HoloHue: Manipulating Smart Lights with Microsoft MR Maps

Garrett Devereux
University of Washington
gdev@uw.edu

Daniel Thor Gudmundsson
ETH Zürich
dgudmundsson@student.ethz.ch

Kevin Shao
MIT
kshao23@mit.edu

Laura Wülfroth
ETH Zürich
laura.wuelfroth@inf.ethz.ch

Abstract

With the growing increase of smart devices, Mixed Reality developers are creating applications that give users a vast array of ways to interact with and manipulate their homes. In the realm of smart lights, the current approach has been to have a connecting phase, then display holograms to visualize and modify the physical lights. However, with no data storage or localization techniques, this results in an application that requires a lot of set up time to initially connect and position all of the objects, and no way to save the positions of the virtual objects for future use.

To provide a remedy to this dilemma, we propose HoloHue, a Mixed Reality application aiming to connect to smart devices, visualize them through augmented reality, give a clear and intuitive way for users to interact with and manipulate the objects, and store mappings of the objects to allow for persisting locations of 3D objects. With the help of Microsoft's new pre-released MR Maps mapping and localization technology, HoloHue is able to easily store mappings of environments and allow users to re-enter their environments with their content in the same location.

Through our research, we provide real-world demonstrations of HoloHue that illustrate possible use-cases and testing of the functionality in different environments. User studies also help support the user-experience and overall intuitiveness of the app. Our research provides an example of what is possible with MR Maps, as well as a framework and application that we hope will be used and extended to other realms for future work.

1. Introduction

Humans constantly work to interact with and modify the world around them. Whether bringing groceries home, turning up the heater, or flipping on a light switch, we are continually changing the area around us to create an environment more suitable for comfortable living. As the advancement of technology continues to press forward, the presence and accessibility of smart devices in homes increases and provides humans with more and easier ways to increase comfort. From smart doorbells to smart speakers, and smart appliances to smart lights, a house can now be controlled digitally from anywhere. Additionally, with the growing popularity of Mixed Reality and the release of the Hololens 2, there are now more ways than ever for a human to interact with their surroundings.

Motivated by this, we propose *HoloHue*, a Mixed Reality application to provide humans with a simple and intuitive way to manipulate their smart lights. *HoloHue* is differentiated from previous work through our new contribution: the integration of Microsoft MR Maps, pre-released mapping and localization technology. With this in mind, *HoloHue* has four main areas of focus: (*i*) connecting to smart devices in the real world, (*ii*) visualizing them through augmented reality, (*iii*) providing a clear and intuitive way for the user to interact with and manipulate objects, and (*iv*) store mappings of the environments, allowing the locations of the virtual objects to persist when the application is closed and re-localize when re-entering the environment. With a clear system structure and real-world demonstrations of application [3], we hope to inspire future work utilizing MR Maps and smart devices.

2. Related Work

Manipulating Smart Lights. There are a few similar Mixed Reality Applications to *HoloHue*, the most popular being AfterNow's *Hue Lights* [2]. This application lets you move around virtual lights linked to Philips Hue Lights and then allows you to change their color and brightness with gestures or voice commands. However, there is no support for the mapping of environments meaning that the location of the virtual lights does not persist after the application is closed.

Persisting 3D Location. Previously, the standard for persisting 3D content was Microsoft's Azure Spatial Anchors [4]. However, this came with its own difficulties and overhead. As our project is in collaboration with Microsoft Mixed Reality and AI Labs, we were tasked to move away from this and test out the new MR Maps technology.

3. Design Principles

One of the most important aspects of any application is good user experience. We decided early on to make the most of the Hololens by designing our app around its most unique features, i.e. direct manipulation of objects, voice commands, and eye tracking, working to use as few buttons as possible for the best experience.

When designing how the user should interact with the virtual lights, we found the technique of *Bodystorming*. In a nutshell, bodystorming is using physical props to represent digital objects, user interfaces, and user experiences [8]. We used a variety of objects we had at hand at the time e.g. a water bottle, and pretended that this was our future virtual light object that would be used to control the physical lights. We imagined that this digital object had been designed by someone else and the only thing that we knew is that it could control lights. Our task was then to figure out which commands were possible and what they did. Using this approach, we quickly found some intuitive commands such as double tapping to turn a light on, using certain hand gestures to control brightness, and attempting to use voice commands. This formed our base understanding of how a user might expect to use our application.

Another thing we had to keep in mind was how to integrate the HomeAssistant system into our application in a user-friendly way. *HoloHue* is not meant to manage the smart devices connected to HomeAssistant, so we decided that our app should only display lights that are already connected. This means that we treat HomeAssistant as the source of truth and will only modify the state of lights that are already displayed in the system. To make this clear, we needed to create a user-friendly way to display the HomeAssistant information. Using bodystorming as before, the most intuitive way we found was to have a menu or catalog displaying a collection of the lights. This menu could then be opened and closed in a seamless way using hand gestures, similar to how we use the wrist to open the main menu of the Hololens.

These principles were the main driving points behind the design of *HoloHue*. However, it is important to note that due to late access to Microsoft MR Maps, we did not have any information on the mapping and localization and could not consider it in our design meetings. To combat this, we worked to develop modular components that could be developed in parallel and permitted the integration of the MR Maps solution in the last few weeks of development.

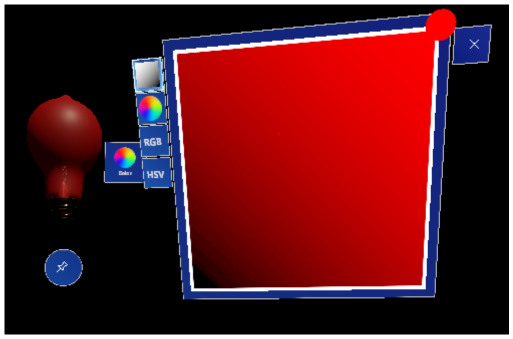

Figure 1. The Virtual Light, ColorPicker menu, and pin

4. Method

HoloHue is broken up into 5 main components that work to accomplish the four areas of focus. These components are Virtual Lights, Storage, Mapping and Localization, the Home Assistant Interface, and the User Interface that ties everything together. We implemented *HoloHue* for the Hololens 2 using the game engine *Unity*. To speed up development, we utilized the *Mixed Reality Toolkit* (MRTK) which provides components and functionalities for mixed reality applications. [6].

4.1. Virtual Lights

The core functionality of *HoloHue* is the interaction between a user and a virtual light. As described in the design, we used bodystorming to identify the best and most intuitive ways to interact with the virtual light. In this section, we will cover those features in more detail.

The most basic feature of the virtual light is object manipulation, or the user's ability to move the object around in space. After initial testing of the app, we discovered when a user tried to change the state of a virtual light (e.g. toggling on and off switch), they would often accidentally move the light as well. This provided some frustration, as it was quite difficult to perform a simple task without messing with the position of the light. To fix this, we added a pin button that toggles whether the light can be moved or not (see Figure 1). From a user perspective, this adds an extra click when placing a virtual light, but this trade-off gives users a clear way to re-position lights resulting in a much smoother experience.

In order to turn a light on and off, we figured that either single-tapping or double-tapping a light was the most intuitive action. We experimented with both, but in the end decided to go with double-tapping as it felt easier, smoother, and was more enjoyable to use.

The third and final basic feature of the virtual light is changing the color. During the bodystorming session, we worked to come up with novel ways of color manipulation, but we always ended up at the same place: color palettes and color wheels. In theory, we felt that a color palette was probably the best way for a user to change colors as it is used in most other applications that offer some color functionalities. Additionally, we felt that while not as intuitive, giving the user RGB or HSV sliders would provide a precise level of color customization if needed. When implementing these features, we found that MRTK already had an experimental color component ColorPicker that used these ideas, which we utilized along with some adjustments [5].

So far we have covered the basic functionalities of the virtual light, but throughout development we had more ideas we wanted to explore. One of them was controlling a light's brightness by simply looking at it and using some hand gesture. This would remove any requirement of lining up the hand pointer and precisely clicking a button, making manipulating lights faster and easier. In order to accomplish this, we had to implement eye tracking in our application which comes out-of-the-box in MRTK by just changing some of the project's set-up. Secondly, we had to come up with a hand gesture that was easy and intuitive to use, and make sure that it was simple enough to implement in limited time. Our solution was to use the Hololens supported hand tracking data and read the average curl of the user's fingers. We could then use this value to set the brightness percentage of the lights, allowing the user to curl their fingers to increase the brightness, or lay their hand flat to reduce it. This proved to be a simple yet effective way to increase the smoothness and novelty of the user experience.

Finally, we wanted to utilize the Hololens's voice recognition capabilities. From our personal experience using the Hololens, voice commands can be very convenient and often faster than using your own hands. However, they also have their downsides, e.g. difficulties recognizing your commands or you might be in an environment you are not comfortable talking in [9]. With this in mind, we wanted to keep the commands short, simple, precise, and ensure that a user was never required to use a voice command. This resulted in us using voice commands to change the color of a light. For example, by pointing at a virtual light bulb and saying "red" or "blue" etc., the light will change to that color. We initially planned to use the same eye gaze solution as we used for the brightness gesture, but that was not possible due to the architecture of the virtual light component in Unity. This is because a voice command is only triggered when an object is in focus, but for some reason that was not working properly when we just used eye gaze. This was most likely caused by various sub-components of the virtual lights that appeared at different levels. Trying to fix this bug had cascading effects by breaking other functionalities which proved to be hard to fix. Instead, we mitigate this by requiring the user to point at a light and then use the voice command. From a user's perspective, this is not as good as using the eye gaze, but including this additional feature gives the user more flexibility than leaving it out would and fits in our limited development timeline. We leave the solution to this problem as future work.

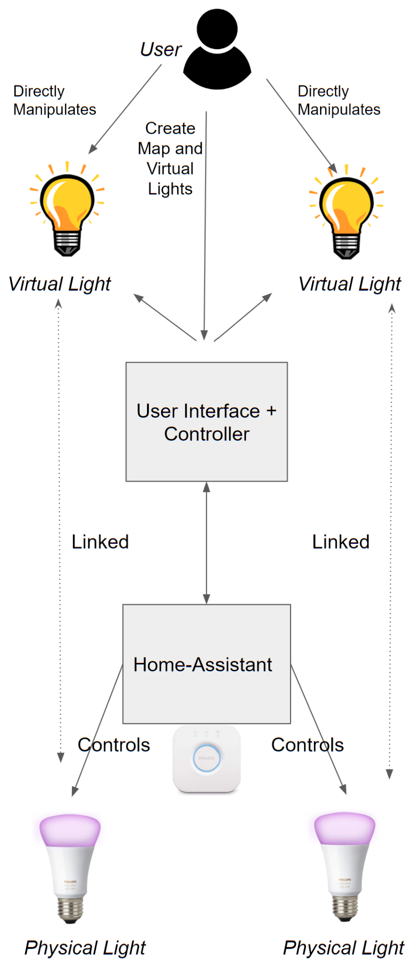

Figure 2. System Overview of *HoloHue* components.

Putting everything together, we end up with a virtual light which supports multiple functionalities, utilizing the most novel features of the Hololens. To get a better idea of what it all looks like and feels like together, see our demonstration video for comprehensive examples of how to manipulate lights [3].

4.2. Storage and spatial anchors

In order to have persistent data between sessions, we needed data storage along with an *Azure Spatial Anchor* resource for the mapping functionality. After weighing our options, we decided to use the document store database Azure Cosmos DB since members of our group were already familiar with it. This also provided a flexible way to store and access data along with the possibility of using multiple devices at the same time.

4.3. Mapping, Localization, and Anchoring

The Mapping and Localization technology was provided by Microsoft in the new pre-released *Mixed Reality Maps* (MR Maps) technology. It allows a user to create a map of their environment and provide spatial anchors which can be used to persist content between sessions. When the user opens *HoloHue*, they can either enter an already existing map or create a new one. During map creation, the user has to walk around their space and make sure that the MR Maps collects enough data in order to construct a cohesive map of the environment. We provide the user with some feedback in the form of feature points detected, which should give a clear indication of which parts of the environment have been mapped and which parts have not. Once the user is satisfied, they can finish the mapping session and the Azure servers will finish constructing and optimizing the map.

Once a map has been created, the user can enter that map and start querying MR Maps for spatial anchors. When a virtual light is created and placed in space, we query MR Maps for the nearest spatial anchor, which is then used as a reference point for our virtual light. Essentially, we compute the relative location of the spatial anchor and a transformation to the new position of the virtual light and store that in our content database along with information about the anchor. Every time the virtual light is moved, this process is repeated and the record in our content database is updated.

If a user enters an existing map, we begin by querying our content database to see if the map already contains any virtual lights. If so, the light records will be loaded into memory, but we do not render its hologram until its spatial anchor is discovered through localization. During this process, MR Maps can trigger a callback function once an anchor is discovered in the environment, and if that anchor has some virtual lights attached to it, we render those virtual lights at the previously saved location. This completes the Mapping and Localization portion of the application that sets *HoloHue* apart from previous work.

4.4. Home Assistant Interface

As we had access to various smart lights connected to a Raspberry Pi running the Home Assistant OS, we chose to utilize the Home Assistant open-source interface [1]. Home Assistant provides a RESTful API to get information and send requests to their web front end, which can be reached at the URI *http://homeassistant.local:8123/api*. Since the URI of Home Assistant is always the same, our application can conveniently find the web frontend if there is a Home Assistant in the same network. To communicate with the API our application sends an HTTP request containing an authorization token and a JSON file with the information needed for the request. We implemented four functions to handle the requests needed

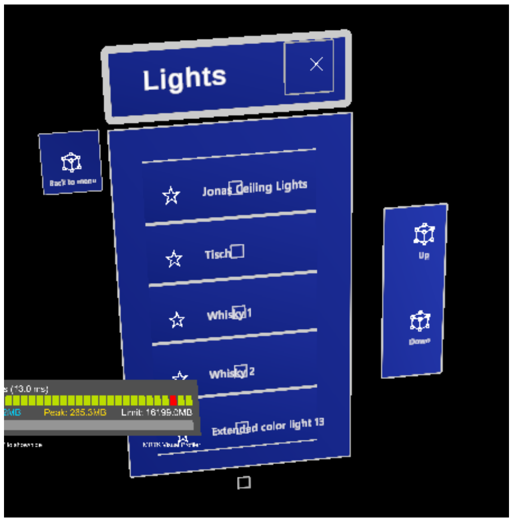

Figure 3. The light menu

to control the lights according to the states of the virtual lights. The first one, `GetAllStates()`, is called when the main lights menu is opened the first time and sends a request to the Home Assistant to get all lights and their states, where a state stores if the light is on or off, its brightness, and its current RGB color. Additionally, we have a similar function `GetState(Id)`, which gets the state of one specific light associated with the light Id. To toggle the light similar to a light switch we implemented a function `toggleLight(Id)`, which sends a request to Home Assistant to turn the light on/off. The last function we implemented is used to update the color and brightness of the light connected to Home Assistant. `ChangeLightState(Id)` sends a request containing the color and brightness of the lamp associated with Id and is called at a frame if the state of the virtual light changes compared to the previous frame or when the state is changed using hand gestures or voice commands. The functions could easily be extended to work with other home automation systems as well, such as Philips Hue, by adjusting the requests according to their API.

4.5. User Interface

There are two kinds of user interfaces in *HoloHue*. The first is the main menu which displays the available environments and provides a way to create a mapping of a new environment. The second is what we call the light menu where we display the smart lights available in the HomeAssistant (see Figure 3). We reuse a lot of components for both of these UI's since they share a lot of features. Their design is heavily influenced by typical mobile designs when it comes to displaying a collection of items, that is, a scrolling object collection where each item can be selected, and an "add" button at the bottom of the screen.

When designing the interfaces, we wanted to minimize the number of buttons so as to not distract from the main purpose of this application, interacting with lights. Using this motivation, we included the option to hide the UI when not in use and open it only when the user needs it. This inspired us to use a type of menu termed *Hand Menu* in MRTK. Hand menus can be opened by simply looking at your palm [7]. We choose to keep the menu open until the user manually closes it, so they can simply drop down their hand after they opened the menu to close it. In order to further improve the user's experience, the menu will follow them around so they do not lose track of it.

The light menu displays a collection of all the lights available in HomeAssistant. Each item can be clicked to open a dialog containing more information about the light and if it has an already linked virtual light. If the light is not linked, the user can choose to link it which will spawn a new virtual light in front of them. They can then position the light where ever in physical space and interact with it. If the light is linked, the user is given the option to un-link it and remove the corresponding virtual light object. We these options, we can guarantee a one-to-one or one-to-none relationship between physical lights and virtual lights.

Despite being initially influenced by mobile design, we found that the experience was not the same in the world of mixed reality and had to make some adjustments. Since we were working with a scrolling object collection, we assumed that a user would simply use their fingers to scroll directly on the collection as they do on a smartphone. However, this is where one of the challenges of designing in the virtual space came to light. What might work on a mobile phone or a laptop, did not necessarily work in a virtual space. We found that scrolling directly on the object collection was frustrating because sometimes we would accidentally press one of the items instead of scrolling and vice-versa. To fix this, we added two scroll buttons to the right, one for scrolling up and one for scrolling down. This made the whole scrolling experience much smoother than before. Overall, this process outlined the differences between virtual and mobile spaces and made us really consider how a user will interact with an object in correspondence to what space they are in.

5. Results

In order to evaluate our results, we need to look at metrics of functionality and user experience to determine if we successfully met our 4 focus points. Goals *(i)* connecting to smart devices, *(ii)* visualizing them through augmented reality, and *(iv)* storing mappings of environments can quite easily be evaluated by testing to see if the application does what it is supposed to. In our Demonstration Video we clearly show how to use each feature of the app with clear feedback that everything is implemented and working correctly [3].

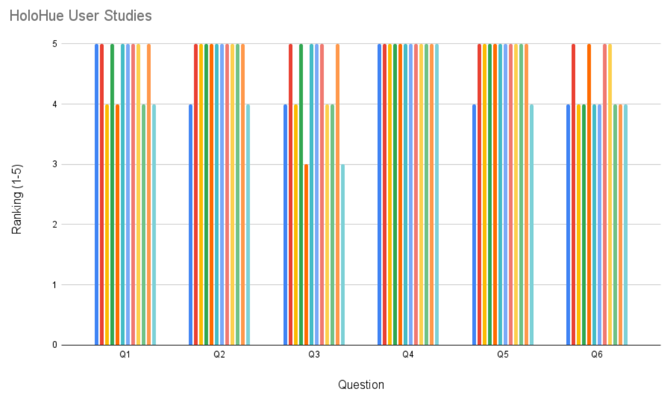

Figure 4. Results of the *HoloHue* User Study

For goal *(iii)* providing a clear and intuitive way for the user to interact with and manipulate objects, we worked to conduct user studies to get feedback on what users thought of *HoloHue*. With limited time and resources, we were only able to get 12 test subjects, who used a virtual version of *HoloHue* which was not connected to Home Assistant. In the experiment, we provided a pre-mapped area and asked the subject to link a virtual light. They then were prompted to manipulate the light using all of the ColorPicker options as well as the voice commands and hand gestures. Finally, at the end of the study, we asked them a series of questions to gain insight into their experience.

In the study, there were six questions that asked the user to rank a statement on a scale of 1 to 5, where 1 represented strongly disagree and 5 represented strongly agree. The statements were (*i*) "HoloHue makes it easy to manipulate lights", (*ii*) "I enjoyed the numerous different options of changing lights", (*iii*) "The catalog and gestures provided a smooth and comfortable experience", (*iv*) "HoloHue was intuitive to use", (*v*) "HoloHue was enjoyable to use", and (*vi*) "If you had easy access to a Hololens and a multitude of smart lights, would you download HoloHue". The results of the study are shown in Figure 4 above.

The first important finding is that we met the focus of being intuitive, with 100% of users strongly agreeing that *HoloHue* was intuitive to use. All of the users also agreed that *HoloHue* made it easy to manipulate lights, with 66.7% strongly agreeing with a ranking of 5 and 33.3% with a ranking of 4.

Additionally, we asked each user to rank all of the features provided to manipulate the lights (Color Palette, Color Wheel, RGB/HSV Sliders, Voice Commands, Knuckle Curl, and Double-Tap). The Knuckle Curl hand gesture seemed to be the most popular, making it in the top 3 of 100% of users, and was the top answer for 33% of users.

The next most popular feature was the Color Wheel which made it in the top 4 of 100% of users and was in the top 2 of 42% of users. During this testing, no one seemed to have a problem with any feature, and the main response we got was that it was really intuitive and easy to use. The only thing that strayed was the smoothness of the experience, with some of the buttons being hard to press and frustrating users, causing 16.7% of users to rank the smoothness and comfortability of the app as a 3, 33.3% ranking it a 4, but still a majority 50% of users with the top ranking of 5.

6. Future work

HoloHue offers many different ways for a user to interact with virtual lights. However, there is still work left with more features and improvements possible. We are aware of a few issues that mostly are related to bugs in the MRTK package. For example, a scrolling object collection that is used in most of the user interfaces is buggy at times. Because of this, pressing buttons in the collection sometimes does not work because the click event most likely does not get propagated down to the button. If this happens, it can be mitigated by trying to close and reopen the menu, or by grabbing the menu and shaking it around. We still are not quite sure why this works.

There is also room for improvement in the user interface. At the moment, the UI can sometimes display stale data, since we do not always update it with the most recent data from the HomeAssistant. This can be easily fixed by simply updating the light collection stored in memory after each request and then updating the UI accordingly. Furthermore, we are currently just using the basic icons found in MRTK which have a lot of limitations. Adding some custom icons would help improve the UI and make it more clear to users what each button does.

Additionally, there is a dialog when a user tries to create a new map that prompts the user to remove all holograms before starting mapping. If the user does that while the app is open, the Hololens will lose all orientation and will no longer display any holograms, forcing the user to reset the Hololens. We consulted our supervisors about this, and they were as confused as we were. To mitigate this, we instead clear all holograms before opening *HoloHue* and then proceed to the mapping mode and simply ignore this dialog. However, Microsoft released a new version of MR Maps which according to them does not require the user to remove all holograms. With limited time, we have not yet been able to factor this release into our project.

The points mentioned above are the most important fixes that we are aware of, but there is also room for other improvements. The most important is to add a tutorial with help functionality. For example, it might not be obvious to new users how to open the light menu by looking at their palm. It is also not obvious which voice commands are possible and which are not. Moreover, different users might have different opinions of which colors should be used. This became evident during one of our demos where a user tried to say the color “Turquoise”, which we had never considered as one of the voice command colors.

Another feature we would like to look into is extending a map. The MR Maps package offers the possibility to extend an already existing map, but due to previous issues with removing holograms, we decided against this. With the new version of MR Maps available, we no longer have to worry about this issue and can allow users to extend their already existing maps with a few small changes.

Finally, one of the most important improvements is better user feedback during mapping sessions. Right now, we display feature points in spaces that have been mapped properly. This gives a good indication to computer scientists who have some experience in computer vision of what is happening. The same could not always be said about a regular users, since to them they are only seeing some weird points in space and may not understand why a plain white wall gets almost no points and a cluttered desk gets many. A future version of our application should therefore give the user instructions on where to walk, where to aim the Hololens, and notify them when they can move one. This should give the user more confidence in the quality of their mapping procedure.

7. Conclusion

We present *HoloHue*, a Mixed Reality application aiming to connect to smart devices, visualize them through augmented reality, provide a clear and intuitive way for users to interact with and manipulate the objects, and store mappings of the environments to allow for persisting locations of 3D objects. Demonstrations and experiments across a variety of environments show that *HoloHue* successfully meets these four focus points while building upon previous work with the new integration of Microsoft MR Maps.

It is important to note that with limited time, we were only able to collect a small amount of user data to base our results. In the future, more user studies should be conducted in more robust environments, possibly with in-depth studies of users utilizing *HoloHue* unprompted in their homes. Additionally, there is more to be done in the development of *HoloHue* with improvements suggested above. Overall, we hope the research provided helps to inspire future work in the realm of MR Maps and smart devices and would love to see *HoloHue* used in the real world.

References

[1] Homeassistant. http://www.home-assistant.io. 4

[2] AfterNow. Hue lights, 2017. https://www.microsoft.com/en-us/p/hue-lights/

9n6jxjf9f4g9 ? SilentAuth = 1 & activetab = pivot:overviewtab. 1

[3] Shao Wuelfroth Devereux, Gudmundsson. Holohue demonstration video, 2022. https://www.youtube.com/watch?v=6uHrTjHrINY. 1, 3, 5

[4] Microsoft. Azure spatial anchors. https://azure.microsoft.com/en-us/products/spatial-anchors/#overview. 2

[5] Microsoft. Microsoft colorpicker. https://learn.microsoft.com/en-us/dotnet/api/microsoft.mixedreality.toolkit.experimental.colorpicker.colorpicker?view=mixed-reality-toolkit-unity-2020-dotnet-2.8.0. 3

[6] Microsoft. Mixed reality toolkit. https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2. 2

[7] Microsoft. Hand menu, 2022. https://learn.microsoft.com/en-us/windows/mixed-reality/design/hand-menu. 5

[8] Microsoft. Thinking differently for mixed reality. 2022. https://learn.microsoft.com/en-us/windows/mixed-reality/discover/case-study-expanding-the-design-process-for-mixed-reality?source=recommendations. 2

[9] Microsoft. Voice input, 2022. https://learn.microsoft.com/en-us/windows/mixed-reality/design/voice-input. 3