Visualization Research Center - University of Stuttgart (VISUS)

Bachelorarbeit

Cubihand: A Handy Toolkit to Interact with Distant Objects in Mixed Reality

Angela Kächele

Course of Study:	Medieninformatik
Examiner:	Prof. Dr. Michael Sedlmair
Supervisor:	Xingyao Yu, M.Sc. Aimée Sousa Calepso, M.Sc.
Commenced:	May 18, 2021
Completed:	November 18, 2021

Abstract

Selection and manipulation of distant objects in immersive environments is highly discussed topic in the HCI community. The current Augmented Reality systems still offer a lot of room for exploring methods of direct selection and manipulation by gestures of distant objects that are small or obscured. Interaction via raycast is imprecise and time-consuming at greater distances from the object. The Cubihand design developed in the course of this work for HoloLens 2 transfers the selection and manipulation process to an immersive controller in the form of a virtual cube, through which the user can more easily select distant objects that are difficult to target at. First, an user can target and select a distant object with the help of the cube. Afterwards, the object takes over all manipulations performed on the cube and can thus be translated, scaled and rotated. In a study, the design of Cubihand was tested on 11 participants and compared with the standard gesture interaction of HoloLens 2. The time required for the task was measured and the participants were asked to complete both a SUS and a NASA TLX questionnaire. Although the analysis showed significant differences in the timing and scoring of the questionnaires in favour of the standard gesture system, participants gave positive feedback on the potential of the design. This offers many opportunities to expand and improve the design of Cubihand.

Contents

List of Figures

List of Tables

Acronyms

ANOVA analysis of variance. 36

AR Augmented Reality. 13

C# C Sharp. 25

HMD Head-Mounted Display. 13

MR Mixed Reality. 13

MRTK Mixed Reality Toolkit. 21

NASA-TLX NASA Task Load Index. 31

Prefab prefabricated model. 25

SUS System Usability Scale. 31

VR Virtual Reality. 13

1 Introduction

The field of Augmented Reality (AR) is being used increasingly, both in the private and in the industrial sector. With the increasing number of users who have not necessarily had much experience with different areas of Mixed Reality (MR) and thus also AR, the systems must be as easy to use as possible. Both the direct manipulation of nearby objects and the manipulation of distant objects must be simple and intuitive. This criterion is already fulfilled for nearby objects, but precision and speed may be reduced when selecting and manipulating distant objects [WHB+18]. Small size or occlusion of the distant object increases this problem. Even using so-called raycast techniques, with which a distant object can be targeted, selected and manipulated, these problems occur, as will be explained in more detail later on. In the context of this work, an alternative to the previous standard gesture system of so-called see-through Head-Mounted Display (HMD) devices is developed with „Cubihand" and tested in the context of a user study. Instead of using a raycast, the user can select a distant object with the help of a virtual cube and then manipulate this object with direct gestures of the hands on this cube.

Users can choose between many AR or MR systems. A simple tablet can already provide an AR experience by extending the view through the camera and thus on the display. However, the interaction space is limited to the surface of the tablet in this case. If the user wants to engage in a three-dimensional experience, they can use a so-called HMD instead of a normal display. Here, the image is projected in the immediate proximity of the user's eye. The simplest version of a HMD is a pair of videos glasses, which are mainly used in the field of Virtual Reality (VR), which project a completely virtual room to the user. An AR version of these are video glasses that use additional cameras to record the room around the user, project it to the inner display and thus be able to augment the given space. When using these, however, the user's spatial, real perception can still be disturbed by delays of the camera or a restricted field of vision, for example. The simpler version of these are see-through HMDs, which have a transparent display onto which the immersive space is projected. Compared to camera-based MR systems, the user's real hands, for example, cannot leave their position, but the user gets a better feeling for the imersive world seeing and using their own hands. Interaction with this type of HMD is discussed in this work. While with see-through HMDs, the manipulation of objects that are in immediate range of a user can be performed easily and intuitively after a short briefing, a selection of objects that are more distant from the user already turns out to be more difficult [WHB+18], as already mentioned above.

The selection of distant objects, for example in the see-through HMD Microsoft HoloLens 2, is archieved by using raycast and pointer. The tip of the ray projected from the hand serves as a cursor with which the user targets an object and then manipulates it using the same gestures they use to manipulate nearby objects. However, this can be problematic especially with smaller objects or even those that are obscured by other virtual objects.

As an example, one might imagine a scenario in which the employees of an architectural studio are sitting around a large table together with clients. Everyone present uses the HoloLens 2 to project the architectural project in the middle as a 3D model for everyone to see. A customer now wants to make a suggestion for a change to the model, but would have to grab an object on the back of the model to do so. The object is therefore hidden by other parts of the model. With the standard gesture system of HoloLens 2, the customer has the option of either standing up and walking around the table so that the cursor of his pointer is no longer blocked by the object. Or he could ask another person on the opposite side to move the object for him, but this could lead to misunderstandings and would therefore be not effective. In this situation, a system that uses a controller to enable object selection without the pointer would serve a useful purpose. Especially an immersive controller, which means a controller that would only be visible and usable through HoloLens 2 system, would not have to be held seperately and could therefore be used at any time if needed.

The Cubihand system developed as part of this work takes up this idea - with the help of an immersive cube, a user should bee able to select small or hidden objects that are no longer in the direct interaction radius or within reach of the user. The selection of an object as well as translation, scaling and rotation should be made possible for the user by operating the cube. In the above-mentioned scenario, the customer could use the cube to move an immersive plane, with which the selection of the hidden object would be possible, and after this selection, changing the object by manipulating the cube. Translation, scaling and rotation on the cube would be transferred to the object to be manipulated. This would give the customer the possibility to easily represent his wishes without having to rely on the help of others.

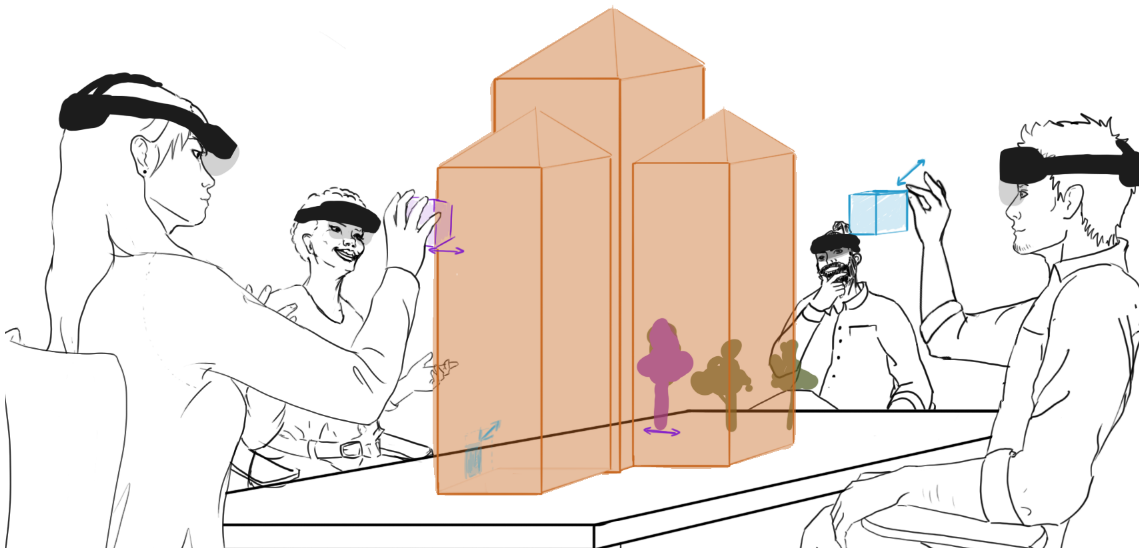


Figure 1.1: Illustration of the example scenario for the use of Cubihand

The goal of this work is to find an approach for an alternative to the existing interaction systems. With Cubihand, an immersive controller is to be provided that enables a more precise and reliable manipulation of distanced object than the standard system of the HoloLens 2, without having to access additional tools such as handheld controllers. The controller will be implemented in the Unity 3D Game Engine [Tec21b]. After the implementation of this controller, a user study is conducted to test whether users perceive this new type of selection and manipulation useful and can actually archieve better results in terms of speed and user satisfaction. For this purpose, both the time to complete the task and the usability of the design and its workload will be measured and compared to the corresponding results of the standard gesture system.

The following section provides an overview of previous work and its approaches to the manipulation of distanced objects in immersive space. Section three discusses the concept and design of Cubihand while the fourth section explains the methods for the actual implementation of this design. In section five, the study used to test the concept and implementation of Cubihand and the evaluation of its result is described. Finally, section 6 concludes the results of this study. Strengths and weaknesses of this work will be discussed and ideas for further development will be adressed.

2 Related Work

While many studies have already looked at the interactions of nearby objects, Whitlock et Al. in [WHB+18] empirically explore which type of interaction in AR with objects at a greater distance archieves the greates success in terms of accuracy, speed and personal perception of preference. Three different interaction options are avaiable: voice-activated interaction, interaction through gestures (as it is also used with nearby objects on the HoloLens 2) and interaction through external devices held in the hand. This is tested on three types of manipulation such as translation, rotation and selection. Particular challenges in the manipulation of distant virtual objects include a limited viewing angle, altered movement sequences in the manipulations or ambigious pointing at objects. The greater the distance, the greater the influence on accuracy. For example, errors are amplified by the viewing angle over distance. Although interaction by gesture promises a higher error rate due to the inaccuracy of the pointer at distance, this and interaction by additional input devices are both the fastest and most accurate. Interaction through speech input archives lower success, while interaction through gestures is considered the favourite among the subjects. According to this approach, it is wortwhile to go further into favoured interaction possibilities of gestures and to find a solution for the disadvantages of these.

In [BH97], Bowman and Hodges examine various methods for manipulating objects in VR space that are more than an arm's length away from the user. They present a total of six possibilities, two of which are based on the raycast technique. This method is also used today, in a more sophisticated form, with the HoloLens 2 when the user wants to manipulate distant objects. Here, a „light beam", in other words a drawn line, which is projected into the room starting from the user's hand or index finger, with which they can select an object. In the work of Bowman and Hodges, however, simple raycasting fails because of the limitations in manipulation, for example there is no method to rotate the object around all axes. Translation from and towards the user is archieved in this work by an extended form of raycasting, „reeled raycasting", i.e. in a form of a fishing rod, the user can move the object towards and away from him.
Other methods of object manipulation were also mentioned in the paper, such as the simple extention of the virtual arm. Although raycasting made it easier to reach for an object than the other methods, users preferred the other methods for object manipulation. This led to the development of a mixed method, called „HOMER" (Hand-centered object manipulation extending raycasting), in which the user first selects the object using a ray. The virtual hand is then projected onto the object, allowing manipulation until the object is released. The manipulation is relative to the user's movements, so that, for example, the movement effort for distant objects is not too great. For more precicse movements, the use of a mouse is recommended.

Bellarbi et Al. [BZO+17] deals primarily with the selection of distant objects in VR or AR space. Here, a further method to the „HOMER"-Technique proposed by Bowman and Hodges is presented: If the user has selected an object that they want to manipulate but is too far away, they can zoom in on the object by pointing at it. As long as they are pointing, the field of vision is enlarged until the object is within the user's direct reach, allowing direct manipulation with the standard gestures.

Despite augmented reality, the users arm is displayed in virtual form, which means that this arm can move around within the field of view as desired. The real arm is at an angle at which the user cannot see it. Compared to the „HOMER“-Technique, this method archieved greater efficiency and lower error rate on average in the test. The problems of inaccuracy and hidden objects can be solved well with this method, but this requires additional camera technology in order to be able to represent the arm virtually and show it correctly to the user. With the HoloLens, this type of technology is not given, as the user has a free radius of vision, so they have to rely on seeing the real hands. However, this method shows that alternatives to pure raycasting for selecting distant objects, as is also used in the „HOMER“-Method, are desirable.

With the HoloLens 2, als already mentioned, a form of raycasting is also used for distant object manipulation. In the direct movement radius, so on nearby objects, two different methods can be used for object manipulation: On the one hand, the user has the possibility to move the object by simply grasping with one hand, to rotate it by grasping it and turning the hand, or to scale it by grasping it with two hands. This form of manipulation is called Object Manipulator [Mic21d] and can also be used with objects that are further away. Instead of using the hand for translation and rotation or the hands for scaling, the manipulations are transferred to the distant objects by pointing and grasping with raycast beams [Mic21e]. There is a cursor at the end of the ray. During simple hand pointing, the cursor has the shape of a circle. If the user points at an object and starts to perform a manipulation, the cursor becomes a dot [Mic21b]. In addition, a so-called Bounds Control [Mic21a], a transparent box with handles, can be placed around the object. Here the user can pick the surfaces, edges and corners and thus translate, rotate or scale the object with only one hand. This interaction type is also used when the object is further in space, more than 50 centimetres away from the user. Then the user points to the surfaces, edges or corners of the object using raycast and can thus manipulate it. In the case of distant objects that are also not large, so that edges and corners are close together, for example, this can lead to inaccuracies in the operation and ultimately make it more difficult. Also, as Whitlock et Al. already mentioned, the further away the object, the more inaccurate the pointer will be.

The above-mentioned work shows that although users prefer the use of gestures to other input methods such as voice input, they prefer other gestural methods to the Raycast when using this input method with distant objects. It is therefore necessary to find an alternative to the raycast and Pointer system used by HoloLens 2. Although there are already approaches that form an alternative to the raycast, these cannot be implemented for see-through HMD systems; Here, it is not possible to change the position of the user or their hands, as the user's hand is not projected to the field of view. It is therefore worthwile to implement an approach that can be used without a system with video glasses and where the user can thus use their own hands and field of vision to select and manipulate a distanced object. Cubihand is taking this approach.

3 Design

The design of Cubihand is intended to create a new way of selecting and manipulating distant objects in AR, especially those that are small or hidden by other virtual objects. The most important element in this type of interaction is a cube that enables both the selection and manipulation of distanced objects, which will be described in more detail later in this chapter. The cube is placed in the user's field of vision near their dominant hand. So the cube within the users reach and moves with the camera in its initial position in front of the user, unless it is being manipulated by the user, and thus does not move out of sight by itself.
With this cube, the user has the possibility to control almost the entire selection and manipulation process. The cube itself can be manipulated in its position, size and rotation. When the cube is released after an interaction, it returns to its original position and resumes its original size and rotation orientation. The cube is coloured in an easily perceptible colour, a strong violet, so that it is clearly visible even in a well-lit room. Its size is chosen so that it is as small as possible to cover as little of the field of vision as possible, and at the same time large enough to be easily grasped by the user without the user acidentially performing an unintented type of manipulation. The size is thus chosen to be about the size of the user's hand, which is not fully opened, and larger than the user's fist. This makes a total of 12.5 centimetres each in width, height and depth. One face of the cube points directly to towards the user. The distance between the cube and the user is chosen so that the user can push the cube away from him or pull it towards him without it disappearing from his field of vision, that is, about 40 centimetres from the position of the users eyes.
The user starts in a given scenario in a room with various objects of different sizes and distances from the user. The distance of the objects is such that the user can no longer manipulate these objects with their hands. The user wants to select one of these objects in order to finally be able to manipulate it with the help of the cube. Cubihand has two states for this, the scanning state and the manipulation state. Initially, Cubihand starts with the scanning state, which allows the user to select the object. After selecting the desired object, it switches to the manipulation state. After being done with the manipulation, the user can toggle back to the scanning state to chose another object. A concept of the design displayed in Figure 3.1.

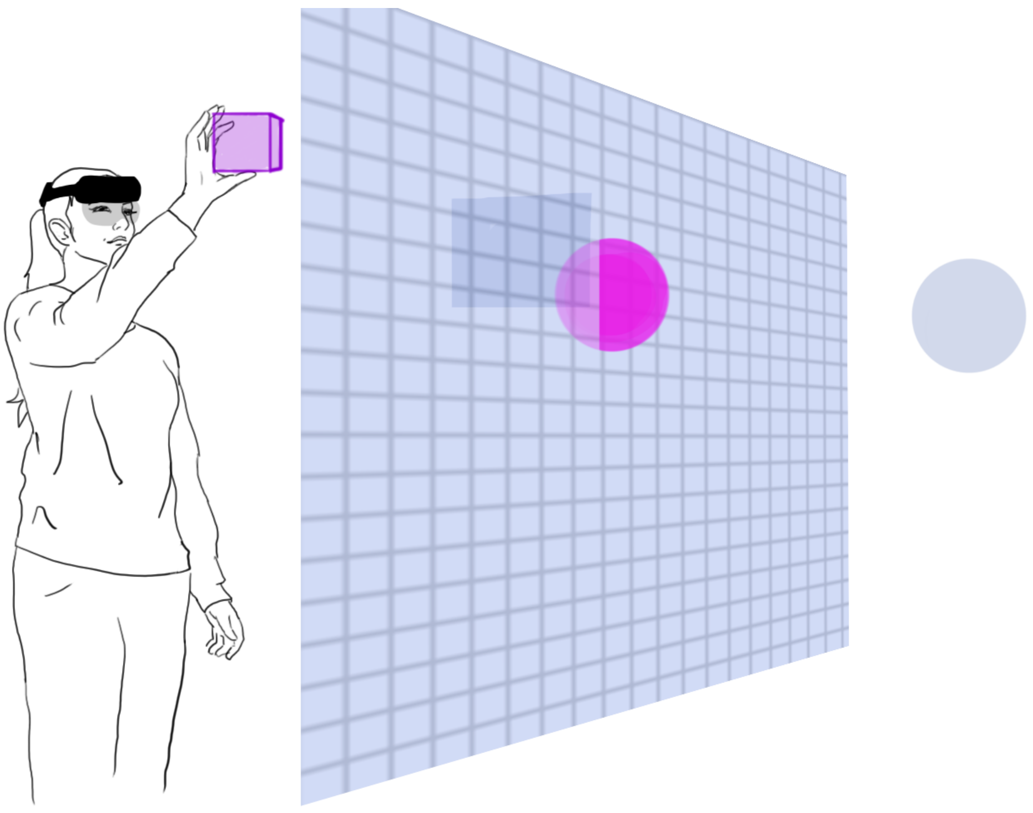


Figure 3.1: Concept Art for the use of Cubihand in the scanning state where the user moves the plane with the cube until the chosen object gets intersected with the plane.

3.1 Scanning state

Initially, the object could be selected by simply touching it with the pointer supplied by Microsoft as standard, in other words by pointing at an object with a ray emanating from the hand and clicking on it by means of a grasping movement, the so called pinch. However, especially with distant objects, major problems with the precision of the pointer were found in the previously mentioned work, so that the user may need several attempts to select an object or even select a wrong object that is close to the object to be selected. This effect is also intensified by a small size of an object, as the area to be grasped is reduced even further. There is also the possibility that the user accidentally selects manipulations that they did not intend when the standard interaction options for the objects are activated by default, so that, for example, they rotate the object around an axis instead of translating it. In addition, objects that are hidden by another virtual object in front of them cannot be seen and cannot be selected directly with the pointer. To do this, the virtual object must first be moved or the user has to move themselves, which can mean additional effort for the user. For this it is obvious to first set all objects in the room transparent and at the same time not selectable, which means that the colliders of the objects are deactivated in each case. Only through a certain condition should the objects to be manipulated be set opaque and selectable. In Cubihand, this condition is archived by a plane, the so called scanning plane, and by the contact with it. This plane is initially located in front of the user behind the cube and moves with the user's field of vision, which means that it is linked to the user's position. It has a transparent grid in order to be able to perceive its movements in space more clearly along the z-axis, so away from and towards the user. Due to the transparency, objects are not hidden by the grid lines.

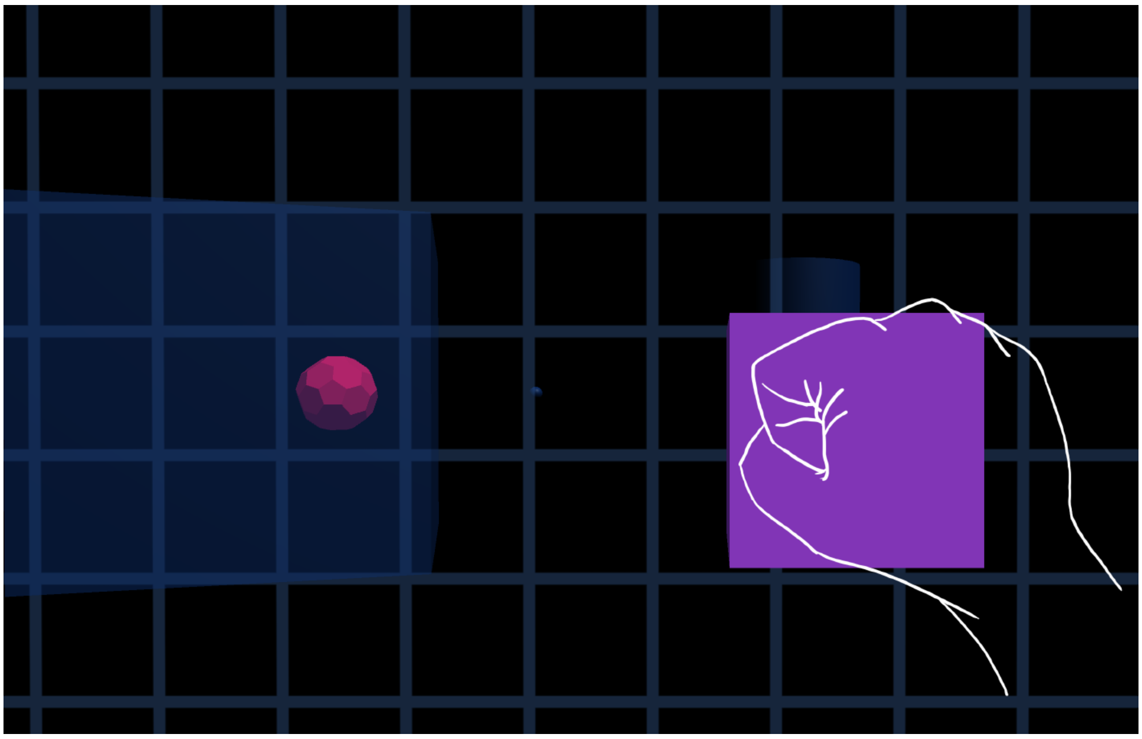


Figure 3.2: Screenshot of scanning process with intersecting object

In addition, it is possible to move the cube in front of the user by grasping it. In doing so, the cube can not only be translated, but also rotated and scaled with two hands - however, the latter two types of manipulation do not matter at this state and are therefore ignored. If the user now grabs the cube with one hand and moves it forward away from them, the plane also moves forward away from the user through the space. If the user pulls the cube towards them, the plane moves towards the user. The distance of the cube from its starting position is decisive: the further the user moves the cube from its original position, the faster the plane moves through the space. Conversely, the plane moves slower if the user moves the cube only minimally. Movements of the cube that affect axes other than the users z-axis are not perceived. This allows the user to control the movements of the plane more precisely. The plane moves through all objects it touches on its way through space. However, as soon as an object comes into contact with the plane, the surface of this object changes from transparent to opaque, which means it is more clearly visible. In addition, the colour of the object changes to make an intersection with the plane more obvious. If the user releases the cube, the plane settles on that position relative to the user. In Figure 3.2 this process is shown in an example scene. Any object that is in intersection with the plane can be selected. The selection is done by using the standard pointer of the Mixed Reality Toolkit (MRTK), which means that the user points at the object, or the object is targeted by the light beam starting from their hand and selects the object by means of a pinch gesture. Cutting objects that are behind other objects can thus now be selected through the front object without any problems, as the front object does not have a collider at this point. Small objects can also be selected better by using the scanning plane, as the pointer can only be used to select and not to manipulate the object. This means that when pointing to the object, there are no other options for manipulating the object that the user could accidentally select instead of the intended selection of the object. The successful selection of the object is indicated to the user by the disappearance of the scanning plane. As soon as this plane has disappeared, which means that the selection of the object was successfull, Cubihand switches to the manipulation state.

3.2 Object Manipulation state

With the selection of the target object and the transition to the object manipulation state, all manipulation types that are carried out on the cube are translated to the target object. For this purpose, another, more precise manipulation type [Mic21a] is activated for the cube. The manipulation type already activated during the scanning process already allows complete manipulation of the cube, so translation, scaling and rotation were previously possible, but two hands were needed for scaling and rotation around only one axis was possible, but imprecise and most likely the user would rotate around the axes, too. The now additionally activated manipulation mode makes it possible to perform all three types of manipulation on the cube and ultimately also on the target object with only one hand, as can be seen in Figure 3.3. Handles are projected onto the corners and edges of the cube as soon as the user's hand is near the cube. If the user grasps the handles at the corners of the cube, they can scale the cube and thus target the objects scale. If the user grasps the handles at the edges of the cube, they have the possibility to rotate the cube and the target object around only one axis. The translation of the cube and the object is done as before by grabbing the cube and moving it.

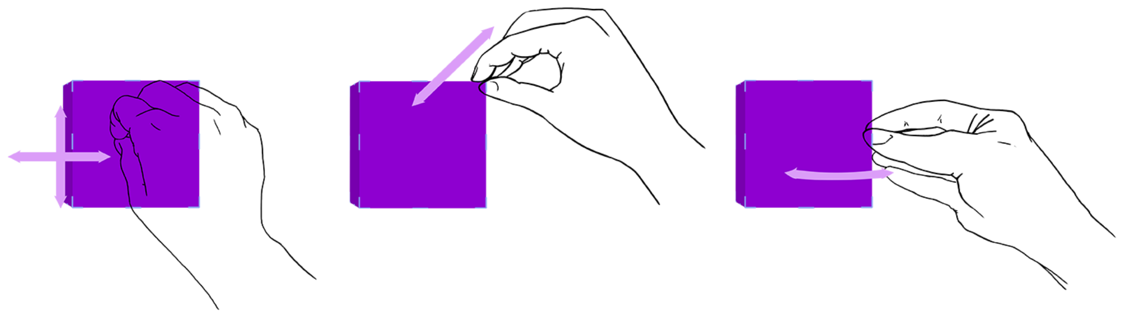


Figure 3.3: Sketches of translating, scaling and rotating the cube with BoundsControl. The user can modify the cube over the surface and the handles on the corners and edges for translation, scaling and rotation.

The translations on the cube are not transferred absolutely, but relatively to the target object. The further away the target object is from the user, the faster it moves in relation to the cube. Nevertheless, the user needs several steps for larger translations, which means that the user has to move the cube several times from its starting position to the desired direction, since the user has a restricted field of vision due to the cube and thus the target object can quickly be obscured by the cube during the translation. Performing the translation in one movement, while the user may not be able to see the target object for a longer period of time, could therefore lead to unwanted placements of the object or even to the object getting lost in space. The scaling of the target object is absolute to the scaling of the cube for similar reasons. Although the user has two options in the execution of the scaling - one or two-handed - they only differ in the number of hands required, not in the precision. In relation, the scaling of the target object would react too sensitively and the user could accidentally scale the object much too large or even invisibly small without intending to do so, and possibly lose the object. Thus, for larger scales, the user has to scale the cube several times, but has full control over the process and thus archieves greater precision. The rotation of the target object is exactly the same as the rotation of the cube, as the user has two different options for the rotation, which differ in their precision. To rotate the object quickly and roughly, the user can reach for the cube and rotate it around several axes simultaneously with an intuitive hand rotation. If more precise rotation is required, the user has the option of grasping the respective edges of the cube and rotating it around one axis.

The types of manipulation can be carried out in any order. If the user releases the cube after an interaction, it resets itself to its original position, size and orientation, as already mentioned in the selection process. However, the target object that simultaneously performed this interaction retains the changed position, size and orientation. If the user has completed the desired manipulations on the object and wants to end the process, they can deselect the object by selecting it again with the pointer and the following pinch gesture. The successful deselection is indicated by the appearance of the scanning plane and thus the change to the scanning state.

4 Implementation

The implementation for Cubihand was carried out in the Unity 3D Gaming Engine version 2019.4.18f1. Among other capabilities, this development environment enables the implementation of MR, so AR and VR applications for various plattforms, including the HoloLens 2. In Unity, C Sharp (C#) is used as the programming language for its scripting API. This project was build for Universal Windows Platform with ARM architecture.

4.1 Mixed Reality Toolkit

In order to be able to develop applications for the HoloLens 2 in the Unity 3D Engine, it is recommended to use the MRTK [Mic21f]. This tookit offers various packages that can be imported into Unity. On the one hand, these provide the basic functions that are necessary for the HoloLens 2. In addition, further packages offer prefabricated models (Prefabs)[Tec21a], audio functions and examples that can be used as orientation. Pointer functions, audio-visual feedback, prefabricated and already functional models for buttons, for example, are included in these packages. If necessary, the latter can be modified and adapted to one's own wishes. Components that are necessary for manipulating objects can be easily added to the objects to be manipulated and can even be changed visually. In addition, the MRTK offers functional shaders for the HoloLens 2 in order to optimise the representation of the objects in the applications. For the development of Cubihand, the MRTK version 2.7.2. was used. This is compatible with the Unity version already mentioned above. Three MRTK packages were imported in this project: the foundations package, the extension package and the example package for orientation and familiarisation with the MRTK. All of the materials in this projects are using MRTK shaders. Also, MRTK offers default profiles, for example for camera, input or spatial awareness. Some profiles have been adjusted, such as the camera profile for its display settings or the input profile for its pointer range.

4.2 Cube

The most important object in the implementation of the project was the cube. As a `GameObject`, it uses the main camera as its parent and thus always takes over its position with certain offset to this camera. As components for the cube scripts from the MRTK were used in addition to the scripts written for this project, which will be described in more detail later. To be able to grab a `GameObject`, both a `Collider` and the `NearInteractionGrabbable` script [Mic21c] are needed. The `ObjectManipulator` script makes the object translatable and rotatable with one hand and scalable with two hands. `BoundsControl` adds handles to the `GameObject` on which the user can perform the same manipulations one-handed on corners and edges. Together with the ConstraintManager, which also manages the MinMaxScaleConstraint to define a minimum and maximum size for the cube, these scripts form the base for the cube.

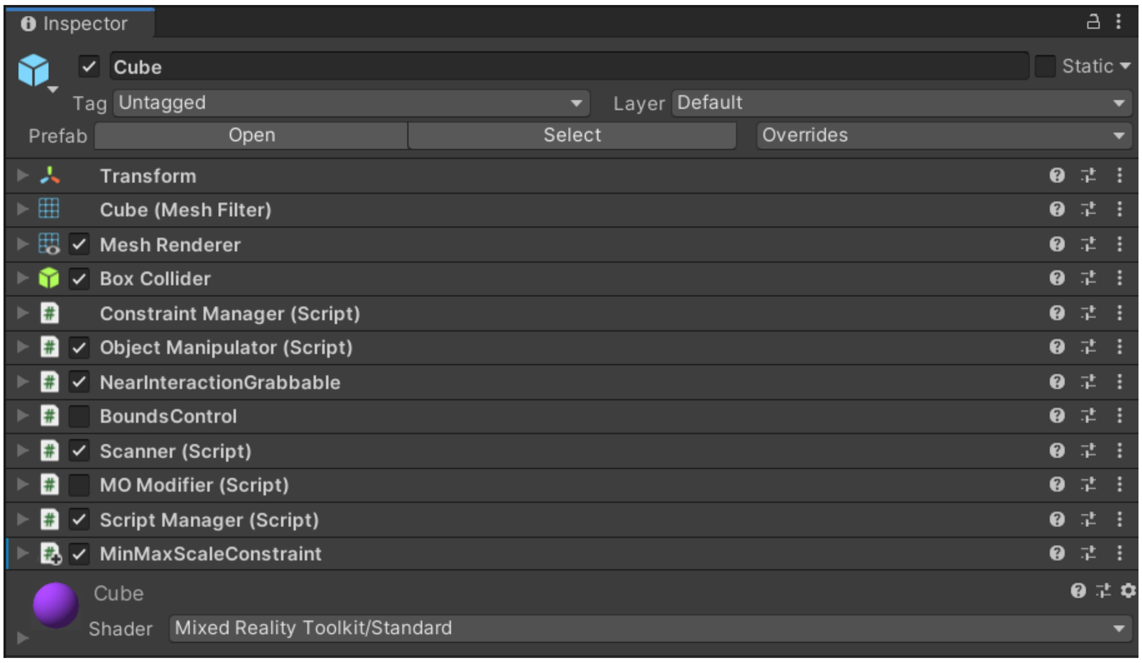
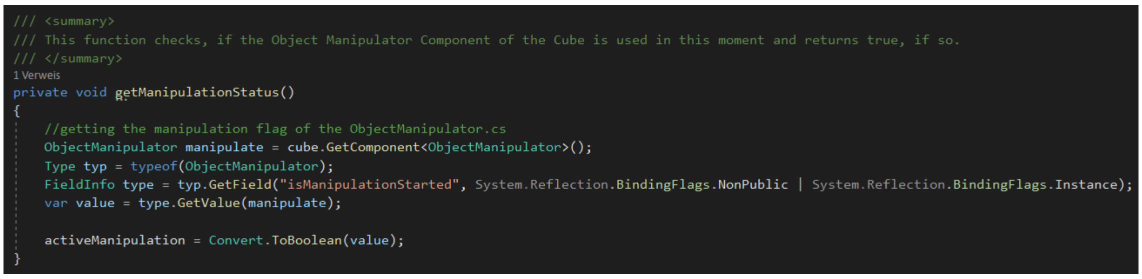


Figure 4.1: Cube Inspector with all its scripts used in the implementation and which scripts are set active initially. So the Script Manager, as well as the Scanner Script and other MRTK Scripts are activated initially. MOModifier and BoundsControl are being activated after the selection of an object.

4.2.1 Scanner Script

Unlike some other scripts, the Scanner script is initially set to active, as you can see in Figure 4.1. Right at the start of the script, the cube is set as a child of the camera to its position, alignment with an offset and the original size of the cube. As long as the cube is not manipulated by the user, this reset is done every frame so that the cube is always correctly aligned with the user. At the same time, every other object in the scene that is a SceneObject in the scene hierarchy of Unity is initially assigned a transparent material and the Collider is deactivated for every object. In the LateUpdate function of the script, the manipulation status of the cube is checked. If the status of the ObjectManipulator script is set to active, the manipulation status of the Scanner script is also set to active. Since the ObjectManipulator does not offer a getter-Method for this variable, it must be requested like shown in Figure 4.2.

If the manipulation is active, the position of the scanning plane changes according to the movements of the cube along the z-axis. The speed of movement of the scanning plane depends on its distance from the user - the further away, the faster the scanning plane moves, up to a threshold value. In addition, during an active manipulation, the Colliders of all scene objects are set to active in order to be able to register a collision of these with the scanning plane. For this purpose, the collision of



```
/// <summary>
/// This function checks, if the Object Manipulator Component of the Cube is used in this moment and returns true, if so.
/// </summary>
1 Verweis
private void getManipulationStatus()
{
    //getting the manipulation flag of the ObjectManipulator.cs
    ObjectManipulator manipulate = cube.GetComponent<ObjectManipulator>();
    Type typ = typeof(ObjectManipulator);
    FieldInfo type = typ.GetField("isManipulationStarted", System.Reflection.BindingFlags.NonPublic | System.Reflection.BindingFlags.Instance);
    var value = type.GetValue(manipulate);

    activeManipulation = Convert.ToBoolean(value);
}
```

Figure 4.2: Function in Scanner script to retrieve the manipulation status from ObjectManipulator script

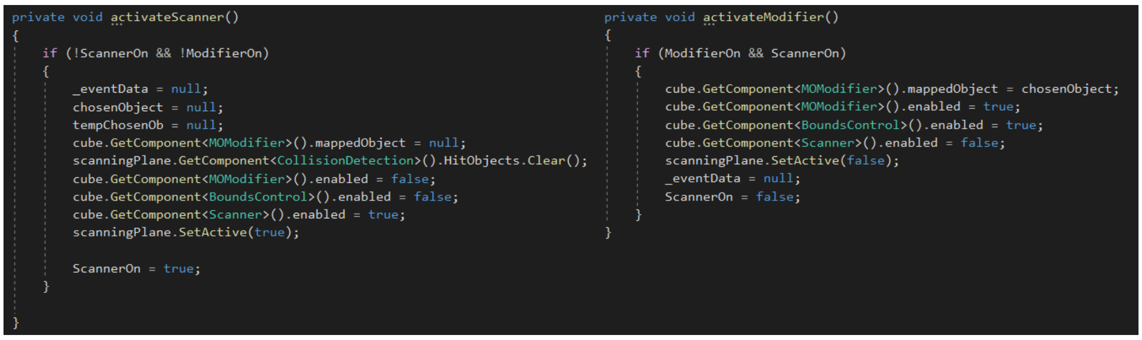
the scanning plane with the objects in the scene is requested every frame. The scanning plane has its own script for this process, which will be described in more detail later. Here, a list of objects is provided that actively intersect the scanning plane. The material of these objects is set to an opaque material that is easily perceivable in colour. If the manipulation is ended, the cube resets itself to its initial position with initial values. The scanning plane retains its current position relative to the user and the `Collider` of all unintersected objects are deactviated again so that no false collisions with pointers can occur. The `Collider` of all objects that are still in the intersection with the scanning plane at the end of the manipulation remain set to active so that the user can select these objects with the pointer.

4.2.2 Mapped Object Modifier Script

The `MOModifier` (Mapped Object Modifier) script is initially deactivated. It is only activated by the `ScriptManager` script when the user selects the object to be manipulated; this process will be described in more detail later. If the `MOModifier` script is activated, the cube is set as a child of the main camera, as with the `Scanner` script. In addition, the initial position and rotation of the object to be manipulated is saved, so it can be modified later. In the `LateUpdate` function, the distance of the object to be manipulated from the user and thus the scaling movement speed is calculated first. The further away the object is from the user, the faster it will be moved later during translation. Furthermore, the manipulation status of the cube is calculated here as well. If manipulation via the `ObjectManipulator` is active, which is requested the same way as in Figure 4.2, the corresponding boolean is set to true. In addition, the manipulation status of the `BoundsControl` is also requested here. `BoundsControl` offers events for its manipulations, so that these can be easily retrieved via listeners. If one of the manipulation events of `BoundsControl` is active, its corresponding boolean is set to true. If one of the two manipulation types is active, the translation, rotation and scale of the object to be manipulated are controlled by the manipulations on the cube. The translations of the cube are translated depending on the distance to the object to be manipulated. The further away the object is from the user, the greater the translation of the object compared to to the translation of the cube. The scaling and rotation difference of the cube to its starting values are each transferred absolutely to the object to be manipulated. If neither of the two types of manipulation is active, the position, size and rotation of the cube are reset to their original values. However, the position, size and rotation of the manipulated object are preserved.

4.2.3 Script Handler

The `ScriptHandler` script controls the activity of the `Scanner`, `MOModifier` and `BoundsControl` scripts. Initially, besides the `ScriptHandler` only the `Scanner` script is activated in this implementation, as has been shown in Figure 4.1. In this script, there are two boolean values `ModifierOn` and `ScanningOn`, which enable the control of the change between scanning and manipulation state of the Cube. Initially, both values are set to `FALSE` and thus the scanning state is activated once, in which `ScanningOn` is set to `TRUE` again, as displayed in Figure 4.3 (`activateScanner()`). The two boolean values are necessary to ensure that the state change functions are only executed once each, since every function is called once per frame with the `Update` function otherwise.



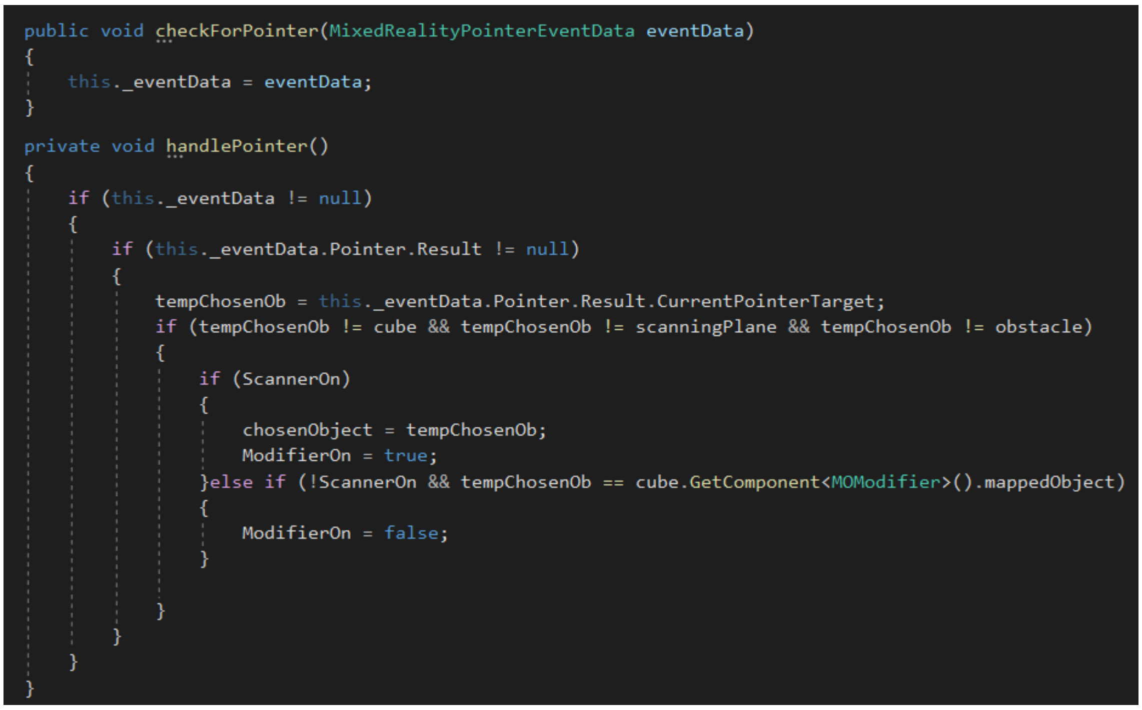
```
private void activateScanner()
{
    if (!ScannerOn && !ModifierOn)
    {
        _eventData = null;
        chosenObject = null;
        tempChosenOb = null;
        cube.GetComponent<MOModifier>().mappedObject = null;
        scanningPlane.GetComponent<CollisionDetection>().HitObjects.Clear();
        cube.GetComponent<MOModifier>().enabled = false;
        cube.GetComponent<BoundsControl>().enabled = false;
        cube.GetComponent<Scanner>().enabled = true;
        scanningPlane.SetActive(true);

        ScannerOn = true;
    }

}
```

```
private void activateModifier()
{
    if (ModifierOn && ScannerOn)
    {
        cube.GetComponent<MOModifier>().mappedObject = chosenObject;
        cube.GetComponent<MOModifier>().enabled = true;
        cube.GetComponent<BoundsControl>().enabled = true;
        cube.GetComponent<Scanner>().enabled = false;
        scanningPlane.SetActive(false);
        _eventData = null;
        ScannerOn = false;
    }
}
```

Figure 4.3: The two functions which activate different elements for both scanning and manipulation states

The core of this script is a function that waits for a pointer event and saves its `eventData` when the event arrives. This event occurs when the user is pointing at an object and selects it with the pinch gesture. The event is provided by MRTK with the `PointerHandler` script which is attached to the parent `GameObject` of all scene objects. The `eventData` is forwarded to a function that checks whether the script is currently in the scanning or manipulation state. If the script is in the scanning state, the `ModifierOn` variable is set to `TRUE` and the new object to be manipulated is taken from the pointer's target. In Figure 4.4 the functions for this process are displayed.

The new object to be manipulated is tranferred to the function for changing to manipulation state (see `activateModifier()` in Figure 4.3), where it is passed to the `MOModifier` script as the `MappedObject` after `MOModifier` has been activated. In addition, the `BoundsControl` script is activated and the `Scanner` script with its scanning plane is deactivated. The `ScannerOn` variable is then set to `FALSE`, so the scripts are only activated once when the state changes. The script is now in the manipulation state until another pointer event is intercepted. If this takes place, the script switches back to the scanning state, deactivating the `MOModifier` and `BoundsControl` scripts and activating the `Scanner` script with its scanning plane. The object to be manipulated is also reset.



```
public void checkForPointer(MixedRealityPointerEventData eventData)
{
    this._eventData = eventData;
}

private void handlePointer()
{
    if (this._eventData != null)
    {
        if (this._eventData.Pointer.Result != null)
        {
            tempChosenOb = this._eventData.Pointer.Result.CurrentPointerTarget;
            if (tempChosenOb != cube && tempChosenOb != scanningPlane && tempChosenOb != obstacle)
            {
                if (ScannerOn)
                {
                    chosenObject = tempChosenOb;
                    ModifierOn = true;
                }else if (!ScannerOn && tempChosenOb == cube.GetComponent<MOModifier>().mappedObject)
                {
                    ModifierOn = false;
                }

            }
        }
    }
}
```

Figure 4.4: Functions for the pointer event to be catched and processed depending on the state of the script

4.3 Other Scripts

In addition to the scripts that are attached to the cube as components, other objects in this implementation are also using scripts. As already mentioned, the objects in the scene, respectively the parent `GameObject` of them, have been provided with a `PointerHandler` script of the MRTK. This script outputs `eventData` for various pointer events, for example when a pointer click is executed by a pinch gesture or an object is dragged by a pointer. For this implementation, however, only the `OnPointerClicked` event is important, so only this event is intercepted as described above.
A `CollisionDetection` script was created for the scanning plane. In this script, the two `Collider` functions `OnTriggerEnter` and `OnTriggerExit` of the plane `Collider` are used. If the `Collider` of another object touches the `Collider` of the plane, the `OnTriggerEnter` function is called. This adds the object of the other `Collider` to a list. If the `Collider` of the other object leaves the `Collider` of the plane, the `OnTriggerExit` function is called. This removes the object of the other `Collider` from the list. The list can be retrieved externally via a getter method and is thus retrieved by the `Scanner` script in order to change `Collider` and material of intersected objects.
For the parent `GameObject` of the scanning plane, the `TransformScanningPlane` script was created. This script lets the `GameObject` and its child, the scanning plane, take over the orientation and position of the camera and thus of the user. On the one hand, the position of the camera is taken over completely and on the other hand, the rotation around the y-axis of the camera. Rotation around the z- and x-axis of the plane remain as they are.

5 Evaluation and Discussion

The design of Cubihand was tested by carrying out a user study. In course of this study, a total of three measurements were taken and evaluated individually.

5.1 Study Design

A total of eleven people (2 female, 9 male) between 18 and 31 years old took part in this study. The sample contains mainly students, in order to ensure a familiar contact with media. In addition, the majority of these test participants had experience in dealing with computer games, so that a good learning rate at spatial understanding in AR could be archieved. In comparison to the Cubihand interaction system, the test subjects were offered a reference interaction system based on the HoloLens 2 standard gestures. Thus, the subjects had to test both types of interaction in the same given scene. With the order changing for each subject, the study was carried out in a within-subject design. In order to compare the two different types of interaction, the participants had to complete a task in two conditions, each with the same scene and a different type of interaction. In Condition C_1, participants had to use the Cubihand interaction system while with Condition C_2, they had to use the HoloLens 2 standard gesture system as reference. For each of the three runs of both conditions, the tim eneeded for that run was measured. Thus, a total of six time measurements were taken. In addition, the participants were asked to fill out two questionnaires after completing three runs of each interaction type. The questionnaires were the System Usability Scale (SUS) and NASA Task Load Index (NASA-TLX). Thus, both of these questionnaires were answered twice per participant.

5.1.1 Task Description

A total of seven objects were placed in the scene of the task. Three of these objects were to be manipulated by the user, while three other objects identical in shape marked a goal state. The last object was placed in the room as an immovable obstacle. The objects to be manipulated and the obstacle were displayed in a transparent blue, as set per default for scene objects in Cubihand. The transparency served to detect objects behind obstacles. The obstacle itself represented an object which, although it appeared to be an object to be manipulated, may not be moved and was therefore set immobile. Nevertheless, the obstacle, like the other three objects to be manipulated, has a collider to prevent the pointer from showing through. The three copies of the objects to be manipulated, which mark the goal state, were coloured green transparently and placed to the left of the user. In addition, the objects were partially changed in their size and rotation so that the user also had to carry out these manipulations in order to be able to create the same state with the objects to be manipulated. The exact positioning of the elements is displayed in Figure 5.1.

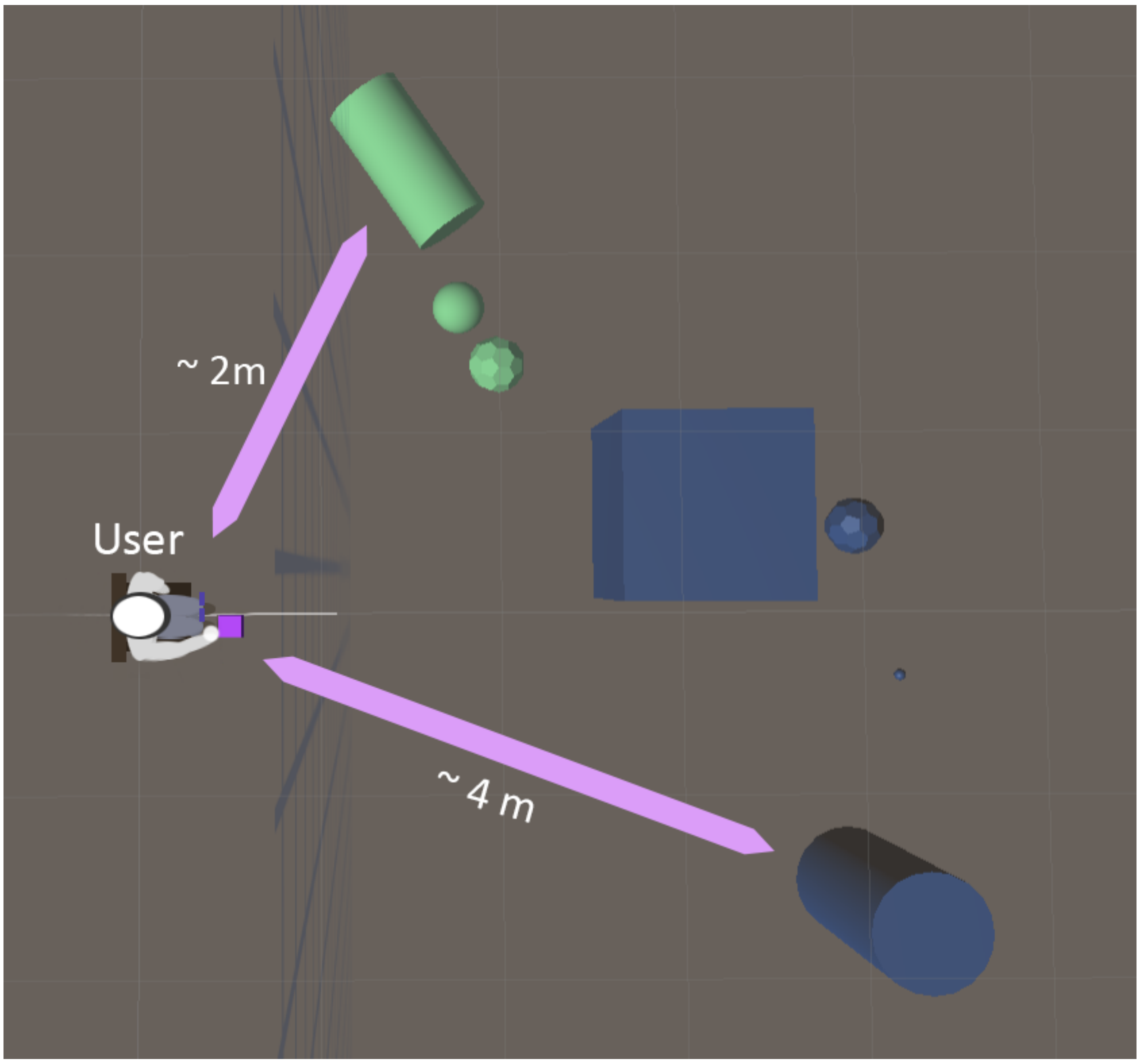


Figure 5.1: Study scenario shown from above. The participant sits about two metres away from the goal states, where the objects are up to four metres away from the user. The buttons are placed directly in front of the user.

Ideally, all three objects to be manipulated overlapped exactly with the objects of the goal state at the end of the task. The transparency of the goal states allowed the objects to be manipulated to still be recognised. In addition, the goal states did not have a collider, so that re-selecting the objects to be manipulated in them did not pose a problem. Users therefore had to use their own judgement to determine whether the objects to be manipulated matched the goal state, as it is displayed in Figure 5.2.

In order to archieve the same position, rotation and scaling of the goal states with the objects to be manipulated, various combinations of these manipulations had to be made. For the first object, a platonic body, only translation is required. The second object to be manipulated, a small sphere, required both translation and magnification to match the goal state. The third and final object, a large cylinder, needed to be reduced in size, rotated around the z and y-axis and translated. The difficulty of these manipulations is initially determined by the selection of the objects. Here, during the design

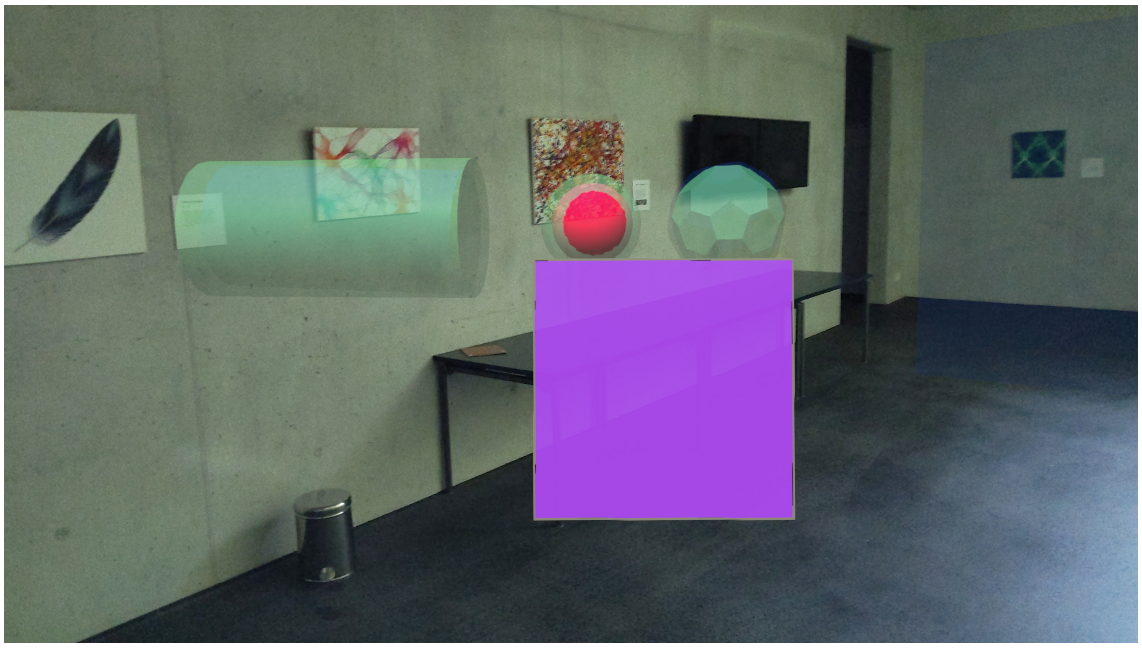


Figure 5.2: Example for placing the objects in their goal states with the sphere still being selected.

of the scene, the difficulties in manipulating distant objects documented in the previous works were adressed, in other words, dealing with small, distant or obscured objects. Thus, the platonic body was placed behind an obstacle, that had transparency but an active collider to intercept a pointer when pointing to the object behind it. Therefore the participant had to change their position to reach the object with the pointer by default. In addition, the small size of the sphere and its long distance from the user was deliberately chosen to also make its selection by the pointer more difficult. The cylinder had an easy selectable size, but all three forms of manipulation had to be performed on it, whereby the rotation in particular was made more difficult by the non-uniform shape of the cylinder, as this form of rotation required greater precision. In the starting position, there were two buttons below the user. The „Start“-Button which the user had to press before beginning the task, started a time measurement in seconds. If the user presssed the „Stop“-Button after completing the task, the result of this time measurement was saved in the application’s log file. This could be downloaded and read out from the browser interface of the HoloLens 2 after the attempt was completed.
Before starting both tasks in both conditions, each participant received a briefing on how the applications for both conditions work. On the one hand, the functionality of the gestures in the HoloLens 2 were explained to them, as they are used exclusively in the reference condition and partitially in the Cubihand condition, such as pointing and pinching. The Cubihand condition was also explained in its mode of function, such as moving the scanning plane by using the cube, selecting and manipulating the desired object. Before the participants had to start their tasks and time measurements, they were given a practice run of the task per interaction type. In addition to the actual task, the participants were advised to carry out the tasks in the sitting position, which was their starting position, and to remain seated, if possible.

5.1.2 Measurements

As already mentioned in the design of the study, a time measurement was taken during each run of both types of interaction. The measurement was started and stopped manually by the user so that any loading times or simply looking around in the scene were not added to the time measurement. Three time measurements were taken for each type of interaction, resulting in a total of six measurements per participant. In addition to the time measurement, the participants were asked to fill out the SUS and NASA-TLX questionnaires. For each condition, they were asked to fill out both questionnaires. The SUS questionnaire by John Brooke [Bro+96] measures the handling and usability of a system or, in this case, the type of interaction. In ten questions, participants can vote on the usabiliy of a system with five answer options each, which are values between one („Strongly Disagree“) to five („Strongly Agree“). These values are finally calculated to a total score. This makes it possible to compare the usability of both interaction types Cube and standard gesture selection and manipulation. The NASA-TLX questionnaire measures the workload that participants experience during their tasks [HS88]. The participants can vote on a scale from -10 („Very Low“) to 10 („Very High“) on how much they rate their Mental Demand, Physical Demand, Temporal Demand, Performance, Effort and Frustation during the task. The individual subscales of this questionnaire can also be compared well with each other between the different interaction types.

5.1.3 Hypotheses

H_1: There will be no significant difference in completion time between using Cubihand and the standard gesture selection and manipulation.

When using the standard gesture system of HoloLens 2 to select the distant small or covered objects, the participant is dependent on the use of the raycast and pointer system. The participant must first point to the object with their hand and the select it by doing a grasping gesture. Small objects require a high degree of precision, as the participant must hit the surface of the object exactly with the pointer's cursor. If the participant does not do this and accidentally grabs a corner or edge of the object, they will unintentionally manipulathe the object instead of selecting it. Wrong manipulations would have to be undone under certain circumstances. So the smaller the object, the more precisely the participant must be able to guide the pointer. The cursor can already slip if the participant wants to access the object after aiming and moves their hand too much in the process. So in the above study scenario, the user may need several attempts to grab the small sphere and therefore take longer to select the object.

The same problem occurs when the user has to manipulate the same sphere or the bigger cylinder. For this, the cursor must be aimed exactly at the corners of the objects if the user wants to perform a one-handed scaling. Even a two-handed scaling would again be problem with the small sphere, as even the pointer cursors of both hands would have to hit the surface of the sphere exactly. By using Cubihand as the interaction method, the user will not encounter such a precision problem, as they only had to release the cube as soon as the plane hits the object. Targeting the small sphere is also easier because the hitbox of the sphere is larger due to the lack of cursor-sensitive, manipulatable corners and edges, and thus there are no unintentional manipulations that might have to be undone.

By covering the platonic body with another virtual object so that the participant cannot even aim at the object with the standard gesture system pointer and its cursor, the user is forced to stand up in order to manipulate the object. Due to the actually short distance, on which no physical obstacles are to be found, and due to the direct grasping and manipulation with the hands which the user will perform, it can be expected that the participant saves time as a result. Overall, it can be assumed that both the time saved by using Cubihand and the time saved by standing up when using the standard gesture system balance each other out. This should generally result in the participants taking the same time selecting and manipulating objects through Cubihand than with the standard gesture system. From this, the hypothesis H_1 can be formed.

H_2: The user rating of Cubihand regarding System Usability and Task Load will be better than the respective ratings of the standard gesture selection and manipulation.

Not only the time taken by the participant while using the standard gesture system, but also the user satisfaction should be affected by the problems described above. Here, the frustration of selecting the small sphere in particular could increase, as the participant may experience several failures before sucessfully selecting and manipulating the sphere. The participant also has to concentrate hard while controlling the pointer. Furtehrmore, the fact that the participant has to stand up to perform one of the three selections and manipulations could have a negative impact on the user experience with the standard gesture system.

With Cubihand, on the other hand, the user can remain seated and still select and manipulate all three objects. Both the selection of the small sphere and the selection of the platonic body should be easier through the plane, as the speed of the plane and therefore the intersection with the objects can be adjusted precisely. The satisfaction of the participant should increase with that. All manipulations can be performed directly with the hands on the cube, so the participant does not have to expend much effort to aim at the objects and control them with delicate movements. The usability and the workload caused by the interaction type is determined with the help of SUS and NASA-TLX questionnaires. Since Cubihand is expected to make the system easier to use with distant objects and thus more user-friendly and easier to complete tasks, hypothesis H_2 can be raised regarding these two questionnaires.

In order to be able to test the two hypotheses, in the next section both the data on time measurement and the data from the evaluation of both questionnaires were analysed.

5.2 Evaluation

5.2.1 Data Evaluation

In the evaluation of the data set, the type of interaction was used as a factor in this study, so a distinction was made between cube and standard gesture selection and manipulation. The latter was described as Pointer in the evaluation for the purpose of a simpler overview.
In order to be able to test the first hypothesis H_1 „There will be no significant difference in completion time between using Cubihand and the standard gesture selection and manipulation.“, the data on time measurement were analysed. For the evaluation of the time measurement, the fastest and thus lowest time value in seconds of three measured values per interaction type was used. No average of the values was formed, as the participants mostly had a good learning rate, as shown in Figure 5.3. Despite a previous round of testing, the participants needed considerably longer in the first attempt than in the next two attempts. The fastest time value is therefore more representative than the average including the first attempt and will be referred to as the score in the process of this evaluation.

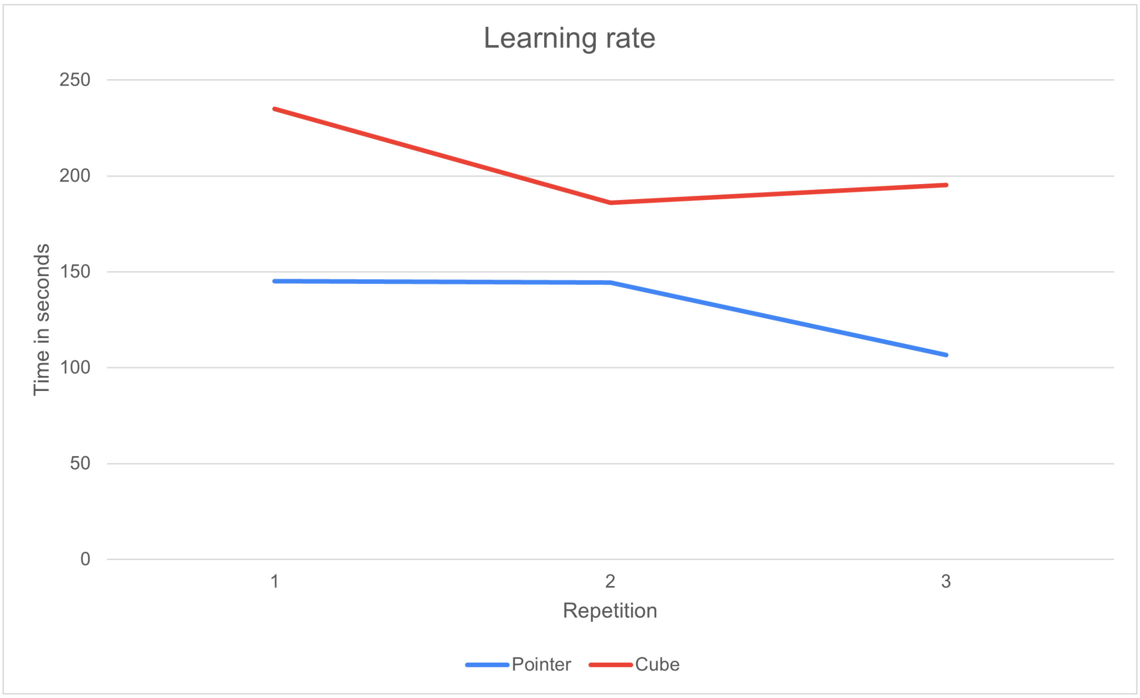


Figure 5.3: Learning rates of both interaction types visualised with the average of the three time measurement values. The average time needed for the first repetition of each interaction type was higher than in the following repetitions.

First, the score of both types of interactions was checked using a Shapiro-Wilk test to see if the measured data are normally distributed. If the p-value of the test is above the significance level of 0.05, it can be assumed that the data is normally distributed. This is a requirement for analysing the data with analysis of variance (ANOVA), or in this case, since there are only two variables, a pairwise t-test.

factor	statistic	p-value
Cube	0.927	.380
Pointer	0.908	.231

Table 5.1: Values of the score data in Shapiro-Wilk test. Both data sets are normally distributed.

Since the p-value of both types of interaction in the Shapiro-Wilk test is above .05, as you can see in Table 5.1, it can be assumed, that a normal distribution is given for both data sets.
In the further process, the artithmetic mean and the standard deviation of the scores of both factors were calculated. As displayed in Table 5.2, the mean of the pointer is clearly below the mean of the cube.

				Confidence Interval		Absolute Values	
factor	n	mean	standard deviation	min	max	min	max
Cube	11	176.	29.7	158.4487	193.5513	57.88113	170.2596
Pointer	11	98.6	33.1	79.03951	118.1605	124.1112	273.0285

Table 5.2: Descriptive statistics of the score data. Mean of the standard gesture system and 95%CI indicate, that the completion time with using the standard gesture system is significantly lower than the completion time with Cubihand. Therefore, the standard gesture system outperforms Cubihand regarding the completion time.

A pairwise t-test was conducted to compare the effect of both interaction types on the time measurement score. The results indicate that there was a significant difference in the score speed in completing the task with Cubihand ($M = 176, SD = 29.7, 95\%CI = [158.449, 193.551]$) compared to completing the task with the standard gesture system ($M = 98.6, SD = 33.1, 95\%CI = [79.04, 118.16]$), $t(10) = 5.634, p = .000$.

In addition, the mean and the 95% confidence intervals of the interaction types regarding the time score were calculated. As can be seen in both Table 5.2 and Figure 5.4, mean and the upper and lower bound of the confidence interval of the Pointer are lower than the values of the mean and confidence interval of the cube.

It can therefore be cleraly stated that the participants performed better over time with the use of the standard gesture selection and manipulation than with the use of Cubihand. Accordingly, hypothesis H_1 is disproven, since there is a significant difference in the execution time.

To be able to test the second hypothesis H_2 „The user rating of Cubihand regarding System Usability and Task Load will be better than the respective ratings of the standard gesture selection and manipulation.“, the data of the SUS questionnaire as well as the data of the NASA-TLX questionnaire were processed and analysed. The results of the SUS questionnaires were evaluated individually. For this purpose, the formula provided for the evaluation of a SUS questionnaire was used. The questions of the SUS questionnaire are numbered from 1 to 10. The formular differentiates between the sum of the answers of the non-even-numbered questions U and the sum of the answers of the even-numbered questions E. With these two variables, the formula is as follows: $score = ((U - 5) + (E - 25)) \cdot 2.5$

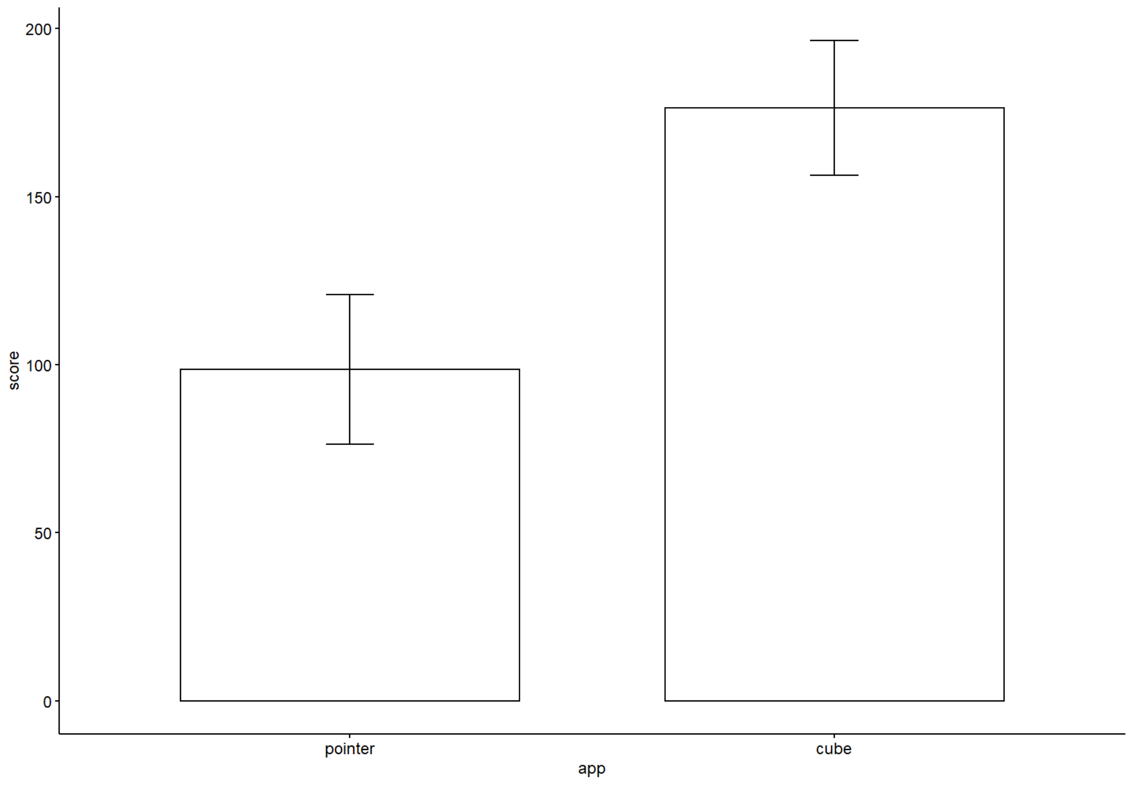


Figure 5.4: Mean and confidence intervals of the score of both interaction types, for a more detailed description see table above.

This results in two SUS scores for each participant, one for Cubihand and one for the standard gesture selection and manipulation. These scores were first run through a Shapiro-Wilk test, as was done for the time measurement, to guarantee the normal distribution of both data sets. Likewise to the time measurement, both p-values exceed the significance level of $p = .05$, as shown in Table 5.3. Thus, both data sets of the SUS scores are normally distributed and can be used for further analysis.

factor	statistic	p-value
Cube	0.896	.163
Pointer	0.924	.350

Table 5.3: Values of the SUS data in Shapiro-Wilk test. Both data sets are normally distributed.

Here, as well, the artithmetic mean with the corresponding standard deviation and confidence interval was calculated for both types of interaction. Table 5.4 and Figure 5.5 clearly show that the mean of the pointer with its confidence interval is above the mean of the cube with its confidence interval.

A pairwise t-test was conducted to compare the effect of both interaction types on the SUS score. The results indicate that there was a significant difference in the SUS score in completing the task with Cubihand ($M = 59.5, SD = 18.8, 95\%CI = [48.39, 70.61]$) compared to completing

				Confidence Interval		Absolute Values	
factor	n	mean	standard deviation	min	max	min	max
Cube	11	59.5	18.8	48.39011	70.60989	32.5	82.5
Pointer	11	81.1	9.71	75.36186	86.83814	67.5	95

Table 5.4: Descriptive statistics of the SUS data. Mean and 95%CI of the standard gesture system indicate, that the SUS score with using the standard gesture system is significantly higher than the SUS score with Cubihand. Therefore, the standard gesture system outperforms Cubihand regarding the SUS score.

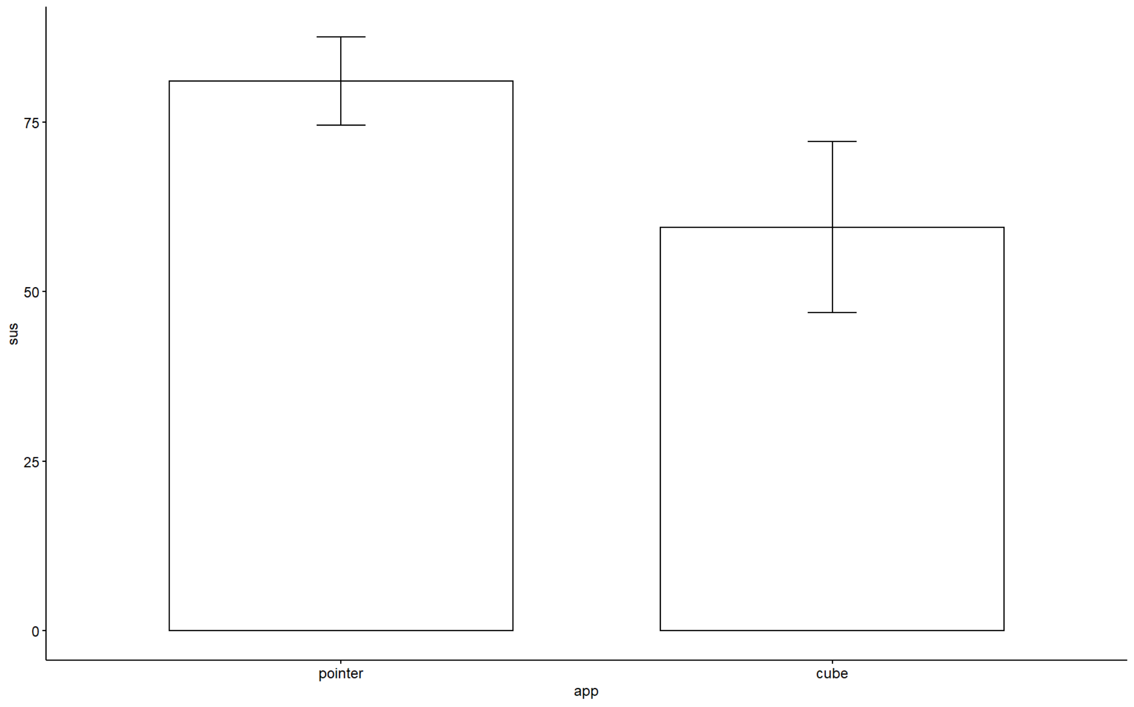


Figure 5.5: Mean and confidence intervals of the SUS of both interaction types, for more detailed description see table above.

the task with the standard gesture system ($M = 81.1, SD = 9.71, 95\%CI = [75.362, 86.838]$), $t(10) = 3.147, p = .01$.

This means that there is also a significant difference between the two data sets. Since the SUS questionnaire ideally aims to archieve the highest possible score of 100 points, standard gesture selection and manipulation also scores better among participants than Cubihand.

However, to clarify the validity of the second hypothesis, the NASA-TLX questionnaire is also evaluated. The structure of this questionnaire with its five independent subscales enables a separate evaluation of each individual scale. For the evaluation, the ranges of the scale, on which the participants could vote from a value of -10 to 10 for each question, were divided into four ranges. The first range, „Low“ applies to a value from -10 to -6. This is followed by the range „Low-Medium“, which applies from -5 to 0. „Medium-High“ is the range from 1 to 5, while „High“ applies from 6 to 10.
Figure 5.6 shows how many participants voted for each question within each range. The lowest possible score is desirable, so the ideal value is -10. At first glance it is clear that more participants voted in the lower ranges for standard gesture selection and manipulation than for Cubihand. The participants not only voted more in the two lower ranges of the scale, but even more in the lowest ranges.

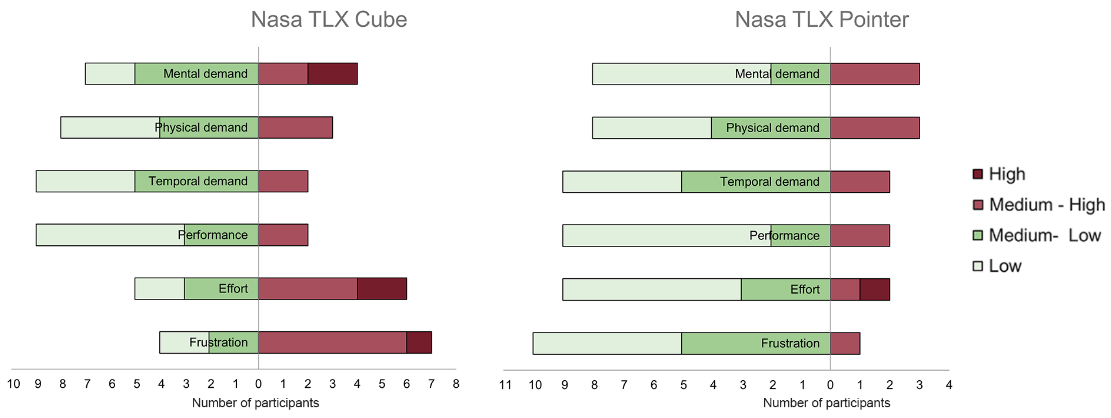


Figure 5.6: NASA-TLX results of both interaction types, classified in four ranges from „Low“ to „High“.

Overall, in Figure 5.6 both types of interaction are similar in Physical and Temporal demand. In the other subscales Mental Demand, Performance, Effort and Frustration, Cubihand scores less well than the standard gesture selection and manipulation. Thus, the workload of Cubihand seems greater.

In order to test whether the differences in NASA-TLX subscales are significant, a Shapiro-Wilk test was first carried out to check this data set for a normal distribution. It was found that the data are not normally distributed, which means that $p < 0.05$ applies partially. An evaluation using ANOVA is therefore not possible. An alternative for evaluation is the Wilcoxon test for two dependent samples. This was carried out for each subscale of the NASA-TLX questionnaire. The relevant results of the Wilcoxon tests across all subscales are listed in Table 5.5 for the purpose of clarity.

In all tests, the subscale scores were compared between the two interaction types. From the results of the Wilcoxon tests, it can be seen that here are no statistically significant differences between the NASA-TLX scores of the condition for Phyiscal Demand, Temporal Demand, Performance and

subscale	factor	median	T	Z score	p value
Mental Demand	Cube	8	64	-2.72	.007
	Pointer	4			
Physical Demand	Cube	8	25	-0.2	.838
	Pointer	5			
Temporal Demand	Cube	5	27	-0.49	.624
	Pointer	6			
Performance	Cube	4	19	-0.07	.943
	Pointer	4			
Effort	Cube	9	40.5	-2.08	.038
	Pointer	4			
Frustration	Cube	11	40	-1.23	.22
	Pointer	6			

Table 5.5: Results from the Wilcoxon test for each of the TLX subscales. On Physical Demand, Temporal Demand Performance and Frustration, there are no significant differences between both interaction types. On Mental Demand and Effort, there are significant differences, where the standard gesture system outperforms Cubihand.

Frustration. All p-values of these subscales are below the critical value of $p = 0.05$. It can also be seen that there are statistically significant differences between the NASA-TLX scores of the conditions for Mental Demand and Effort. Based on the results of the evaluation on Figure 5.6 and the median of the two interaction types in the respective subscales, it can be seen that the Pointer scores lower on average and thus better than Cubihand.

Both the results of the evaluation of the SUS questionnaire and the results of the individual subscales of the NASA-TLX questionnaire cleraly show that the participants had better a experience with standard gesture selection and manipulation than with Cubihand. Usability scored lower with Cubihand and the workload was greater. The participants showed a cleraly higher level of frustration and had to put more effort when using Cubihand. It can be deduced from this that the users prefer the use of standard gesture selection and manipulation to the use of Cubihand, which would disprove hypothesis H_2.

5.2.2 User Feedback

The feedback from the participants during the completion of the tasks was diverse. Many users mentioned that they liked the idea of Cubihand. They perceived the selection with Cubihand in particular more pleasant, especially that of the hidden and the small object. They did not have to stand up, which was more comfortable, and the small object was easier to select. Especially with the small object, users were often tempted to stand up as well in order to manipulate it directly with their hands without a pointer, instead of continuing to waste time trying to grasp it. Standing up for both objects while using the reference condition also had a positive effect on time performance, so the time measurements were smaller partly because of this.

Another reason for the shorter time measurements when using the standard gesture selection and manipulation was, according to the participants, the rather cumbersome handling of the manipulation with the cube. The manipulation was often not as intuitive as expected, especially the many small movements during translation were more of a hindrance than a help. It was not helpful that the cube took up too much of the field of vision, according to the participants. The field of the vision of the HoloLens 2 was already not very large. It was difficult that during the translation or other manipulations of the object by the cube, one could not see the object at all and thus had no control over the object at that moment. The participants were therefore often forced to search for the object again before they could continue with the manipulation. With the pointer, on the other hand, the translation could be done much more intuitively and quickly as soon as the object was grasped.

According to the participants, the speed of the scanning plane when moving in the selection process could have been another reason for the higher time measurement when using Cubihand. The participants did not perceive the plane to be too fast or too slow during the selection of the object themselves, but from the initial position to the objects further back in the space, the plane moved a little too slowly. Regarding the scanning plane, it was mentioned that the movement would be easily recognisable though the grid.

6 Conclusions and Future Work

6.1 Conclusion

With Cubihand, an alternative to the standard gesture system of see-through HMDs was developed for the selection and manipulation of distanced objects. In the process, the design of Cubihand was tested with a user study on the HoloLens 2 and compared with the standard gesture system of this HMD. The time needed by the participants to complete the task with the help of the respective interaction type was measured. In addition, the participants were asked to complete a SUS and NASA-TLX questionnaire for each interaction type and thus evaluate the different interaction types in terms of usabilty and workload. A total of eleven participants took part in this study. Both the completion time results and the results of both questionnaires indicated that the standard gesture system performed better than Cubihands in terms of completion time, usability and workload. However, it is noticable, that the percieved time spent on the NASA-TLX questionnaire does not differ significantly between the two types of interaction, but the actual time measured does. Either the participants do not perceive that they need more time when using Cubihand, or the number of participants is too small to detect a significant difference. With a larger sample, more significant differences in the NASA-TLX subscales might emerge.

The participants gave positive feedback about the idea of the design of Cubihand. In particular, the lower score of Cubihands in terms of usability was not due to the idea itself, but rather to the one aspect of the translation of the cube. Especially the selection of objects with Cubihands they perceived as easier than with the standard gesture systems pointer. In addition, when using Cubihand, the participants preferred one-handed manipulation of the cube, while with standard gesture system, two-handed manipulation of the objects was largely performed.

Unfortunately, during the course of the study it became clear that a time allocation of the one hour per participant was very short. Despite the fact that all participants had experience with media and most had experience with computer games, it was initially very difficult for the participants to get started with the HoloLens 2. The limited time ensured that there was no time to go through Microsofts gesture tutorial. This increased the likelihood that the gestures would not be executed correctly despite instruction and a test run for each app. It was striking how often the users did not reach for objects clearly recognisable for the HoloLens 2 or let go of them again. The right feeling for handling the HoloLens 2 and its hand recognition only develops after a rather longer, familiar handling of the system. It is precisely this experience with the gestures of the system that is actually a prerequisite for the correct handling of both the standard gesture system and the resulting Cubihand. Participants who had to start directly with the Cubihand app in the course of the within subject design of the study needed a very long time in the test run to get used to everything at once, which was very overwhelming. And even then, they mostly did not perform the gestures in a reliably controlled manner in the first or even in the further time-measured trials. Accordingly, for a new study in the further course of the design, more time per participant should definitely be planned in order to enable a tutorial beforehand and to largely equalise the conditions for all subjects.

It would also be interesting to compare the participants with regard to their experience with computer games or even with AR or VR systems. It would certainly be more informative to know to what extend this experience affects the handling or intuition and the spatial imagination. Most users of AR glasses eventually use these systems over a longer period of time, so that a familiar handling is given.

6.2 Future Work

In the course of the user study, in addition to feedback on the current implementation, participants also made suggestions for improving the system. It was generally suggested to implement a mixed system in which the user could decide for himself which method should be used for selection and manipulation. In this way, larger distant objects, that are easier to select with the pointer, such as the cylinder in the study, can still be selected and manipulated with the standard gesture system and its pointer. At the same time, however, one could also access hidden or very small objects with Cubihand that would otherwise force the user to stand up or at least require a greater amount of time and precision to grasp and ultimately manipulate. In a scenario such as the one presented in the study, a mixed system would thereby take advantage of both types of interaction.

In addition, if the system were to evolve, more work would need to be done on the implementation of the manipulation through the cube. For example, a way would have to be found so that the cube interferes as little as possible in the user's field of vision. Activating it on demand, for example in a mixed system as described above, would take up less of teh users field of vision, at least in the initial phase. During an active manipulation by the Cube, it would also be a good option to cover the Cube with a largely transparent material during this time. This way, the users would still cover part of their field of vision with their hand, but the three-dimensional cube would be transparent and at least a part of the field of vision would no longer be covered by it. The mapping of the Cube onto the object would also have to be designed more precisely. Much more fine-tuning would have to take place, such as with translation. This should no longer require many small mvoements, but should be as intuitive as using the pointer to move the object in one motion. The transparency of the cube would also make this more possible without the user losing the object. To make the active mapping of the object to the Cube visible, one could connect both objects with a Ray, similar to how the hand has a connection to the object through the pointer when it is selected and manipulated with the standard gesture system. Furthermore, a modulated speed of the scanning plane could be considered. Its speed is currently already dependent on the distance to the user and thus becomes faster at greater distance to the user, but still feels too slow when the object to be selected is further back in space. At the same time, the speed cannot be increased further, as selecting a small object would otherwise require too much precision. According to participants of the study, it would be worth considering the possibility of switching between to levels of speed, such as higher movement-speed and a more precise and lower selection speed.

With all these ideas, a further development of Cubihand has great potential to provide a useful and simple alternative for especially the one-handed selection and manipulation of distand small and hidden objects in a simple in precise way.

Bibliography

[BH97] D. A. Bowman, L. F. Hodges. “An Evaluation of Techniques for Grabbing and Manipulating Remote Objects in Immersive Virtual Environments”. In: *Proceedings of the 1997 Symposium on Interactive 3D Graphics*. I3D ’97. Providence, Rhode Island, USA: Association for Computing Machinery, 1997, 35–ff. ISBN: 0897918843. DOI: 10.1145/253284.253301. URL: https://doi.org/10.1145/253284.253301 (cit. on p. 17).

[Bro+96] J. Brooke et al. “SUS-A quick and dirty usability scale”. In: *Usability evaluation in industry* 189.194 (1996), pp. 4–7 (cit. on p. 34).

[BZO+17] A. Bellarbi, N. Zenati, S. Otmane, H. Belghit, S. Benbelkacem, A. Messaci, M. Hamidia. “A 3D interaction technique for selection and manipulation distant objects in augmented reality”. In: *2017 5th International Conference on Electrical Engineering - Boumerdes (ICEE-B)*. 2017, pp. 1–5. DOI: 10.1109/ICEE-B.2017.8192012 (cit. on p. 17).

[HS88] S. G. Hart, L. E. Staveland. “Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research”. In: *Advances in psychology*. Vol. 52. Elsevier, 1988, pp. 139–183 (cit. on p. 34).

[Mic21a] Microsoft. *Mixed Reality Design: Bounds Control*. https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/features/ux-building-blocks/bounds-control?view=mrtkunity-2021-05. Accessed on 2021-10-28. 2021 (cit. on pp. 18, 22).

[Mic21b] Microsoft. *Mixed Reality Design: Cursors*. https://docs.microsoft.com/de-de/windows/mixed-reality/design/cursors. Accessed on 2021-09-24. 2021 (cit. on p. 18).

[Mic21c] Microsoft. *Mixed Reality Design: Near Interaction Grabbable*. https://docs.microsoft.com/en-us/dotnet/api/microsoft.mixedreality.toolkit.input.nearinteractiongrabbable?view=mixed-reality-toolkit-unity-2020-dotnet-2.7.0. Accessed on 2021-10-28. 2021 (cit. on p. 25).

[Mic21d] Microsoft. *Mixed Reality Design: Object Manipulator*. https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/features/ux-building-blocks/object-manipulator?view=mrtkunity-2021-05. Accessed on 2021-09-24. 2021 (cit. on p. 18).

[Mic21e] Microsoft. *Mixed Reality Design: Point And Commit*. https://docs.microsoft.com/de-de/windows/mixed-reality/design/point-and-commit. Accessed on 2021-09-24. 2021 (cit. on p. 18).

[Mic21f] Microsoft. *Mixed Reality Toolkit*. https://docs.microsoft.com/de-de/windows/mixed-reality/mrtk-unity/?view=mrtkunity-2021-05. Accessed on 2021-10-28. 2021 (cit. on p. 25).

[Tec21a] U. Technologies. *Prefabs*. https://docs.unity3d.com/Manual/Prefabs.html. Accessed on 2021-10-28. 2021 (cit. on p. 25).

[Tec21b] U. Technologies. *Unity Game Engine*. https://unity.com/de. Accessed on 2021-10-28. 2021 (cit. on p. 14).

[WHB+18] M. Whitlock, E. Harnner, J. R. Brubaker, S. Kane, D. A. Szafir. "Interacting with Distant Objects in Augmented Reality". In: *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 2018, pp. 41–48. DOI: 10.1109/VR.2018.8446381 (cit. on pp. 13, 17).

All links were last followed on November 15, 2021.

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