Final Year Project

Expanding MAAP Annotation

Adam Hilario

Student ID: 18202271

A thesis submitted in part fulfilment of the degree of

BSc. (Hons.) in Computer Science

Supervisor: Anthony Ventresque

UCD School of Computer Science

University College Dublin

March 14, 2024

Table of Contents

GitLab Repository: https://gitlab.com/adhilario1/MAAPAnnotate2022

Abstract

The goal of this project, and is outlined in this paper, is to make the current iteration of the MAAP Annotate software less an individualistic experience to one that would support collaboration between peers. This involves enabling the ability to have persistent annotation data, that can exist outside of the instance of the running application, and allowing it to be shareable through conventional means of file sharing, as well as through cloud services. Additional consideration was made to the longevity of the application by creating a port of the application to the HoloLens 2, Microsoft's current flagship augmented reality device.

Chapter 1: **Introduction**

Stone carvings and paintings on megaliths are some of the earliest recorded forms of early human's artistic expression. Being able to analyse and interpret these pieces would offer insight as to how their creators' daily lives and communities. The first iteration of this project enabled the digital cataloguing of the art as 3D Objects and the ability to annotate such work through in an Augmented Reality (AR) environment.

The Megalithic Art Analysis Project, or MAAP, is a multidisciplinary project involving multiple universities in Ireland and France.[1] The focus of the Computer Science and Information technology side of the project is to facilitate digital archaeological research by allowing researchers to analyse megalithic art without the art or the researcher being present. This is accomplished using a combination of different technologies from the algorithm designed to analyse the 3D models to the Microsoft Kinect, originally developed for gaming, whose motion and depth sensing capabilities scan the megalith and collect the data for 3D modelling.

Just as stone age cultures used megalithic art as a mode of expression to communicate ideas, we currently use social media in a similar way. Other than being poetically symbolic, modelling the mechanics of annotation system after various social media platforms has its uses. Since an element of this system will be simultaneously open-source and an academic tool, using stratification methods such as groups or communities as a subset of the overall user base as seen in platforms like Facebook will be an overall boon to the structure of the system. However, as we are beginning to see the major pitfalls of content moderation, or lack there of, on sites like Facebook and Twitter, creating an open-source content platform that aims to create a collaborative, constructive environment with a focus on research will need a method of recognising deconstructive content.

While the current iteration of the project serves its function in its ability to distinguish megalithic art apart from natural weathering as well as create 3D models of the megaliths, there are still aspects of the current program that need to be improved for ease of use purposes. The primary issue is the current version of the annotation system does not support public annotations, in that the annotations that one researcher might make are stored locally and only accessible by the researcher that made them. Therefore, the solution would be to create a platform and database to create and store annotations that can be viewed, commented, and edited by researchers globally, functioning similarly to research focused "social network." If time permits, other work can be made towards researchers' ease of use. This would include separate the scanning and processing aspects of data collection, so making expeditions to collect data does not require hauling heavy equipment and can be done on portable technology, such as a smart phone.

Chapter 2: Project Specification

2.1 Core Requirements

- Create a database for third party cloud storage of annotations
- Make annotations comment-able and editable.
- ~~Create two tiered annotation environment, one that is open-source and another that is invite-based for the university/research level~~
- Port the original iteration of the MAAP Annotate Software from HoloToolKit using the Windows Universal Platform to the Mixed Reality Toolkit using OpenXR

2.2 Discretionary Requirements

- Create user environment to cater towards use with the Microsoft HoloLens. (e.g. speech dictation)
- ~~Update current code for compatibility with current Unity release.~~ Covered by the port under Core Requirements.
- ~~Further sub-strata of aforementioned two tier environment. (e.g. location, university, popularity)~~
- Enhance original tools to enhance user experience and reorganize menu layouts.

2.3 Advanced Requirements*

- ~~Create live messaging system to facilitate real-time collaboration.~~
- ~~Install some form of content moderation.~~
- ~~Enable a HoloLens free option, that would still utilise the 3D models but could be manipulated by mouse.~~
- Create a controller for saving and loading annotations.
- Enable PC Remoting to run the application on a PC and stream to an AR headset.

*Due to the porting requirement, the original direction of the project was unfeasible. Therefore the advanced and much of the discretionary requirements needed to change.

Chapter 3: Related Work and Ideas

3.1 MAAP Annotate

The original MAAP Annotate paper[2] outlines the implementation of the Annotation in its original state. The paper begins by outlining the motivation to create the annotation system as to create a digital system that can replicate megaliths in 3D space and to allow the tagging, altering and manipulation of these models without altering the original. The next portion of the paper describes proper techniques on how to use the system such as available tools and recognised gestures, in addition to an evaluation of how test users assessed the design of the tool. Evaluation of the program was conducted by 10 archaeologists across diversified strata at University College Dublin. After being introduced to the Annotation System and the HoloLens itself, testers were allowed to explore the program at will. Feedback was recorded according to System Usability Scale, which had a positive reception with a score of 80.75 out of 100.

The main critique of the Annotation was the drawing selection tool was hard to use, with 60% rating it a 3/5, meaning the experience was middling. It should therefore be the focus, in terms of basic improvements, to address some these concerns. The most crucial of these improvements should focus on the provided tools, which is what most test users seemed to take issue with.

Revisiting this section after the fact, much of the interface has been overhauled to cater towards a better user experience. Overall, the HoloLens 2 is better at recognising gestures than the previous generation, and has hopefully rectified this. Other issues include functionality of the lamp tool which the users had felt lacked some functionality in how effective it actually was. As will be explained later, in the report, lighting has been approached an entirely different manner which should address the issues had.

There were many suggested fixes within the original MAAP Annotate paper, but acknowledged the severe limitations introduced by the 1st generation HoloLens's lack of onboard memory. It is then the goal of this project to somehow rectify this, either by working to find more efficient methods to the suggested solutions or research means of offsetting the work done by the HoloLens onto a secondary machine.

3.2 Microsoft Mixed Reality Toolkit

Microsoft's Mixed Reality Toolkit[3] is the company's replacement of the original HoloToolKit that was implemented in the first iteration of MAAP Annotate outlined above. The Toolkit, specifically built for Unity is what handles the signal received from the HoloLens and interprets them into meaningful actions. It is the core driver behind all the functionality of the HoloLens and is the basis of the application as a whole. The package itself has multitudes of prefabricated assets, scripts, and components that do not exist in its predecessor. The most appealing aspect of this new toolkit is its shift in focus away from the HoloLens as a proprietary hardware, but instead of one of many that can run mixed reality software. This is an overall boon to the project, even if just from an philosophical standpoint, in that limiting access to software, and specifically an

academic software such as this, based purely on device is keeping those who do not have to an expensive luxury device such as the HoloLens, is to keep them from participating in the discourse. It would therefore be incredibly ironic to develop an application meant to facilitate an open dialogue between peers while passively barring people from the discussion solely based upon there ability to buy a HoloLens 2, which is nearly $4000 USD. Much of this open-source support is facilitated by OpenXR, which will be discussed in the next section.

3.3 OpenXR

OpenXR, not to be confused with Valve's OpenVR, is a royalty-free API developed by Khronos Inc. and has received contributions from AMD, Facebook, Google, Intel, and of course, Microsoft to name a few. The API enables access to a multitude of mixed reality devices by providing the core pose prediction, frame timing, and spatial input functionality to the device as a level of abstraction from what WindowsXR utilised by the HoloLens 1, which communicated with the HoloLens input directly, and thus was coded to interpret input signals as if coming specifically from a HoloLens device. However, this is not to say that cross-platform development was not possible with the HoloToolKit/HoloLens 1 but it would have involved installing each respective devices APIs for which you were aiming to develop for. Given that memory was an issue in the original MAAP Annotate Development for the HoloLens 1, OpenXR offers a lightweight solution to cross-platform development. To use an analogy, if Unity is the engine, the Mixed Reality Toolkit are the axles, wheels, steering column, etc. and OpenXR is the steering wheel and all of the controls available to the end user. There may be slight variations to the transmissions and what is going on under the hood from car to car, but turning the wheel to the left still makes it go left.

From a development standpoint, there is not much to think about in terms of how to implement because to do so would just be to use the Mixed Reality Toolkit and its automatically built-in. However, it must be decide if cross-platform is something that is feasible for this application. There was already concerns posed in the original report for MAAP Annotate, so it is difficult to determine processing and memory capabilities if we abstract from a specific platform. What makes this even more difficult is that, amongst commercial AR devices, the HoloLens 2 is fairly robust. However, when developing software we must take the weakest link in terms of hardware, and so cross-platform support is yet another feature that cannot be implemented until the memory issue is solved.

3.4 Philosophy

Ethics

Since the emergence of (A/V)R, there has been some legitimate concerns as to the ethics of integrating these technologies in everyday life. In an article originally published in the journal *Presence: Teleoperators and Virtual Environments* in 2005[4], Behr, et al, while focused on VR, there concerns pertained to physical and psychological harm such as motion sickness, information overload, and "reentry into the real world." However, in 2005 there were not many widely available headset for the consumer market, and so, without trying to demean the authors work, much of the article reads as hearsay, especially when considering the vast advances in XR technology made since then. Needless to say, the landscape has changed drastically. Despite this, there concerns with the likes of motion sickness and information overload, which in the case of this application

could lead to eyestrain and headaches, is valid. As to motion sickness, this that simply cannot be avoided. It would be nearly impossible to account for what triggers motion sickness across the general population. However, a huge advantage that the HoloLens has by being augmented rather than virtual reality in that it is much of what a user would being seeing is their real environment and the digital object are built in such a way that they behave more like real objects than anything else. As to information overload, this just further cements the need for a well designed UI. Many other publications have come out with concern as to the affect on users sociability after exposure to extended time in VR environments[5], but these critiques do not hold water, given that, one, the HoloLens is used for augmented reality environments, and the whole purpose of this project is to make a collaborative application in AR. Therefore, the "detachment from reality" that people were once so concerned with, while more valid for services like Facebook's Metaverse, is not as applicable in this scenario.

In a paper title, "Ethics Emerging: the Story of Privacy and Security Perceptions in Virtual Reality"[6] from USENIX, or The Advanced Computing Systems Association, has concerns geared more towards security in regards to information, which is discussed more in depth in the *Data Consideration* section, as well as harassment in a virtual or augmented environment. What the authors assert is that while most developers handle data security, they fail to consider how to police behaviour in a virtual environment, where there may not be any third party witnesses. This concern is the most legitimate in relative to this project specifically, since, as mentioned before, it the prime directive of this project to increase interactions between users of AR systems. The difficulty, however, and it is even a concern in the first place, is that there obviously no cut-and-dry way of keeping people safe. Asking users' to self police is not a legitimate remedy to the problem because the people who would adhere request probably not the perpetrators of these sorts of actions to begin with. However, limiting access based off a presumption of potential misbehaviour not only raises much more ethical concerns in regards to profiling, but also runs antithetical to this application's purpose. I preliminary solution may be to not have connections with other users automatic but an opt-in invitation based system, not unlike how people currently conduct private-public events. One can send invitations to trusted individual to attend their birthday party, but technically everyone is allowed to go to Chuck E. Cheese, but not everyone is allowed to have the pizza they bought for their friends.

Design and Collaboration

The focus of this project is to expand the current iteration of MAAP Annotation, which is still primarily a tool for the individual, truly collaborative. It would be easy enough to achieve this by making the notation files exportable, allowing for anyone with the program to be able to load the files and being able to use services like Github to share as we would with any code. However, other than being inconvenient, the simple plan does not necessarily meet the motivation for why MAAP annotation needs to be collaborative. The philosophy as to why the annotation system would benefit from a social media type system is discussed in the paper: "Collaborative research in art, design and new media - challenges and opportunities."[7] The primary focus of the paper was to explore how the interaction between STEAM disciplines benefit each discipline in their own right, with one of the examples being how patients in a welsh hospital diagnosed with depression benefited from "dream films" made by local artists.

While none of the research discussed in the paper is directly related to any technique that will be employed within the project, it provides inspiration and justification for an open source system. It would be simple enough to build software that was meant to be shared internally amongst other researchers. Many problems that arise otherwise such as content moderation, due to the built in professionalism and lack of anonymity among peers, could be avoided. However, this limits the collaboration to people whom the original researcher has considered. By making it open-source, we can gather insight from unexpected disciplines. For example, we can intuitively reason that archaeologists, anthropologists, and sociologists may have insight on megalithic art. However, by

nature of the medium, someone with a background in engineering may have useful insight about a piece of art that may not be anticipated. Therefore, the immediate obstacles created by making a public system work, in the long run may be worth it given the perceived benefits. I hope to address potential solutions to the aforementioned problems next.

Chapter 4: Data Considerations

While this project does not collect or use private data, it does connect to either the default database or one privately owned by the user and should therefore be made aware that the app will pull and write entries to the linked Azure Blob Storage Database. The app does not create new tables, nor is it allowed to make any alterations to file systems in both the cloud or local directory other than to write to them. However, the application does need permission to gain access to the Azure cloud which can be given through a private key found in online Azure portal. While the app does not share this access key, nor store it outside the session, it is the input of the key from the *New Database* button that poses the greatest security risk

The data stream by which the PC Remote App sends information to the headset is stated to be unsecured according to the MRTK documentation and should not be used over as network not trusted by the user. This should not be an issue since all of the interactions between application and database will be done from the PC running it and Microsoft's servers, which is protected. The user will be most vulnerable when entering the connection key to link an alternative database. Once when the key input information from the HoloLens 2 when using the built in virtual keyboard is transmitted to the PC, and again when the frame containing the visual representation of the previously input characters. However, in reality more complicated than it would seem, and this is mainly due to how PC Remoting works. The only information being exchanged by the HoloLens and the PC are the positions of the inputs and their pointers in the applications "world space" from the headset and the visual representation of the current state of the application from the PC in the form of a frame (essentially a picture). Therefore, when the user is giving input to the virtual keyboard, it is merely sending it's relative position to the headset, and sending this back to the PC, combines this with its own understanding of the current state of the program and handles interactions with this in consideration. The only way this data is useful to a malicious individual would be if they knew the exact position of the virtual keyboard, allowing them to make the same calculations that the users' computer is doing to handle interactions and collisions. However, the PC is never transmitting this data. The actual vulnerability is the fact that it is just streaming image data, not dissimilar to a service like Twitch or YouTube, and so anyone with the right codec can intercept and decode the image data. While it is always recommended, in general, to conduct sensitive transactions on the internet over a secured network, but obviously with the intention of using this software at universities, this cannot always be guaranteed. Currently, this system relies on timing and obscurity for safety, meaning that the window of vulnerability is not only brief, but also the likelihood of someone wishing to interfere in a Remote PC MRTK application is incredibly slim. This is obviously barely passable justification as the application exists currently, so this will have to be rectified in the future. If the concern still exists among users currently, rebuilding the application to run natively on the HoloLens is just a matter of changing a few settings in Unity. The implemented features should still work if this is the case.

Chapter 5: Outline of Approach

5.1 Setup and Initial Work

The initial work undertaken was to learn the Unity Environment along side the HoloLens Hardware and Software Development Kit (SDK). This included downloading and installing the most recent versions of Unity, Microsoft Studio, and the HoloLens Emulator, which allowed for a streamlined development process, negating the need to constantly build and deploy to the physical hardware. However, it is at this point when it was discovered that as of 2019 [8], the original HoloLens toolkit, HoloToolKit for Unity, has since been made obsolete with the Mixed Reality Toolkit (MRTK). Furthermore, Microsoft no longer recommends using the Windows Mixed Reality Plug-in, Windows XR, for Unity Versions past 2019, with long term support for the device itself set to end in mid-2022.[9] This is due to Microsft's apparent shift toward an software focused, open source development strategy with the integration of MRTK and OpenXR, a royalty-free access standard designed for cross platform support. It was then necessary to decide whether it was more important to follow the initial plan set out for this project, or ensure that the work conducted would have longevity passed last year, and it was decided on the latter. Despite this initial setback, the HoloLens 2 and MRTK offer a larger breadth of methods for human-computer interaction and ancillary technologies that would facilitate ease of use and quality of life choices.

The technical specifications for the new port of the MAAP Annotate Software, alongside the original components are as follows:

5.2 Porting

There are many ways in which MRTK is based on the HoloToolKit and where obvious inspiration was drawn, however it was important, and so remains, to make the distinction between the former being a replacement of the latter, rather than it's direct progeny. What is meant by this is that, while MRTK does expand on functionality that was lacking in the HoloToolKit, there are also instances where MRTK either completely overhauled the original functionality of, what would appear to the uninitiated, identical or similar libraries, or completely do away with integral parts of the original Toolkit.

Much of these differences centred around the vastly improved input system of the MRTK over the HTK, as well as the MRTK's approach to interacting with digital 3-dimensional space in general. Firstly, where the original HoloLens, and by extension the HoloToolKit, relied primarily on hand gestures and a proprietary remote, the MRTK has the built in capability of hand, eye, and gesture tracking, in addition to traditional methods of mouse and keyboard, and controller. With the diverse input methods, MRTK utilises a pointer-based[10] input management system that is inherited by the different methods, offering a level of abstraction that allows for a centralised, uniform method of input handling that is irrespective of the specifications of any one method. In essence, the MRTK pointer functions as a persistent ray cast that originates from the input device, allowing it, whether it be hand, eye, or controller, to act similar to a cursor in 2-dimensional space, extrapolated to a third dimension. This differs from the HTK which relies on specific tailoring

by the developer on how to interpret the changes in state of the input to perform designated tasks. Additionally, with the implementation of the OpenXR environment over WindowsXR, and the abstraction from hardware specific inputs, MRTK is not tethered to any particular platform. This allows for the ported MAAP Annotate to be deployed on a number of virtual and augmented reality devices that support OpenXR.

The following outlines the specific changes and additional features to the tools originally found in the first iteration of the MAAP Annotate software:

5.2.1 System and Interface Design

How PC Remote Applications Work and Their Benefits

The most significant change from an architectural point standpoint is the change to a PC Remote Application, enabled by the MRTK.[11] These types of applications are more common in VR games that require more powerful resources than are available on any given headset. The original MAAP Annotate, and most HoloLens application prior were required to be built and deployed for and onto the HoloLens device. By making it a PC remote applications, it is ran from a PC and streamed to an augmented reality device on the users' network. Much of the technical description as to hoe this system works has been covered in the *Data Consideration* section, so the following primarily serves to explain the setup and why the decision was made to make such a change in the first place.

In order to run a PC Remote Application, users are required to have a computer capable of running the application and an AR Headset compatible with the application. Currently, this iteration of the application has only been tested for machines running Windows 11 using a HoloLens 2 emulator, although applications built on Windows 10 or 11 can be run on the other operating System. Additionally, while this was initially built and tested a Windows Machine, it should be compatible on MacOS and Linux machines, as per the Unity Build settings. From the HoloLens 2, or its emulator, the Holographic Remoting app needs to be installed on the device which is available from the Microsoft Store, which is a preinstalled application on the headset. Upon running the Holographic Remoting app, the user will see a screen with the IP address of the headset or emulator. Then from the running desktop application, a text field can be seen located in the upper left cornernext to a button that reads connect. Entering the IP address from the Remoting app into this text field will and, upon a successful connection, will begin streaming data from the PC to the HoloLens.

Primarily, this serves to make the application more mutable in regards to the operating system of the computer running the application and the hardware to which the application is being streamed. A secondary benefit that does not service the software in its current state, but since the computational and graphical processing is performed by a machine that will most likely be more powerful than any A/VR headset available, as the application becomes more robust in terms of features and capability, it will not be limited by the power of the viewing device. However, rather than speculative functions of a remote application, the iteration of MAAP Annotate as outlined in this paper, in addition to offsetting the processing power to a secondary device, the application also uses PC Remoting as a work around to the cumbersome method of storing local files on the HoloLens. On an application native to the headset, files would normally store on the device and the user would have to manually move the files from the device to a computer. Through Remoting, since the application is already running on a PC, the saved file will be stored directly onto the computer, circumventing the need for a file transfer. Alternatively, given the source code, it is still possible to build the application to run natively on the HoloLens 2 with small changes to the build settings. Instructions for which can be found in the MRTK Documentation.

Interface Design and Navigation

Porting the application from the first generation HoloLens allowed for a complete overhaul of the user interface while still being recognisable from the previous iteration. TO CHANGE: Thus features are accessed through navigation menus that are dynamically activated as needed.

Upon loading the application, users are met with the stone to be analysed, a pinch slider to control a virtual overhead lamp, and a three button menu containing toggle controls to enable note creation or object manipulation, with the third button opening an options menu. The *Annotation* button opens a sub-menu that appears above the stone with toggle buttons which enable text-based annotations, a pen tool, and square selections. The *Options* sub-menu contains the features to save the current annotations, load a previous annotation, load a new stone (unimplemented), or link a new database. Each of these functions are either new to the MAAPAnnotate System or are now implemented from the previous iteration. The function of the aforementioned features will be further described later in the report. While this new menu system adds additional layers a user must navigate in order to enable key functionality, the central motivation was to reduce the amount objects cluttering the users' field of view. Mixed reality already being a new environment for most people, generation of a flood of menus and tools upon load could overwhelm new users. Additionally, with many of the menus being manipulable, having them be persistent throughout regardless of the current tool selection would interfere with said tool given that the same input pointer would be used for both.

In future iterations, the user could instead be met with a preliminary setup menu on load that will allow them to choose the stone and annotations. This would be a feature similar to that found in word processing software that can store cache data to restore previous session. However the application currently only contains an example Megalith for the purpose of demonstrating features of the application. This is for the same reason the *New Stone* button has yet to receive functionality.

5.2.2 Tool Update and Design

The Mixed Reality Toolkit offers many additional, intuitive functionality that the HoloToolKit simply lacked. By examining the tools provided by the MRTK out of the box, it becomes apparent that the original iteration of MAAP Annotate was inhibited by the functionality of the first generation HoloLens and the HoloToolKit.

From this point, the changes made during the port will be referenced relative to the AR Toolkits since the with OpenXR, MRTK is no longer specific to the HoloLens 2 hardware and the differences have less to do with the devices than the logic that powered them. It must also be stressed that the improvements enabled by the MRTK environment and the comments in reference to this are not meant to disparage the development team of the original MAAP Annotate, but rather to demonstrate the quality-of-life improvements that the MRTK has facilitated since its introduction.

Deprecated Tools

A. Flashlight - In the original application, the flashlight necessary as there was not a persistent light within the scene. While this was the most optimal solution for the system at the time, it was not usable in conjunction with other features since the lamp was only functional while holding the flashlight and so a user would have to constantly hold the lamp in one hand in order to gain the visibility it offered. By having a persistent lamp source in the world space, illuminating the stone can become an after thought, while also having the ability to adjust the intensity of the light source as the user would see fit.

B. Projection - The decision to deprecate this tool was derived from its competition with the new option given to the Pen. While the functionality is still useful, since users can now adjust the width of the marking the pen makes, it can now perform the same actions as the projection but now with the flexibility of making shapes free-hand. Therefore, given the option of simply widening the pen tool or switching back and forth between the pen and the projection, would be easier to simply adjust the width of the pen tool.

Manipulation

Where much of the original manipulation functionality had to be handled individually via menu buttons, the Object Manipulation script provided by the MRTK allows for a more tactile user experience that functions more similar to how a user would interact with an object outside of augmented reality or in familiar preexisting digital environments. The functionality of Manipulation in 3-dimensional space is still relatively similar as it was in the original implementation of the annotation software from the perspective of the user, but can now be done directly to the current megalith, offering a more tactile experience that is more analogous to viewing a real stone.

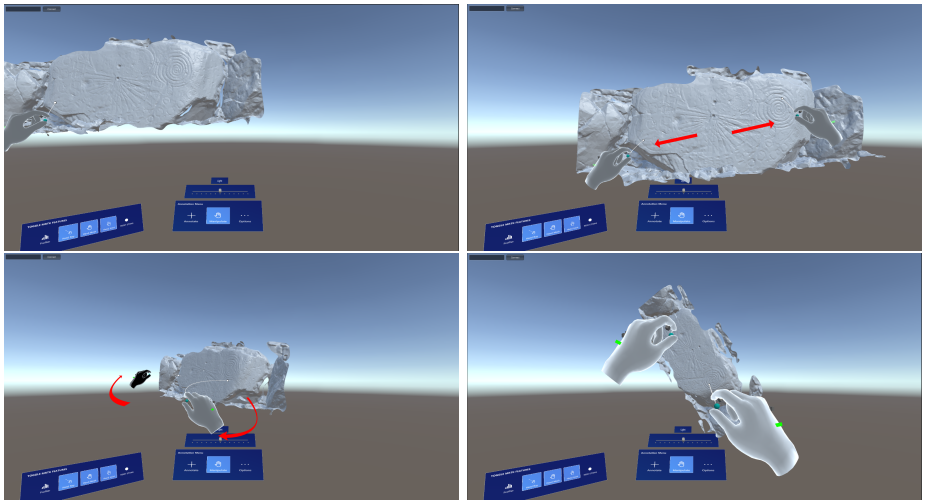


Figure 5.1: *Manipulation Methods (left to right, top to bottom): Movement, Resizing, and Rotation along the Y and Z-axes*

Scale and rotation are handled with the introduction of the users second hand. By grabbing the stone and moving the hands closer or further apart, the stone will scale to match the grab points relative to the input devices (hands). This is a similar gesture to zooming in and out on a smartphone, a gesture most users will already be familiar with. Rotation is handled in much the same way. When the user grabs the stones with both hands, the script will register the distance between the grabbed object and the input devices. Then when the input devices, in this instance, hands, move throughout 3D space, the distance between the object and input will remain constant, thus rotating the object. The built-in rotation from MRTK allows for the object to be rotated along all three axes rather than solely the vertical Y-axis that is seen in the original application.

From a technical aspect, MRTK handles manipulation by casting a ray that originates from the hand(s) upon detection by the HoloLens in the forward direction of the user in a process aptly named Raycasting. In short, a raycast can be thought of as a beam constantly being emitted, or cast, from the input source for some predetermined length. Then when an object is placed between the ray's origin and its maximum range, the "beam" is recognises the shorter travel distance and registers a hit, which contains information about the ray, such as its length as well as the object it collided with, such as its position in space and even the type of object. MRTK uses the information given by these raycasts to perform manipulations, and in fact, all other tools provided. As briefly mentioned prior, on a collision with the object to be manipulated, all information regarding said

object's spatial information is now all relative to the ray. So when the ray's origin moves, so will the ray, and in turn, the object itself.

Annotation

There are three methods of annotation available to the user in MAAP Annotate: Marking, Projection, and Text. The inherent function of these methods was changed minimally during the port, however they are features that many users would come to expect in similar drawing applications. The first is the provision of pinch sliders that allow for users to change the colour, size, and opacity to the Marker. Since MAAP Annotate cannot yet link textual annotations to specific markings made by the user in the stone, colour coordination provides the ability to distinguish markings that can then be referenced in text. In future iteration, the selection tool can be implemented to select annotations on the stone to edit or link them to the text. Users now also have access to Undo, Redo, and Clear functions for the Marker, functioning similarly to any existing digital drawing tool on the market, such as MS Paint.

The textual annotations received minimal improvement, simply creating what is essentially a virtual notebook that the user can use to separate their notes into pages and title them. Navigation of the notebook can be done through a button menu associated with the annotation panel in the 3D environment. This includes flipping through the notebook page by page, and having the ability to remove the page currently being viewed. Text annotations can also be searched via the name title of the annotation or its number. A major advantage that the HoloLens 2 provides users' in terms of "quality of life" features is that voice-to-text is integrated into the virtual keyboard, eradicating the need to cumbrously type letter by letter. In future iterations, it would be possible to search for annotations through key phrases contained within the text or tags, however this is not implemented in the version covered by this report.

5.3 Collaborative Annotation

The original intent of the work conducted during this project was to improve the collaborative capabilities of the MAAP Annotate software. While this was initially envisioned to take the form of an augmented reality social media adjacent platform, existing third party support and interconnectivity is limited in the current version of the MRTK. The most applicable SDK available was privately developed for. The decision was ultimately made against the implementing these libraries due to the lack of credibility in terms of security and official support by either Microsoft or Unity. While this project does not use private or sensitive data itself, the act of saving and loading annotation data, both locally and remotely, involves accessing private storage owned by the user. It therefore would have been irresponsible to potentially open a back-end into user's private information via an unverified third party SDK. However, this has not limited the capability of the project in its functionality, merely limited the available cloud services to those owned by Microsoft, namely Azure. Additionally, Azure has other services outside of storage that will be beneficial to the application in the future and so the integration will be easier to implement.

5.3.1 Saving and Loading

The *Save* and *Load* actions involve the current scene data into an array of binary data that can be then uploaded to Azure Blob Storage, the details of which will be discussed in the next section, and stored locally in binary *.dat* files.

Saving

Upon pressing the save button, located in the options sub-menu, the user will be greeted with a text box to enter the prospective name of the session data they are attempting to save. After submitting the name, the Annotation Controller, which is connected directly to the cloud storage, will search the storage tables for an entry by this name. If there already exists an annotation within the database, the user will be notified, but will not be disallowed from overwrite the entry. Regardless of whether or not there is a preexisting annotation of the same name, the users will be prompted with a new window where they can edit or insert the description field of the Azure Blob Storage table. The user then has the option of completing the save/upload process or aborting the save through the corresponding buttons. If the user decides to desert the save, they will be prompted with a third window that requests confirmation for the action.

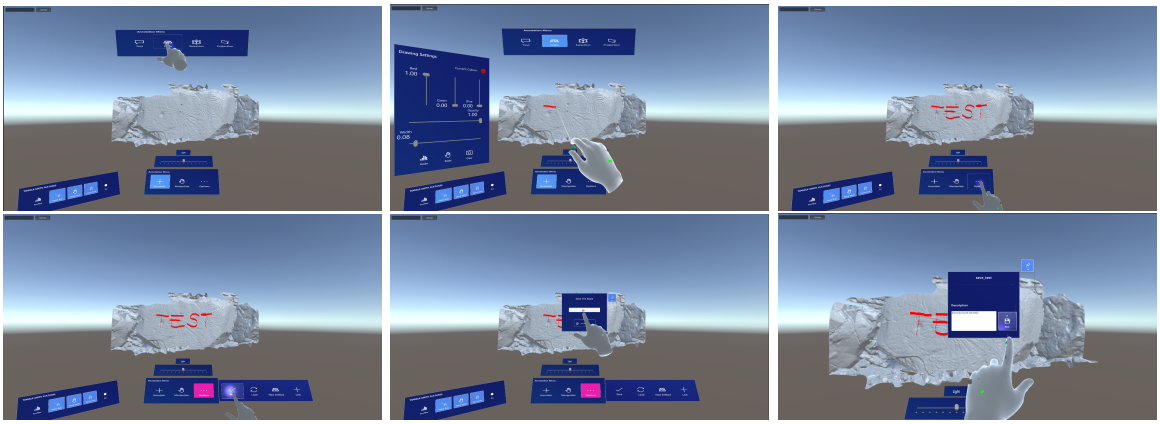


Figure 5.2: *Creating and Saving a Drawn Annotation*

Unity 3D objects are not natively serialisable, despite being comprised solely of serialisable elements, and so the first step in the reading and writing to a binary file, the object attributes must first be converted to custom object that have the *System.Serializable* designation given to the new object. However, a quirk of the *BinaryFormatter* class is that, for reasons that are yet to be revealed, cannot serialise objects that inherit characteristics from a super or abstract class. Therefore, eschewing any knowledge gained concerning the best practices for object oriented programming, all annotation types (manipulation of the artifact, text annotations, and custom markings) must all be handled by one class. This new object class, aptly called *SerialisedObject*, stores all possible data, from the name of the game object to each vertex in a line, if applicable. The object also contains a flag field that denotes which type of annotation the serialised object originated from. This will help decode the saved data later.

From there, the *Annotation Controller* iterates through every annotation, converting them to *SerialisedObject*s and adding it to a list of serialisable objects and is used as input to *BinaryFormatter.Serialize()*, which converts the list of objects to formatted binary data. The Binary Formatter, for one converts the list objects to binary as implied, but also contains a header that keeps track of the original structure of the object list, but also the structure and data fields of the object that constitutes the list type. This header is partially responsible for why inherited objects cannot be serialised, since the header would then have to store the inheritance data, which it does not permit, either by design or allocated space in the header. Nevertheless, the converted data stream is simultaneously written into a local file as well as uploaded to the blob storage. As will be explained in depth in a later section, as was learned through experimentation during the development process, it is important that the entire file stream is uploaded to the cloud storage, since the file is not actually stored in the database. What is meant by this, is that services like Google Drive will take a file upload, read it, and store all the data, header included, from the file. This would also include information on how to read the file header by the file extension. Blob storage works differently in that it leaves the responsibility reading the file data to the script that

is connecting to it. Standard file IO methods do not include the header info when parsing file contents, and this is why saving to the file and cloud occur in parallel rather than in series.

Loading

The cloud support has enable the implementation of a prospective feature from the previous iteration of MAAP Annotate. Users are now able to load annotations from previous sessions made by anyone who has stored on in the connected Azure Blob Storage table. In the same options menu, from which users' can save their annotations, is the *load* button. Upon activation, the Azure storage is queried for the names of the annotation, which serve as the Row Keys within the tables, and generates a dynamic list of buttons which, when pressed pulls the annotation blob data. This data comes in the form of a stream of binary data and is fed to a custom controller that interprets it as Unity 3D objects which comprise the annotations.

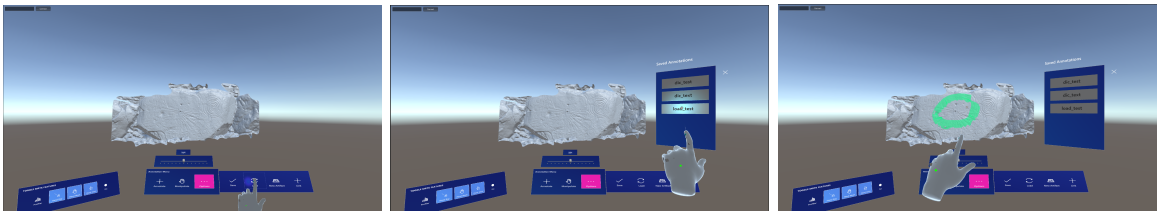


Figure 5.3: *Steps to Load*

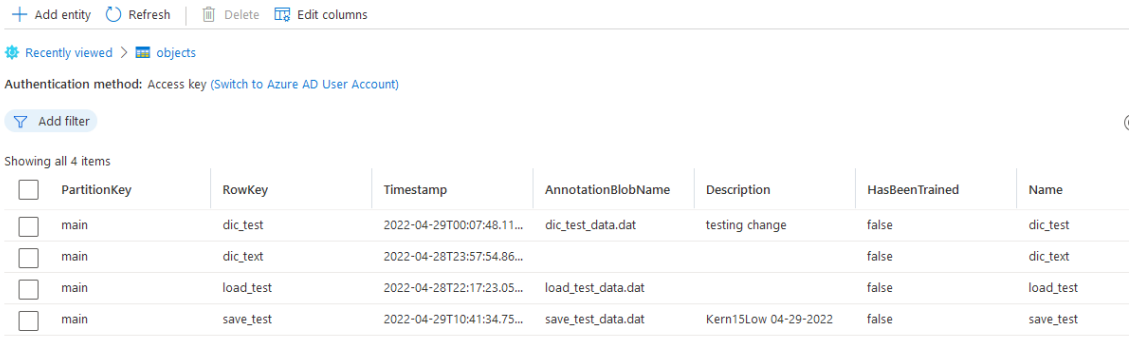
The controller then creates a list of objects of the *SerialisedObject* class from the byte array as dictated by the header. The list is then iterated object by object, using the aforementioned type flag from the *SerialisedObject* class to dictate what these what type of annotation each object represents.

5.3.2 Cloud Support

The *save* and *load* functions heavily utilise Azure Blob storage for managing persistent data across sessions. Since the *save* function merely converts the session state to a binary data stream/file, it could be stored in any commercially available storage without any change. Currently, the only inhibition as to the users' choice of cloud storage service is the lack of official support for third parties. Regardless, Azure Blob storage is perfectly suitable for the what MAAP Annotate needs from a cloud service.

Azure Blob Storage is a table-centric database service that allows for the storage of unstructured data. It is so unstructured to where file types are not required to store file data. The documentation nearly recommends that the blob key contains an extension, simply for the developers benefit. Similar to non-SQL databases, it does not limit table entries by a predetermined size or data type. It is able to do so by storing all data as segmented byte arrays. The term "Blob" (or **B**inary **L**arge **OB**ject) storage is derived from this structure, because it is analogous to amorphous binary blobs. It functions similarly to the page system of digital memory systems but on a user visible level. Blob Storage has a primary table that logs the *Row Key* (name), *Time Stamp*, an optional *Description*, and a pointer to an entry on a separate'that actually contains the blobs, in the form of a string called the *Annotation Blob Name* (obviously the column title is not standard). This type of storage allows for incredible flexibility and offers the potential to implement more annotation types in the future, such as images. While there are other major services that offer similar blob storage services such as Amazon's *Simple Storage Service (S3)* that would be just as suitable, using Microsoft's Azure service facilitates more direct compatibility with other Azure services that would useful to implement in the future such as Azure *Spatial Anchors*[12] which allow multiple people to view

the same 3D Object simultaneously by keeping track of each person's spatial coordinates relative to the central object and constructs an identical virtual environment for member. While having the capability to do so, this service has not been implemented in the current version of MAAP Annotate due to the inability to test the spatial anchors in this environment.



PartitionKey	RowKey	Timestamp	AnnotationBlobName	Description	HasBeenTrained	Name
main	dic_test	2022-04-29T00:07:48.11...	dic_test_data.dat	testing change	false	dic_test
main	dic_text	2022-04-28T23:57:54.86...			false	dic_text
main	load_test	2022-04-28T22:17:23.05...	load_test_data.dat		false	load_test
main	save_test	2022-04-29T10:41:34.75...	save_test_data.dat	Kern15Low 04-29-2022	false	save_test

Figure 5.4: Blob Table Example

Chapter 6: Project Workplan

6.1 Workplan

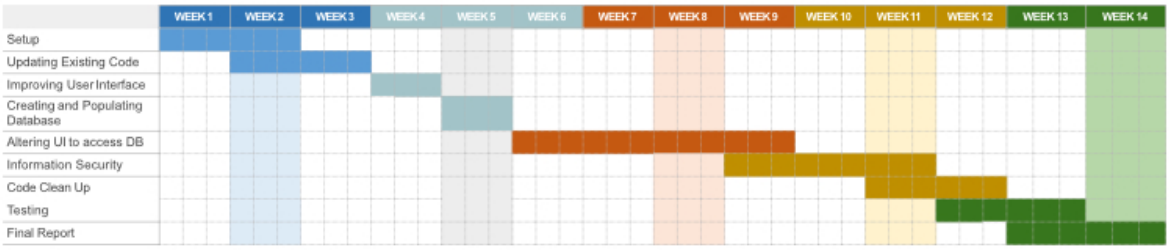


Figure 6.1: Project Timeline

The preceding Gantt chart outlines the intended project timeline. The order outlined above is as follows:

- Setup
- Updating Existing Code
- Improving User Interface
- Creating and Populating Database
- Altering UI to Access DB
- Information Security
- Code Clean Up
- Testing
- Final Report

6.1.1 Setup & Updating Existing Code

Initial Plan

This part is primarily to install necessary software and drivers, as well as gaining access to various services that may be needed such as AWS cloud services. Three weeks have been allotted to these tasks because this project is a continuation of work from previous projects, and so the existing code may be using older versions of software, such as Unity, and have allocated such time to update code as necessary, and replace any depreciated libraries that may or may not exist.

After Implementation

Obviously, the need to port the application to an entirely new software and hardware environment changed these plans significantly. Originally, this was only intended to take around two or three weeks. However, since the original conception of the MAAP Annotation application in 2017, Microsoft has released the HoloLens 2, and created a new SDK called the Microsoft Mixed Reality Toolkit in 2019. Since then, the HoloLens and its SDK, the HoloToolKit, has since been made obsolete, and as of 2022, is no longer supported. Therefore, what was assumed to be a simple version update became a full platform port from the HoloLens to its successor. Additionally, since these plans were made before in depth development, it was not yet known that neither HoloToolKit, nor the Mixed Reality Toolkit, have proprietary AWS support. As previously mentioned, this motivated the decision to switch to Azure services, although this ended up not having an effect on development since they offer much of the same functionality. Both services also offer a free trial to new accounts.

6.1.2 Improving User Interface

Initial Plan

To improve the UI is not to imply that the existing version of the project is lacking anything. Rather, what is meant by this is that some steps will need to be taken to optimise the interface for online use. This may include creating classes and assets to access annotations from a data base, annotation management tabs, etc.

At this time, only a week has been allocated to this task because a bulk of the work will work in conjunction with the database, and so much of the work done in this step is to preempt what may be needed on the user end to interact with the DB.

After Implementation

Due to the port, this step ended up being inherent in that process, so no actual time was specifically allotted in order to complete this.

6.1.3 Creating and Populating Database

Initial Plan

For this, AWS DynamoDB will most likely be used for this since there is an available license through UCD and will be the most compatible with other AWS services anticipated to be used. Since the system that will be used for the database and the data itself already exists, the work required in this step is to simply create the DB with the necessary partitions and tables.

This section of the project is slightly different, in that this step is relatively atemporal compared to the steps up to this point. Meaning that this in theory can be done right now with no impact to how the other steps are conducted. However, it is placed this late in the Gantt chart because this is the latest this work can be done since the next step is allowing users to query information from the database from the UI and is the bulk of the work.

After Implementation

As mentioned before the switch was made to Microsoft Azure fairly early in development. Like AWS, Microsoft offers free service to students.

Setting up the database took much less time than anticipated since the structure of blob storage is generally already predetermined. Therefore, the database design was not an issue and it was simply a matter of initialising the blob storage and entering the connection key to the database controller script.

6.1.4 Altering the UI to Access the Database

Initial Plan

In this step is where the full functionality of this project comes from. Combined with the initial work done in the UI from two parts prior, the total time allotted will be five weeks. In this five week period, time will be taken for planning, analysis, design, implementation, testing, and revision.

What this involves is to create tools that are easy to access and use in the HoloLens environment. What this means is to ensure that users will be able to navigate the environment as they would with any social media but operating with the knowledge that an augmented reality environment does not necessarily translate design-wise from a desktop experience.

After Implementation

During development, this step remained, for the most part, unchanged. Altogether, this involved creating the 3D object that users' will interact with in order to save and load annotation, implementing the MRTK classes that enabled Azure capabilities, and the scripts used to interpret binary data.

The only significant changes made, as mentioned prior, was the approach to how users interact with each other's annotation data. It now serves as more of a shared drive than a social media, which has its benefits and hindrances that will be discussed later, but it does forgo the need for immediate content moderation since annotations will mainly be created and shared by individual with at least a working relationship.

6.1.5 Information Security

Initial Plan

Making a collaborative annotation means collecting even the smallest amount of info, even purely for the sake of tracking contributions and being able to differentiate from the open-source and academic users. Creating a comprehensive digital security system would be a project in and of itself, so the one to be used will be a pre-built service. An initial example of this would be to allow users to log in through their e-mail client such as Google. This way, the only information taken by the application is that which is given permission by the email client and the liability for privacy lies else-ware. Therefore a majority of the time taken in this step will be research into the most compatible system and the implementation of the decided upon service.

After Implementation

Much of what was considered and implemented in regards to user was discussed in a previous section. However, since the application no longer has user login or registration, personal information is no longer being used in the way it was in the original plan. The services implemented in this application handle security on their end and so this step was not as involved as previously thought. The rest of the development went on as planned

6.1.6 Code Clean Up

This is the time allotted to clean up any unused methods, combine or separate sections of code where applicable and clarifying any areas that may be used in the future.

6.1.7 Testing

This is the time when the application is examined for bugs and any feature improvements that may need to be considered. These improvements will not be any major performance updates, unless otherwise broken, as much of the primary functionality should have been tested throughout.

6.1.8 Final Report

Notes and Data will be compiled into the final report at this time. However, if done responsibly, the report will be added to and updated regularly throughout the entire semester.

Chapter 7: Summary and Conclusions

At this stage in the development of MAAP Annotate, considerable additions have been made with the goal of one day making this tool an invaluable asset to academic research.

Porting to the Mixed Reality Toolkit and HoloLens 2

While the result of the work conducted is not as what was originally intended, it was none the less necessary. The decision to port the software to new devices is vital to the longevity of a program, especially when it involves cutting edge technology, as augmented reality is, that is constantly being revised, updated and improved.

PC Remoting

Additionally, the port also allowed for the compatibility with services and platforms that would have taken a considerable amount of work in the HoloToolKit, but are inherent to the Mixed Reality. This of course includes, but is not limited to the switch to a streamed remote application. This in conjunction with the OpenXR platform lowers the cost barrier to using the application. While, the HoloLens, both its first and second generations, is a great asset to this software, and perhaps the best in the market for what it is geared toward, it is priced as such. So the first step into making MAAP Annotate an open source platform, was to lower the barrirer of entry to run it in the first place.

UI Redesign

Porting ended up being a boon to the project, especially when it came down to redesigning the User Interface. In 2017, when this was project software was first designed, augment reality as a commercially available tool had only been out for a year or two. Therefore, best practices when developing an augmented 3D environment had not been established. Since then, the MRTK has been released with a careful consideration as to how people interact with a virtual space and were implemented likewise.

Cloud Storage

Despite the shift in focus away from a social media type platform, the work that was not unfruitful in terms of making a collaborative platform. Users' are now able to create and store persistent data that can be shared with others. While not the service originally intended to be implemented, the Azure Service and readily available SDK made connectivity more easily achievable with room to grow.

Final Thoughts and Considerations for Future Work

It was the intention of this project to expand the functionality of the preexisting MAAP Annotate software. Although not executed as originally envisioned, MAAP Annotate is farther down the path to being a fully fledged digital working environment that attempts to virtualise collaborative discovery that would otherwise mandate in person interaction. In hindsight, the expectation to

create a fully functioning social media platform centered around archaeological analysis with content moderation that has yet to be comprehensively implemented in exiting services was perhaps fuelled somewhat by overzealous ambition.

However, having now gained in-depth experience of working with mixed reality technology for an extended amount of time, this intention might be the wrong direction. With the use of services available to mixed reality platforms, it is more fitting to create what is essentially a mixed reality work space that functions no differently from any other physical room, but is not bound to visual location. Through the use of technology like Azure Spatial Anchors, digital objects can exist in the physical world, with shared attributes across multiple devices, without being specific to a real world location. What at one point seemed like complete science fiction, such as *The Matrix* or *Ready Player One*, are already being implemented by companies like Facebook with the Metaverse. While not being completely virtual like the examples give, taking this application in this direction could potentially be more valuable than a social media-like approach (The irony of shifting focus from one Facebook product to another one is not lost). While social media does enable the sharing of work and ideas, as many have learned, can be an isolated experience. On the other hand, creating a virtual "cyberspace" geared towards academic collaboration is better analogue to how this type of work is conducted currently.

Chapter 8: Acknowledgements

I would like to take this section to thank Dr. Anthony Ventresque for his assistance and advice during the development of this project, as well as the original development team for providing the foundation of this work.

Bibliography

1. Ventresque, A., Keane, M., O'Sullivan, M. & Normand, J.-M. https://maap.ucd.ie/.
2. Barbier, J. e. a. *MAAP Annotate: When Archaeology meets Augmented Reality for Annotation of Megalithic Art* in *VSMM 2017 - 23rd International Conference on Virtual Systems and Multimedia* (Dublin, Ireland, Oct. 2017), 1–8. https://hal.inria.fr/hal-01626057.
3. Polar-Kev. *MRTK-Unity Developer Documentation - Mixed Reality Toolkit* https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/?view=mrtkunity-2021-05.
4. Behr, K.-M., Nosper, A., Klimmt, C. & Hartmann, T. Some Practical Considerations of Ethical Issues in VR Research. *Presence: Teleoperators and Virtual Environments* **14,** 668–676. eprint: https://direct.mit.edu/pvar/article-pdf/14/6/668/1624365/105474605775196535.pdf. https://doi.org/10.1162/105474605775196535 (Dec. 2005).
5. Kenwright, B. Virtual Reality: Ethical Challenges and Dangers [Opinion]. *IEEE Technology and Society Magazine* **37,** 20–25 (2018).
6. Adams, D. *et al. Ethics Emerging: the Story of Privacy and Security Perceptions in Virtual Reality* in *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)* (USENIX Association, Baltimore, MD, Aug. 2018), 427–442. ISBN: 978-1-939133-10-6. https://www.usenix.org/conference/soups2018/presentation/adams.
7. Liggett, S., Earnshaw, R. A., Thompson, E., Excell, P. S. & Heald, K. *Collaborative research in art, design and new media - challenges and opportunities* in *2015 Internet Technologies and Applications (ITA)* (2015), 503–508.
8. qianw211. *Choosing a unity version and XR plugin - mixed reality* https://docs.microsoft.com/en-us/windows/mixed-reality/develop/unity/choosing-unity-version.
9. Evmill. *Hololens 1st (GEN) release notes* https://docs.microsoft.com/en-us/hololens/hololens1-release-notes.
10. CDiaz-MS. *Controllers, pointers, and Focus - Mixed Reality Toolkit* https://docs.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/architecture/controllers-pointers-and-focus?view=mrtkunity-2021-05.
11. Hferrone. *Holographic remoting overview - mixed reality* https://docs.microsoft.com/en-us/windows/mixed-reality/develop/native/holographic-remoting-overview.
12. Thetuvix. *Shared experiences in mixed reality - mixed reality* https://docs.microsoft.com/en-us/windows/mixed-reality/design/shared-experiences-in-mixed-reality.