UAANAV: A HoloLens 2 Guided Tour Experience

Final Project Report

Ashton Curry & Rane Murphy

05/02/2024

Table of Contents

Abstract

This project focused on creating a navigational application for the Microsoft HoloLens 2 mixed reality headset. By integrating augmented reality, users can interact with a blend of holograms and the real world. Designed primarily for new or prospective students and staff, the application serves as an interactive guide to the University of Alaska Anchorage (UAA) campus and a promotional use case for the Alaska Data Science and Artificial Intelligence Lab at UAA. It offers a dynamic and educational method to discover the campus while providing insights into the history and culture of UAA's notable points of interest. The holograms form a trail, resembling breadcrumbs, with interactive elements such as photos and games. Additionally, the application includes a map creation mode, enabling future developers to enhance the tour experience by designing their own maps and adding new points of interest.

1. Introduction

Wayfinding is used in our daily lives to get from one place to another. Applications like google maps allows people to plot routes among many roads and trails. These apps are highly effective in getting people where they need to go, oftentimes finding the most efficient and shortest route. However, these applications rely on typical GPS positioning for localization which is poorly adapted for indoor use due to the complexity of indoor spaces combined with weaker GPS signaling while indoors[1]. Therefore, a new approach must be taken to effectively guide people in indoor spaces.

There are a few other problems that are proposed by the clients that this software was built for. The University of Alaska Anchorage (UAA) and the Alaska Data Science and Artificial Intelligence Lab (ADSAIL) are beneficiaries for the software that was built. ADSAIL is a small lab on the UAA campus that many students do not know exist. The software created is aimed at promoting the lab and its potential for others to develop and create using the technologies the lab gives access to. Additionally, the software targets UAA to provide campus tours or guidance around the campus to new or prospective students and other visitors.

The ADSAIL lab contains three Augmented Reality devices developed by Microsoft called the HoloLens 2. Previous research has shown that the HoloLens 2 can be an effective device to develop applications for wayfinding [2]. The features of the HoloLens 2 includes sensors and spatial mapping abilities that aid in developing wayfinding applications. An existing service from Azure called Azure Spatial Anchors (ASA) can be used to create virtual markers, where game objects such as a navigation node, pictures, and even games can be placed and interacted with on the referenced ASA.

To provide a solution to the problems of wayfinding, promoting ADSAIL, and guided UAA tours, this project aimed at creating a HoloLens 2 navigation application that provided engaging, interactive, and educational content. The application's working title is UAANAV: A HoloLens 2 Guided Tour Experience. The application has two modes: a tour mode and a map creation mode. In combination, they allow for maps with any desired content to be created by tour designers and experienced by interested parties.

2. Description of the planning process

The planning for the UAANAV application was spread across six phases. Each phase was a stepping stone onto the next phase, iterating through phase development as the application was built. There were times where a phase was revisited, such as research, to help incorporate later phases of the process.

2.1 Phase 1: Initial Concept Development

The first phase involved brainstorming sessions where a UML diagram was created to help visualize all the pieces that would be put into the UAANAV application. The concept of how the application would work was also workshopped during this phase. There were discussions about making a treasure hunt type application at first. This evolved into a tour guide application after defining what the client would practically use the application for.

In addition to brainstorming what the application would do, the objectives of the project were also defined during the first phase. Because the client chosen was UAA and ADSAIL, it was decided that the application would promote ADSAIL and provide UAA with campus navigation in an educational, interactive, and engaging tour for prospective students and visitors of the campus.

2,2 Phase 2: Research

After defining what the application would look like and the objectives the application would accomplish, the next phase was to research how to implement the project. This involved looking up other similar wayfinding projects using the HoloLens 2 device, in order to better understand if it was possible to implement and take inspiration from other projects. Then the tools of the project were researched. This involved working through official tutorials for the use of the HoloLens 2 device, reading documentation, and going through tutorials for the game engine and to set up the software development kits for the project libraries. This phase was revisited throughout the project, as it was required when deciding how to implement each requirement.

2.3 Phase 3: Scenario Development

Scenarios were developed for the tour application. This is where it was decided how a user would walk through the tour, and how they would interact with the hologram elements. Scenarios included a new student walking through a building they have never visited before, or a visitor that has walked through the building but is interested in particular exhibits inside the building, like the various projects displayed throughout the campus or points of interest like the hockey rink and basketball court. This scenario building provided information on what requirements the application needed to successfully allow the user to navigate, learn, and interact with the application.

2.4 Phase 4: Defining Requirements

The first three phases provided the information needed to define the requirements of the application. This is where functional requirements such as points of interest that are dynamically able to change between types of nodes were conceptualized. Another consideration made was to split the application into two modes: a touring mode and a map creation mode. The two modes provided the means to tackle functional, non-functional, system, and user requirements.

2.5 Phase 5: Prototypes and Testing

Throughout the implementation of the application, many tests and prototypes were created. The development usually started with adding an additional feature or updating a feature, uploading it to the HoloLens 2, then using the device to work with the feature and decide if any changes needed to be made. This phase often required going back to the research phase and requirements phase to rework the application.

2.6 Phase 6: Future Development

After the application was in its final iterations, there were considerations for its future. Additional features of the application were defined, current features that need improvement were highlighted, and a developer manual was created. The hope is that future developers can further enhance the application and implement many more tour maps of campus with the tools this project leaves behind.

3. Requirements

Because the project was not mentored and there wasn't a client to report to, we acted as our own clients, coming up with specifications to add or consider as the project was built. The Requirements can be divided into 4 sections: functional, non-functional, system, and user.

3.1 Functional Specifications

The application has two core functionalities. A navigation component, and a point of interest component. The navigation component consists of the ability to direct a user towards a waypoint or point of interest, with the ability to update the user's location in real-time or through button interaction. The points of interest component provides location based objects that a user can interact with. These points of interest can display photos, games, and other content anywhere along the navigation route. An important function is that points of interest and waypoints can be dynamically changed to any content the tour creator wants it to be, allowing the tour to have a point of interest swap between any type of media deemed appropriate.

User interaction is another functional requirement. The user can interact with buttons at points of interest to change the hologram or to move on to the next waypoint or point of interest. The user can also play games at points of interest that might require eye movement or hand interaction.

3.2 Non-Functional Specifications

Some non-functional specifications are also important. One aspect is the loading of the map, which must load in about 5 to 10 seconds to give a fast response for the user. Additionally, the response from interacting with hologram objects needs to be near instantaneous so the user feels like their actions have done something. Usability is also important, which is that the design of the tour is easy to navigate, and the content is intuitive to interact with in the augmented environment. The reliability of the HoloLens 2 also needs to be able to run the application for the duration of the tour which is dependent on the battery life of the headset.

The HoloLens 2 application also needs to scale well, which is easy enough to just download the application on multiple devices. The only consideration is the querying of the database when creating the map, but this should be fine since it isn't a big dataset for any one map. Finally, there is compatibility. The application needs to work on future firmware updates to both the hololens 2 application and the development environment for the software maintenance.

3.3 System Specifications

There are four hardware specifications to take into consideration:

- Hardware
- Software
- Network
- Database

The hardware needed for the application is the HoloLens 2 itself. The software required includes the operating system that the HoloLens 2 runs on and the development environment that the UAANAV application uses, which is Unity. Additionally, the third party libraries that are used for the software are the Azure software development kit and the Mixed Reality ToolKit. A network connection is needed to run the application so that positional data can be retrieved from the cloud database that is used. The database consists of Azure Spatial Anchors (ASA) and Azure Table Storage (ATS). This is to store and retrieve the spatial information for the game objects in the map.

3.4 User Specifications

Users are required to have the ability to operate the HoloLens 2. Some training and calibration are required before a user might feel comfortable in the augmented reality environment. Technical proficiency is needed to use the headset effectively. The user also needs the physical abilities to interact with the holograms. They will need to have hands and be able to make gestures with their hands. They will also need the ability to see and move their eyes for eye gazing interactions.

4. Description of Design

4.1 UI

The application made heavy use of premade UI elements provided by the MRTK. These were mostly used on the developer end, but the slate in Figure 1 was also featured in the games, and buttons were employed at many POIs for engagement or advancement. More than simply accelerating development, it was a practical necessity as they have been optimized to handle events and visual display much more efficiently than if we implemented the features ourselves. Furthermore, using the traditional Unity UI canvas standalone was not an option as it did not appear on the Hololens at all.

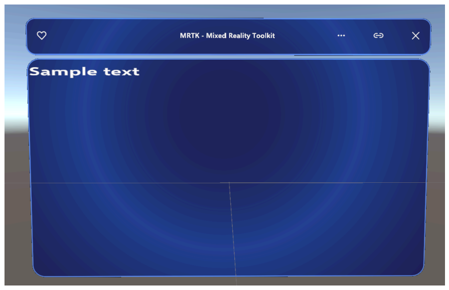
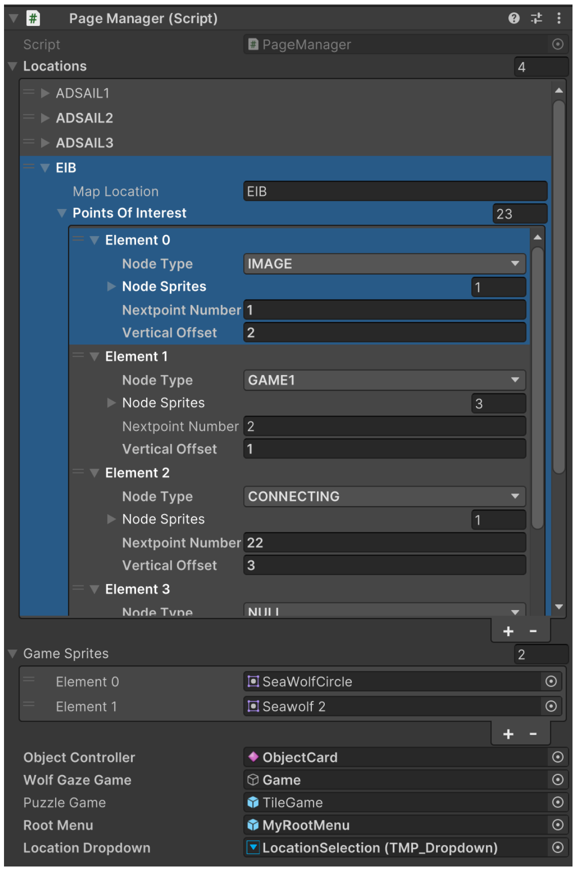


Figure 1: Simple slate UI developed by MRTK

Besides slates and buttons there were other UI features and functionalities used throughout. Text was used to provide instructions to users and developers or as labels for button functionality. For this purpose Unity's built-in TextMeshPro was used for its flexible display options and easy to use interface. More subtle UI implementations were in the form of event handling triggered through button presses. Unity makes this process trivial to work with through drag and drop functionality. All game object behavioral changes and public function calls can be linked to these events in the editor.

4.2 Data Structures

The central data structure spanning all aspects of the application is the PageManager. It takes the cloud-originating information from the Azure databases and, combined with developer inputs, gives form in the real world that users can observe and interact with. As depicted in Figure 2, this single point of contact stores all dynamic information related to the function and appearance of each POI. It incorporates a developer interface that allows effortless drag and drop functionality and data input with minimal training required. From this interface, users can select POI type, any number of images to cycle through, and the next POI that it should direct the user to.



```
Page Manager (Script)
Script                      PageManager
Locations                   4
  ADSAIL1
  ADSAIL2
  ADSAIL3
  EIB
    Map Location            EIB
    Points Of Interest      23
      Element 0
        Node Type           IMAGE
        Node Sprites        1
        Nextpoint Number    1
        Vertical Offset     2
      Element 1
        Node Type           GAME1
        Node Sprites        3
        Nextpoint Number    2
        Vertical Offset     1
      Element 2
        Node Type           CONNECTING
        Node Sprites        1
        Nextpoint Number    22
        Vertical Offset     3
      Element 3
        Node Type           NULL
Game Sprites                2
  Element 0                 SeaWolfCircle
  Element 1                 Seawolf 2
Object Controller           ObjectCard
Wolf Gaze Game              Game
Puzzle Game                 TileGame
Root Menu                   MyRootMenu
Location Dropdown           LocationSelection (TMP_Dropdown)
```

Figure 2: PageManager setup

Figure 3 represents an abstraction of a sample tour that can be developed. In this view it can be seen that the tour is effectively traversing a linked list with connection “breadcrumbs” and various POIs that bridge the start and end nodes. In the case that the end is chosen to lead back to the start, a circular linked list is created.

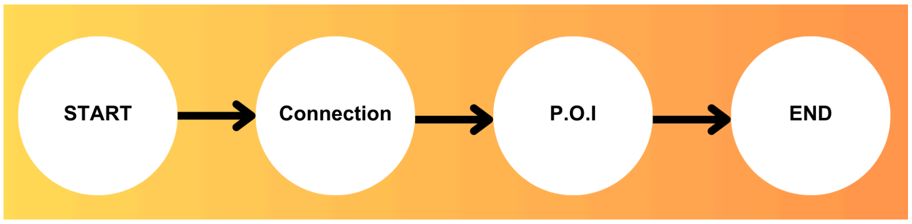


FIgure 3: Abstract tour node design

4.3 Architecture

There are two modes for our application: the tour mode and the map creation mode. The overall system architecture for the tour mode is represented in Figure 4. The tools that were used to create the application were Unity for the game engine, MRTK which is the UI library provided by Microsoft, and the Azure Software Development Kit for interacting with the Azure Spatial Anchor (ASA) service and the Azure Table Storage (ATS) service.

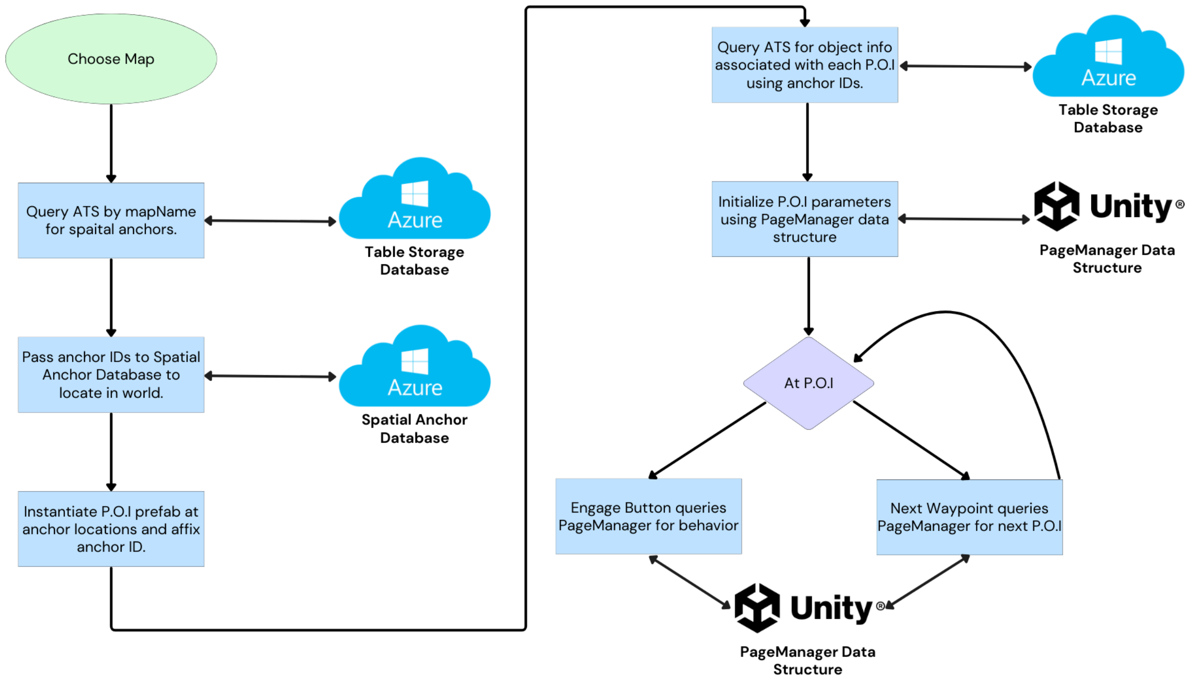

Choose Map
Query ATS by mapName for spaital anchors.
Azure
Table Storage Database
Pass anchor IDs to Spatial Anchor Database to locate in world.
Azure
Spatial Anchor Database
Instantiate P.O.I prefab at anchor locations and affix anchor ID.
Query ATS for object info associated with each P.O.I using anchor IDs.
Azure
Table Storage Database
Initialize P.O.I parameters using PageManager data structure
Unity
PageManager Data Structure
At P.O.I
Engage Button queries PageManager for behavior
Next Waypoint queries PageManager for next P.O.I
Unity
PageManager Data Structure

Figure 4: Tour mode architecture

The user starts the tour process by interacting with the MRTK developed UI, as seen in Figure 5. A map location is selected from a list of created maps, which are named after the building that the tour takes place in. The program then queries the ATS for spatial anchors, which are passed to the ASA cloud for the locations of the points of interest. The ATS is then used again to create game objects though the PageManager data structure at the various points of interest. The user is then able to walk to points of interest and engage with the UI there to either display different information, play a game, or move on to the next point of interest in the tour.

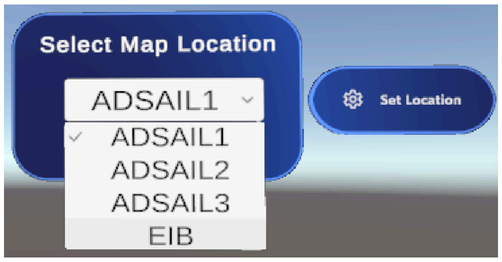
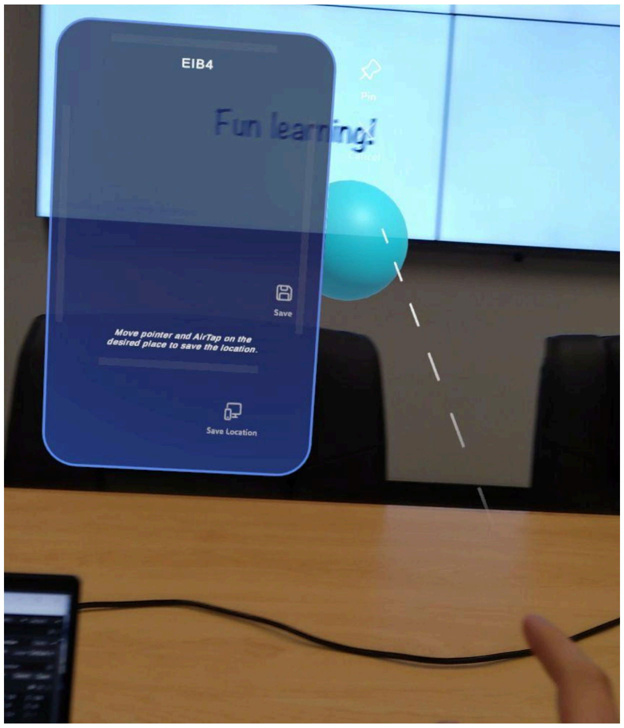


Figure 5: Map location selector

The architecture for the map creation mode involves the PageManager. The data structure as described earlier is used to insert data for game objects at the points of interest along the tour. It is largely a plug and play system for developers, and allows for the creation of many maps with any number of nodes. After the map is created, it will be added to the list for the map locations. Figure 6 below shows the UI for configuring the tour locations.



Move pointer and AirTap on the desired place to save the location.

Figure 6: Map creation UI

The two cloud components, ASA and ATS, were represented as databases. The ASA service is its own managed database, of which there was only a need to know the spatial anchor id to reference it. The rest of the database attributes are used for spatial localization. The ATS database was needed to persist anchors across sessions and we designed it. Table 1 shows the structure of the database table, which is named ‘objects’ in the cloud service. Each entry in the table is a spatial anchor, which can be represented uniquely by its ID. Then, we used the mapName field to identify which map the spatial anchor belongs to, and a name field to associate a unique spatial anchor with a game object. This also allows people to name their points of interest for reference later when adding game objects to the PageManager.

Spatial anchor ID	name	mapName
Abcd1234	EIB0	EIB
dcba4321	EIB1	EIB

Table 1: Azure Table Storage database representation

4.4 Algorithms

Our project only consisted of two main algorithms, with most of the project being UI and interactive experiences. The first algorithm involves the Azure cloud to move around spatial anchor data. The second algorithm is directed at the chevron arrow guide for users to follow to waypoints on the tour.

We can visualize the algorithm for the spatial anchor data in Figure 7. The holoLens 2 will load up our application, then when the user wants to create a map they use the ASA service to get the location data. The spatial anchor that is created is then uploaded to the ATS service after the user saves the information for the map the anchor is part of, the location name, and the spatial anchor ID is automatically placed. The more important and complex part of the algorithm is getting that data and using to load game objects, this is a step-by-step representation of this part of the algorithm:

1. User selects 'load-all' for the map they have chosen
2. The HoloLens 2 queries the ATS for all the spatial anchor IDs with the specific mapName attribute that was chosen by the user.
3. The spatial anchor IDs are then saved into a list, as well as their names.
4. The spatial anchor ID list is used by the application to query for the location information held by the ASA service
5. The location information and the spatial anchor name is then used by the PageManager to create game objects in the real world

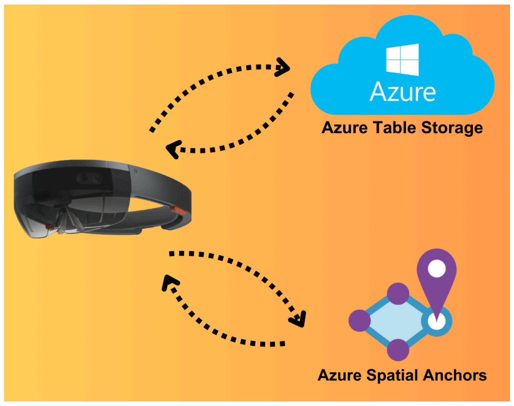


Figure 7: Abstract cloud algorithm to get spatial anchors

The last algorithm is implemented for wayfinding between points of interest using an arrow/chevron. This algorithm is activated when the 'Advance' button on a point of interest is touched, which triggers a query to the PageManager to lookup the next point of interest or waypoint which is saved within the data attributes for the current node by using the name field of the anchor. The query checks against all active anchors and passes the location of the next node to the arrow/chevron object, which then uses the MRTK library to point the arrow towards the next game object on the list.

5. Software Development Process

Most of the development process occurred in bursts of productivity with dormant periods of research in between. The process generally worked as follows: uncover a new resource that proposed a manner to implement what we wanted, exhaust all considered pathways with that resource, hit a dead end most of the time, and do more research. After multiple iterations we eventually had enough pieces that worked to have a functional MVP.

5.1 Testing and Debugging

Testing and debugging were both much more complicated than other projects in our experience for multiple reasons. Time required is on the forefront of reasons as every part of the process takes a long time. Setting up the environment is a time consuming and grueling process that must be done for every computer that works on the project. Then the process of building the project in Unity and deploying to the Hololens requires around 5 minutes on the lower end with Rane's computer requiring closer to 20. And that rebuilding and deploying must be done for every change ranging from a brand new feature to changing a mistaken flag from false to true.

Another problem we encountered was when using in-editor testing. There is an advanced emulator that can be used but requires Windows 11. This was not an option for Rane and was not used at all in lieu of MRTK emulation tools. In-editor testing is much faster with these MRTK tools than a full deployment but cannot capture the experience accurately. The sense of spatial scale is skewed and the most important features we care about such as querying the cloud and loading holograms cannot be performed. Furthermore, we learned that traditional standalone UI elements are not functional in the Hololens and must be nested in other objects instead.

The tests typically ran as follows:

- Deploy a new feature or multiple if they did not have the same dependencies
- Interact with UI elements noting responsiveness and appearance along the way to new feature
- Test new feature(s) while making note of ease of use and spatial position
- After typical unsuccessful trial theorize where the process went wrong

- After trying multiple theories return to problem from a different angle
- Make many tiny visual adjustments with each test

5.2 Planned vs Actual Work Schedule

The original plan outlined in our design document identified nice, clean time tables for each phase of the project. Naturally there is going to be overlap to phases and that was identified in our original Gantt chart, but it was clear, for example, that implementation should be completed by the end of March. It is not a surprise to most people with experience working on bigger projects that things don't always go according to plan, but we were surprised by how far development deviated from schedule.

Research, implementation, and testing all extended to the very last week. The design phase also went longer than projected as our understanding of what was possible developed, but we generally had the core concept identified relatively early on. Essentially we built it as we flew it as the saying goes. This was a natural and inevitable reality as there were many interconnected parts to the project that we had to learn about with each of us having experience in the different areas but not quite the whole picture. Each time we broke through a new barrier with research we reevaluated the best way forward and how to deliver the best product possible in the end.

5.3 Additional Challenges

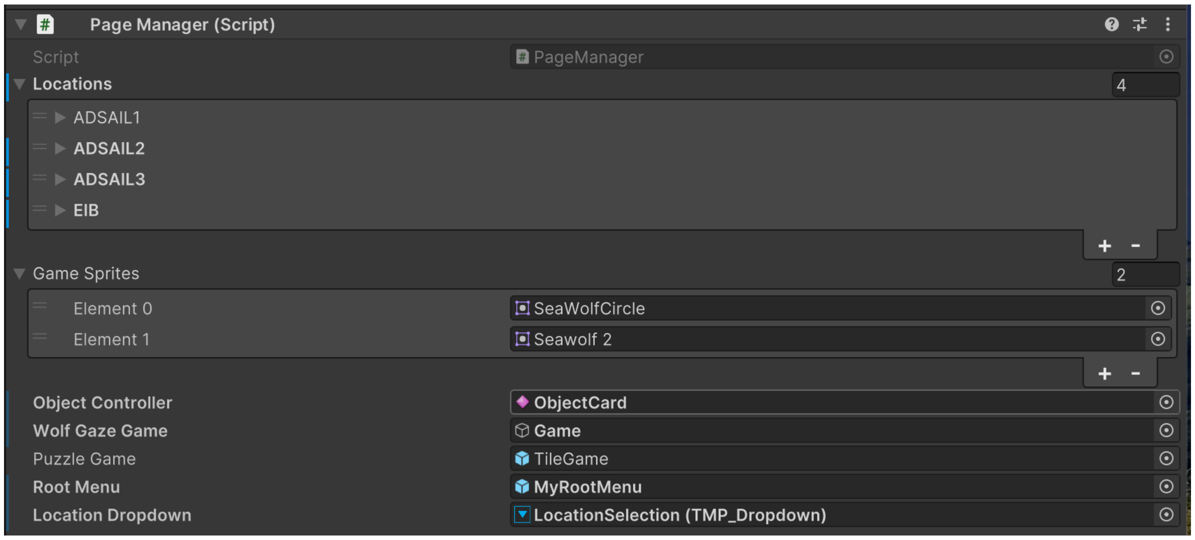
AR is not a new technology but wearable AR has not yet reached a critical mass of users. Economics plays a huge role in this. The Hololens is thousands of dollars which makes it infeasible for the typical hobbyist developer to use. Since it is so expensive and has little application outside industry or other niche areas it has not been widely adopted by consumers. This leads to less product support and documentation than would be available for a flourishing device. Throughout the development cycle we were hard pressed to find up-to-date documentation or tutorials that explained how a particular feature worked or could be used. This led to countless hours of research wasted on deprecated experimental features or other dead ends. Our hope is that we can save future developers from this pain with the provided resources.

6. Results

The UAANAV application finished with our initial goal: a minimum viable product that can guide users on a tour through points of interest in a map. There were no direct clients of our project, so we made our own goals and felt good about what we achieved with the requirements set for our program. We believe that the project demonstrates both the map creation and the tour mode application to the point of proving the concept that indoor navigation with the HoloLens 2 is both feasible and scalable. Most of the requirements were met with the final product, while some can be developed further.

6.1 Final Product

There are several figures to represent the core of our program. Starting with the map creation, a user can create and manage maps through the PageManager, as shown in Figure 8, inspector window found within the inspector interface on Unity. This is currently the only way to create new maps and dynamically update various nodes that are located on the map.



```
Page Manager (Script)
Script                PageManager
Locations                                   4
    ADSAIL1
    ADSAIL2
    ADSAIL3
    EIB
Game Sprites                                2
    Element 0         SeaWolfCircle
    Element 1         Seawolf 2
Object Controller     ObjectCard
Wolf Gaze Game        Game
Puzzle Game           TileGame
Root Menu             MyRootMenu
Location Dropdown     LocationSelection (TMP_Dropdown)
```

Figure 8: Page manager for creating and managing maps

After creating a new map, users can update the map with spatial anchors by opening the application on the HoloLens 2, choosing the map name that they created, and start placing down the nodes in the location they want the points of interest or connecting nodes to be at. Then, the user can come back to the page manager and attach objects to the new nodes that were created and add content. This will then be loaded into the map during the tour phase.

After the map is created and node types are established, the application can go into tour mode. Figure 9 depicts some of the P.O.I's that are possible to see along the tour, which includes games, pictures, or slideshows users can interact with.

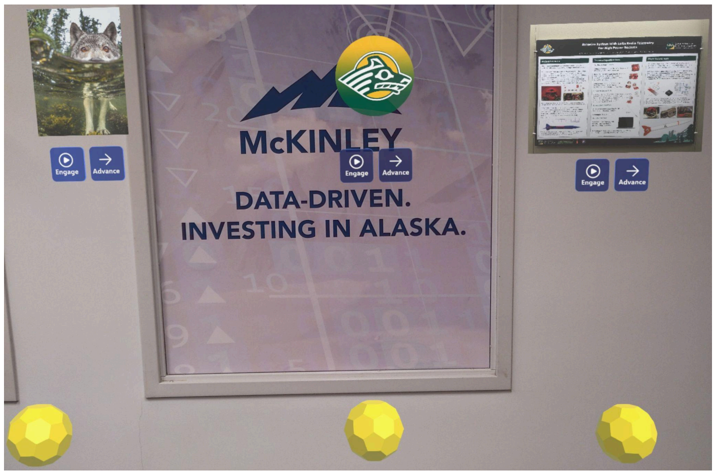


Figure 10: Different types of Points Of Interest

We also wanted to create games that leverage the technology of the HoloLens 2, in particular the eye gaze feature and the hand interaction feature. To do this, we created an eye gaze game where the user must look at various rings and wait for the seawolf to grow to the size of the ring to showcase the concept that the application can sense eye direction and interact with hologram objects in relation to eye movement. The game can be seen in Figure 11. Additionally, there is a tile puzzle game where a user has to tap on tiles to solve the picture puzzle, which demonstrates the hand interactivity with hologram objects. Figure 12 is a snapshot of the game in action.

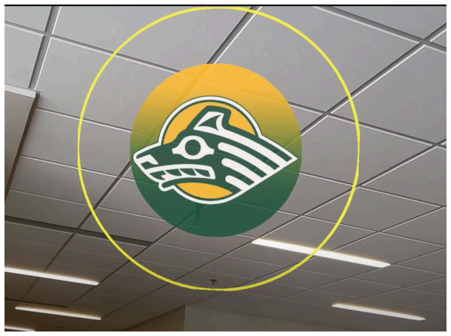


Figure 11: Eye gaze ring game

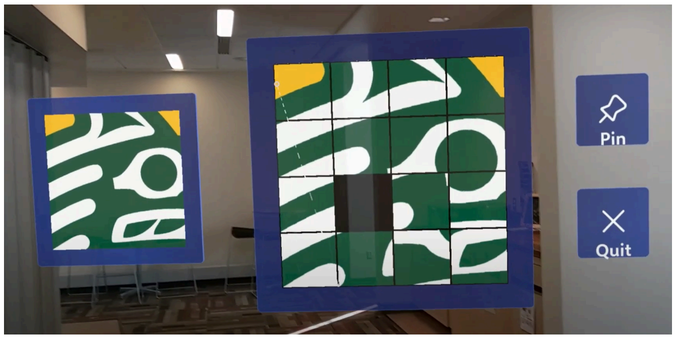


Figure 12: Tile game for hand interaction demonstration

6.2 Future Work

The application turned out well given the complexity of putting together the cloud components, the MRTK library, and the Unity game engine. We hope that other developers will further add to our project, as many things can be improved upon. One such idea is to implement the map creation process into a user interface and intuitive process that someone can do through the HoloLens 2 application instead of how it works now through the Unity game engine.

The PageManager can also be updated. It currently is organized well, but adding different points of interest, such as video, text boxes, and audio could be implemented so that users can have more diverse experiences. The organization for game objects also could be improved on here, where a file system or a cloud storage database needs to be designed and implemented. In combination with working on the PageManager, the chevron/Arrow indicator can be updated to use the data structure to guide a user through the tour experience. It currently is not user friendly, and only directs users to the next node if the ‘Advance’ button on a point of interest is pressed.

The UI also could use a lot of work. The current implementation of holograms, menus, buttons, and guiding arrows are all basic and supposed to represent the proof of concept. We found that UI is the most important aspect of our project, since that is what users interact with and follow. A clear tutorial on how to go through the tour is also needed, and a way to train users quickly before using the HoloLens 2 device since features like pinch click are not known unless we tell the user.

7. Conclusions and Lessons Learned

UAANAV was created using Unity, the MRTK library, and the ASA and ATS cloud services. The objective of our application was to provide a working proof of concept for using the HoloLens 2 as a tour guide application for new and prospective students at

UAA and to promote ADSAIL and its potential for the developer community. We felt that our application met the requirements we set out to achieve, with a lot of room for improvement and modification for future development.

The project was very challenging. It involved lots of research into Azure, the MRTK library documentation, official Unity tutorials for using the game engine, and official Microsoft tutorials on developing applications using the HoloLens 2 and spatial anchors. This project would not have been as developed if it weren't for the fact that a team member had years of Unity game development experience before starting. The HoloLens 2 and MRTK library had outdated and sparse documentation to help guide us to our goals, leaving us with having to lean on open source projects with the same ideas. With that being said, the experience gained from designing, implementing, and iterating through phases of development were worthwhile and ended in efforts that we feel proud of and can be used in a practical setting with future contributions and tweaks.

One thing that we would change is having a lot more time dedicated to user testing and unit testing. Our implementation of the project took a lot longer than anticipated, and left us using time dedicated to play testing on developing our project. When we presented the project to an audience, it was clear that the UI and game elements would need to be modified to make a more intuitive and fun experience. Because of this, we recommend that those who work on this project in the future work with people unfamiliar with the HoloLens 2 while developing features to get a better feel for what the application needs in terms of UI and functionality.

8. References

[1] G. Dedes and A. G. Dempster, "Indoor gps positioning-challenges and opportunities," *in VTC-2005-Fall. 2005 IEEE 62nd Vehicular Technology Conference, 2005.*, vol. 1. Citeseer, 2005, pp. 412–415.

[2] Brunnnner, X., Schalbetter, L., & Wu, T. (2020). *HoloNav: A mixed reality indoor navigation system.*

Appendix A: Heuristic Evaluation Table

Heuristic Issue (1-10)	Severity (1-4)	Description of Problem	Source (person)	Action Taken

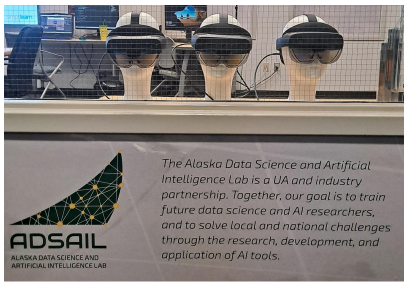
H8	1	allow tap with finger, not knuckle; better sensitivity	Cam	This was not directly implemented, it is a feature of the HoloLens 2. Can't directly influence touching .
H8	1	too long to wait for the seawolf logo to expand	Jaren	Created variable that allows developer to change speed at which logo expands
H1	2	text & icons seemed a bit small	Mya	Not implemented.
H3	4	pop up seemed jittery	Kai	Only in video, not implemented.
H4	1	text is sometimes squished	Eleonora	Only in video, not implemented.
H8	2	arrow said turn left, but you walked straight	Connor	Arrow needs work, but not implemented
H2	2	depth to button seems off	Mary	This is only apparent in video, not in actual use. Not implemented
H1	2	feedback so you know where your line of sight is	Robert	We thought about this, but one feature of the holoLens is that it automatically tracks your eye gaze. It would be hard to create a dynamic reticle that showed where the holoLens would think you are looking at and also extremely distracting. Not implemented
H4	1	document panel says engage and advance; non-standard menu names	Garrett	This wasn't implemented, but it's recognized this should change with future iterations.
H3	2	button pressing difficult	Antonio	The buttons are easily pressable in person, so not implemented or already resolved.

H8	1	poor image quality, e.g. wolf and posters	Anthony	This is only true in the video, much better while using HoloLens
H8	2	add zoom option for images	Hiromi	Can dynamically change image resolution in pageManger
Q		Do the activities scale with user height? (A = yes)	Quinton	Implemented.
H5	1	add height max to avoid straining necks	Quinton	The game has field of view parameters that can be changed.
H2	3	needed some instructions for users	Witmer	User Manual Implemented.
H3	2	requires calibration for each person	Cam	No way to change this. Not Implemented
H4	2-3	remove HoloLens open menu icon	Robert	Implemented. Added a finished mapping button to remove menu
H8	1	animate/smooth menu box opening	Connor	Not implemented.
H8	2-3	floating icons/chevrons visible through walls	Cam	Can't change this. Not implemented.
H1	2	Better readability on buttons, different button with less shadowing so text is clearer	Mika	Not implemented.
H4	2	"Engage" seems to mean different things at different stations -- better name for this button?	Witmer	Not implemented. But recognized this needs to change in future iterations

Appendix B: User/Developer Manual

Requirements

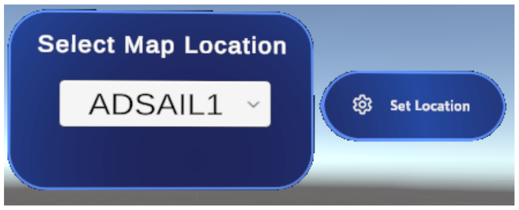
Users will need to have access to ADSAIL located on the third floor of the Rasmuson Hall building at UAA. They will need access to the lab or have a lab tech come in and give them access to the device. There are three HoloLens 2 devices there that already have the application installed on them:


ADSAIL
ALASKA DATA SCIENCE AND ARTIFICIAL INTELLIGENCE LAB
The Alaska Data Science and Artificial Intelligence Lab is a UA and industry partnership. Together, our goal is to train future data science and AI researchers, and to solve local and national challenges through the research, development, and application of AI tools.

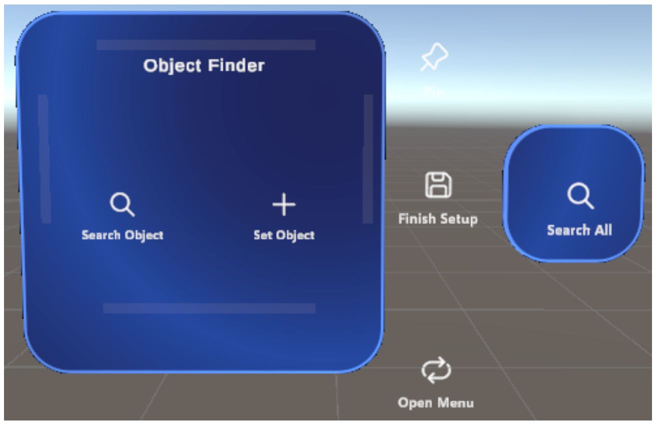
There are paper manuals on how to operate the HoloLens 2 next to the devices themselves which gives users the credentials to use the ADSAIL Microsoft account. Users will also require access to a Microsoft Azure account. Developers have additional requirements listed at the end of this section.

Setting Location

To start the app, navigate to the apps section of the microsoft menu while wearing the HoloLens 2. Select '_Capstone-UAA-NAV' app, it should be the first app in the list. After, the program will launch and bring you to the map selection screen:



You can use your hands to tap or pinch on the dropdown arrow to see a list of maps already created. To later display holograms and if you're in the ADSAIL lab, select the ADSAIL3 map. Once the map is selected, use your hands to tap or pinch tap the 'set location' button. This will bring up the object finder menu:



Finding Nodes

If you want to display the nodes in ADSAIL, tap/pinch the ‘Search All’ icon. Once you tap it, look around the room carefully and wait 5 - 10 seconds. The anchors should load and the game objects will too. If not, try to tap the ‘Search All’ Icon again. You can remove the menu by clicking ‘Finish Setup’ but will not be able to pull it back up without restarting the application.

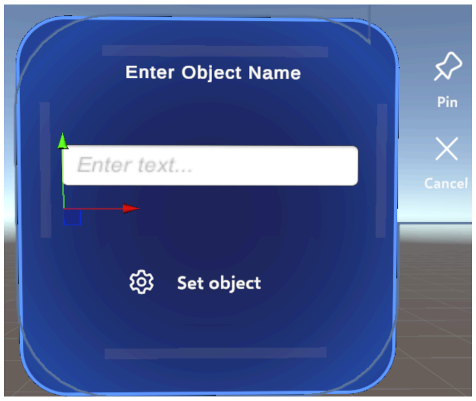
Once you load up holograms you can interact with the blue square mesh buttons for the game objects to work. Right now, in ADSAIL3, there should be a picture object, a tilegame object, and a slideshow. Play around with them to get a feel for the application!

Creating New Locations

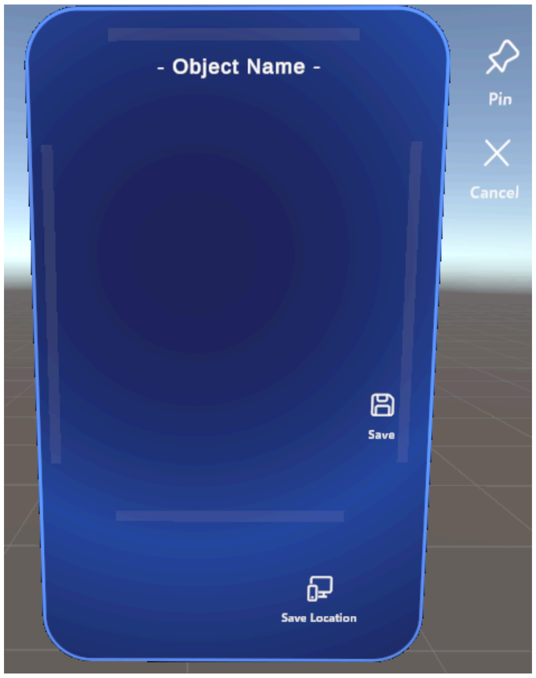
To add new map locations, in the editor hierarchy navigate to SelectLocation > Canvas > LocationSelection and go to the “Dropdown - TextMeshPro” component. Under “Options” click the plus button and enter the name of the new map location in the newly created space. Then navigate to the PageManager in the hierarchy and go to the “Page Manager (Script)” component. Under “Locations” click the plus button, expand the newly created entry, and enter the same new map location name. This name is stored as a string for comparison purposes so they must appear exactly the same in both places and is not restricted from using special characters.

Placing New Nodes

If you want to set up new points of interest or nodes, you can tap/pinch the ‘Set Object’ icon which will bring you to a object setting menu:



You can enter the name of your object here by tapping/pinching the text box, this will pull up a hologram keyboard to type in. For simplicity and reference, the application is built using a number system 0-(n-1) where n is the number of points you want on the map. After you set the name, hit enter on the keyboard. Then tap/pinch the ‘Set Object’ icon. This will pull up the object menu:



To set the point, tap/pinch the ‘Save Location’ Icon which will generate a blue sphere mesh to visualize the location of your point of interest. Tap/pinch to set the ball, it will turn into a cube and ask if you want to save the location, if you want to continue to move it to another location, select ‘No’, else you can select yes and set new spatial anchors up. After adding new nodes to a map location update the PageManager accordingly to reflect them. To do so, navigate to the PageManager in the hierarchy and go to the “Page Manager (Script)” component. Under “Locations” expand the map location that you added the nodes to. Change the “Points Of Interest” value to reflect the new number of nodes (using 0-(n-1) naming convention).

Adjusting Points of Interest

If you want to change node types or their features navigate to the PageManager in the hierarchy and go to the “Page Manager (Script)” component. Under “Locations” expand the map location you want to make the changes on. Each element in this list correlates directly to nodes in the selected location. For example, “Element 2” in the list addresses the point of interest named “2”. If a mistake is made and a number is skipped when creating nodes, the slot that would have been used must still be occupied in the list but will be ignored.

Expand the element you wish to make changes on and choose the node type from the dropdown. Enter a value into “Nextpoint Number” that the node will direct users to when using the “Advance” button (has no effect on “NULL” or “CONNECTING” nodes). If the node type is set to “IMAGE” one or more sprites may be loaded under “Node Sprites” and “Vertical Offset” may be used to adjust the image heights. The “Engage” button will be enabled automatically if there is more than one image supplied. Game selections will not be affected by sprite input as they have their own already.

Interacting With Microsoft Azure

The application relies on Azure Spatial Anchors and Azure Storage Table so a Microsoft Azure account is required. Follow the referenced tutorial to get a new account set up if one is not supplied. The account information is required in two locations in the editor hierarchy: DataManager > Data Manager (script) and AnchorManager > Spatial Anchor Manager (Script). The tutorial should be able to provide the details of where to find the relevant information in your account.

Saved table information for points of interest can be modified if need be. This may be useful if a naming mistake is made or for renaming a location. In either case keep in mind the name reflected in the table is a concatenation of the map location and the individual node name and must reflect accordingly. For example the point of interest located at EIB with the given name “2” is internally stored as “EIB2” and must be in the Azure Table Storage.

To modify table information directly, log in to the Microsoft Azure Portal and select the storage account resource linked to the application. Select “Storage browser” then “Tables” then “objects”. The table for all points of interest in the application will be displayed. Each entry can be right-clicked to edit or delete. Not all attributes can be modified such as “RowKey”. “RowKey” reflects the original name given to each node and will be different from the “Name” attribute if it has been modified after creation. This does not present a problem unless you wish to create a new node that will conflict with the pre-existing key in which case it will have to be deleted. The only attributes currently used in the application are “MapName”, “Name”, and “SpatialAnchorId”. Of these, only “MapName” and “Name” should ever need to be modified if at all.

Exiting

If you want to exit the application, hold up your right wrist in front of your eyes, and a Microsoft Windows icon should appear. Tap it, and the Microsoft Windows menu should pop up. Tap the Home icon at the bottom of the menu to exit the application.

Additional Resources

Unity Learn: Unity Learn

Azure Spatial Anchors Tutorial: Azure Spatial Anchors tutorial

HoloLens 2 Development Tutorial for environment setup, MRTK toolkit, and the Azure SDK for spatial anchors and table storage with Unity: Tutorial

MRTK documentation: MRTK Docs

Azure documentation: Azure docs

Github: Capstone Github

Developer Additional Requirements

A solid foundation in Unity is a must to further develop this project. At a minimum, prospective developers should have completed or have experience equivalent to “Unity Essentials” and most of “Junior Programmer” (complete up to Unit 5) from Unity Learn to be able to effectively engage with and enhance the application. These two pathways together are slated for 14 weeks to complete and does not account for the additional learning required of the other project components.