

Visualisation Research Center

University of Stuttgart  
Universitätsstraße 38  
D-70569 Stuttgart

Bachelorarbeit

**Interaction concepts for  
collaborative reviewing and editing  
of large-scale 3D-models in AR/VR  
applications**

André Wruszczak

<b>Course of Study:</b>	Softwaretechnik
<b>Examiner:</b>	Prof. Dr. Thomas Ertl
<b>Supervisor:</b>	Michael Becher, M.Sc. Dr. Guido Reina
<b>Commenced:</b>	July 9, 2018
<b>Completed:</b>	January 9, 2019



## **Abstract**

Even though a large number of Augmented reality (AR)/Virtual reality (VR) applications already exist in the entertainment industry, their usefulness still need to be further examined outside of gaming and movies. Due to the current limitations of AR hardware, modern concepts for user interfaces and user experiences have to be evaluated and designed in order for them to become a sufficient base for developing precise, efficient and enjoyable tools. Especially the difficulty of properly visualizing large-scale 3D-models on the current screens of AR devices, demand a new set of methods on how to look at and navigate around them. As opposed to AR, is VR an isolating experience, requiring the conception of new collaborative approaches to allow working productively in a team. For this purpose the note and management tool is recommend in order to provide methods for sequential and parallel collaboration in AR/VR applications.

## **Kurzfassung**

Durch die große Anzahl von AR/VR-Anwendungen im Unterhaltungssektor, stellt sich die Frage ob diese Hardware auch für Projekte und Applikationen außerhalb der Spiele- und Filmbranche benützt werden könnte. Durch die aktuellen Limitierungen der AR-Hardware müssen neue Konzepte für die Bedienung von Applikationen konzepiert und entwickelt werden um eine ausreichende Basis für ein produktives, präzises und angenehm bedienbares Werkzeug darstellen zu können. Insbesondere große 3D-Modelle lassen sich sehr schwer auf den bisher noch sehr kleinen Displays von AR-Geräten abbilden, weshalb neue Arten der Betrachtung und Navigation gefunden werden müssen. Anders als bei AR ist VR eine isolierende Erfahrung, weshalb insbesondere neue kollaborative Ansätze gefunden werden müssen um eine produktive Zusammenarbeit im Team ermöglichen zu können. Für diesen Zweck wird das Notiz und Management Tool vorgeschlagen.



# Contents

<b>1</b>	<b>Introduction</b>	<b>11</b>
<b>2</b>	<b>Design Foundations</b>	<b>13</b>
2.1	Safety . . . . .	13
2.2	Comfort . . . . .	13
2.3	Familiarity . . . . .	13
2.4	Immersion . . . . .	14
2.5	Environment . . . . .	15
2.6	Input Possibilities . . . . .	15
2.7	Billboards . . . . .	17
2.8	Movement . . . . .	17
2.9	Instructions . . . . .	17
2.10	Feedback . . . . .	18
2.11	Spatial Mapping . . . . .	18
<b>3</b>	<b>Evaluation of User Interface (UI)/User Experience (UX) interaction models in AR/VR</b>	<b>19</b>
3.1	Differences AR and VR . . . . .	19
3.2	Inputs . . . . .	19
3.3	User Interfaces . . . . .	21
3.4	Additional Concepts . . . . .	28
<b>4</b>	<b>Existing Tools</b>	<b>31</b>
4.1	Unity . . . . .	31
4.2	SteamVR . . . . .	32
4.3	Vuforia . . . . .	32
<b>5</b>	<b>Implementation</b>	<b>33</b>
5.1	Requirements . . . . .	33
5.2	Technology and Libraries . . . . .	33
5.3	Collaboration . . . . .	34
5.4	Note Tool . . . . .	34
5.5	Management Tool . . . . .	35
5.6	Input . . . . .	35
5.7	Optimizations . . . . .	36
<b>6</b>	<b>Results</b>	<b>37</b>
6.1	Library of useful UI Elements . . . . .	37
6.2	Tools for Reviewing and Editing . . . . .	41
6.3	Collaborative Work . . . . .	45
6.4	Difficulties . . . . .	46

<b>7 Summary</b>	<b>49</b>
<b>8 Future Work</b>	<b>51</b>
<b>Bibliography</b>	<b>53</b>

## List of Figures

2.1	Comfort zone diagram for VR users by Mike Alger [VDMfVR15] . . . . .	14
2.2	Optimal field of view as depicted in the mixed-reality design guideline by Microsoft [MRGuideline] . . . . .	15
3.1	Milgram’s reality-virtuality continuum, as seen in [CF11]. Source originally from Wikipedia: <a href="https://en.wikipedia.org/wiki/File:Virtuality_Continuum_2.jpg">https://en.wikipedia.org/wiki/File:Virtuality_Continuum_2.jpg</a> . . . . .	19
3.2	List design improved for VR . . . . .	22
3.3	List design improved for AR . . . . .	23
3.4	Button designs by Mike Alger as depicted in [VDMfVR15] . . . . .	24
3.5	Notification design improved for AR . . . . .	26
3.6	Notification design improved for AR . . . . .	27
3.7	Information panel design improved for VR . . . . .	28
3.8	Information panel design improved for AR . . . . .	29
6.1	List implemented for VR . . . . .	37
6.2	List implemented for AR . . . . .	38
6.3	Notifications implemented for VR . . . . .	39
6.4	Notifications implemented for AR . . . . .	40
6.5	Detail view implemented for VR . . . . .	40
6.6	Detail view implemented for AR . . . . .	41
6.7	Main menu implemented for AR/VR . . . . .	42
6.8	Before the manipulation tool was used to filter the list and model . . . . .	42
6.9	The resulting model after the manipulation tool was used . . . . .	43
6.10	A miniature version of the large-scale 3D-model that allows for quick navigation . . . . .	44
6.11	An element has been selected and a new note is being added to it . . . . .	45
6.12	The view tool allows for hiding objects as well as highlight them . . . . .	46





## List of Abbreviations

**API** Application programming interface. 45

**AR** Augmented reality. 3

**BIM** Building information modeling. 28

**HMD** Head-mounted display. 30

**LOD** Level of detail. 36

**UI** User Interface. 5

**UX** User Experience. 5

**VR** Virtual reality. 3



# 1 Introduction

It is often necessary to build a small-scale physical representation of a structure in order to make communicating about the structure easier for all parties involved. Similarly, AR/VR could be used for collaborative work on such structures, while replacing the need for a physical representation, especially as a 3D-model of the structure usually already exists. Furthermore these AR/VR representations could allow for an easier integration to already existing tools while enabling a type of collaboration that does not even necessarily require all participants to be there at the same time and location. So far, AR/VR applications are often used to impress potential clients and visitors by providing a new and exciting experience to them. Yet, the number of AR/VR applications that are regularly used in businesses is still marginal.

The current lack of standards and established guidelines for AR/VR applications result in users not being able to become familiar with interaction concepts and therefore decreases the chance that they will want to use these applications for business purposes. That being said, being able to examine small details in the 3D-model while maintaining a clear view over the whole model, as well as allowing users to inspect the 3D-model from any perspective they can imagine could be a significant improvement to current workflows. Nevertheless, do AR/VR applications still face technological problems, mostly due to hardware limitations. Especially the HoloLens is still severely limited in its capabilities and as a result several already existing design and interaction concepts had to be adapted in order to properly work for the HoloLens too. In contrast, VR applications usually manage to fully visualize large-scale 3D-models but because of the isolating elements of VR, designing with collaboration in mind can be difficult as well.

The goal of this thesis is to find and examine current design guidelines and interaction concepts and to develop a useful library of UI elements that could improve collaborative reviewing and editing of large-scale 3D-models with a focus on architectural needs. Furthermore, toolkits that offer a starting point for implementing user interfaces for AR/VR applications were examined and evaluated as well. In order to be able to visualize large-scale 3D-models on the HoloLens, optimization methods as well as alternative rendering possibilities were examined. Likewise, a note and management tool were developed to enable users in VR to collaborate effectively with their team.



## 2 Design Foundations

Although AR/VR designs are still considered to be in a constant state of change, some design guidelines on how to handle common problems for AR/VR have already been proposed by the producers of current AR or VR devices. It is important to note that although typical UI/UX designs for desktops and mobile devices might not directly apply to three dimensional designs in AR/VR, they are still of relevance and the aforementioned guidelines heavily rely on these standards as well as on already traditionally three dimensional design concepts to be found in package designs, interior design, architectural design or similar.

### 2.1 Safety

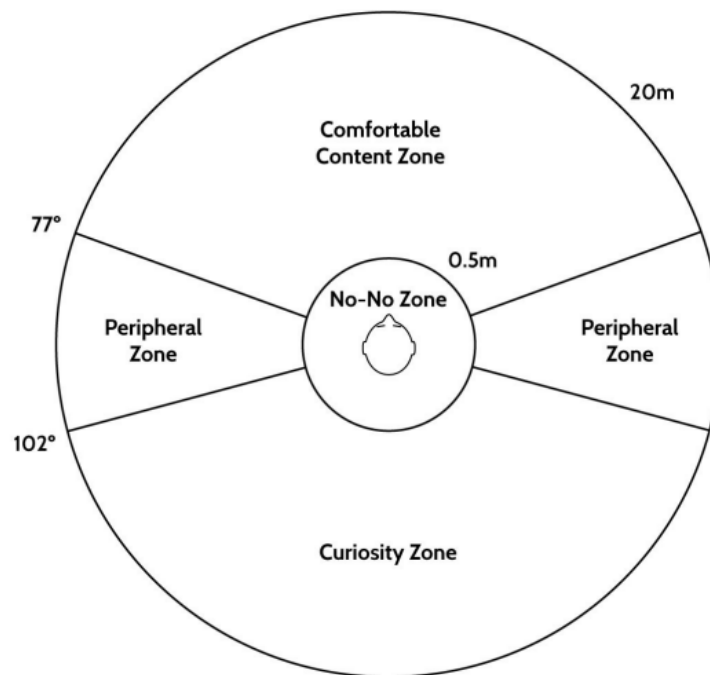
First and foremost the safety of the user and anyone in their close vicinity while using a AR or VR device should be of utmost importance. Therefore applications that require large and fast arm movements are to be discouraged. Dizziness and nausea can also occur by extensive use of VR devices, although some people are more susceptible to these discomforts than others.

### 2.2 Comfort

There are many things to consider when designing AR/VR applications for business purposes. In order for any application to be worth using for professional purposes, it should provide a significant improvement over other tools as well as allow the user to comfortably operate with it and without demanding too much physical effort. Therefore input methods that are physically straining for most users are to be discouraged. Especially for VR devices using the rotation of the head should only sparingly be used as well. Mike Alger neatly summarized which areas around the user are suitable for placing any sort of interactable content in the following diagram Figure 2.1. Furthermore placing elements too high above the user's head or too close to their feet could potentially encourage uncomfortable neck movements as seen in Figure 2.2.

### 2.3 Familiarity

Although it is not necessary to use familiar two dimensional design concepts for three dimensional designs it could greatly reduce alienation, allowing the user to get to know the new possibilities of AR/VR incrementally without overwhelming them. Quill is a perfect example for a familiar, yet functional three dimensional design approach, it heavily lends

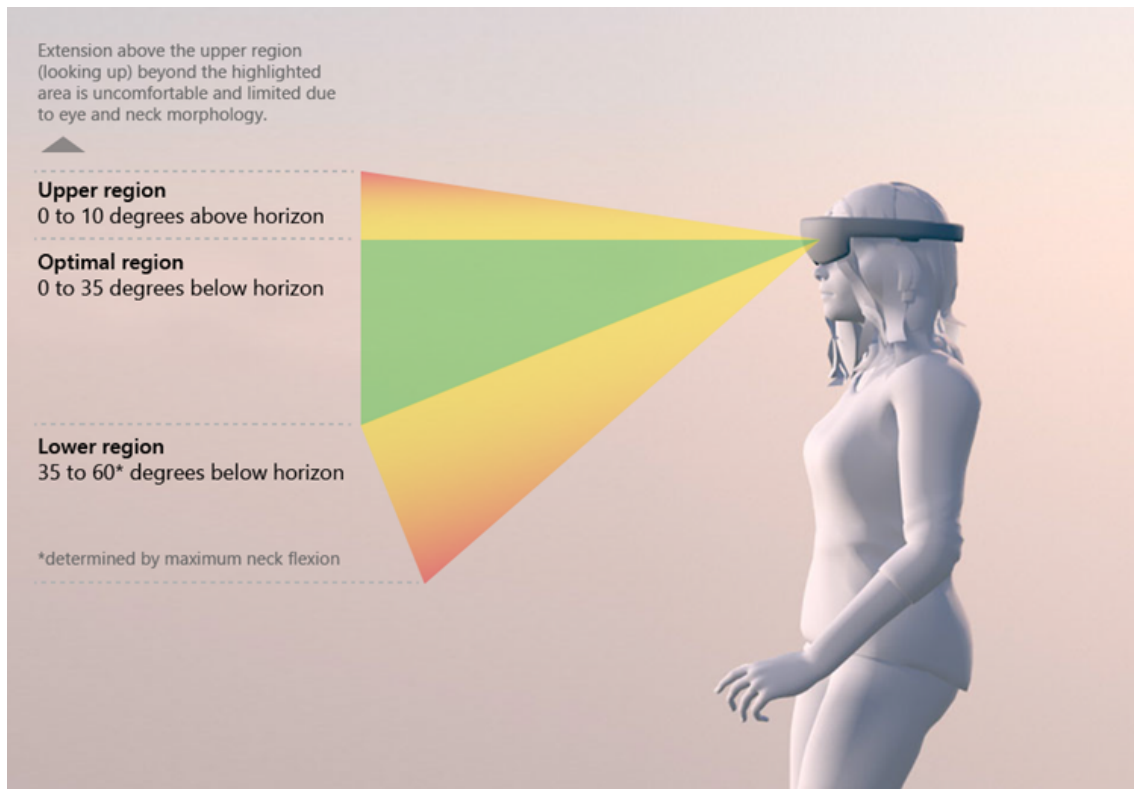


**Figure 2.1:** Comfort zone diagram for VR users by Mike Alger [VDMfVR15]

concepts from the for most users very familiar 90's era of UI design, where all buttons and panels were designed to resemble their formerly used mechanical counterparts [Quill18]. This heavily increases the productivity of first time users and could therefore increase the application's retention rate as well.

## 2.4 Immersion

Immersion is an important aspect of VR design, especially in the entertainment sector should the user fully immerse themselves in the media and potentially even forget their surroundings. This is often not desirable for business applications and tools as the user is most likely using the AR/VR application with a very specific purpose in mind and total immersion of the user could potentially distract from that purpose. Additionally, increasing immersion often comes at the cost of decreasing comfort and functionality. For example while raising ones arm and looking at the wrist is a highly immersive and familiar process for most users in order to get the time, it's a lot more time and energy consuming to simply glance at the time indicator at the bottom right of any desktop screen.



**Figure 2.2:** Optimal field of view as depicted in the mixed-reality design guideline by Microsoft [MRGuideline]

## 2.5 Environment

The real world environment around the user whenever a AR/VR application is used needs to be considered in the design process as well. It is important to avoid placing interactable objects at positions that are hard to reach, especially the distance to the user plays an important part as seen in Figure 2.2. By placing everything in the comfort zones of the user, they do not have to crouch in order to pick something from the ground or jump in order to pick something up from the ceiling. AR devices like the HoloLens even offer spatial recognition by scanning the actual environment of the user in order to allow attaching virtual elements to key objects in the user's environment in contrast to a random position in the air.

## 2.6 Input Possibilities

### 2.6.1 Gaze

The most intuitive and straight forward approach for interaction involves the gaze of the user. By including an indicator point in the center of their screen, the user immediately sees where his gaze is directed at and can now easily highlight or select objects by just

looking at them. VR as well as AR applications support this method, yet it is important to note that this indicator point should almost always rely on the head position and not the individual eye position in order to avoid eye strain for the user.

### 2.6.2 Gestures

As most people are accustomed to use their hands as an input device, be it with a mouse, keyboard or a touchscreen, it is not surprising that VR and AR technology tries to build on this familiarity. VR offers gesture control with LeapMotion or other trackers [LM18]. It is important to note that this approach is rather limited so far, especially overlapping hand gestures tend to cause tracking issues and some users might not be able to perform all gestures either. Additionally one application should only use very few distinct gestures in order to avoid accidentally recognizing a similar yet different gesture. As the HoloLens so far does not include any sort of support for motion controllers it relies heavily on gestures for complex inputs. Only a very few distinct gestures are commonly used, like the "air tap" and "bloom" [MRGuideline].

### 2.6.3 Gamepad-Controller

Although gamepads are mostly associated with the entertainment sector, some companies recently started using these input devices for a comfortable and familiar way to achieve complex maneuvers, which is especially useful whenever two different rotary axis are needed, as it is often the case with camera movements [MIA18].

### 2.6.4 Motion controllers

So far there is no support for the HoloLens for motion controllers, but most VR devices heavily rely on these for complex inputs. Especially grabbing objects can be seen as a direct transfer from the simple drag and drop motion normally required with the mouse or touch inputs. Furthermore, pointing at things with a controller or hand is a familiar movement for most users and by providing the user with a curved arc instead of a straight line, tasks like teleporting over a small obstacle become a lot easier.

### 2.6.5 Voice Commands

Simple tasks could also be mapped to voice commands in order to select something has been highlighted before with the gaze of the user. Voice commands are getting more and more familiar reliable and familiar through the growing number of smart home devices, but are still rather error prone. Due to the risk of misunderstands, important or not undoable actions should not be activated by voice commands.



## 2.7 Billboards

Billboarding UI elements is a common technique used in AR/VR applications to ensure that UI elements always face the user at all times. This is often advisable to ensure that the user can actually see the elements and does not have to move to a specific position with a specific head rotation to actually properly see them.

## 2.8 Movement

### 2.8.1 Real Movement

The most obvious way for the user to move in AR/VR is by actually having them move in their real environment. This is only possible if the AR/VR device actually supports motion tracking either with inside-out tracking of the HoloLens [HololensTracking17] or external tracking sensors like the lighthouse [Lighthouse16] of the HTC Vive [Vive18]. It is important to note that the AR/VR space of users, as well as their mobility can significantly vary.

### 2.8.2 Teleportation

A commonly used method for allowing movement in AR/VR applications is to let the user manipulate their position with an input device, effectively teleporting them to any given position in sight. Ideally this also allows for rotations, preventing users from having to move and circle around an object if they want to inspect any object from different angles.

### 2.8.3 Locomotion

Another way to allow movement is Locomotion. This is a steady movement of the user's position, controlled by an input device. In order to avoid motion sickness locomotion is often used in combination with the vignette design although vignetting might not necessarily improve symptoms for motion sickness [NBW18].

## 2.9 Instructions

Due to most users not yet being familiar enough with AR/VR applications, even common and basic instructions on how to operate the application should be clearly explained at least once. Ideally the attention of the user is naturally guided to these instructions and meanwhile other inputs are locked to the user. Additionally the rest of the surroundings might have to be made less visible in order to assure that the user actually pays attention to the instructions. Furthermore a tested way to guide the attention of the user to the instructions are guiding arrows, however, introducing transparency to the arrows in order to avoid blocking the view for the user is advised [RP17]. Similarly, sounds as well as a grey overlay effect for the whole view can be used to guide the user.

### **2.10 Feedback**

Considering haptic, visual or auditive feedback is very important whenever designing enjoyable AR/VR applications. The user should always see an immediate reaction to any valid interaction finished. Similar to application and web design the response time should be held as low as possible to allow for a responsive and proficient use of the application.

In AR/VR applications this effect might be even more important as users are used to immediately seeing an effect in their real world surroundings whenever they decide to touch an object or move.

### **2.11 Spatial Mapping**

Spatial Mapping is an important tool for AR applications to anchor objects to objects in the real world. This can be achieved in several ways, it is important to allow the user to move objects around whenever logical to do so in order to cater to all sort of different working spaces the AR user might have. Additionally allowing the user to attach an object on top of a working desk could potentially let the user feel familiarity with the tool by resembling the former workflow of the task.

## 3 Evaluation of UI/UX interaction models in AR/VR

### 3.1 Differences AR and VR

Prior to discussing different interaction models in AR/VR it is important to note the differences in environments they target. As seen in Figure 3.1, AR is a lot closer to real environments and expands upon it by adding auditorial and visual components. Additionally AR hardware scans the real environment in order to provide the possibility of even modifying existing objects in it. In contrast, VR applications usually do not interact with the real environment at all and instead provide a whole new experience. As a result AR applications unlike VR applications allow for simple collaborations with people at the same time and location, by not completely blocking their real environment and therefore other people from their view.

Another big difference is the current amount of input possibilities for both sort of devices. The HoloLens does not support motion trackers yet and gestures by themselves are a rather limited form of input, as currently only very few gestures with only one hand at the same time, are supported. Meanwhile the VIVE allows for several different input possibilities, either with the motion controllers or other similar devices coupled with the VIVE motion tracker.

### 3.2 Inputs

Most inputs can be generalized as either selecting, submitting or dragging an UI element.



**Figure 3.1:** Milgram's reality-virtuality continuum, as seen in [CF11]. Source originally from Wikipedia: [https://en.wikipedia.org/wiki/File:Virtuality\\_Continuum\\_2.jpg](https://en.wikipedia.org/wiki/File:Virtuality_Continuum_2.jpg)

### 3.2.1 Select

In order for the application to register which element the user wants to interact with, an element needs to be able to be selected. This could be achieved with the following AR/VR inputs.

**Gaze:** The most straightforward way to select an element is by selecting it whenever the user looks at it for a set minimum period of time. The problem with this approach is that it is difficult to avoid accidental selections as the user pretty much always looks at something within the application. Additionally selecting smaller objects can pose quite a difficult task as it requires to steadily and precisely gaze at it for a set amount of time. Finally transparent objects as well as a group of objects can not be selected intuitively.

**Laserpointer:** By adding a laserpointer either from the tip of the motion controllers or head position of the user it is a lot easier to quickly select specific elements. In AR the laserpointer could be a line from the head position to the "tap" finger position that extends further beyond in order to allow precise selections. This approach might not be very immersive but is quite familiar to most users as it highly resembles the selection model on a PC.

**Collision:** A different approach to detecting if a user wants to select something could be by collision. Whenever the user virtually touches an element it gets selected. The problem with this approach is that small elements that are close to each other could be selected and it would be difficult to select only a smaller subset at the same time. Whenever an element is selected there should be a visual and auditive feedback given to the user.

### 3.2.2 Submit

After an element was selected, it is now possible to submit any action associated with it. This could be a new input or an expansion of the already mentioned inputs.

**Fuse:** After the user has been looking at a specific element for the set minimum period of time a countdown could be started and visually represented to the user, this approach is commonly known as a fuse button. Once the countdown is completed the action is triggered and the element was submitted. However this process is rather time consuming and could quickly become tiresome to the user.

**Trigger Button:** Whenever the user selected an element with the laserpointer from the tip of the motion controller, simply pressing a button in the controller could submit the the element. For AR devices this could be achieved with another air tap or multiple air taps in quick succession.

**Target zone:** If the user virtually collided with the element it could be possible to move it to a designated target zone in order to trigger the desired action.

Most users should already be familiar with these interaction methods. Nevertheless should immediate visual and auditive feedback always be provided in order to let the user know that the approach has successfully worked. This feedback could be coupled with a notification or similar UI elements.

### 3.2.3 Drag

Whenever the user needs to scroll, rotate, move or scale an element a drag interaction is required. For the most part selecting an element, then submitting it and moving the input device to the desired position should suffice in order to complete a drag motion. Yet some of these interactions could possibly be further improved with two inputs at the same time.

**Rotate:** In order to rotate an element the user could potentially grab two anchor points of any desired element and now move both inputs around the center of it. For VR devices this should not be a problem as most of the times two motion controllers or hands are supported. For AR devices bimanual inputs are not yet officially supported even though it is possible to implement such two-handed gestures [BimanualAR18].

**Scale:** The same methods could be used for scale, but instead of moving both inputs around the object, they need to be moved further apart from one another. For effective use of this method it is generally advised to implement a sort of indicator of the current scale between both inputs.

## 3.3 User Interfaces

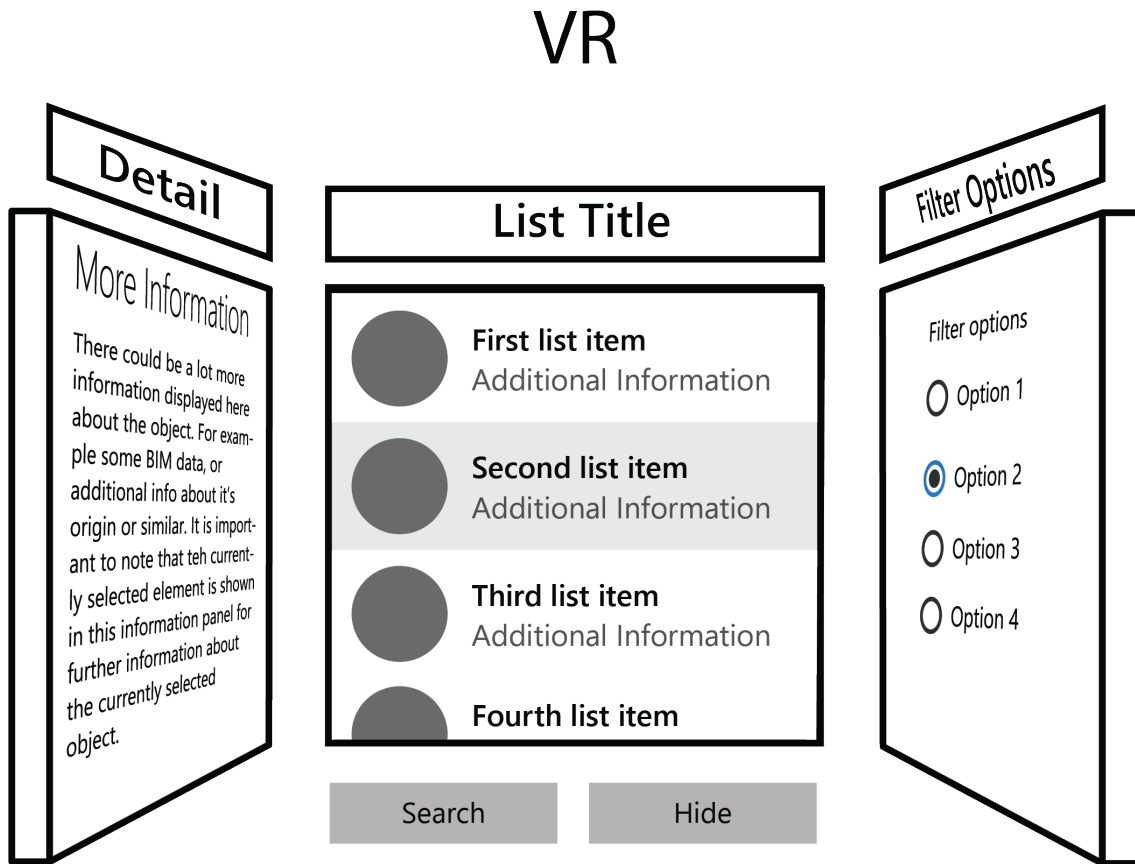
### 3.3.1 Lists

The material design guideline defines a list as "continuous, vertical indexes of text or images" [MaterialList18]. Accordingly a list is most commonly known as a scrollable, vertical list of elements, which could potentially be interacted with. Generally speaking a list includes a title and name of each element and also has a search function in its header. The most straightforward way to design a list for AR/VR is to create a simple 2D panel that includes a header as well as a scrollable vertical list of elements. This list has a fixed position somewhere in the virtual or augmented environment.

Lists allow for a general overview over a set of elements. Additionally lists often increase the viewability of certain elements by allowing the user to filter and sort through them. Users should also be able to get more information about a single element by selecting it or similar.

#### Improved for VR

In VR small interactable boxes and texts are often hard to interact with, therefore the list has been decluttered by separating it into three panels in order to be able to increase the size of each element as seen in Figure 3.2. In order to decrease the space this list uses up, both the detail and the filter options panel could potentially be hidden and only appear whenever the search button is submitted or one of the elements in the list is selected accordingly.



**Figure 3.2:** List design improved for VR

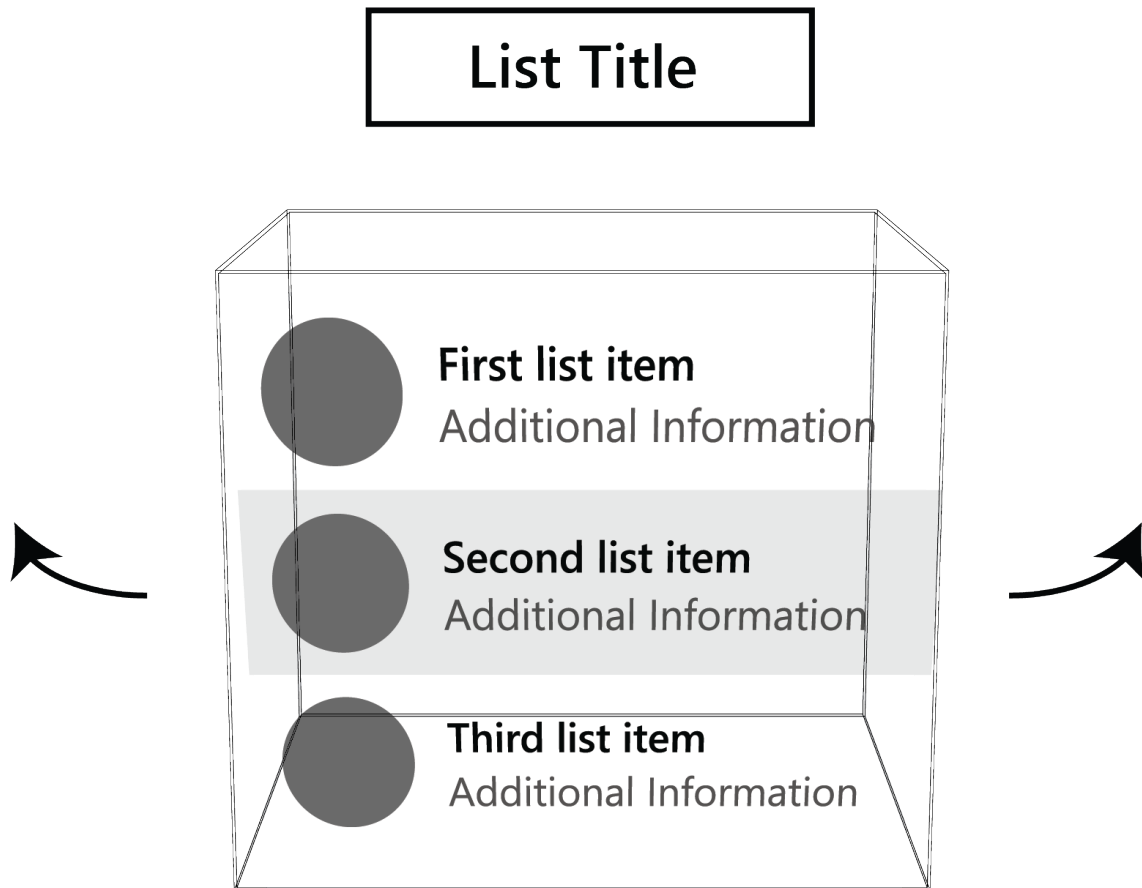
This representation of a list should allow the user to have an overview over the list, as well as a single element for comparison and enable rapid sorting and filtering by immediately seeing the result of the filter, sort and search options in the list panel.

All panels are billboards and should therefore always be facing the user in order to avoid forcing the user to move to a specific spot in order to be able to read the texts. Moving and placing the panels in a relative position to the user should be implemented to avoid blocking important views on an object of interest whenever the list is open.

### Improved for AR

In contrast to the open and big panels of the VR list, the AR list has been condensed into a single cube as seen in Figure 3.3. The cube allows for simple one handed interactions by dragging it in a direction. The list can be scrolled by either dragging the cube up or down. Whereas dragging it right shows the filter options panel and left the information about the currently selected element in the list. Other than in the real environment, the cube does not necessarily need to be limited to only have 4 faces when rotating it right or left, although including more than 4 faces could be counterintuitive to users and not be discovered unless explicitly visually shown or explained to the user.

# AR



**Figure 3.3:** List design improved for AR



**Figure 3.4:** Button designs by Mike Alger as depicted in [VDMfVR15]

Finally this cube could be positioned anywhere around the user and should not block the view of actually important objects.

#### 3.3.2 Buttons

Buttons normally have two main variations. Big, horizontal buttons, including the action the button triggers as text or small, circular buttons that depict an icon instead of the text describing the action. Furthermore both variations of buttons can have different states, they can be highlighted, pressed and deselected. The user can interact with the buttons by either colliding with their motion controller, pointing or gazing at it and selecting it.

##### Improved for VR

Mike Alger proposed a way to include the three dimensionality of VR to increase immersion and give additional feedback to VR buttons [VDMfVR15]. His design as shown in Figure 3.4 emphasizes the physical aspect of the button and expands it by having to move the button further than a certain point on the local z-axis. Particularly this movement along the z-axis should reduce accidentally submitting a button as the user not only needs to select and collide with it, but even push it through the threshold. Personally I'd only use the big horizontal variation of a button with the text included in VR as bigger buttons are more visible and easier selected and submitted.

##### Improved for AR

In contrast to my personal preference for bigger horizontal buttons in VR the opposite should be more beneficial for AR devices. With the severely limited screenspace, always showing the bigger buttons is a waste of space. Instead only the smaller, circular should be enough for most applications. Additionally the smaller buttons can still transform into their bigger counterparts whenever selected, in order to confuse confusion and clarify their use and action.



### 3.3.3 Notifications

Notifications are small panels with text that notify the user about actions in the applications. Coupled with other visual and auditive signals can notifications provide important and instant feedback to the user.

These notifications normally appear from the bottom of an application and disappear very quickly. However it is not always the case that the user faces the origin of the notification in AR/VR applications, therefore additional method should be implemented in order to assure that the user registered and acknowledged the notification before it disappears.

#### Improved for VR

According to the aforementioned design guidelines for VR it is not recommended to attach any UI element to the screen of the user in order to avoid discomfort.

This is why the notification appears in between the UI element the user was using and the user as seen in Figure 3.5. It is important to note that the notification is not anchored to the user, instead it behaves like any other object in the virtual environment. Due to the closeness to the user the notification panel should hide other elements behind it, using a visual overlay for all elements behind the notification could further enhance this effect.

Should the user not currently face the notification, transparent guiding arrows and a auditive clue will guide the user to it. Otherwise the notification disappears after the user looked at it for a specific amount of time.

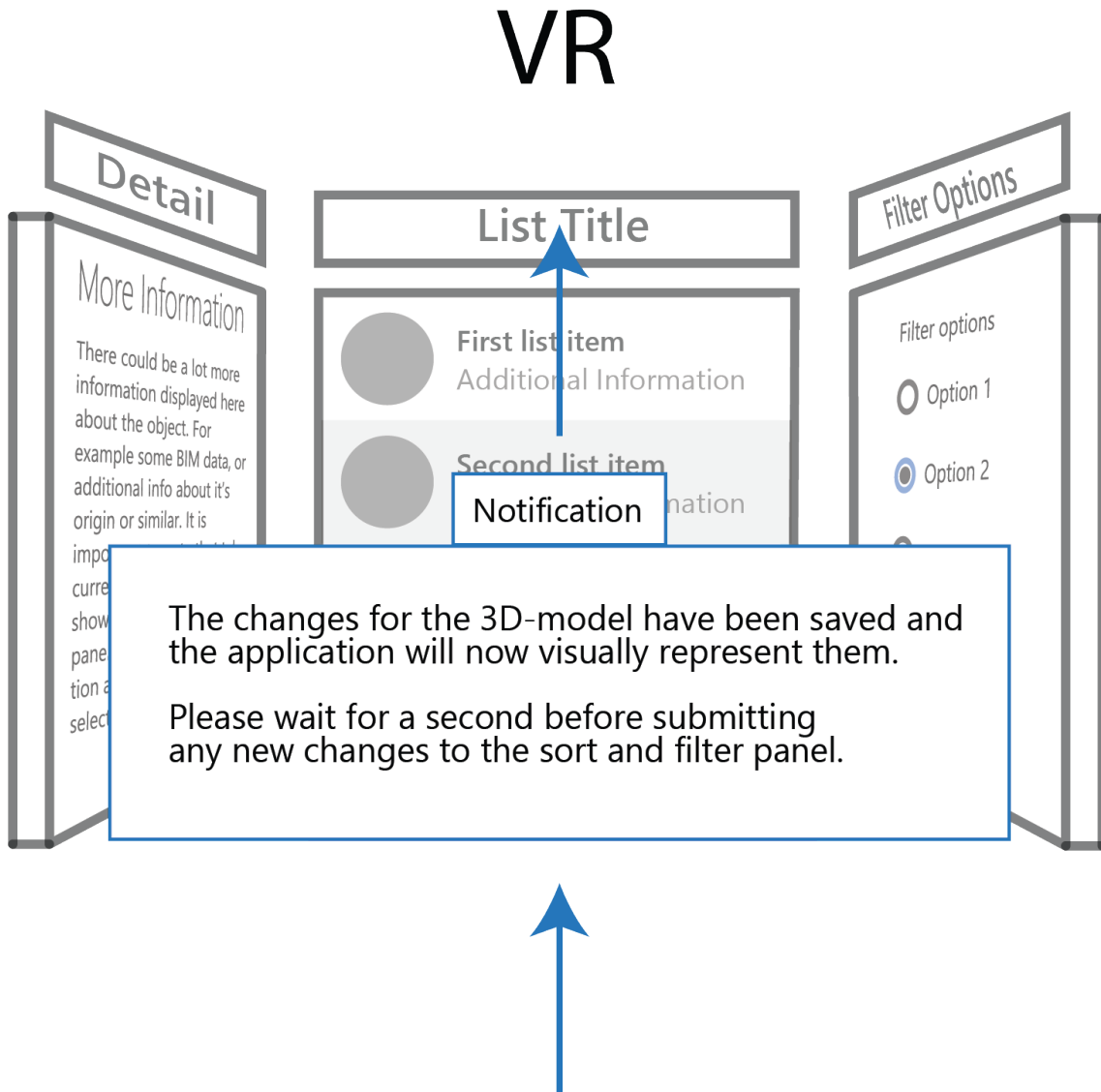
#### Improved for AR

Even though it is theoretically possible to anchor an UI element to the screen of an AR display it is still not recommended. Instead the AR notification always appear above the "notification stand" as seen in Figure 3.6. The stand can be repositioned and attached to physical objects in the real environment by using spatial mapping. Similar to the VR approach, notifications should only disappear once the user spent the appropriate amount of time looking at it and guide the user to it with transparent guiding arrows and sounds.

### 3.3.4 Detail View

The detail view appears whenever the user selects an object in the application. This view could offer more information about an object, include notes from other users or previous uses or allow for manipulation of the object.

As a rule should the object that is shown in the detail view be somehow visually modified to clarify the connection to the detail view. This is important to avoid accidentally manipulating or commenting on the wrong object, because another object was selected in the first place.



**Figure 3.5:** Notification design improved for AR

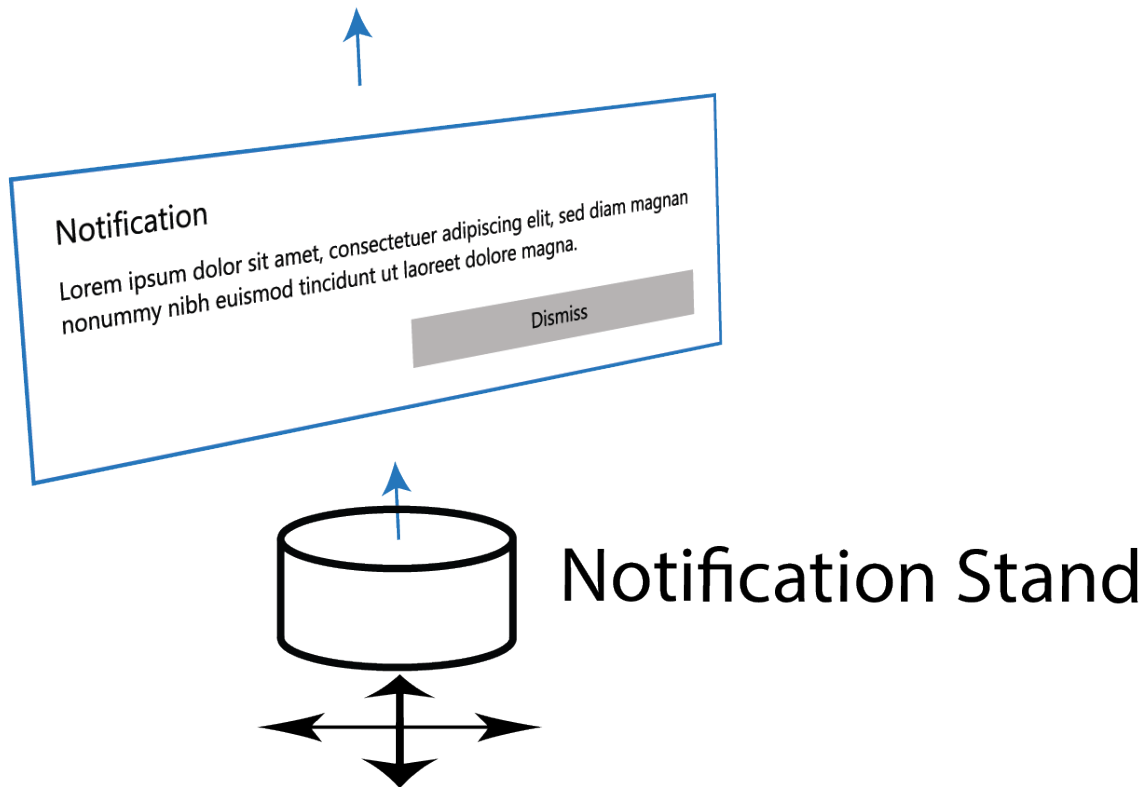
### Improved for VR

In VR a small resizable and rotatable cube with the selected object inside will be shown in front of the user whenever that user selects an object as seen in Figure 3.7.

Similarly to the list representation in VR are three separated parts shown to the user. A detail panel that includes additional information about the selected object as well as a note panel that contains all notes that users have created for the object.

In the middle of both panels is a cube, containing the selected object. The cube can be rotated and scaled by selecting its vertices.

# AR



**Figure 3.6:** Notification design improved for AR

Underneath the cube are several buttons located that could potentially allow for further manipulations of the object or usage of the note panel.

## Improved for AR

Instead of the three separated parts only the rotatable and scalable cube, containing the selected object is shown to the user as seen in Figure 3.8. This cube can be moved and attached to objects in the real environment with spatial mapping.

The buttons are of the small and circular variation and expand whenever selected. In contrast to the VR version, are more buttons available. Pressing on the additional buttons show either the detail or note panel accordingly on the face of the cube, facing the user.

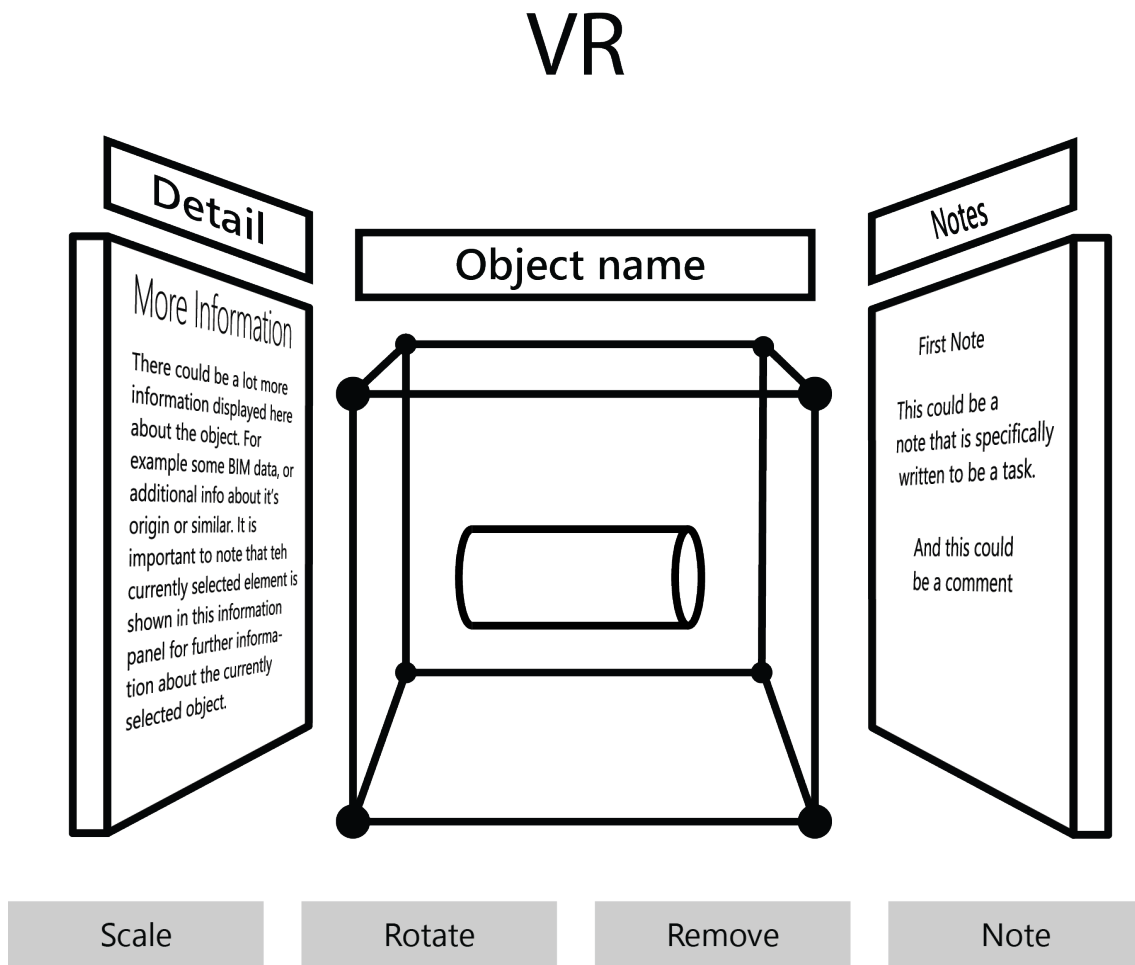


Figure 3.7: Information panel design improved for VR

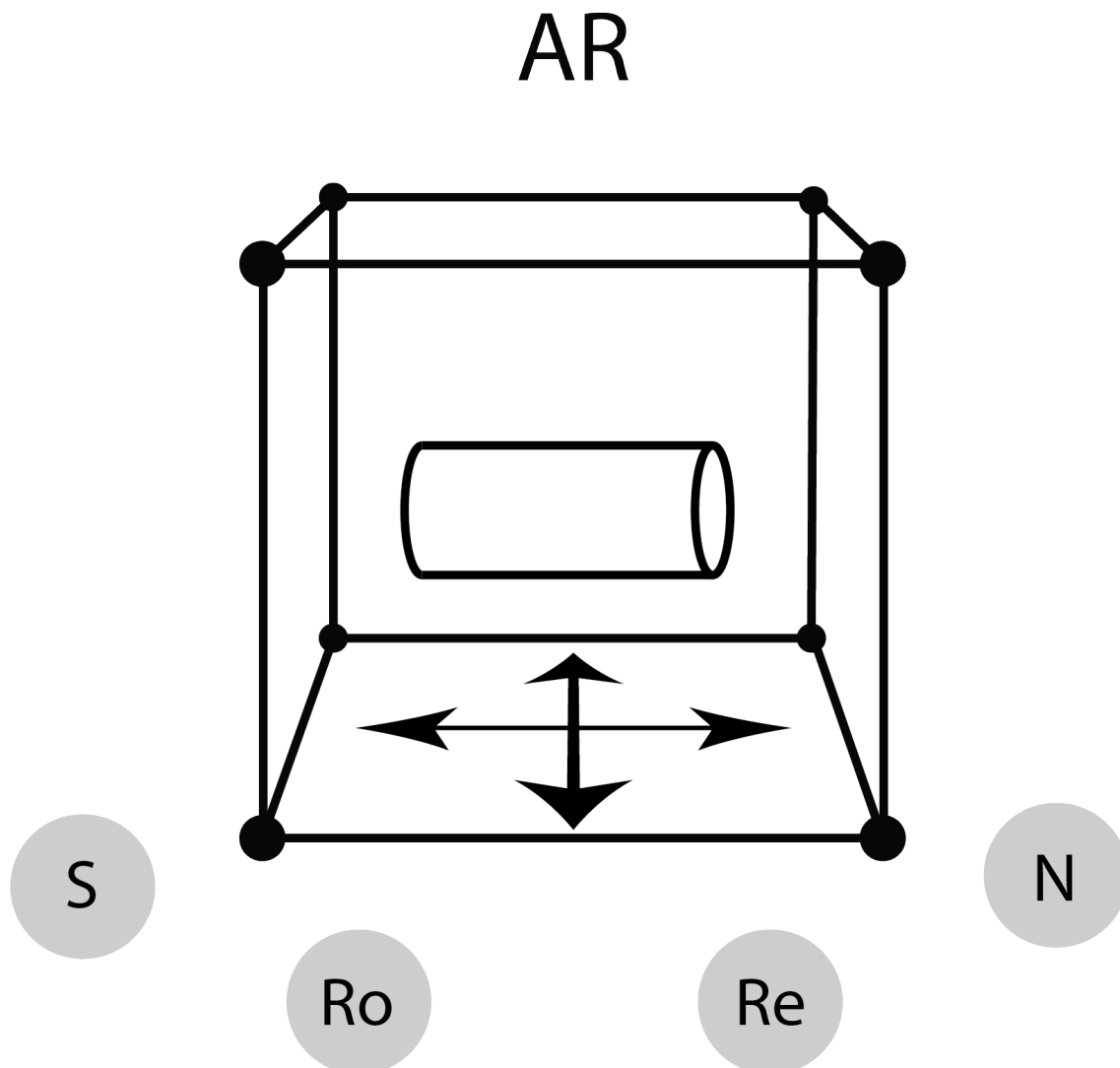
### 3.4 Additional Concepts

#### 3.4.1 Main Menu

Whenever the user might not use motion controllers or similar as an input device a main menu could provide useful actions to the user. These actions could be displayed as a small menubar with several buttons, visible to the user at all times.

#### Improved for AR/VR

Instead of a small menubar, a cube similar to Figure 3.3 could be used with the same interaction principles. Unlike other objects this cube could follow the user around and whenever an object is selected, the cube could display the buttons instead of the detail view. The cube can be positioned at any set relative distance to the user.



**Figure 3.8:** Information panel design improved for AR

### 3.4.2 Exploding view

The exploding view allows for an easier selection of smaller elements hidden underneath or behind other elements [ExplodedViews17]. By using the "Explode" Voice command while looking at a model, or alternatively pressing the explode button either on a motion controller or the main menu. The exploded view includes a list, containing all elements that are currently exploded above the model. By filtering the elements in the list as well as the elements above the model get filtered accordingly. This could potentially be used to allow the user to filter through the model for objects with specific Building information modeling (BIM) values.

### 3.4.3 Smart Controller

Because of the limited screen space for AR another display for showing the panels that needed to be collapsed to buttons or otherwise in the aforementioned designs could vastly improve the user experience.

Although using a tablet device or similar for this purpose could be very helpful, it is important to note that in order to allow for an external device to display the data the user currently selected, a network connection or similar must exist between the Head-mounted display (HMD) or PC and the target device.

As a result the user could potentially start the application on a tablet device and select the "Smart Controller" mode for it. The tablet then becomes an additional input device and serves as an additional panel.

## 4 Existing Tools

### 4.1 Unity

Since the tools were to be implemented in Unity, all tools in this section are available as plugins for or already part of Unity [Unity].

#### 4.1.1 VRTK

VRTK is a collection of helpful prefabs and scripts for developing VR applications in Unity [VRTK]. It also includes a VR Simulator for testing. It was originally intended to exclusively support VIVE and Oculus development processes but has since expanded its support to other VR devices as well. It is Open-source software but so far mainly developed by a single developer and even though Oculus recently supported the development financially, it is unclear if and how long the development will continue. Currently VRTK does not offer ready to ship solutions for common interactions or design problems, instead it gives developers a solid framework to build upon - even though in my personal experience extending VRTK scripts can be rather tiresome and more documentation would be appreciated. Finally VRTK mostly focuses on fun interactions that should be used as a starting point for developing games, precision and efficiency are normally not the focus.

#### 4.1.2 NewtonVR

A commonly cited alternative to VRTK is NewtonVR [NewtonVR]. NewtonVR was developed in order to allow developers to quickly implement physic based interactions in their VR applications. Development has not been continued since January 2018 although it has not been officially discontinued yet. NewtonVR can be considered a helpful addition to VRTK but in my opinion it does not provide enough helpful tools that have not been already covered in other, more extensive toolkits.

#### 4.1.3 EditorXR

EditorXR is a project developed internally by the Unity team [EditorXR]. It allows users to run the Unity Editor as a VR application and is a helpful tool for tweaking minor details at the end of development. So far it is suffering from severe performance issues on most systems and is not actively being developed even though it periodically receives bug fixes or similar. Nevertheless EditorXR offers interesting methods on how to design UI for professional use.

### 4.1.4 Windows Mixed Reality Tools

Microsoft's Mixed Reality Tools offer a wide variety of tools for developing AR/VR applications in Unity [MRTK]. In addition to several UI components according to Microsoft's "Fluid design guidelines" it also provides an input system that supports gestures and gaze inputs and controller support for motion controllers. Additionally it allows for quickly testing out applications with the built in emulator. Their coordinate system and spatial sounds as well as spatial mappings allow for a solid foundation to build upon for developers that just started developing for AR/VR. MRTK is also available for other platforms beside Unity.

Personally I found the MRTK to be rather useful in the beginning, but some of its features are tightly coupled with other scripts that hindered extending upon it.

### 4.2 SteamVR

The SteamVR plugin is a necessity for developing VR applications in Unity [SteamVR]. It provides the necessary connection to the SteamVR software in order to receive tracking data of the user. It is constantly developed, even though sometimes changes can break the most recent version due to frequent name changes. Additionally it includes several examples on how to interact with objects in VR as well as 3D-models for the VIVE controllers and animated hands.

In my experience the connection to the SteamVR software and the tracking data works almost flawless, but the examples are poorly structured and not developed with the design philosophies that Unity encourages and are therefore hard to build upon. Sometimes even changing an exposed variable in the Unity inspector went as far as break several other components as well.

### 4.3 Vuforia

The Vuforia Engine is already integrated in Unity and allows for developing AR applications [Vuforia]. Vuforia provides several helpful tools such as the AR Camera and images that need to be scanned by the AR device in order to render the desired 3D-model on top of it. Consequently it is perfectly well suited for speeding up development for AR applications targeted at smartphones and tablets, but it does not offer a lot of help for HoloLens development.



# 5 Implementation

## 5.1 Requirements

**Navigation Tool:** The user should be able to get closer to the 3D-model as well as further away from it. It should be possible to create persistent points of interests that the user can quickly navigate to also a list of all objects that are part of the 3D-model should function as a way to quickly inspect each and every one of these. In VR basic movement by teleporting or locomotion should enable the user to inspect different angles of the 3D-model better. In AR similar methods allow the user to inspect the entirety of the model without having to walk the actual distances.

**View Tool:** In order to be able to look at the 3D-Model from all angles and different distances the user should be able to make parts of the model invisible and highlight other parts of interest. Also a miniature model should help the user view the bigger picture without necessarily having to move or zoom on objects at all.

**Manipulation Tool:** A very simple sort of brushing and linking should be implemented as well, allowing the user to immediately see what impact changing certain variables could have on the 3D-Model.

**Selection Tool:** Any object that is part of the 3D-Model should be selectable in order to see any data associated with it like for example the BIM of that object. Additionally notes and tasks need to either be attached to objects or reference them.

**Note Tool:** The user should be able to create notes and attach them to a specific position or to an object itself. These notes can either be written down as text or recorded as speech. It is important that these notes are persistent and can be easily seen by all other users, as they should be able to comment on these. Notes can consist of warnings, errors, todos and reference data of the object it is attached to.

**Management Tools:** Users should be able to directly create tasks to the project management tool used, like for example Trello. These tasks could potentially be notes themselves but don't need to be. Tasks should also be able to include data of any object it references. This data could for example be the BIM of a structural object.

## 5.2 Technology and Libraries

**Unity:** Unity was used for implementing the tools as introduced in the requirements. Unity allows for rapid prototyping in AR/VR applications by simply using OpenVR or MRTK or other tools that are available for Unity. All scripts for Unity were written in C#

using C# 6 and .Net 4.x compatibility. A private Github repository was used for source code version control. The project is developed, built and executed primarily on Windows 10 (64-bit).

**Zenject:** Zenject is an open source dependency injection framework primarily developed for Unity [Zenject]. It was primarily used in order to decouple code according to the model-view-controller design pattern by using its signal based system.

**Scriptable Objects:** were used to allow for a simple way to extend the provided tools and UI and to allow for persistence.

**Windows Mixed Reality Toolkit:** was used mostly for its UI elements and prefabs as well as the input keyboard.

### 5.3 Collaboration

The implemented tools can be used in collaboration with other users in order to work on large-scale 3D-models either sequentially or at the same time.

#### 5.3.1 Sequential

In sequential collaboration one user can use the application and modify it or add notes to specific parts of the element, which get persistently saved. Another user can then load the savefile afterwards and see the notes and modifications made by the other user.

#### 5.3.2 Parallel

Parallel collaboration is achieved by letting several users look at the model at the same time. This was achieved by setting up a simple server-client model with UNet [Unet] Additionally it potentially allows for the use of holographic remoting for the HoloLens. One caveat is that networking with UNet tends to be quite error-prone, therefore in order to avoid false updates of manipulated elements, only one user can effectively manipulate objects and use all tools. Others can only get more information about a model, add notes and use the navigation tool.

### 5.4 Note Tool

The note tool is important for both types of collaboration as it allows the user to write or record notes and attach these to elements in the model. The persistence is achieved by using Unity's scriptable objects and saving and loading them as .json files on runtime.

### 5.4.1 Adding Notes

By clicking on the Add Note button, one can either write a text, by selecting the text panel. A small keyboard should appear in front of the user, allowing them to type out the note. Alternatively the user could click the record button and voice record a note message. The voice message gets transformed into text by the IBM Watson speech sdk available in Unity [IBMWatson] and can be listened to as well.

### 5.4.2 Overview and searching notes

In order to get a quick overview for all notes in the application the note button in the main menu can be used, displaying a list of all existing notes. These notes can be filtered and the user can search for specific notes by using an input text field. The overview allows for adding, deleting and editing as well as commenting notes.

### 5.4.3 Editing and Commenting

When an existing note has been selected by the user, the note can either be revised and edited or commented on.

## 5.5 Management Tool

The note tool can also be integrated in project management tools like Trello [Trello]. After setting up the integration in the settings of the main menu, any note added will be created on the assigned board. This works in both directions. New comments or notes will be pulled and updated in the application as well.

## 5.6 Input

In order to stay consistent with the Unity way of things the Unity interaction handlers have been used. The following interaction handlers were originally intended to be used with mouse or touch only, but now also support the necessary AR/VR inputs.

### 5.6.1 Select

**IPointerEnterHandler:** will be called whenever the user either points at an element with the motion controller laser or with the air tap laser. This sets the `EventHandler.CurrentlySelectedElement` to the desired element.

**IPointerExitHandler:** called whenever the user leaves the collider of an element with the motion controller laser or air tap laser. Sets `EventHandler.CurrentlySelectedElement` to null.

**IPointerClickHandler:** Gets called whenever the submit button is clicked or air tap is triggered and `EventHandler.CurrentlySelectedElement` is not null.

### 5.6.2 Manipulation

**IPointerBeginHandler:** As long as the select button has been pressed and not released yet - or the air tap respectively, the `IPointerBeginHandler` is called. This can be used for all sort of dragging motions.

**IPointerEndHandler:** Whenever the select button has been released or the air tap has been released the `IPointerEndHandler` is called. This was used for dropping elements or terminating rotations, scalings and other manipulations of objects.

## 5.7 Optimizations

### 5.7.1 AutoLOD

AutoLOD is an experimental tool released by the unity team that handles automatic LOD generation as well as certain scene optimizations [AutoLOD]. It was mostly used for its scene optimization capabilities and its tools for handling LOD in Unity. Additionally there are tools for merging textures as well, in order to reduce draw calls. Ultimately the automatic Level of detail (LOD) generation was dropped in favor of the naive approach of disabling elements by size and letting the user decide which objects should be displayed or not.

### 5.7.2 Naive approach LOD

Because of the severe hardware limitations of the HoloLens and size of the given model, a system for level of detail had to be implemented as well. By default elements with a small collider volume don't get shown in the scene unless the user get's close enough to it or modifies the filter settings in the manipulation tool.

### 5.7.3 Edge collapsing

A script for edge collapsing and therefore mesh simplification was used in order to allow for the amount of colliders used in the project to work properly [MeshSimplification].

## 6 Results

### 6.1 Library of useful UI Elements

#### 6.1.1 Lists

Figure 6.1 shows the implementation of a list, that was specifically designed for VR. By selecting and submitting an element of the list in the middle panel, the detail panel to the left immediately changes its title and text contents to the corresponding values of the selected element. Scrolling is achieved by dragging up or down the middle panel, a scroll bar might be helpful in order to visually inform the user that this interaction exists. The apply button under the filter panel applies the current filters to the list and potentially to the 3D representations of the list elements as well. The search button causes the MRTK keyboard to appear and allows the user to search for a specific list element. In the event that the user needs to hide one of the panels, the header of the list can be selected and submitted in order to hide the panel. Closing the entire list view can be achieved by selecting and submitting the hide button below the center panel.

Figure 6.2 shows the AR version of the same list. In order to cater for the smaller screen size the three panels have been collapsed onto the faces of a cube. The cube can be rotated left and right in order to switch between the panels. Rotating the cube slightly up and

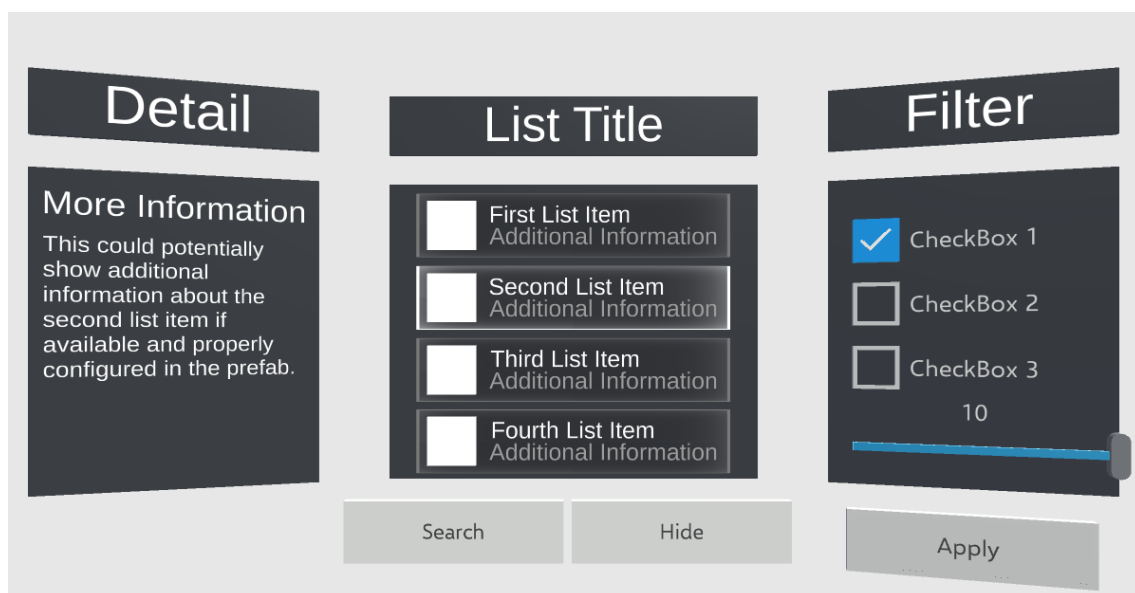


Figure 6.1: List implemented for VR

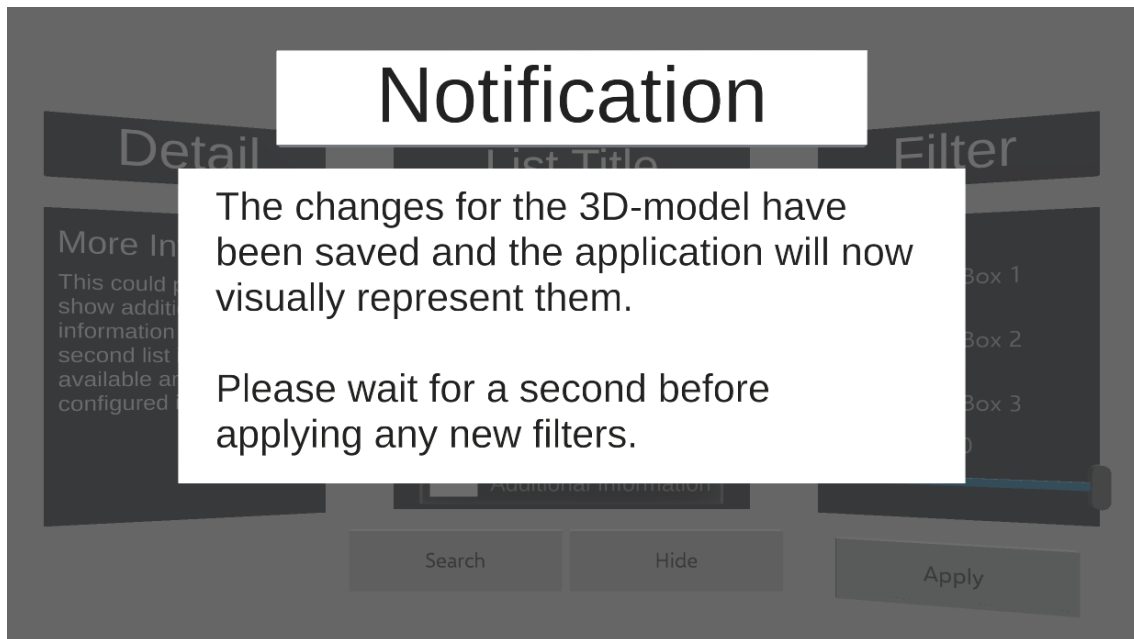


**Figure 6.2:** List implemented for AR

down can trigger actions for the panel that is currently facing the user, such as scrolling or applying the current filters. With the cube it is neither necessary nor possible to hide one of the panels.

### 6.1.2 Notifications

Notifications can be used to inform the user as seen in Figure 6.3. While the notification slides up from below to the height of the user's head, the rest of the application gets darker in order to let the user know that a notification is appearing. Additionally guiding arrows coupled with a sound are used to guide the user to the notification if they are not facing it already. In order to limit distracting the user from their work this should be used sparingly and only whenever vital informations to be communicated.



**Figure 6.3:** Notifications implemented for VR

In AR notifications appear on top of the notification stand as seen in Figure 6.4. The notification stand can be positioned anywhere in the real environment of the user. Similarly to the VR notification, guiding arrows as well as a sound are used in order to guide the user to the stand whenever a new notification appeared. In both cases will the notification disappear after the user has looked at it for a configurable amount of time.

### 6.1.3 Detail view

The detail view is a variation of the list as it includes the detail panel, but instead of the actual list, the object is shown inside of a bounding box and notes where the filters were shown, as can be seen in Figure 6.5 for VR. The buttons below allow for quick manipulations of the currently selected object as well as a switch to the note tool with the "Add Note" button. The connection to the highlighted object in the 3D-model is visually represented by coloring both, the actual object that was selected as well as the copy that is shown to the user in the detail view green and connecting both with a green line.

Slight changes to the detail view had to be implemented for AR depicted in Figure 6.6. The rather long toolbar was replaced with smaller buttons and the note panel is not shown at all until the note button was used. In addition needed the spherical handles of the bounding box containing the object to be bigger in order to be easily selectable in AR.

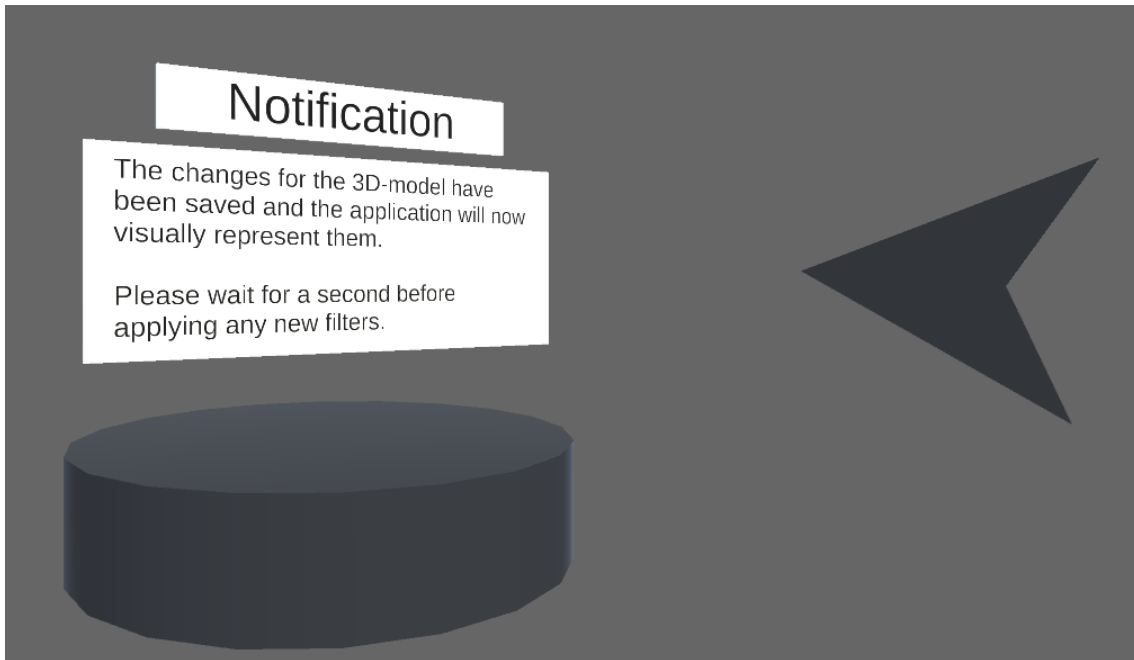


Figure 6.4: Notifications implemented for AR

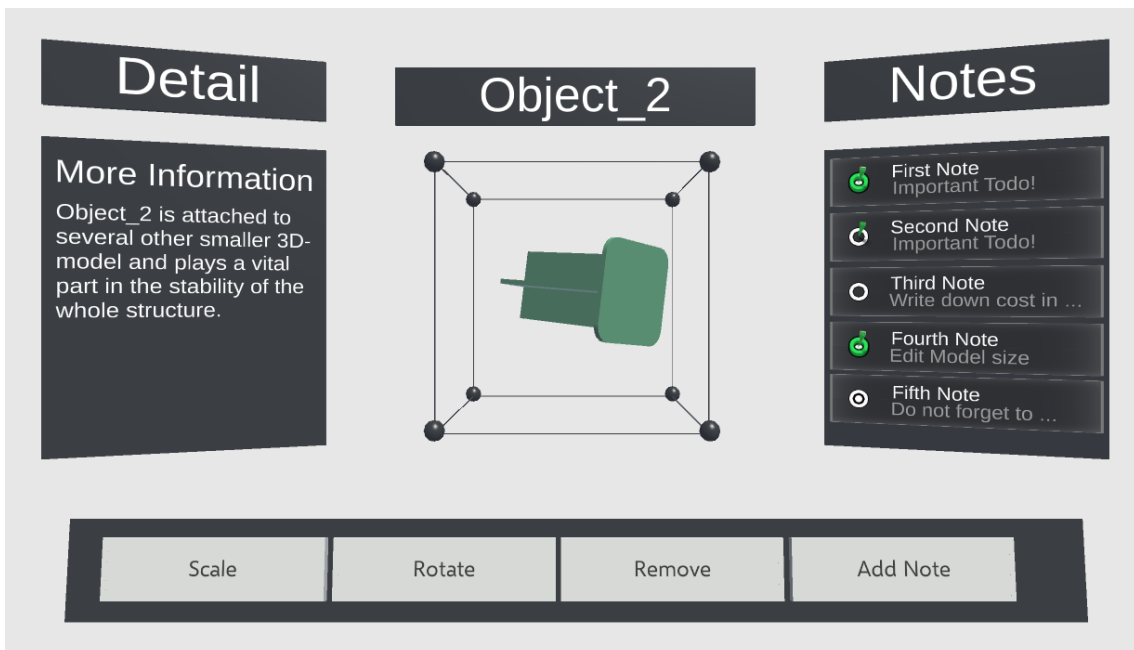


Figure 6.5: Detail view implemented for VR



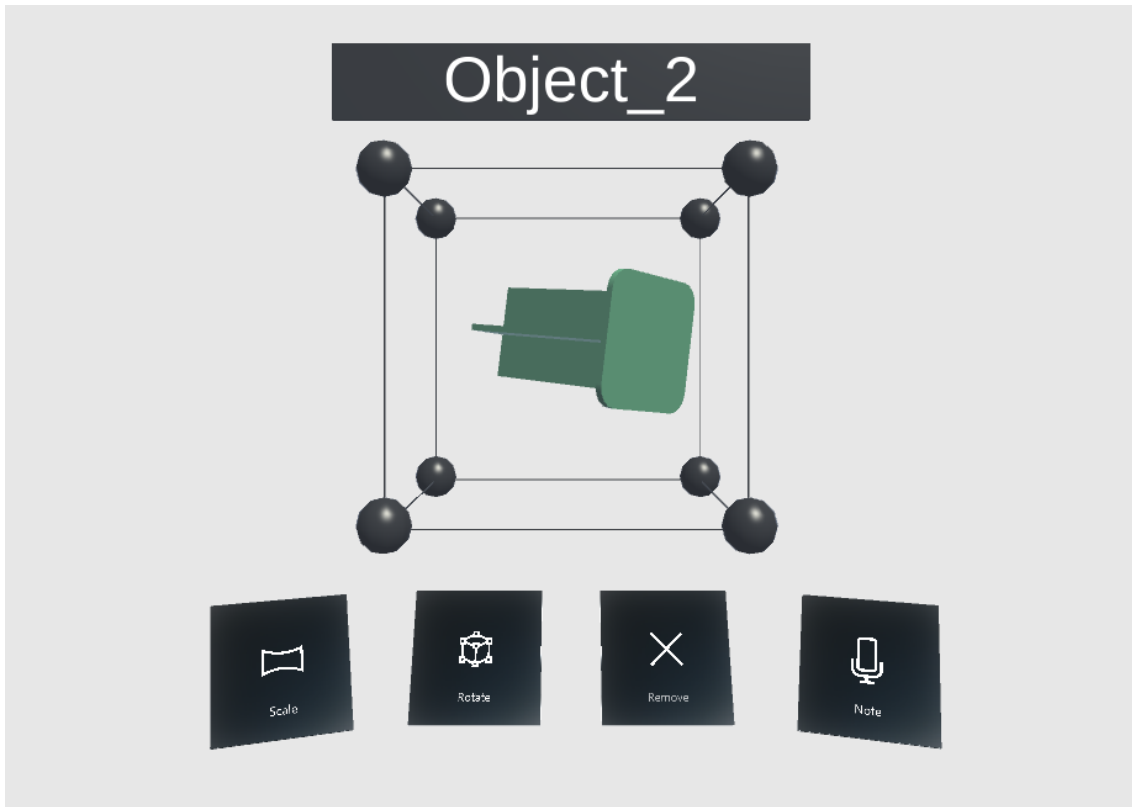


Figure 6.6: Detail view implemented for AR

### 6.1.4 Main Menu

The main menu can be shown to the user whenever one of the tools need to be accessed as seen in Figure 6.7. This menu is similar for AR and VR seeing that the original VR version was more time consuming to use due to it being a very long toolbar of horizontal buttons whereas the smaller version is easier to use and obstructs the view of the user to a lesser extent.

## 6.2 Tools for Reviewing and Editing

### 6.2.1 Manipulation Tool

The manipulation tool can be used to filter out unwanted parts of the 3D-model. In Figure 6.8 all elements are still shown due to the fact that the checkboxes corresponding to bigger groups of objects in the 3D-model are selected. Additionally the slider filters out all objects that are larger than the current value of the slider. In this case no objects were filtered out as no object is actually larger than the value.

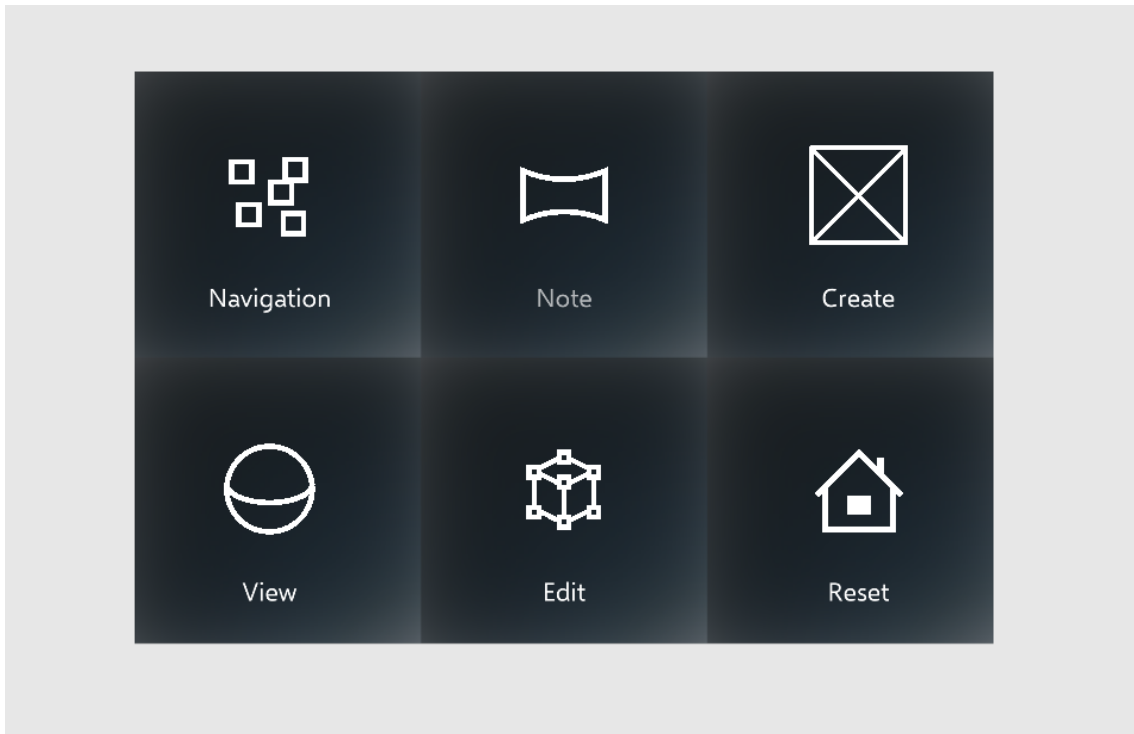


Figure 6.7: Main menu implemented for AR/VR

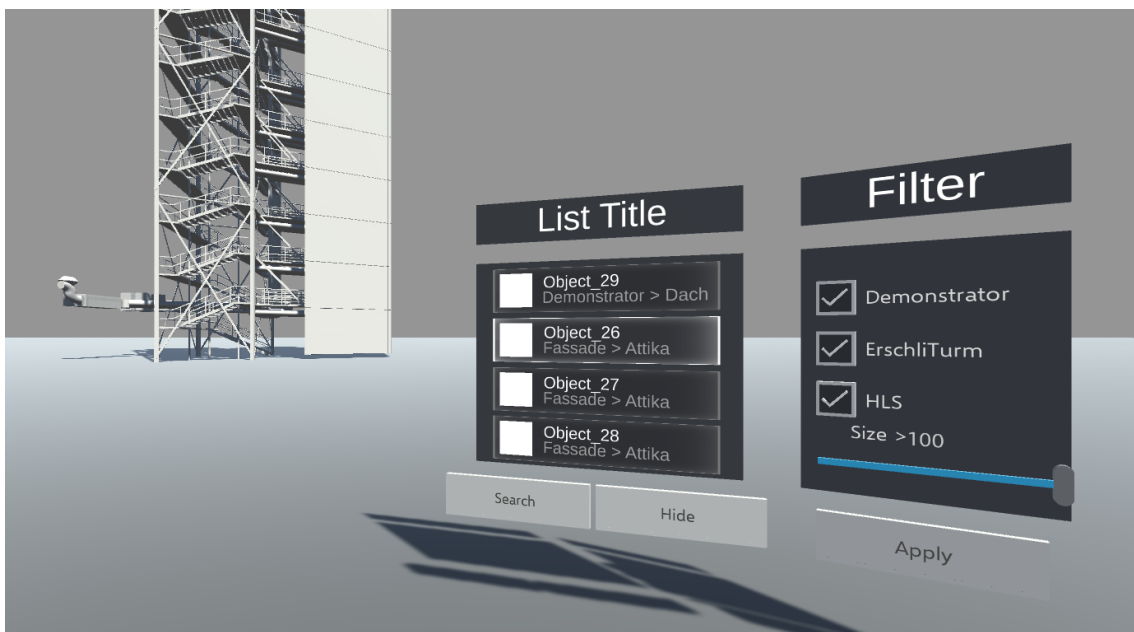
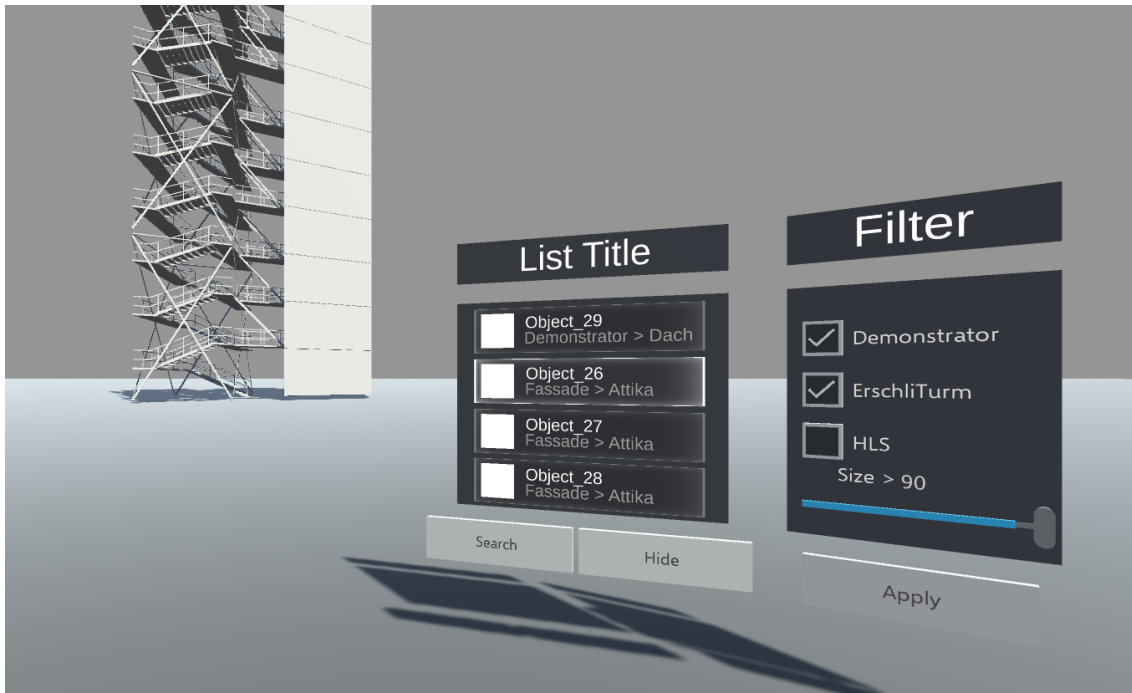


Figure 6.8: Before the manipulation tool was used to filter the list and model



**Figure 6.9:** The resulting model after the manipulation tool was used

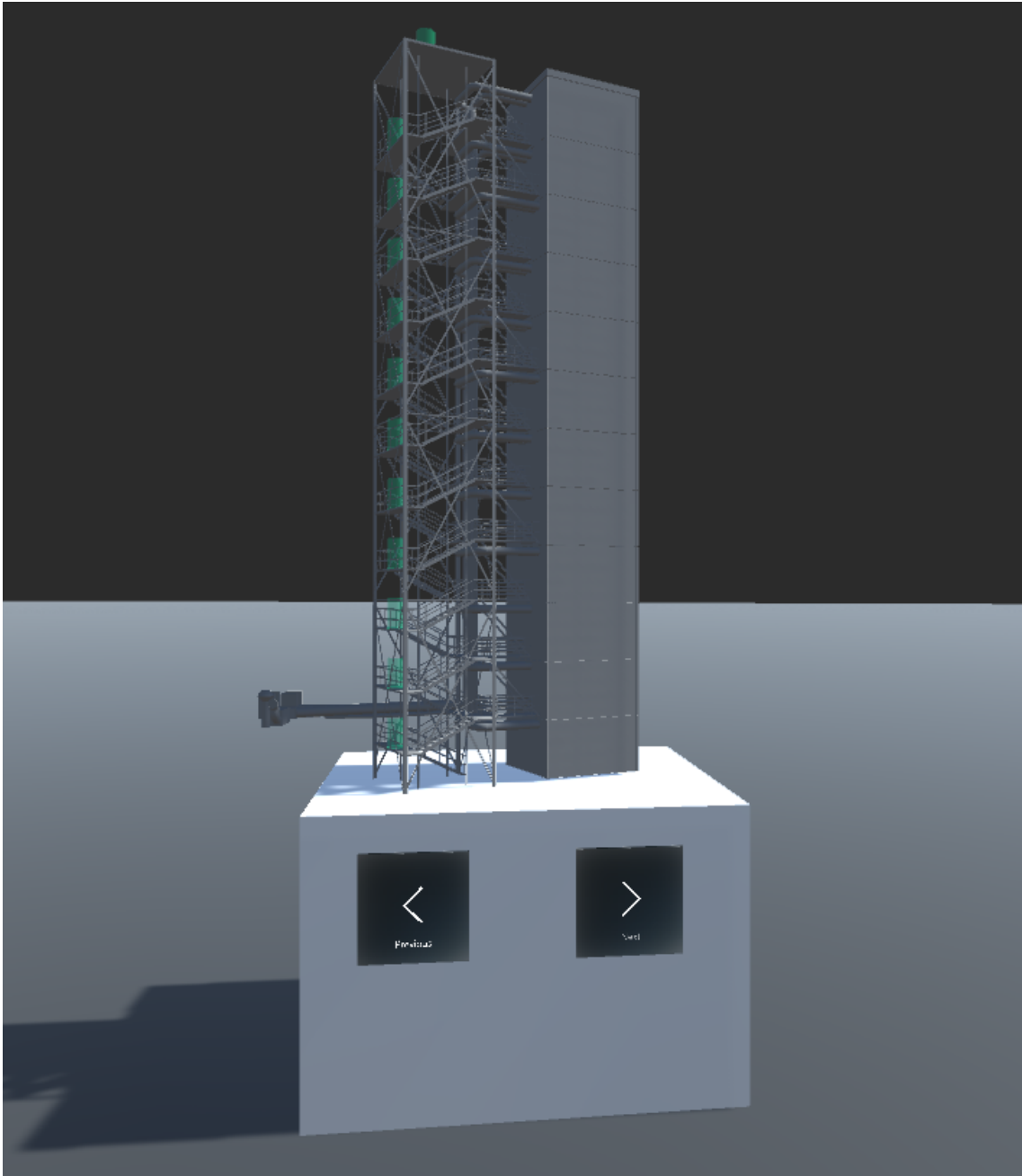
Figure 6.9 depicts the same example but after the manipulation tool was used to filter out unwanted objects. The "HLS" part of the 3D-model has been unchecked and is therefore not rendered anymore identically to all other objects that are larger than the slider value.

### 6.2.2 Navigation Tool

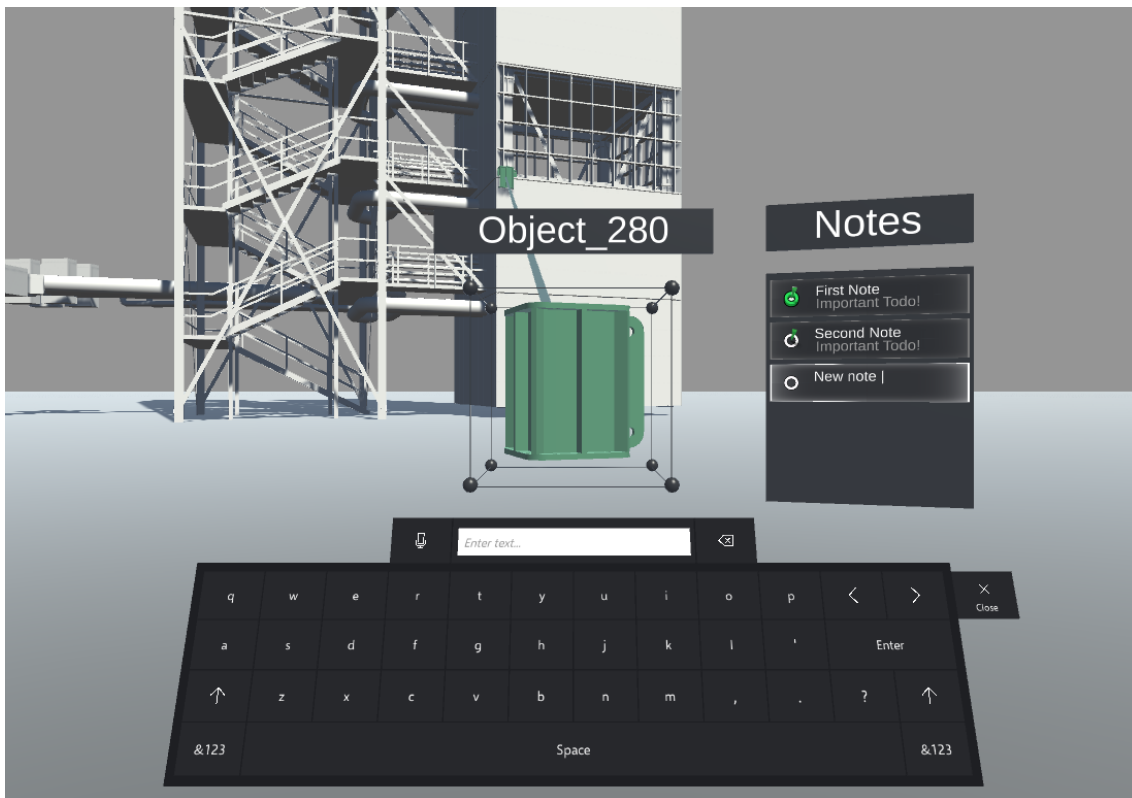
The navigation tool can be used to quickly move around large-scale 3D-models as seen in Figure 6.10. A miniature model is used that allows the user to teleport to any position by simply selecting and submitting. Additionally points of interests were added to the miniature model that allow for a teleport to a specific position and rotation by selecting and submitting on the green cylinders that can be seen in the model. Alternatively the buttons below the model can be used to quickly jump from one point of interest to the next as well.

### 6.2.3 Note Tool

The note tool can be selected from the detail view whenever an object was selected and submitted as shown in Figure 6.11. A new note can be created by using the "Add Note" button below the notes panel and typing out the title and description of the note. Clicking on the icon next to a note changes its state, these states can be configured in order to support different workflows. Whenever a note is created a corresponding task will be added to the corresponding Trello board should an integration exist.



**Figure 6.10:** A miniature version of the large-scale 3D-model that allows for quick navigation



**Figure 6.11:** An element has been selected and a new note is being added to it

#### 6.2.4 View Tool

The view tool can be used to hide or highlight objects from a 3D-model Figure 6.12. This allows the user to easily look through otherwise blocking objects and highlight objects for later use or emphasis. Highlighted objects can be turned to normal objects by highlighting them again. In order to show hidden objects the show button must be used, indicating outlines for all objects that were previously hidden.

### 6.3 Collaborative Work

These tools can be used to allow for collaborative work on a large-scale 3D-model. It is important to emphasize that whenever more than one user work on a 3D-model at the same time, only the first one is allowed to use all tools, other users are restricted to the navigation tool only in order to avoid common difficulties with multiuser applications.

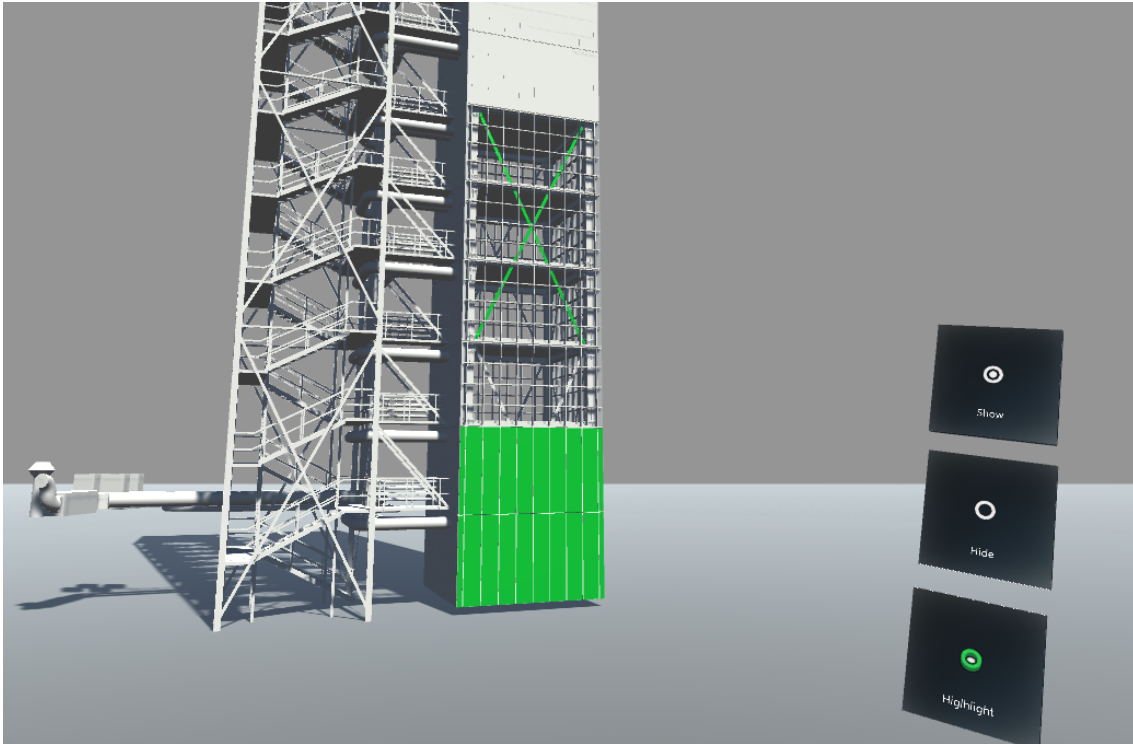


Figure 6.12: The view tool allows for hiding objects as well as highlight them

## 6.4 Difficulties

### 6.4.1 VR-First Design

By planning the interactions and designs for VR first, many changes had to be made afterwards to properly accommodate these designs to AR. Generally speaking most UI/UX designs for AR could work the very same way for VR. Yet VR designs definitely do not necessarily work seamlessly for AR devices as the input as well as the screen size of these devices can be severely limiting. Additionally several design guides for VR devices already exist, while there are not as many AR design guidelines publically available. The Windows AR design guideline mostly focuses on how to use its Application programming interface (API) and implement common interfaces, most recently a big overhaul of several design elements was even implemented in the Windows Mixed Reality Portal [WMRPDesignChanges19] that were expanding upon the fluid design guideline which is yet to be updated.

### 6.4.2 Pitfall: Existing Tools

The existing toolkits mostly only offer usable prefabs for very narrow use cases as well as exclusively for either AR or VR devices. Additionally these tools tend not to work very well together and could lead to problems whenever multiple toolkits are used in the same project. Therefore most of the tools ultimately only used for inspiration in UI/UX design after I had wasted a lot of time on trying to make them work for my use cases.

### 6.4.3 AR Hardware limitations

The biggest difficulty was definitely developing for the HoloLens. The screen size severely limits UI/UX Design and the input possibilities were limiting enough to cause the proposition of using another device for an additional input device as well as to expand on screen size.

### 6.4.4 AR and VR are not yet properly released or commonly used

Most of the released AR devices do focus mostly on the visual representation of data on the glass and not on gesture input or any sort of spatial mapping like the HoloLens does. Therefore the amount of AR glasses with similar capabilities to the HoloLens is marginal causing support to be severely limited, and so far there is no indication that this will change anytime soon.

A common problem, that is not particularly limited to AR/VR hardware is that whenever a newer product is released, users with less common CPUs or GPUs are often not able to properly make them work on their systems on release, potentially even leading to product returns. In fact, quite recently VIVE issued an official statement that they received multiple issues about the VIVE Wireless Adapter and therefore accepted returns from users with Ryzen-based PCs [VIVEAMD18]. Finally the HoloLens does not even work with all versions of Windows 10, as "N" versions do not include necessary software, only to be fixed by the installation of another Windows version [WMREG17].





## 7 Summary

Before interaction concepts were evaluated and designed, common definitions, design foundations and input possibilities were researched and explained. With this in mind common designs from current design guidelines were used and modified in order to properly work with the individual requirements of AR and VR applications. Additionally the most common user interactions were defined and several fitting methods for handling AR and VR input were explained.

Then existing toolkits and plugins were examined and evaluated, leading to the conclusion that while all of them offer a starting point for getting into AR and VR development with Unity, they are rarely modular or extensible enough to build a full scale application upon, instead they are mostly suited for rapid prototyping.

This concluded in the implementation of several UI elements for AR/VR applications in Unity that comply to the design principles, fulfill the requirements and were developed with modularity and extensibility in mind. In order to successfully implement the UI elements and their functionalities for the currently available AR/VR hardware, several methods on how to optimize large-scale 3D-models and on how to visually represent these on the HoloLens and VIVE were examined and explained. Given a large-scale 3D-model, developers could use the developed library of UI elements in order to quickly allow their users to navigate around and within their model, inspect specific elements, interact with the model and to collaboratively review and edit it by adding or modifying existing notes that can also be integrated on their Trello board.

The resulting library was developed as part of an Unity Project in Unity 2018.2.7 and implements the requirements as stated in that chapter. Zenject, SteamVR and MRTK are all free assets on the Unity Asset store that were used in the development process of the project.

Finally common difficulties from my personal experience were documented and elaborated upon.



## 8 Future Work

Conducting an eye-tracking user study by letting the participants follow an example workflow where they would have to work quick and precise as well as collaboratively could help getting more insights about the advantages and disadvantages of the different interaction concepts and implementations. In order to be able to properly evaluate the eye-tracking data, three-dimensional heatmaps with temporal aggregation could be used for the visual analysis. It would also be interesting to compare the different navigation and interaction interfaces with a similar taskload on a PC.

Most of the implementation details that are specific to the current hardware limitations of the HoloLens would need to be reevaluated as soon as the next version of the HoloLens is available. Especially performance optimizations like the dynamic level of detail system might have a negative impact on usability for the AR application and could therefore potentially be reduced or left out entirely. At the present time a thorough investigation of the capabilities of holographic remoting frameworks for AR could be evaluated.

The library of useful AR/VR UI elements and interactions has been developed with modularity and extensibility in mind and could certainly be further expanded upon, especially with the ever growing releases of new displays and input devices and consequently new requirements.

Connecting several input devices, for example tablets or smartphones as "Smart Controllers" with the actual AR/VR application might further improve the collaborative aspect by being able to share snapshots of the three dimensional aspect on the smartphone or guide an unexperienced user through the model with the tablet. Additionally moving the miniature model out of the AR/VR application and onto a separate device might be a helpful way of increasing possibilities for collaboration.

Finally the note and management tool could be further developed to offer integrations for all sorts of common management systems. The possibility to export and import notes might also be of interest for some projects and might be further improved by using more input devices.



# Bibliography

- [AutoLOD] AutoLOD. *Automatic LOD generation + scene optimization*. URL: <https://github.com/Unity-Technologies/AutoLOD> (cit. on p. 36).
- [BimanualAR18] *Bimanual Gestures on the HoloLens*. 2018. URL: <https://www.youtube.com/watch?v=XXhjZJoGo1Y> (cit. on p. 21).
- [CF11] J. Carmigniani, B. Furht. “Augmented Reality: An Overview”. In: July 2011, pp. 3–46. DOI: [10.1007/978-1-4614-0064-6\\_1](https://doi.org/10.1007/978-1-4614-0064-6_1) (cit. on p. 19).
- [EditorXR] Unity. *Editor XR for Unity*. URL: <https://github.com/Unity-Technologies/EditorXR/> (cit. on p. 31).
- [ExplodedViews17] Y. W. Zhutian Chen Huamin Qu. *Immersive Urban Analytics through Exploded Views*. 2017. URL: [http://chenzhutian.org/upload/papers/2017\\_immersiveexplodedviews.pdf](http://chenzhutian.org/upload/papers/2017_immersiveexplodedviews.pdf) (cit. on p. 29).
- [HololensTracking17] *Inside-out tracking*. 2017. URL: <https://docs.microsoft.com/en-us/windows/mixed-reality/enthusiast-guide/tracking-system> (cit. on p. 17).
- [IBMWatson] IBM. *VR speech sandbox vive*. URL: <https://github.com/IBM/vr-speech-sandbox-vive> (cit. on p. 35).
- [Lighthouse16] Oliver. *Lighthouse tracking examined*. 2016. URL: <http://doc-ok.org/?p=1478> (cit. on p. 17).
- [LM18] LeapMotion. *Leap Motions software and hardware*. 2018. URL: <https://www.leapmotion.com/technology/> (cit. on p. 16).
- [MaterialList18] Google. *Material Design Lists*. 2018. URL: <https://material.io/design/components/lists.html#> (cit. on p. 21).
- [MeshSimplification] *Mesh simplification*. URL: [http://wiki.unity3d.com/index.php/Mesh\\_simplification\\_\(for\\_MeshCollider,\\_lossless\)](http://wiki.unity3d.com/index.php/Mesh_simplification_(for_MeshCollider,_lossless)) (cit. on p. 36).
- [MIA18] M. Precision. *MIA + Track Package*. 2018. URL: <http://motorizedprecision.com/mia-track-full-package/> (cit. on p. 16).
- [MRGuideline] Microsoft. *Design for mixed reality*. URL: <https://docs.microsoft.com/en-us/windows/mixed-reality/> (cit. on pp. 15, 16).
- [MRTK] Microsoft. *Mixed Reality Toolkit for Unity*. URL: <https://github.com/Microsoft/MixedRealityToolkit-Unity> (cit. on p. 32).

- [NBW18] N. Norouzi, G. Bruder, G. Welch. “Assessing vignetting as a means to reduce VR sickness during amplified head rotations”. In: Aug. 2018, pp. 1–8. DOI: [10.1145/3225153.3225162](https://doi.org/10.1145/3225153.3225162) (cit. on p. 17).
- [NewtonVR] *Overview of NewtonVR*. URL: <https://newtonvr.readme.io/docs/overview-of-newtonvr> (cit. on p. 31).
- [Quill18] M. Schaefer. *Designing in VR*. 2018. URL: <http://www.mattschaefersdesign.com/designing-in-vr/> (cit. on p. 14).
- [RP17] P. Renner, T. Pfeiffer. “Attention guiding techniques using peripheral vision and eye tracking for feedback in augmented-reality-based assistance systems”. In: *2017 IEEE Symposium on 3D User Interfaces (3DUI)*. Mar. 2017, pp. 186–194. DOI: [10.1109/3DUI.2017.7893338](https://doi.org/10.1109/3DUI.2017.7893338) (cit. on p. 17).
- [SteamVR] Valve. *SteamVR Plugin for Unity*. URL: <https://assetstore.unity.com/packages/tools/integration/steamvr-plugin-32647> (cit. on p. 32).
- [Trello] Atlassian. *Kanban Board*. URL: <https://trello.com/en> (cit. on p. 35).
- [Unet] Unity. *Multiplayer and Networking*. URL: <https://docs.unity3d.com/Manual/UNet.html> (cit. on p. 34).
- [Unity] Unity. *Unity Engine*. URL: <https://unity3d.com/> (cit. on p. 31).
- [VDMfVR15] M. Alger. *Visual Design Methods for Virtual Reality*. 2015. URL: [http://aperturesciencellc.com/vr/VisualDesignMethodsforVR\\_MikeAlger.pdf](http://aperturesciencellc.com/vr/VisualDesignMethodsforVR_MikeAlger.pdf) (cit. on pp. 14, 24).
- [Vive18] Valve. *VIVE*. 2018. URL: <https://www.vive.com/us/product/vive-virtual-reality-system/> (cit. on p. 17).
- [VIVEAMD18] B. by several users. *Wireless adapter incompatible with RYZEN cpus*. 2018. URL: [https://community.viveport.com/t5/Technical-Support/Update-on-Vive-Wireless-Adpater/m-p/25838?\\_\\_woopraid=0uuBmtqDuznI#M10293](https://community.viveport.com/t5/Technical-Support/Update-on-Vive-Wireless-Adpater/m-p/25838?__woopraid=0uuBmtqDuznI#M10293) (cit. on p. 47).
- [VRTK] *A productive VR Toolkit for rapidly building VR solutions in Unity3d*. URL: <https://vrtoolkit.readme.io> (cit. on p. 31).
- [Vuforia] PTC. *Augmented Reality Platform*. URL: <https://www.vuforia.com/> (cit. on p. 32).
- [WMREG17] Microsoft. *Troubleshooting Windows Mixed Reality*. 2017. URL: <https://docs.microsoft.com/en-us/windows/mixed-reality/enthusiast-guide/troubleshooting-windows-mixed-reality#im-getting-a-the-install-class-is-not-present-or-is-invalid-error-in-device-manager> (cit. on p. 47).
- [WMRPDesignChanges19] I. Dixon. *How Microsoft are polishing the UI in Windows Mixed Reality with 19H1*. 2019. URL: <https://thedigitallifestyle.com/w/index.php/2019/01/02/how-microsoft-are-polishing-the-ui-in-windows-mixed-reality-with-19h1> (cit. on p. 46).

[Zenject]

*Dependency Injection Framework for Unity3D*. URL: <https://github.com/svermeulen/Zenject> (cit. on p. 34).

All links were last followed on December 8, 2019.





## **Declaration**

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature