

# Augmented Reality Home

LOGAN PARKER

[lparker@go.olemiss.edu](mailto:lparker@go.olemiss.edu)

Sponsored by J. Adam Jones, PhD.

Professor and Director of Hi5 Lab,

6623257510, [jadamj@acm.org](mailto:jadamj@acm.org)

## Abstract:

In this project that I am calling AR Home, I planned to create an application for the Oculus Quest 2 which would allow a user to put a VR headset on and augment the world they see. The application should allow for a user to overlay their real furniture with virtual furniture and décor while maintaining a passthrough view of the real environment. This technique of composing virtual objects and the real world is known as augmented reality; when both real world and virtual objects are present in the same view.

The primary motivation for this project was to create an application that would be used by an interior designer who is with a client and wants to very quickly prototype different designs live for the client. This is typically not possible because placing multiple virtual objects in a space requires the use of images that must be taken in advance, then the designer would need use a computer to either painstakingly recreate the space in a 3D modeling software or to simply superimpose these images with virtual objects. Images alone cannot offer multiple viewing angles of the object in place, and a 3D rendering of the space will not be a perfect recreation of the space.

## Report:

After completing ARHome, I have learned an incredible amount about the development lifecycle, time management, overcoming challenges, and communication. This project was a success. By success, I mean that I set out to create a product with certain criteria, and by the end of the deadline, that product was created. I am very happy with the work that I did and the solutions I created to the problems I encountered throughout the project. I think one of the reasons that I was able to reach my goals because I found working on the project exciting.

This is not to say that everything was smooth sailing. I feel like I allowed the project to bring me far more stress than it should have. I devoted an immense amount of time to this project, far more than I ever imagined at the beginning of the semester. Every time I completed a new part of the project, I would think of a “better” way to implement it. This means, at times, I would code something two or three times before settling on an implementation. However, because I devoted so much time to the project, I had no problem completing the project before the deadline. I learned how to manage my time wisely and spend a regular amount of time on the project each day. I do not think that I, personally, am good at continuous development of something for a long period of time. I would much rather spend several consecutive eighteen-hour days working on a project and have it finished than work on something for a long period of time. I believe that this spaced-out development is what made me constantly be stressed about the project.

Furthermore, among the most difficult things throughout the development of this project was the everchanging SDK I was working with. The Oculus Integration Package went through five versions while I was developing ARHome. With each new version, some things would be removed or deprecated, which would require me to change the way my code was written. On the other hand, some things that I had spent hours working on implementing had been added as a built-in function. There was even a week where the SDK was broken entirely

for all developers using the Passthrough API. This meant that no development could be done at all. This is something that you never prepare for, and if it had happened at a different, more inopportune time, I may not have been able to demo anything.

This brings us back to things that I learned during the development process. I learned that working with bleeding edge codebases is exciting, but difficult. I think, honestly, it was the difficulty that made it so exciting. I would be so frustrated, but when I finally figured out how to do something that there is no information about online, it felt so good. Computer science has always been a puzzle to me, and it is the connection of different puzzle pieces that makes me enjoy it so much.

If I were to do this project again, which I probably will, I would go into it knowing so much more about how things should be setup and designed. After completing one iteration of the project and devoting so much time to Unity, I have a much better understanding of Unity and C#. Looking back at the code and general solutions that I created at the beginning of the semester, I can't imagine what was going through my head. The way I was doing things seems so obviously flawed now, but at the time that was the best solution I could think of. So, restarting this project and only keeping a small portion of the work I have done would prove extremely beneficial to the overall quality of the project.

In this redone version, I would like to add a network discovery tool that doesn't take three minutes to complete. I would also like to paginate the inventory system. Right now, there is a hard limit of 16 items that can be added to the inventory. I would also like to add a way for users to save their designs and have them appear in a menu which the user can select designs from. Adding in the new world space anchors that oculus has just implemented would be a great feature to assist with the loading of room designs. Another great addition to the project would be some general aesthetic design on both the companion application and the main oculus application.

In conclusion, I learned so much and had a great time developing ARHome. I had things go poorly and things go great, but in the end, I produced a product that I am very proud of. It can be improved in many ways, but I think that it is only after working on the project so long, I am able to see the weak points of the project and know exactly how to remake them. I am very thankful for the time I have spent working with not only my sponsor, but also the faculty of the department on this project and my other coursework.

## Revised Design Specification:

### User Requirements:

#### Minimum Viable Product:

The minimum viable product of this project should implement camera passthrough, or the ability of the user to see the real world through the eyes of the VR headset's multitude of sensors. The application will indirectly interface with depth data and infrared camera data and create a layer mask which overlays virtual 3D models over real world geometry. The data must

be interfaced with indirectly using Facebook's Passthrough API because depth and live camera footage is not accessible to developers for privacy and security reasons. It should also implement a 3D object placement system, which allows a user to place 3D objects in a virtual coordinate system and manipulate them to correctly overlay real world objects. The application should have a robust set of 3D models preloaded and, if it proves possible, connect to a outside database to load 3D models into the application dynamically. These 3D models will ideally be presented in a simple, yet effective, virtual user interface. Depending on the complexity and limitations of the hardware and software platforms I am working with, implementation of layer masking which allows a mix of real and virtual camera views would be a nice achievement.

#### System Users:

Only one user is required given the nature of the project. The user will be whoever owns a VR headset with each headset having a separate instance of the application with its owner's designs. If I create a way for the user to interact with Sketchfab, the aforementioned 3D model database, the user will require a Sketchfab account.

#### Designs Choices:

Previous applications similar to AR Home have used various head mounted displays (HMD's). There are many products such as the Microsoft HoloLens which are made specifically for Augmented Reality, and there are several VR headsets which have been used in previous work such as the Oculus Quest 2, the older Oculus Quest 1, the PlayStation VR, the Valve Index, and the HP Reverb. When considering the primary use case, portability is a must, and the Oculus Quest and the HoloLens are the only headsets that can be used without being tethered to a computer or game console. This immediately limits the possible headsets to the two Oculus Quest headsets and the HoloLens. The HoloLens offers a far better Augmented Reality experience than the Oculus Quest, but at the time of writing, the HoloLens 2 is twelve times the price of the Oculus Quest 2. The camera feed on the Oculus Quest is limited to black and white whereas the HoloLens does not use a camera feed to recreate the world, but a semi-transparent screen. The Oculus Quest 1 is also cheap, but it has been replaced by the Oculus Quest 2.

The software utilized in this project will be vast. The creation of VR/AR applications can be done from pure code or in both of the most popular game engines: Unity and Unreal Engine. The creation of an augmented reality application like this is quite impractical without the use of a game engine. While there can be greater customization and novelty in an application created without an engine, a game engine allows the developer to utilize things that take entire careers to perfect. This allows the creation of something new rather than wasting time "reinventing the wheel". The Unity Game Engine utilizes the C# programming language to create scripts to control game behavior and manipulate 3D objects. Unreal Engine uses C++. The precedent for projects like this one is to use Unity. It has had XR, which is a term that encompasses AR, VR, and MR, for the longest time and has the most support. A divisive victory for Unity comes from the fact that the Oculus Passthrough API is only implemented in Unity's Oculus Integration Package. This Beta API is the only way that a developer can access the camera feed from the Oculus Quest.

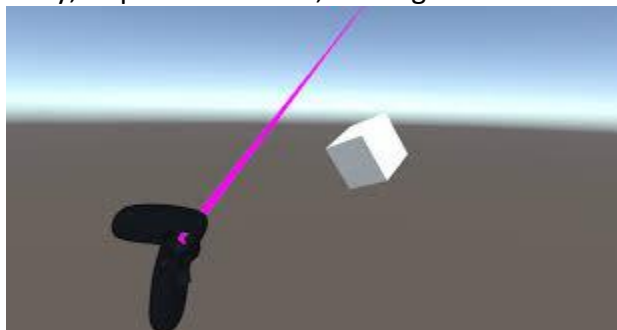
### Design Selected:

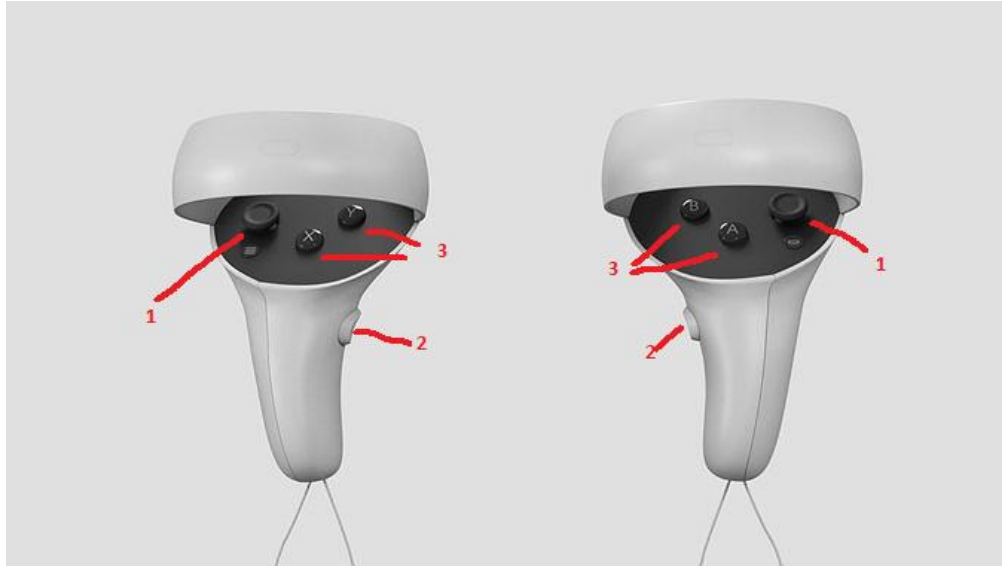
With all of these technologies in consideration, the Oculus Quest 2 was chosen as the primary platform because of cost and ubiquity. When considering the userbase of these headsets, the Oculus Quest 2 has a 30%-40% market share of all VR headsets, and the application can work on both the Quest 2 and Quest 1 further increasing the userbase. The Oculus Quest 2 is the most widely used headset on Steam with making up 32% of all VR headsets. Additionally, it is the most popular headset used on Oculus's own platform. The game engine I have chosen is The Unity Game Engine. It will be used to aide in the development of this augmented reality software (see sec. Development Environment). This means that the majority, if not all, of the programming will be done in C#. C# is an amazingly versatile language that is very similar in syntax to Java. This will allow for easy readability and development. The main API's used will be the Oculus Integration Package, the Oculus Passthrough API, and the OpenXR Backend. The application will be built into an APK for distribution and execution on the Oculus Quest.

The workflow of the application will be: user puts on headset in their space they want augmented, open the application, be presented with a passthrough view of the room, press a button on the controller to open a menu of items, place an item in the virtual space, manipulate the object into position, repeat.

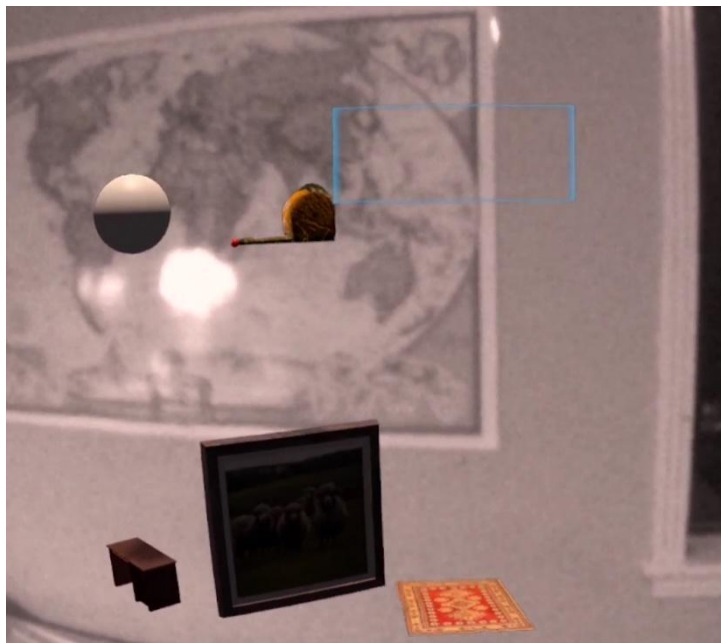
### User Interfaces:

A user will interact with the application using both of the tracked controllers that come with the Oculus Quest. These controllers will be used as a means to position virtual objects in the environment by simply grabbing the objects with the grip controllers (2) as if they were real. The controllers also have traditional buttons (3) on them which will be used to open different menus such as the inventory pictured below. The analog sticks (1) will be used to rotate, scale, and cycle through objects in the inventory. The controls will be fairly intuitive, but also have hints at runtime to remind the user of the control scheme. There is also the component of the Head Mounted Display which will provide information to the user's eyes. Simply put, the application will be interfaced with entirely through pointing a controller so that a ray, as pictured below, coming from the controller is hitting the intended object.





1. Analog stick
2. Grip trigger
3. "Traditional Buttons"



#### Development Environment:

The application will be developed in Unity Game Engine running on a Windows 11 powered desktop PC with a Ryzen 5600x CPU, 16GB of RAM, and an Nvidia Rtx 3080 GPU. The Unity platform is highly extensible and ready to be adapted to current and evolving needs with a powerful C# scripting system, comprehensive API and extensive documentation. It allows the creation of Game Objects which can be completely customized and reused throughout the program. The majority of the work will be done in Unity 2020.3.4f1 personal, but as mentioned

before, Unity utilizes C# scripts to connect game objects and functions. This means that the programming will be done in Visual Studio 2019. Versioning is being done through Unity and by saving the entire source code as a zip on an external drive that is stored in a different location after every major feature addition or bug fix.

#### Deployment Environment:

After development, the application will be deployed onto a 64GB version of the Oculus Quest 2 running a modified version of Android. The Oculus Quest 2 is a lightweight, yet powerful standalone VR headset which utilizes the Qualcomm Snapdragon XR2 SOC. The deployment process is handled by Unity to build the application into an APK file which can then be distributed as a standalone file to Oculus headsets. This application will be supported by the use of screen capture software to demo the work being done to the client.

#### Test Plan:

The test plan will be following the Oculus Developer's guide to VR App Testing. The scripts and individual systems in the application can be tested using unit tests. This will ensure that each unit of the program is working as intended. Where needed, integration tests can be run to make sure that two or more systems work as intended when connecting to each other. The final way that testing will be implemented is through End-to-End human testing because as the experts in this field state, "There are ways to automate these tests, but keep in mind that automating parts of the QA process can require a high degree of skill and resources that you may want to prioritize elsewhere. (Testing 5)"

The End-to-End human test plan will consist of adding a feature to the application, and building it and pushing it to the headset. Then, put the headset on and test the feature for completeness and flaws. A backup of the project will be kept after each major addition to the project to prevent complete loss of work. The version can also be incremented in Unity to help keep track of the latest version. The final step, once it has passed all unit tests, integration tests, and my validation, will be to send the application to my sponsor to test. This will serve the same purpose as beta releases to a small group do in large companies.

#### Bibliography:

"Oculus App Development in Unity | Oculus Developers." Oculus for Developers, 2021, developer.oculus.com/documentation/unity.

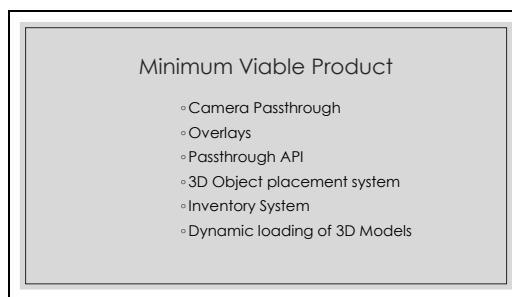
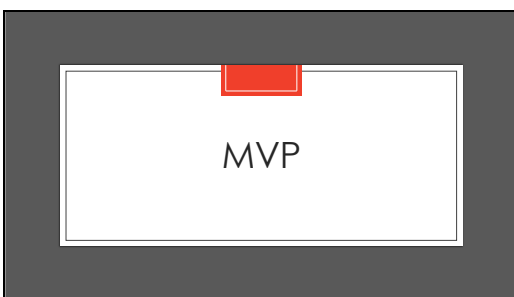
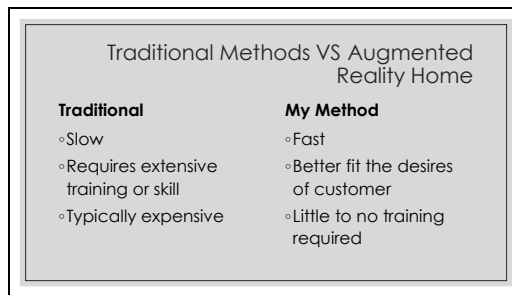
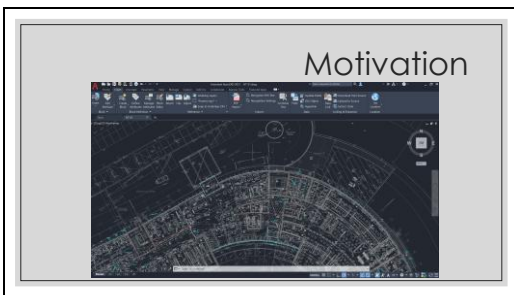
"Unity - Manual: Unity User Manual 2020.3 (LTS)." Unity Documentation, 2021, docs.unity3d.com/Manual/UnityManual.html.

Parker, Logan "World Builder" Oculus Quest 2 Application, 2021, Immersive Media University of Mississippi

Statista. "Steam User VR Headset Share Worldwide 2021, by Device." Statista, 22 Sept. 2021, [www.statista.com/statistics/265018/proportion-of-directx-versions-on-the-platform-steam](https://www.statista.com/statistics/265018/proportion-of-directx-versions-on-the-platform-steam).

Oculus. "VR App Testing Guide - Automation and Performance | Oculus Developers." *Oculus Developers*, developer.oculus.com/resources/automation-performance-testing.

## Final Oral Presentation Slides:





# DESIGN CHOICES

## Design Choices Headsets



## Design Choices Platform

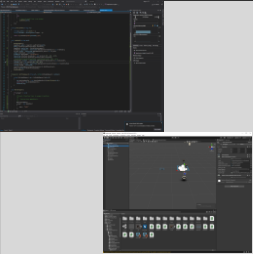


## Design Choices Languages



# TOOLS

## Tools



- Visual Studio (16.8.6)
- Unity3D (2020.3.4f1 Personal)
- Liv.tv (@latest)
- Postman (@latest)
- SideQuest (@latest)
- GitHub (@latest)

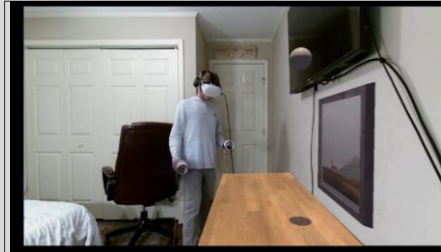
## LIBRARIES

### Libraries

- Oculus Integration Package (@latest)
- GLTFUtility (@latest)
- .Net Framework (4.8.04161)
- IISExpress-Proxy (@latest)



## MIXED REALITY VIEW



## DEMO

## HIGHLIGHTS



QUESTIONS?