NUI-SpatialMarkers: AR Spatial Markers For the Rest of Us

Alex Karduna, Adam S. Williams, Francisco R. Ortega

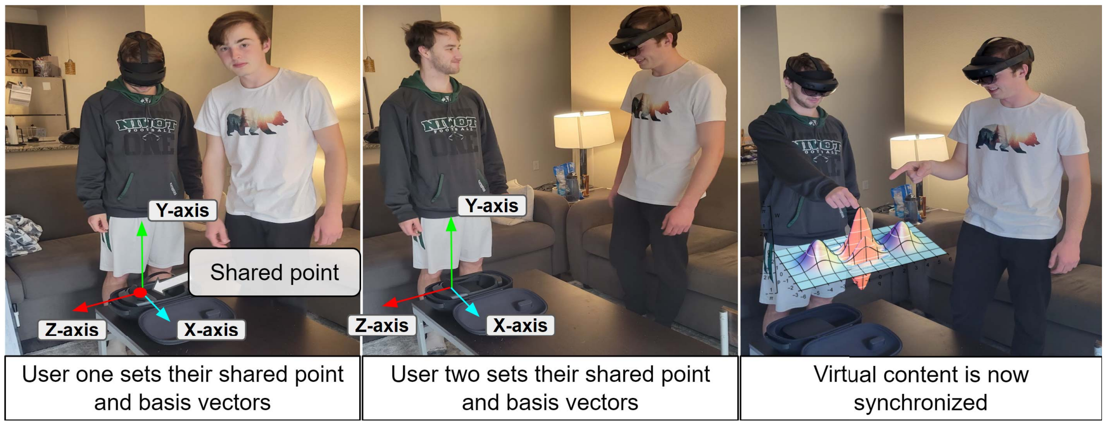
Y-axis
Shared point
Z-axis
X-axis
Y-axis
Z-axis
X-axis
User one sets their shared point and basis vectors
User two sets their shared point and basis vectors
Virtual content is now synchronized

Fig. 1. NUI-SpatialMarkers cross-device synchronization process

Abstract—Multi-user Augmented Reality (AR) development tends to be very difficult and there are few tools that allow developers to easily deal with the unique problems that arise. One of the main problems that arises from multi-user AR is that of aligning objects. In this paper we present a light weight, open source system for solving the aligning problem. This system was specifically designed to make it as easy as possible for developers to modify components to fit their needs.

Index Terms—Augmented Reality, Networking, Spatial Anchors

2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW) | 978-1-6654-8402-2/22/$31.00 ©2022 IEEE | DOI: 10.1109/VRW55335.2022.00022

1 Introduction

Unlike Virtual Reality (VR), in augmented reality (AR), there are both virtual and physical objects. Physical objects are objects that can be seen and interacted with when not using the AR device, while virtual objects can only be seen through the AR displays. The advantage of AR devices over VR and regular reality is that virtual and physical objects can both be seen and interacted with, whereas virtual reality only allows for virtual objects to be interacted with. Alongside this unique ability comes a unique set of problems. One of these problems is that of making multiple separate virtual objects synchronized between simultaneous co-located users.

In a co-located collaborative environment users must be able to see and act upon the same virtual objects. These objects must also have their location and movements synchronized on all of the connected head mounted displays (HMDs). Without this synchronization, both a user's immersion and their understanding of the environment can be degraded. Consider viewing a shared visualization, if one user is pointing at an area, then the other users need to have their HMDs displaying the shared visualization in the same position, otherwise they would not know what was being pointed at.

Recent work has used a multiplayer Unity library in conjunction with image targets to synchronize content between AR devices [2]. Another approach to cross-device synchronization used in VR was to create a 3D model of the real-world environment and then have each connected VR device use it's controllers to record three identical locations which were then used to synchronize the devices' virtual content [1] [1]. Outside of research, systems including Azure's spatial anchors and Google's ARCore cloud anchors solve this problem but they do not allow the developer to control how the problem is solved [2] [3]. Additionally, both ARCore and Azure use resource heavy programs to track physical objects then base the virtual objects location off of those physical objects, a process which can degrade device performance depending on the available compute power.

The major disadvantages of the existing synchronization approaches and tools for cross-device coordinate synchronization is that they can add an unnecessary level of complexity to the project, introduce more computational overhead, or remove the developer's control of how synchronization is achieved. In this paper we introduce a novel Unity compatible synchronization tool called NUI-SpatialMarkers. NUI-SpatialMarkers was developed to be easy use and easy to incorporate into an existing Unity project. It uses limited compute resources and is able to be modified to fit developer and researcher needs. NUI-SpatialMarkers is available for download at `https://github.com/NuiLab/NUI-SpatialMarkers`.

2 Global Coordinate System

The first step in this system is creating the global coordinate system. This global coordinate system is created by picking a point to be the origin and a set of vectors that form the basis of the vector space. Any point could work as the origin as long as all of the AR devices know the location of the point in their own coordinate system, otherwise

- *Alex Karduna is a undergraduate researcher at Colorado State University working in the NUILAB. E-mail: akarduna@colostate.edu.*
- *Adam Williams is a DARPA Post-Doctoral Fellow at Colorado State Univeristy working in the NUILAB. E-mail: Adam.Sinclair.Williams@colostate.edu.*
- *Francisco R. Ortega is an assistant professor at Colorado State University and Director of the NUILAB. Research. E-mail: fortega@colostate.edu*

[1]`https://github.com/felixkosmalla/unity-vive-reality-mapper`

[2]https://docs.microsoft.com/en-us/windows/mixed-reality/design/spatial-anchors

[3]https://developers.google.com/ar/develop/java/cloud-anchors/introduction

there would be no way of converting from the coordinate system of the device to the global system. The real-world point that is used as the origin for the connected AR-HMDs coordinate system is called the shared point.

There are many methods of creating a shared point between several devices including image targets [2] and manual triangulation [1]. This toolkit uses a a relatively simple method for the generation of a shared point to limit setup complexity (Figure 1). That said, this toolkit is made to allow users to insert their own methods for synchronization, allowing them to incorporate the most appropriate method based on their own needs. An example approach that the developer could implement would be to place a Quick Response (QR) code in a location where all of the devices can scan it. As many AR devices will give the position of any QR code scanned, the position of the scanned QR code can be used as a share point. The default method used in the toolkit is to place one of the AR devices on a stable surface and have the system mark the headset's position as the shared location. This process must be completed by each connected AR-HMD.

After all AR-HMDs have their shared point (i.e., coordinate origin) set, the device's coordinate systems must be oriented. To do this a basis or set of vectors that represent the x, y, and z axes are created. While many differed basis vectors are possible, some lead to easier implementation than others. The best basis vectors should be Unity vectors and would ideally be shared between devices. This toolkit uses the y-axis as the first basis vector. Most AR and even VR devices set the y-axis perpendicular to the ground plane, allowing that vector to be instantly shared between devices.

Generating the vector for the x axis will depend on the method used for creating the shared point. In the default method, the vector that is pointing in the direction of the device while it is placed at the shared point is used. In order to make the vector easier to use the y component is disregarded and the vector is scaled to a unit vector. For other methods of getting the shared point, different methods for creating the x axis will need to be used. The key is to ensure that the vector is the same on all devices, has no y component, and has been scaled to a unit vector. Once the x and y axis vectors have been created the last one is the z axis which is just an orthogonal vector to the x axis that has no y component. It is again helpful to scale this vector to a unit vector.

3 Converting Coordinates

Having created the basis, and found the shared point to use as the origin, the coordinate system is done. The last thing step is to figure out the transformations required to convert to and from the local coordinate systems to the global coordinate system. This task can be simplified greatly if all of the coordinate systems share the same scale, which is true for most use cases and is what this tool assumes. With this assumption only three transformations need to be made to synchronize a point between different device's coordinate systems. The first step is to subtract the vector of the synchronized point from the origin of the global coordinate system. This gives the vector from the origin point to the synchronized point. That vector then needs to be multiplied by a rotational matrix in order to orient it relative to the origin of the global coordinate system. After this last transformation has been applied the resulting vector is the vector to the synchronized point in the global coordinate system which can then be sent to the other devices. The other devices can simply apply the inverse of this process to convert the vector into their local coordinate system by using their own rotational matrix and origin point. This process does not change the rotation of the object; however, a similar process can be done on the vectors representing the rotation of the object to rotate it. The vector representing the objects rotation do not need to be subtracted from the origin of the global coordinate system, they only need to have the rotation matrix applied.

The synchronization process is executed by the *converter* component of the NUI-SpatialMarkers tool. Developers can implement variations of the process above to suit their project's needs. These variations can range from adding another transform to account for different scale coordinate systems, to using quaternions to avoid certain constraints on the movement (specifically gimbal lock).

The last steps required are to send the new coordinates to all of the other devices and to receive any sent coordinates while also updating the respective position of the objects affected. This is done using two components, the first is networking component that sends any data passed to it and stores any data received by it. The default implementation of this system sends the data passed to it as user datagram protocol (UDP) packets to the other IP addresses supplied. NUI-SpatialMarkers is configured to work on LAN and can be configured to work on larger networks. The networking component automatically checks every frame to see if there has been any data sent to it, and if so, it reads that data and pushes it into the queue.

The second component used during this step is the object controller. This component will need to be added to each synchronized object. Each object controller will have a unique object identification code (ID). The object controller component has two jobs. First, when the object that the object controller is attached to moves, the controller sends the new coordinates of that object and it's ID to the converter component. The converter then sends the processed coordinates and ID to the the network component which sends those coordinates and the ID as a UDP packet to all other devices.

The object controllers second job is to check if the network sender has any packets that have the same ID as itself. If the network sender does, it pop's them off the queue and sends the coordinates to the converter to be converted back into the local coordinate system. Once converted, the object's transform is updated to match the new coordinates.

The object controller component has been given a couple other quality of life features, including disabling interactions with the attached object when another user is interacting with it, and being able to disable the object completely if the user removes that object. The object controller class is made to be extendable by developers to suit their needs. Possible changes include having multiple objects controlled by the same controller, and making the controllers capable of generating new objects.

4 Usage

This section provides a quick start for adding this package to an existing Unity project.

1. Download the repo inside the Unity project's assist folder [4]
2. Create an empty game object
3. Add all synchronized components as children of that empty object
4. Add the "network" and "converter" scripts to the empty object
5. Get the IP address of the AR devices that will be used and add them to the public list in the network script
6. Add the "object controller" script to each of the objects
    (a) Add the converter script to the "conv" public variable in each of the object controller scripts
7. Add the "orienter" script to the camera/headset (this is typically the main camera in the scene)
    (a) Add the "converter" script to the "conv" public variable on the "orienter" script

If the project does not use the Mixed Reality Toolkit (MRTK) version 2+ an additional step must be taken. In the code for each of the usages of MRTK 2 specific commands there are comments explaining what that code does so that the developer can replace it with a comparable command from the AR development package that they are using.

Once these steps are complete the project is ready to build.

[4]`https://github.com/NuiLab/NUI-SpatialMarkers`

5 Limitations

NUI-SpatialMarkers was built to work for most standard AR use cases; however, there are some cases that are not covered by default. Two of those common cases are, support for devices that have different scale coordinate systems, or coordinate systems which have a y-axis that does not point out from the center of the earth. Another limitation was the choice to incorporate parts of the MRTK. As the MRTK is a common AR development library this should not pose a problem for most use cases, but when a project uses a different AR library this package requires a few extra steps to set up.

6 Conclusion

NUI-SpatialMarkers provides a easy to use and highly modifiable solution for networking and synchronizing across multiple co-located AR-HMDs. This package provides developers and researchers with a quick way to synchronize basic content across AR devices while also providing space for extending the package to fit diverse use cases.

Acknowledgments

This work was supported by the Office of Navel Research (ONR) awards N00014-21-1-2949 and N00014-21-1-2580, by the National Science Foundation awards 2106590, 2016714, 2037417, and 1948254, and by the Defense Advanced Research Projects Agency (DARPA) award HR00112110011. We would like to thank all of these sponsors for their support during this project.

References

[1] F. Kosmalla, A. Zenner, M. Speicher, F. Daiber, N. Herbig, and A. Krüger. *Exploring Rock Climbing in Mixed Reality Environments*, page 1787–1793. Association for Computing Machinery, New York, NY, USA, 2017.

[2] A. S. Williams and F. R. Ortega. Using a 6 degrees of freedom virtual reality input device with an augmented reality headset in a collaborative environment. In *2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, pages 205–209, 2021.