Excel@FIT 2019
http://excel.fit.vutbr.cz

# Multi-user interaction with 3D objects in Augmented Reality

Martin Minárik*, Adam Jurczyk**

**Abstract**
The aim of this paper is to showcase a possible solution that allows multiple people to interact with a single customizable object in a shared augmented reality session.
The work makes use of the new features available in ARKit 2 to share the AR session. It also demonstrates 3D object handling and customization with changes synchronized in real-time to all connected users, as well as resolving conflicts resulting from multiple people interacting with the same single object.
Our goal is to demonstrate the possible use and benefits in presenting work from a designer to a client in real environment with immediate interactive input from the client.

**Keywords:** Augmented Reality — 3D object multi-user interaction — ARKit 2

**Supplementary Material:** N/A

*xminar31@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*
**xjurcz00@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

## 1. Introduction

With the fast-paced evolution of technology, many people living in a modern society have access to a device that enables the user to experience augmented reality. There exist many applications that span multiple fields, from simple games that allow you to interact with mythical dragons, to visual aides that can provide easy access to information for doctors during surgery. Many augmented reality tools can help students with learning about the body by displaying a 3D model that they can view from any angle they would want. However, most of these applications have so far been limited to a single user at a time, which limits their potential usability due to the simple fact that only one person can interact with it on a single device.

Luckily, there has been a lot of progress in the area of AR experience sharing in the recent year or two. Google has introduced Cloud Anchor [1] sharing to their ARCore platform. This service allows users to host an anchor created in their local space via Google Cloud services. However, we found this approach unsuited to our goal, since it requires a live internet connection to Google servers. This could create issues when trying to present in an area with limited or no network connection.

Microsoft HoloLens has also very recently introduced Azure Spatial Anchors [2] as a way to share augmented reality experience. In similar fashion to the ARCore platform, the anchors have to be uploaded to Azure resources and are shared via a web application hosted on Azure. As such they also require a network connection. In comparison to Google though the anchors hosted on Azure do not expire after 24 hours. The service can also be integrated with applications using ARKit or ARCore.

Apple environment on the other hand provides tools to share these anchors without the need for an internet connection(3.1). This allows us to create a smooth experience independent of outside network conditions, since the users' devices handle everything locally.

Our solution is an application that enables multiple users to view the same virtual object, in this case a knight. Each user can interact with the knight and change it's properties such as helmet or weapon or modify the texture of the armor. The changes are shown on every single person's device at real time from their own point of view. This makes it easier to visualize potential ideas to all users during product presentation and also improve the gathering of feedback by allowing the users themselves to make adaptations to the presented model.

The chief challenges that we faced were the act of synchronizing the shared session user-space as well as resolving conflicting user actions when interacting with the same element at the same time. All actions need to be visible with minimal delay everywhere, so any networking messages sent between clients must be simple and without significant overhead. The problems and solutions will be described in the following sections.

## 2. Display and interaction with a 3D model

To display virtual models in 3D we use SceneKit as rendering engine. This comes with limitations since SceneKit supports only few file formats that can represent a 3D object. Supported formats are COLLADA (.dae), Alembic (.abc) and SceneKit's scene file format (.scn). Xcode automatically converts the file to SceneKit's compressed scene format. The compressed file retains its original .dae or .abc extension [3].

The problem is that most models are created in different format and finding model in the right format is difficult. Solution to this is to use 3D modelling tool such as Blender to convert models to the right format. COLLADA is the preferred framework because it is more popular and can display objects formed by a collection of primitives, including triangles, polygons, and spline curves or even a full scene, including lighting, animation, and camera information which is closest to SceneKit's scene file format. It's also important to set the model to it's origin, correctly scale it and optionally delete unsupported nodes as it's not possible to do it later in Xcode.

### 2.1 Using SceneKit

Displaying 3D content in SceneKit [4] is done by creating a scene that contains a hierarchy of the nodes [5] and properties that represent visual elements. The main node is rootNode which defines the coordinate system of the world. Each child attached to rootNode creates its own coordinate system which is relative to node's parent node. A node can be transformed by editing its position, rotation and scale properties or using transform property. The node hierarchy determines spatial and logical structure of the scene. However, determining whether node is 3D object, light or camera is done by attaching objects to nodes. There are three main types of nodes in SceneKit: groundNode, lightNode, cameraNode. All the nodes in our project are using ARKit [6] to get data about the real-world space. Firstly, groundNode displays 2D and 3D objects in scene, in our case displaying a knight in the real-world space. Secondly, lightNode controls shadows and shading of the scene. ARKit can estimate scene lighting based on a captured video frame and change SceneKit's lightning accordingly providing more realistic model. Finally, cameraNode contains information of the viewpoint from which the scene appears. This information is used to rotate the knight to the user who is interacting with it.

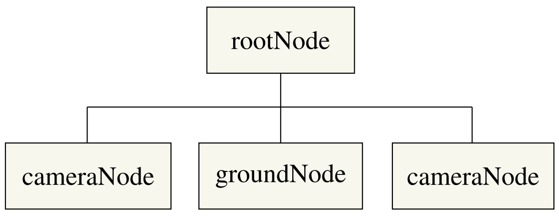

**Figure 1.** Datagram showing node hierarchy in SceneKit.

## 3. Network sharing

The main goal of the network layer is to accurately share information about object position and the changes that occur to it's parts. While there may not be an issue when dealing with a single user changing the object, when multiple people try to do so at once it can create discrepancies and make the whole shared experience unusable, which is not desirable.

In order to deal with these issues, we have decided to use a server-client based approach. One of the devices, usually the one where the session is started, serves as a central hub for all the other peers. Every peer communicates with the host, and the host then distributes the message to the rest of the peers. Because all the communication travels through a central server, it acts as a centralized synchronization unit in

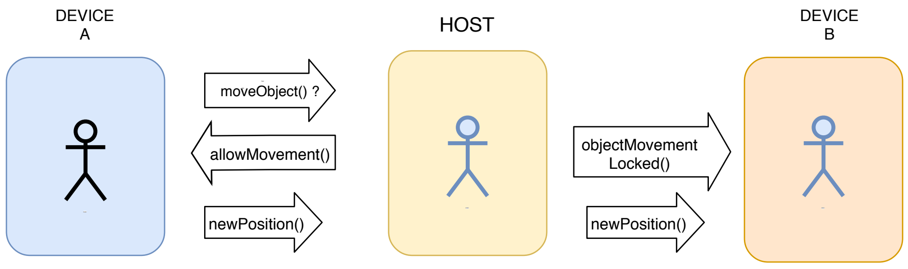

**Figure 2.** User conflict resolution example. User on device A tries to drag the object, which sends a request to the host device. As no one is moving the object, the movement is allowed. Object movement is then locked for all the other users, with the object glowing the color of the action initiator. Device A then transmits new position of the object, which is redistributed to all the other devices.

case there are some discrepancies during object movement. Thanks to that we no longer have to worry about potential differences in individual users' sessions.

When users attempt to modify the same part of the object, e.g. both try to switch the knight's weapon, the server also resolves the conflict and chooses only one of the possible results. The conflict resolution is based on first come first serve basis, meaning the first device to send it's request is chosen. In the case that the operation takes more time (user is dragging the object), the server signals the other peers that an action is in progress, and no one except the action initiator can interact with the object in the given time. This is also shown on the user's device with a glow around the object the color of the given user, or by the greying out of the choice menus.

By indicating the action availability in this manner to all the users, we ensure that everyone always knows what is going on, and thus we prevent user confusion and frustration.

### 3.1 MultipeerConnectivity

MultipeerConnectivity [7] is a native Apple framework that handles communication between devices using underlying networks. It can use infrastructure Wi-Fi, Bluetooth or direct Wi-Fi. The framework also manages the advertisement of a joinable AR sessions as well as its discovery by peers wanting to join. The entire session is managed by an internal object that also stores individual peer ids uniquely identifying the device. Messages are sent using this framework in a binary format, the encoding and decoding of messages implemented using Apple's *Codable* protocol.

## 4. Conclusions

The goal of this paper was to explore and demonstrate the possibility of an augmented reality session with multiple users that can interact with 3D objects placed into the real world. We presented issues we faced and the technologies available in the Apple iOS ecosystem that help us solve them.

We demonstrated a possible implementation of real time interaction with a simple customizable model, as well as outlined a possible solution to handling user action conflicts. While the model we used is simple in it's nature, it is enough to showcase our work. However, it can be swapped for a more appropriate one to suit a different situation.

In the future we would like to explore options to move the hosting of the objects from a local device to a cloud, as it would allow for a more permanent experience without the need to have an initiating presenter.

## Acknowledgements

We would like to thank our supervisor Ing. Vítězslav Beran, PhD. for his help and patience during the creation of our project.

## References

[1] Google. Share ar experiences with cloud anchors. [online], February 2019. `https://developers.google.com/ar/develop/java/cloud-anchors/overview-android`.

[2] Craig Treasure. Share ar experiences with cloud anchors. [online], February 2019. `https://docs.microsoft.com/en-us/azure/spatial-anchors/overview`.

[3] Apple Inc. Scnscenesource. [online]. `https://developer.apple.com/documentation/scenekit/scnscenesource/`.

[4] Apple Inc. Scenekit. [online]. `https://developer.apple.com/documentation/scenekit/`.

[5] Apple Inc. Scnnode. [online]. `https://developer.apple.com/documentation/scenekit/scnnode`.

[6] Apple Inc. Arkit. [online]. https://developer.apple.com/documentation/arkit/.

[7] Apple Inc. Multipeerconnectivity. [online]. https://developer.apple.com/documentation/multipeerconnectivity.