic processor unit (HPU), does most of the work

All is combined into the HL, which can be used fully mobile for about 3 hours on the build in battery. The electronics on the inside are mostly not visible.

Applications

To get an idea of the possibilities of the HL, some applications have been looked into. Those are described here.

Mars environment - JPL / NASA

Data of the Curiosity Rover and satellites orbiting Mars have already lead to a large amount of data about the surface of Mars. Using the HL, NASA/JPL created a 3D visualisation of the surface of Mars that has opened a whole new way of researching, by visually being able to see Mars (Nase Jet Propulsion Lab, 2016). It was even proven possible to augment the surface of Mars in separate, controlled rooms for multiple users, that were able to see augmented versions of each other (see Figure B2).

Holoportation - Microsoft

One of the applications developed by Microsoft is Holoportation, where a virtual copy of a person is projected inside another room as a hologram that can be seen with the HL (Microsoft, 2016b). To create a convincing experience in Holoportation, two rooms need to be set up the exact same way. In one room there is a set of 3D cameras to track events in that room in realtime. The data is processed and used to create a 3D hologram for the viewer in the other room, wearing a HL (and possibly vice versa). As the rooms have been set up the exact same way, there is no need for a virtual environment, and the persons involved can interact with each other in the real environment and context.

Volvo

Car Builder Volvo has teamed up with HL to create a new experience of choosing your car (Erickson, 2015). Before buying or even manufacturing the car, customers can look at holograms of Volvo cars to help them choose or edit specific parts of the car[]. Besides the whole car, Volvo uses some holographic material to show their safety systems to potential customers[]. Although one could consider Volvo to be taking innovative steps, incorporating the HL, questions arise what the advantage of showing a hologram of a car that is available in the same space is. It might just be impressing potential buyers.

While researching several applications that have been developed for the HL, it became clear that most applications are either developed more as a demo showing off the potential of the HL, or by companies using enormous budgets. Most of the impressive applications show an idea of potential of the device, however, with a pretty specific focused app.

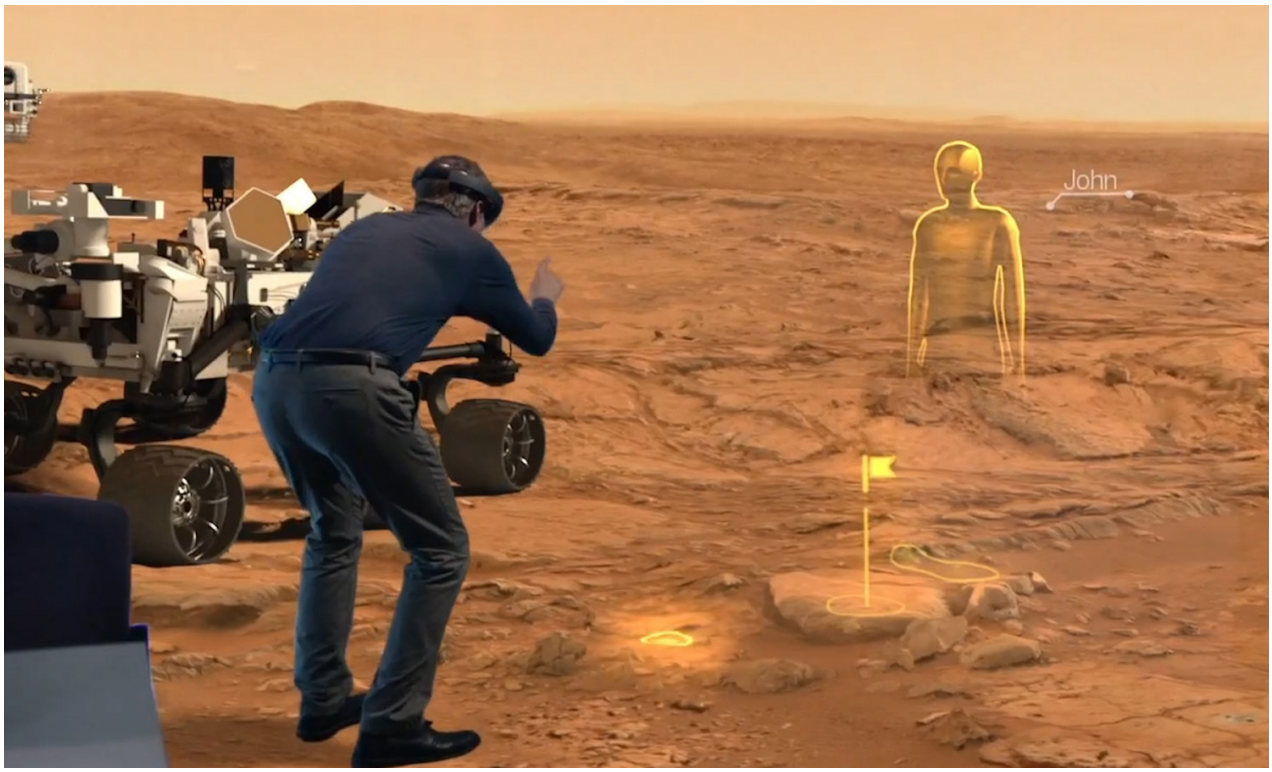


Figure B2. Depiction of the NASA/JPL HoloLens application. Retrieved from Microsoft HoloLens & NASA OnSight YouTub video.

Constraints and limits

The HL has some limits to what it can do. Despite the product functions well, there are some limitations the products has that can lead to constraints when developing applications for the HL.

Battery

The built-in battery of the HL allows the user to move freely through the environment he is in. There is no need for extra cables to power the device and all inputs and outputs are processed on the device itself. This comes, however, with a compromise in the form of a maximum time of use. Using the HL continuously allows for about 2-3 hours of use []. Although this might seem short in the first place, the battery powers a small wearable computer, generating 3D spatial holograms, processes that ask for processing power. It is important to note this maximum time of use when designing applications for the HL. An application that should be used all day long is impossible without recharging the device. The HL does offer the advantage using it while charging, but this does create the need for cables or portable batteries, a trade-off for the freedom that the HL offers in the first place.

Weight

Another point that has to do with maximum time of usage, is the weight of the product. The product does attach nicely to the head of the wearer, with a soft headband, however wearing the HL for longer periods of time can create red imprints on the forehead and nose, where the HL rests. Furthermore, it might be heavy on the muscles of the neck to constantly carry the extra weight. Especially when bending the head to look down, the extra momentum needs to established by the muscles, possibly leading to fatigue.

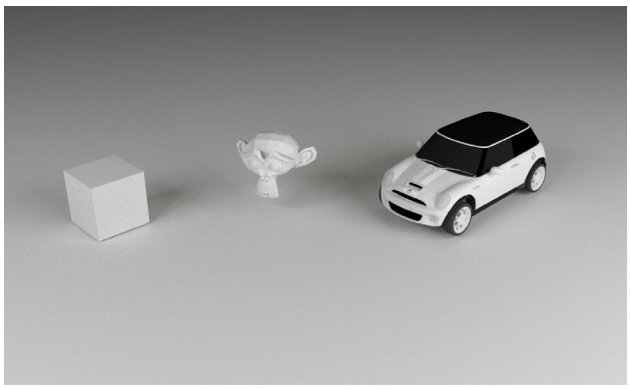


Figure B3: Several virtual objects; left: cube: 8 vertices. Middle: monkeyhead: 507 vertices. Right: car: over 2.000.000 vertices.

Resolution models

The HL can process a maximum of 150.000 vertices safely. This is the total of all visible holograms. Only using visually simple objects, this is not a problem, however, the amount of vertices grows exponentially when using more detailed objects. Creating realistic curvature in a model, only using faces, requires more vertices (see Figure B3).

Limited field of vision

The holograms appearing in front of the eyes of the wearer are limited to the size of the holographic lenses, from which the lightpoints hit the eye. On the HL, this does not match the full field of vision humans have. In fact, it is quite noticeable that the field of vision is quite small compared to the total human vision. Holograms that are too big to fit on the holographic lenses, are cut off visibly (see Figure B4). Looking at small virtual objects, or virtual objects that appear far away, this is not really a problem. When an object is cut off visibly, it can become harder to distinguish what the hologram is and it does detract from the experience of the holograms being really there.

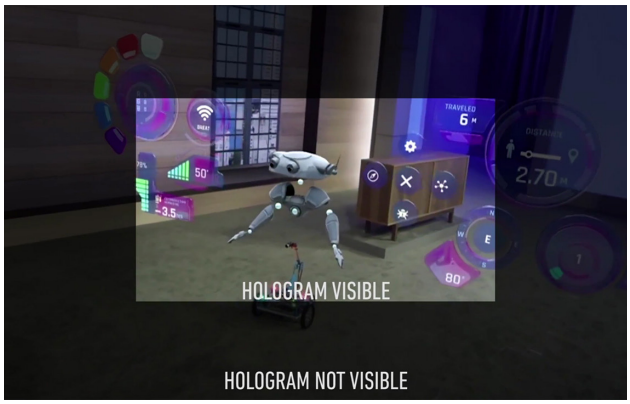


Figure B4: Depiction of field of view while wearing HL.. Retrieved from Polygon.com.

Surrounding

HL uses spatial mapping to create a virtual representation of the real world to place holograms in. This does ask for a surrounding that can actually be mapped. If the user sits in a location that constantly changes, for example due to large streams of people passing by, or moving machines, the HL will not be able to properly map the surrounding.

Overheating

The HL is cooled passively. Tasks that ask a lot from the processor could cause the device to overheat. A built-in safety mechanism ensures the materials of the HL will not melt or break when it is used extensively, by shutting of active applications. This will take place when the average processing power is above a certain criteria for at least one minute.

Optical see through

The optical see through lenses of the HL are transparent, allowing light from the surrounding to be visible to the eyes of the viewer. Lightpoints are added to the observations the viewer makes and they blend with the surrounding, depending on the strength of the light of the surrounding and HL. Basically you could say, that the strongest light source will be best visible. Using the HL in a bright environment, creates holograms that seem transparent, due to the environment being as strong as the light created by the HL. Dark environments are better suitable to project holograms in. A depiction of this is made in [fig], where gradations from white to respectively each color channel to black are made. Note that full black colors will never be visible when a light source is used to project something. Black is the absence of light, and it will always be transparent (see Figure B5).

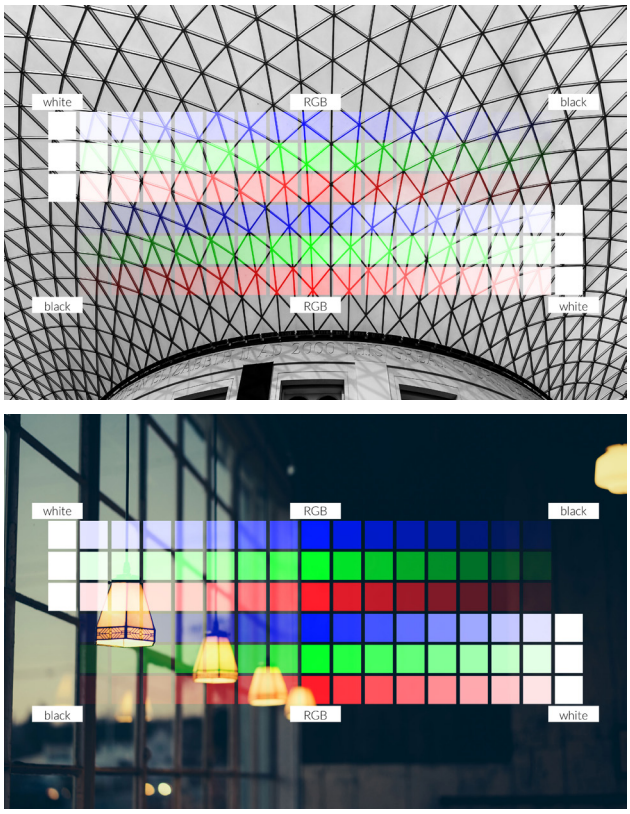


Figure: B5 Depiction of how lightpoints react to light (up) and dark (bottom) environment.

Appendix C
Development for HL

Currently, the only available edition is the developer edition, which is not meant for the consumer sector yet. At the moment of writing this thesis, no plans have been revealed for release of a commercial version of the product. As with most developer editions of products, they are meant for getting to know the product and developing content for it. In order to use the HL at the moment, specific applications have to be written to fulfill a specif goal. The advantage is that the application will suit the exact needs of the developer, in this case, Benchmark. The disadvantage, however, is that the full application will need to be written for the HL. As Benchmark has two developer editions of the HL, it is good to know how an application is created.

Here, developing applications for the HL is described.

According to Vroegop (2016), creating models and applications for the HL takes on 4 steps: design, compose, develop, deploy.

Design
To show content on the HL, models need to be created in 3D. The HL uses models existing out of vertices and faces (surface models). Vertices are points in a virtual coordinate system, that can be connected to each other, forming edges (lines between vertices). Faces are created from 3 or more vertices and form a surface (see Figure C1).

Programs that can be used to create models for the HL

Figure C1: Vertices, edges, and faces.

include:

- Maya
- 3DS Max
- Blender
- Sketchup

In general, CAD/CAM programs are not suitable to be used in the development of surface models, as they create models existing out of filled solids. Surface modeling programs create models of which only the outside properties of the surfaces are used to create visuals.

Compose
Once created, models can be prepared to be used on the HL using either Unity or DirectX. Unity is a 3D environment that is used often for the development of games. Microsoft specifies to use Unity on their website. Unity has created plug-ins for HL developers.

Develop
Visual Studio is used to turn the composed Unity environment into something usable for the HL. Visual Studio is a platform created by Microsoft for development of especially Windows applications. In Visual Studio extra functionality can be coded.

Deploy
Once the application is finished, it needs to be deployed on the HL. Another way to view models on the HL, is by sideloading them into a Microsoft's shared cloud storage system, OneDrive. The models need to be a stored in the specific Filmbox file format (.fbx) in order to be opened on the HL.

3D modellers have extended knowledge on creating models, animations, and working with vertices and faces. Domain experts should know the ins and outs of the domain that the models are being developed for, for example which filetypes can be used. As the HL works on a version of Windows 10, it comes in handy to have a developer for that field, to create extra functionality. AR experts are needed to develop for interaction in AR, as this is a whole different field of interaction regarded to GUIs. As users wearing the HL are able to walk around the real world, new scenarios open up, for example, what happens with an application when the user just walks out of its dimensions? Or how does a digital button look from behind? Finally a 'unity wizard' should combine all input into a application that can be used on the HL.

Appendix D
Application code

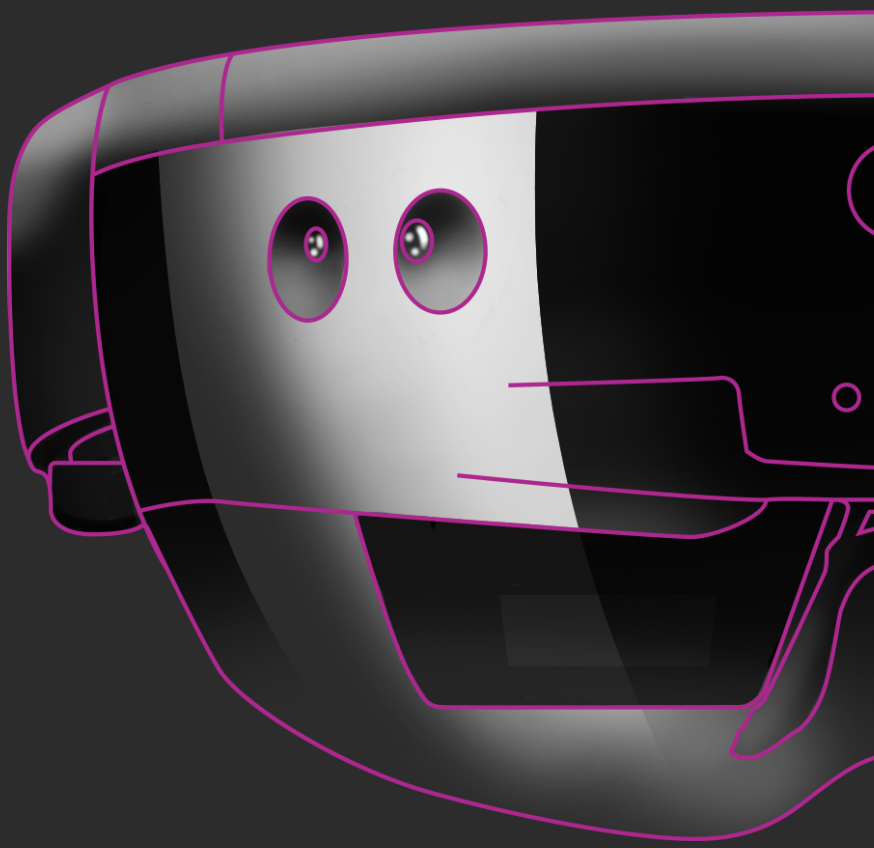
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.VR.WSA.Input;
using UnityEngine.Windows.Speech;

public class HandlerScript: MonoBehaviour {

    private GestureRecognizer _gestureRecognizer;

    private int StepNumber;

    public MovementStepOne movementStepOne;
    public MovementStepTwo movementStepTwo;
    public MovementStepThree movementStepThree;
    public MovementStepFour movementStepFour;
    public MovementStepFive movementStepFive;
    public MovementStepSix movementStepSix;
    //public MovementStepSeven movementStepSeven;

    public GameObject StepOneContainer;
    public GameObject StepTwoContainer;
    public GameObject StepThreeContainer;
    public GameObject StepFourContainer;
    public GameObject StepFiveContainer;
    public GameObject StepSixContainer;
    //public GameObject StepSevenContainer;

    // Use this for initialization
    void Start() {

        HideObjectsOf(StepTwoContainer);
        HideObjectsOf(StepThreeContainer);
        HideObjectsOf(StepFourContainer);
        HideObjectsOf(StepFiveContainer);
        HideObjectsOf(StepSixContainer);
        //HideObjectsOf(StepTwoContainer);

        //Instead of tap recognizer, use different tap
input

        _gestureRecognizer = new GestureRecognizer();
        _gestureRecognizer.TappedEvent += _
gestureRecognizer_TappedEvent;
        _gestureRecognizer.
SetRecognizableGestures(GestureSettings.Tap);
        _gestureRecognizer.StartCapturingGestures();

    }

    // Update is called once per frame
    void Update() {

        // instead of airtap, left mouse click is
simulated.
        if (Input.GetMouseButtonDown(0)) {
            // show button clicked on console
            Debug.Log("Pressed left click.");
            //ObjectShouldMove = true;
            // increment the stepnumber when mouse is
clicked
            StepUp();
        } else if (Input.GetMouseButtonDown(1)) {
            // show button clicked on console
            Debug.Log("Pressed right click.");
            // increment the stepnumber when mouse is
clicked
            StepDown();
        }


        switch (StepNumber) {
            case 1:
                ShowObjectsOf(StepOneContainer);
                HideObjectsOf(StepTwoContainer);
                HideObjectsOf(StepThreeContainer);
                movementStepOne = GameObject.
FindObjectOfType(typeof(MovementStepOne)) as
MovementStepOne;
                movementStepOne.StepOne();
                break;
            case 2:
                ShowObjectsOf(StepTwoContainer);
                HideObjectsOf(StepOneContainer);
                HideObjectsOf(StepThreeContainer);
                //Debug.Log("Step number" +
StepNumber);
                movementStepTwo = GameObject.
FindObjectOfType(typeof(MovementStepTwo)) as
MovementStepTwo;
                movementStepTwo.StepTwo();

                break;
            case 3:
                ShowObjectsOf(StepThreeContainer);
                HideObjectsOf(StepTwoContainer);
                HideObjectsOf(StepFourContainer);
                movementStepThree = GameObject.
FindObjectOfType(typeof(MovementStepThree)) as
MovementStepThree;
                movementStepThree.StepThree();
                break;
            case 4:
                ShowObjectsOf(StepFourContainer);
                HideObjectsOf(StepThreeContainer);
                HideObjectsOf(StepFiveContainer);
                movementStepFour = GameObject.
FindObjectOfType(typeof(MovementStepFour)) as
MovementStepFour;
                movementStepFour.StepFour();
                break;
            case 5:
                ShowObjectsOf(StepFiveContainer);
                HideObjectsOf(StepFourContainer);
                HideObjectsOf(StepSixContainer);
                movementStepFive = GameObject.
FindObjectOfType(typeof(MovementStepFive)) as
MovementStepFive;
                movementStepFive.StepFive();
                break;
            case 6:
                ShowObjectsOf(StepSixContainer);
                HideObjectsOf(StepFiveContainer);
                //HideObjectsOf(StepSixContainer);
                movementStepSix = GameObject.
FindObjectOfType(typeof(MovementStepSix)) as
MovementStepSix;
                movementStepSix.StepSix();
                break;
        }

    }

    private void _gestureRecognizer_
TappedEvent(InteractionSourceKind source, int tapCount,
Ray headRay) {
        StepUp();

    }

    void HideObjectsOf(GameObject ContainingObjects) {
        foreach(Renderer r in ContainingObjects.
GetComponentsInChildren < Renderer > ())
        r.enabled = false;
    }
```

```
    void ShowObjectsOf(GameObject ContainingObjects) {
        foreach(Renderer r in ContainingObjects.
GetComponentsInChildren < Renderer > ())
        r.enabled = true;
    }

    public void StepUp() {
        StepNumber++;
    }

    public void StepDown() {
        StepNumber--;
    }


}


public class SpeechHandler: MonoBehaviour {

    public HandlerScript handlerScript;

    KeywordRecognizer keywordRecognizer = null;
    Dictionary < string, System.Action > keywords = new
Dictionary < string, System.Action > ();

    // Use this for initialization
    void Start() {

        keywords.Add("Reset world", () => {
            // Call the OnReset method on every
descendant object.
            this.BroadcastMessage("OnReset");
        });

        keywords.Add("next step", () => {
            handlerScript = GameObject.
FindObjectOfType(typeof(HandlerScript)) as
HandlerScript;
            handlerScript.StepUp();
        });

        keywords.Add("previous step", () => {
            handlerScript = GameObject.
FindObjectOfType(typeof(HandlerScript)) as
HandlerScript;
            handlerScript.StepDown();
        });

        // Tell the KeywordRecognizer about our
keywords.
        keywordRecognizer = new
KeywordRecognizer(keywords.Keys.ToArray());

        // Register a callback for the
KeywordRecognizer and start recognizing!
        keywordRecognizer.OnPhraseRecognized +=
KeywordRecognizer_OnPhraseRecognized;
        keywordRecognizer.Start();

    }

    private void KeywordRecognizer_
OnPhraseRecognized(PhraseRecognizedEventArgs args) {
        System.Action keywordAction;
        if (keywords.TryGetValue(args.text, out
keywordAction)) {
            keywordAction.Invoke();
        }
    }

    // Update is called once per frame
    void Update() {

    }
}

public class MovementStepOne: MonoBehaviour {

    private int SubStepNumber1;

    public float Speed;
    public float RotationSpeed;
    private float TeleportSpeed;


    //variables step 1

    public Transform ScrewdriverStartPos;
    public Transform ScrewdriverEndPos;
    public GameObject Screwdriver;
    public Transform BracketStartPos;
    public Transform BracketEndPos;
    public GameObject Bracket;



    void Start() {
        SubStepNumber1 = 0;
        Speed = 0.5 f;
        RotationSpeed = 100 f;
        TeleportSpeed = 1000 f;
    }

    void Update() {

    }

    public void StepOne() {
        Debug.Log("MovementStepOne");
        StartCoroutine(DoMovement1());
    }

    IEnumerator DoMovement1() {

        switch (SubStepNumber1) {
            case 0:
                Debug.Log("Substep 1.0");
                SubStepNumber1 = 1;
                break;

            case 1:
                Debug.Log("SubStep 1.1");
                if (Screwdriver.transform.position !=
ScrewdriverEndPos.transform.position) {
                    Screwdriver.transform.
position = Vector3.MoveTowards(transform.position,
ScrewdriverEndPos.transform.position, Speed * Time.
deltaTime);
                } else
                    SubStepNumber1 = 2;
                break;
            case 2:
                Debug.Log("Substep 1.2");
                Screwdriver.transform.Rotate(0, 0,
RotationSpeed * Time.deltaTime);
                yield
                return new WaitForSeconds(2 f);
                SubStepNumber1 = 3;
                break;
            case 3:
                Debug.Log("Substep 1.3");
                Screwdriver.transform.Rotate(0, 0, 0);
                if (Screwdriver.transform.position !=
ScrewdriverStartPos.transform.position) {
                    Screwdriver.transform.
position = Vector3.MoveTowards(transform.position,
ScrewdriverStartPos.transform.position, Speed * Time.
deltaTime);
                } else SubStepNumber1 = 4;
                break;
            case 4:
                Debug.Log("Substep 1.4");
                if (Bracket.transform.position !=
BracketEndPos.transform.position) {
                    Bracket.transform.position =
Vector3.MoveTowards(Bracket.transform.position,
BracketEndPos.transform.position, Speed * Time.
deltaTime);
                } else SubStepNumber1 = 5;
                break;
            case 5:
                Debug.Log("Substep 1.5");
                yield
                return new WaitForSeconds(1 f);
                Bracket.transform.position =
BracketStartPos.transform.position;

                SubStepNumber1 = 0;
                break;
        }


    }

}

public class MovementStepTwo: MonoBehaviour {


    private int SubStepNumber2;
    public float Speed;
```

```
    public GameObject CouplePiece;
    public Transform CouplePieceStartPos;
    public Transform CouplePieceEndPos;

    public GameObject largePartOne;
    public Transform largePartOneStartPos;
    public Transform largePartOneEndPos;

    public GameObject largePartTwo;
    public Transform largePartTwoStartPos;
    public Transform largePartTwoEndPos;



    // Use this for initialization
    void Start() {
        SubStepNumber2 = 0;
        Speed = 1 f;



        largePartOne.GetComponent < Renderer >
().enabled = false;
        largePartTwo.GetComponent < Renderer >
().enabled = false;

        largePartOne.transform.position =
largePartOneStartPos.transform.position;
        largePartTwo.transform.position =
largePartTwoStartPos.transform.position;
        CouplePiece.transform.position =
CouplePieceStartPos.transform.position;

    }

    // Update is called once per frame
    void Update() {

    }

    public void StepTwo() {
        //Debug.Log("MovementStepTwo");
        StartCoroutine(DoMovement2());
    }

    IEnumerator DoMovement2() {

        switch (SubStepNumber2) {
            case 0:
                Debug.Log("Substep 2.0");
                SubStepNumber2 = 1;
                break;
            case 1:
                Debug.Log("Substep 2.1");
                // if statement?
                if (largePartOne.GetComponent <
Renderer > ().enabled == true) {
                    // Debug.Log(largePartOne.
GetComponent<Renderer>().enabled);
                    largePartOne.GetComponent <
Renderer > ().enabled = false;
                }
                if (largePartTwo.GetComponent <
Renderer > ().enabled == true) {
                    //Debug.Log(largePartTwo.
GetComponent<Renderer>().enabled);
                    largePartTwo.GetComponent <
Renderer > ().enabled = false;
                }
                if (CouplePiece.transform.position !=
CouplePieceEndPos.transform.position) {
                    CouplePiece.transform.position =
Vector3.MoveTowards(CouplePiece.transform.position,
CouplePieceEndPos.transform.position, Speed * Time.
deltaTime);
                } else SubStepNumber2 = 2;
                break;
            case 2:
                Debug.Log("Substep 2.2");

                if (largePartOne.GetComponent <
Renderer > ().enabled == false) {
                    largePartOne.GetComponent <
Renderer > ().enabled = true;
                } else
                    yield
                return new WaitForSeconds(1 f);
                SubStepNumber2 = 3;
                break;
            case 3:
                Debug.Log("Substep 2.3");
                if (largePartOne.transform.position !=
largePartOneEndPos.transform.position) {
                    largePartOne.transform.position =
Vector3.MoveTowards(largePartOne.transform.position,
largePartOneEndPos.transform.position, Speed * Time.
deltaTime);
                } else SubStepNumber2 = 4;
                break;
            case 4:
                Debug.Log("Substep 2.4");

                if (largePartTwo.GetComponent <
Renderer > ().enabled == false) {
                    largePartTwo.GetComponent <
Renderer > ().enabled = true;
                } else
                    yield
                return new WaitForSeconds(1 f);
                SubStepNumber2 = 5;
                break;
            case 5:
                Debug.Log("Substep 2.5");
                if (largePartTwo.transform.position !=
largePartTwoEndPos.transform.position) {
                    largePartTwo.transform.position =
Vector3.MoveTowards(largePartTwo.transform.position,
largePartTwoEndPos.transform.position, Speed * Time.
deltaTime);
                } else SubStepNumber2 = 6;
                break;
            case 6:
                yield
                return new WaitForSeconds(2 f);

                if (largePartOne.GetComponent <
Renderer > ().enabled == true) {
                    // Debug.Log(largePartOne.
GetComponent<Renderer>().enabled);
                    largePartOne.GetComponent <
Renderer > ().enabled = false;
                }
                if (largePartTwo.GetComponent <
Renderer > ().enabled == true) {
                    //Debug.Log(largePartTwo.
GetComponent<Renderer>().enabled);
                    largePartTwo.GetComponent <
Renderer > ().enabled = false;
                }

                //largePartOne.
GetComponent<Renderer>().enabled = false;
                //largePartTwo.
GetComponent<Renderer>().enabled = false;

                largePartOne.transform.position =
largePartOneStartPos.transform.position;
                largePartTwo.transform.position =
largePartTwoStartPos.transform.position;

                yield
                return new WaitForSeconds(0.5 f);
                CouplePiece.transform.position =
CouplePieceStartPos.transform.position;
                SubStepNumber2 = 0;
                break;
        }
    }
}

public class MovementStepThree: MonoBehaviour {

    private int SubStepNumber3;
    private float Speed;
    //variables step 3

    public Vector3 StartRotation;

    public GameObject box;
    public Transform boxStartPos;
    public Transform boxEndPos;

    public GameObject bottomBracket;
    public Transform bottomBracketStartPos;
    public Transform bottomBracketEndPos;
    private bool hidden;

    private Vector3 zeroPos;
    // Use this for initialization
    void Start() {
        Speed = 1 f;
        SubStepNumber3 = 0;
        zeroPos = new Vector3(0, transform.position.y,
```

```
transform.position.z);
        box.transform.position = boxStartPos.transform.
position;
        bottomBracket.transform.position =
bottomBracketStartPos.transform.position;
        HideObjectsOf(box);

        box.transform.rotation = Quaternion.Euler(250,
180, 0);
        //   box.transform.rotation = Quaternion.
AngleAxis(250, Vector3.right);
        //  box.transform.rotation = Quaternion.
AngleAxis(180, Vector3.left);
    }

    // Update is called once per frame
    void Update() {

    }

    public void StepThree() {
        //Debug.Log("MovementStepThree");
        StartCoroutine(DoMovement3());
    }

    IEnumerator DoMovement3() {
        switch (SubStepNumber3) {
            case 0:
                SubStepNumber3 = 1;
                break;

            case 1:
                Debug.Log("SubStepNumber 3." +
SubStepNumber3);
                if (bottomBracket.transform.position !=
bottomBracketEndPos.transform.position) {
                    bottomBracket.transform.position =
Vector3.MoveTowards(bottomBracket.transform.position,
bottomBracketEndPos.transform.position, Speed * Time.
deltaTime);
                } else
                    SubStepNumber3 = 2;
                break;

            case 2:
                Debug.Log("SubStepNumber 3." +
SubStepNumber3);
                if (hidden) {
                    ShowObjectsOf(box);
                }
                yield
                return new WaitForSeconds(1 f);
                SubStepNumber3 = 3;
                break;
            case 3:
                Debug.Log("SubStepNumber 3." +
SubStepNumber3);
                if (box.transform.position !=
boxEndPos.transform.position) {
                    box.transform.position = Vector3.
MoveTowards(box.transform.position, boxEndPos.
transform.position, Speed * Time.deltaTime);
                } else SubStepNumber3 = 4;
                break;
            case 4:
                // Debug.Log("SubStepNumber 3." +
SubStepNumber3);
                Debug.Log(box.transform.eulerAngles.x);
                if (box.transform.eulerAngles.x > 270.4
f) {
                    box.transform.Rotate(0.2 f, 0, 0);
                    //box.transform.position = Vector3.
MoveTowards(box.transform.position, zeroPos, Speed *
Time.deltaTime);
                } else
                    SubStepNumber3 = 5;
                break;

            case 5:
                yield
                return new WaitForSeconds(1 f);
                if (hidden == true) {
                    HideObjectsOf(box);
                }

                box.transform.position = boxStartPos.
transform.position;
                box.transform.rotation = Quaternion.
Euler(250, 180, 0);
                SubStepNumber3 = 0;
                break;
        }
        yield
        return new WaitForSeconds(3 f);
    }

    void HideObjectsOf(GameObject ContainingObjects) {
        foreach(Renderer r in ContainingObjects.
GetComponentsInChildren < Renderer > ())
        r.enabled = false;
        hidden = true;
    }

    void ShowObjectsOf(GameObject ContainingObjects) {
        foreach(Renderer r in ContainingObjects.
GetComponentsInChildren < Renderer > ())
        r.enabled = true;
        hidden = false;
    }
}

public class MovementStepFour: MonoBehaviour {

    public GameObject Screwdriver;
    public Transform ScrewDriverStartPos;
    public Transform ScrewDriverEndPos;

    private float Speed;
    private float RotationSpeed;
    private int SubStepNumber4;

    // Use this for initialization
    void Start() {
        SubStepNumber4 = 0;
        Screwdriver.transform.position =
ScrewDriverStartPos.transform.position;
        Speed = 1 f;
        RotationSpeed = -100 f;
    }

    // Update is called once per frame
    void Update() {

    }

    public void StepFour() {
        StartCoroutine(DoMovement4());
    }

    IEnumerator DoMovement4() {
        switch (SubStepNumber4) {
            case 0:
                Debug.Log("Substep 4." +
SubStepNumber4);
                SubStepNumber4 = 1;
                break;

            case 1:
                Debug.Log("Substep 4." +
SubStepNumber4);
                if (Screwdriver.transform.position !=
ScrewDriverEndPos.transform.position) {
                    Screwdriver.transform.position =
Vector3.MoveTowards(Screwdriver.transform.position,
ScrewDriverEndPos.transform.position, Speed * Time.
deltaTime);
                } else SubStepNumber4 = 2;
                break;
            case 2:
                Debug.Log("Substep 4." +
SubStepNumber4);
                Screwdriver.transform.Rotate(0, 0,
RotationSpeed * Time.deltaTime);
                yield
                return new WaitForSeconds(2 f);
                SubStepNumber4 = 3;
                break;
            case 3:
                Debug.Log("Substep 4." +
SubStepNumber4);
                if (Screwdriver.transform.position !=
ScrewDriverStartPos.transform.position) {
                    Screwdriver.transform.position =
Vector3.MoveTowards(Screwdriver.transform.position,
ScrewDriverStartPos.transform.position, Speed * Time.
deltaTime);
                } else SubStepNumber4 = 4;
                break;
            case 4:
                yield
                return new WaitForSeconds(1 f);
                SubStepNumber4 = 0;
                break;
```

```
        }

    }
}

public class MovementStepFive: MonoBehaviour {

    public GameObject Simcard;
    public Transform SimcardStartPos;
    public Transform SimcardEndPos;
    private float Speed;
    private int SubStepNumber5;

    // Use this for initialization
    void Start() {
        Speed = 1 f;
        SubStepNumber5 = 0;
        Simcard.transform.position = SimcardStartPos.
transform.position;
    }

    // Update is called once per frame
    void Update() {

    }

    public void StepFive() {
        StartCoroutine(DoMovement5());
    }

    IEnumerator DoMovement5() {
        switch (SubStepNumber5) {
            case 0:
                SubStepNumber5 = 1;
                break;
            case 1:
                if (Simcard.transform.position !=
SimcardEndPos.transform.position) {
                    Simcard.transform.position =
Vector3.MoveTowards(Simcard.transform.position,
SimcardEndPos.transform.position, Speed * Time.
deltaTime);
                } else SubStepNumber5 = 2;
                break;
            case 2:
                yield
                return new WaitForSeconds(1 f);
                Simcard.transform.position =
SimcardStartPos.transform.position;
                SubStepNumber5 = 0;
                break;
        }

    }
}


public class MovementStepSix: MonoBehaviour {

    public GameObject Screwdriver;
    public Transform ScrewdriverStartPos;
    public Transform ScrewdriverEndPos;

    public GameObject Bracket;
    public Transform BracketStartPos;
    public Transform BracketEndPos;

    private int SubStepNumber6;
    private float Speed;
    private float RotationSpeed;

    // Use this for initialization
    void Start() {
        Speed = 1 f;
        RotationSpeed = -100 f;

        Screwdriver.transform.position =
ScrewdriverStartPos.transform.position;
        Bracket.transform.position = BracketStartPos.
transform.position;
    }

    // Update is called once per frame
    void Update() {

    }

    public void StepSix() {
        StartCoroutine(DoMovement6());
    }

    IEnumerator DoMovement6() {

        switch (SubStepNumber6) {
            case 0:
                SubStepNumber6 = 1;
                break;
            case 1:
                if (Bracket.transform.position !=
BracketEndPos.transform.position) {
                    Bracket.transform.position =
Vector3.MoveTowards(Bracket.transform.position,
BracketEndPos.transform.position, Speed * Time.
deltaTime);
                } else SubStepNumber6 = 2;
                break;
            case 2:
                if (Screwdriver.transform.position !=
ScrewdriverEndPos.transform.position) {
                    Screwdriver.transform.position =
Vector3.MoveTowards(Screwdriver.transform.position,
ScrewdriverEndPos.transform.position, Speed * Time.
deltaTime);
                } else SubStepNumber6 = 3;
                break;
            case 3:
                Screwdriver.transform.Rotate(0, 0,
RotationSpeed * Time.deltaTime);
                yield
                return new WaitForSeconds(2 f);
                SubStepNumber6 = 4;
                break;
            case 4:
                Screwdriver.transform.Rotate(0, 0, 0);
                if (Screwdriver.transform.position !=
ScrewdriverStartPos.transform.position) {
                    Screwdriver.transform.position =
Vector3.MoveTowards(Screwdriver.transform.position,
ScrewdriverStartPos.transform.position, Speed * Time.
deltaTime);
                } else SubStepNumber6 = 5;
                break;
            case 5:
                yield
                return new WaitForSeconds(1 f);
                Bracket.transform.position =
BracketStartPos.transform.position;
                SubStepNumber6 = 0;
                break;

        }


    }
}
```



2017

UNIVERSITY OF TWENTE.

Benchmark electronics