AUTODESK UNIVERSITY

TR123704

Creating Social Presence in AR/VR using Autodesk Forge

Jason Walter
Autodesk

Merten Stroetzel
Autodesk

Learning Objectives

- Understand the value of creating social presence in AR/VR contexts
- Learn about the latest research in creating a sense of virtual presence
- Discover how we used Forge to facilitate collaboration
- Discover how we enabled collaboration in VRED Professional and Stingray

Description

This talk will focus on recent work on how to build immersive experiences with support for collaboration. We will focus on three aspects of collaboration: A) Capture of human (e)motion B) visualization of captured data C) network distribution of data. We will illustrate these aspects using prototypes that we constructed over the past year. And we will demonstrate how these prototypes led to various pilot projects and how they utilize Autodesk Forge.

Speaker(s)

Jason is a principal software developer on the Design & Creation Product team. His focus is on real-time 3D application development focusing on AR/VR. His interests are on intuitive user interfaces in AR/VR and network collaboration. During his tenure at Autodesk, Jason has worked on a number of real-product and projects. Most recently he has focused on real-time network collaboration in AR/VR and helped develop the Future of Making Things demonstration in automotive collaborative VR.

AUTODESK UNIVERSITY

The importance of social presence in AR/VR

In almost every category of AR/VR there is added and essential value to enabling social presence and collaboration.

- **Design** – design reviews are an essential part of product development. Physical prototypes are often built to get a sense of the product and the fewer physical prototypes the faster a product can be developed. Additionally, design reviews are not a single person effort. They are a collaboration of among multiple stakeholders and closing the communication gap can produce better products.[1]
- **Training & Support** – From the assembly to the maintenance of a product or process there is a need to communicate to individuals involved. AR/VR can reduce training times by providing a mechanism to communicate in a safe and controlled environment. Additionally, you can teleport your most qualified staff to a region without physical travel yet provide training.
- **Work Task Simulation** – In some work environments, tasks are complex and access and training in those environments is difficult. Work task simulation allows you to work in a virtual world that is like the environment. For example, machinery to produce semi-conductors must be produced in a clean-room environment; however, access to those clean-rooms for work task training or simulation is difficult. Having access to a virtual clean room for training purposes is essential for providing skill to the necessary individuals.
- **Marketing** – VR has the potential to engage customers in pre-sales activities to learn about a product or solution. Collaboratively engaging that customer in the VR helps communicate the product or solution message
- **Ergonomics** – In factory situation, facilities need to get build to accommodate the need for products development. In an assembly line, you want to know if a pod or station is ergonomically sound for an individual. (1) Likewise this requires feedback from multiple stakeholders.
- **Self-Visualization** – Self-awareness in VR is essential for context. This is natural in the real-world but often lost in VR. Creating virtual presence is important for self-visualization and awareness.

[1] https://haptic.al/virtual-reality-reduces-the-production-time-for-car-manufacturers-9778d7a48733

AUTODESK UNIVERSITY

Latest research on virtual presence

Many companies are dedicating resources to virtual presence. A lot of the recent research has been provided by Oculus/Facebook and included in the Avatar API. Some of the most recent talks and influenced the prototype we built for the Future of Making things includes the following talks.

- **Oculus Virtual Presence Talk -** https://youtu.be/0EZn50XCueI
- **Oculus Maximizing VR Presence** - https://www.youtube.com/watch?v=X6XOwtcscnY
- **Oculus Keynote** - https://youtu.be/NLnk2cCirQM?t=814
- **Secrets to great multiplayer games** - https://www.youtube.com/watch?v=30v4zatNIXc
- **Avatars in AltspaceVR -** https://www.youtube.com/watch?v=iWYgnvJoSUo

We have found that representing individuals as stylized characters increasing the believability and avoids the uncanny valley. The key points learned are:

1. **Speech** – especially spatially correct speech – our main mode of communication is via natural conversation. This needs to be transmitted in VR to provide an immersive experience. And we found that that spatially coherent speech increasing the idea of presence.
2. **1:1 – Tracking –** the movement of the body must be indicative of the perforance
3. **Body representation** – body representation is important for providing self-visualization cues.
4. **Eye contact and gaze** – we use gaze and eye contact as indicators of intent.
5. **Blinking** – Blinking is a natural occurrence adds to the effect of someone “being there”
6. **Lip-sync** – the ability to predict mouth moment
7. **Emoting** – indication of happy, sad, frustrated are visual cues that happen naturally but need a representation virtually.
8. **Arms –** providing awareness of body position and arms is better than just a floating head.

Gesturing is also helpful in communicating to the user. We used a very simple gesturing scheme for communicating in VR.

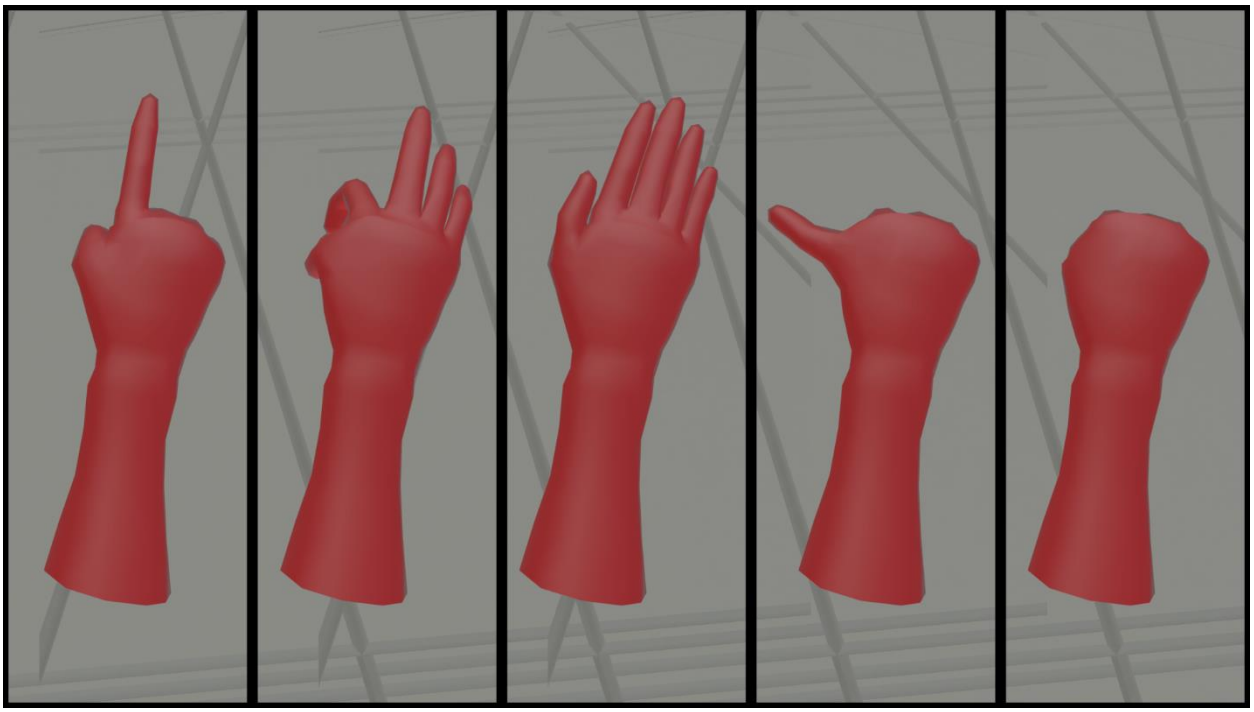

Figure 1: Hand gestures

We choose fixed poses because Autodesk VRED did not support fully articulated hand gestures. The main poses mapped the controller touchpad input:

1. **Finger pointing** – Up position on the touch pad.
2. **OK gesture** – down position on the touch pad.
3. **Hand open** – center position on touch pad.
4. **Thumb Up** – left position on the touch pad.
5. **Fist** – right position on the touch pad.

Controllers are also physically represented so that we know which controllers the users are using. This allowed us to ensure that the participants are using the right tool and we can use the visual hand gestures to point to parts of the controller to communicate how the tool is used.

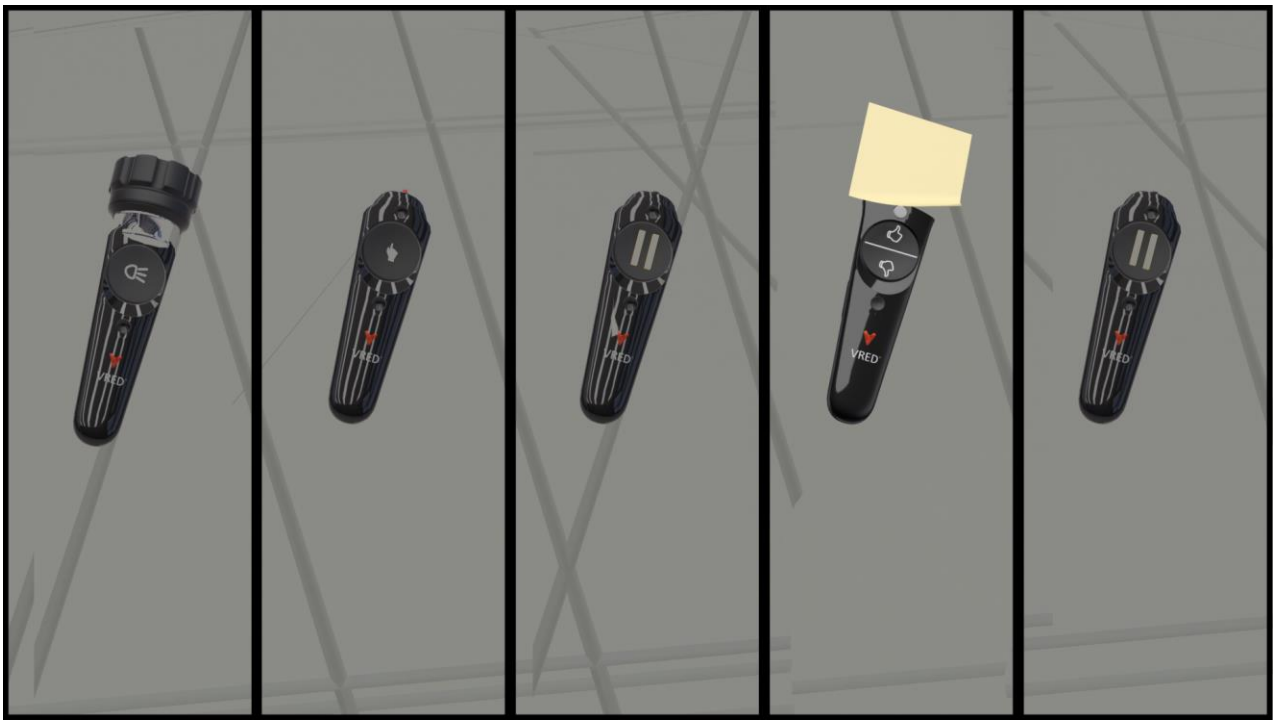

Figure 2: Controllers

We also favored direct interaction versus indirect experiences. Everything had a physical purpose. We found this to allow the user to "understand" the VR environment quicker without having to explain a complex menu. We implemented this in several places within our demonstration.

1. **Opening / closing doors** – users could open and close the doors by using touch
2. **Internal material choices** – users could select material swatches in VR to change the interior color.
3. **Exterior material choices** – the user could touch speed forms that represented the exterior color.
4. **Tool pickup** – tools were acquired by physically touch

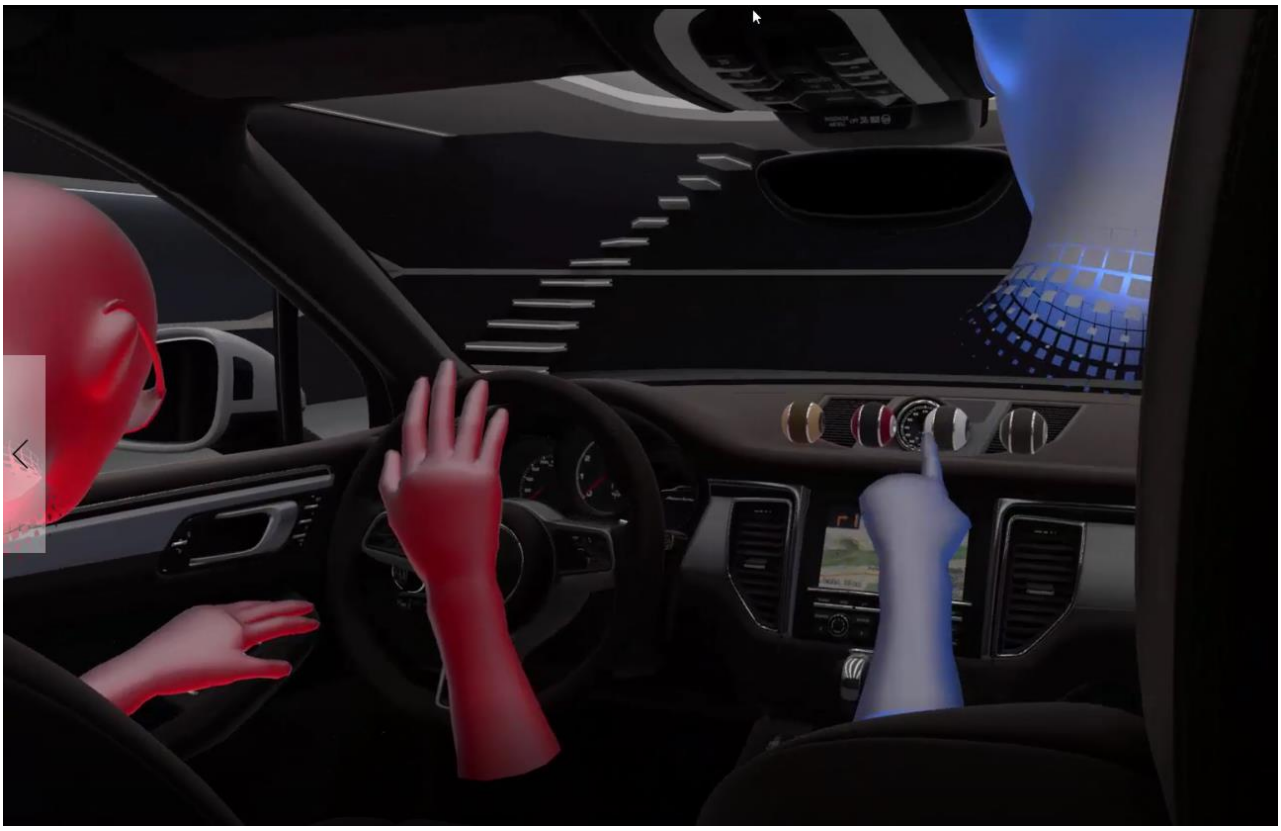

Figure 3: Direct interaction to change fabric. The balls represent different interior material choices.

Network communication

To facilitate efficient communication between multiple Autodesk VRED Professional, we built a multi-layer application interface. This evolved over multiple attempts to do efficient communication between multiple users of Autodesk VRED Professional.

Initial versions leveraged a direct communication protocol that made it difficult to scale beyond two collaborators. This direct communication happened directly within Autodesk VRED. The final version of the app involved a separate application that handle the data transport and ran as a separate process. This allow us to asynchronously handle the data transportation.

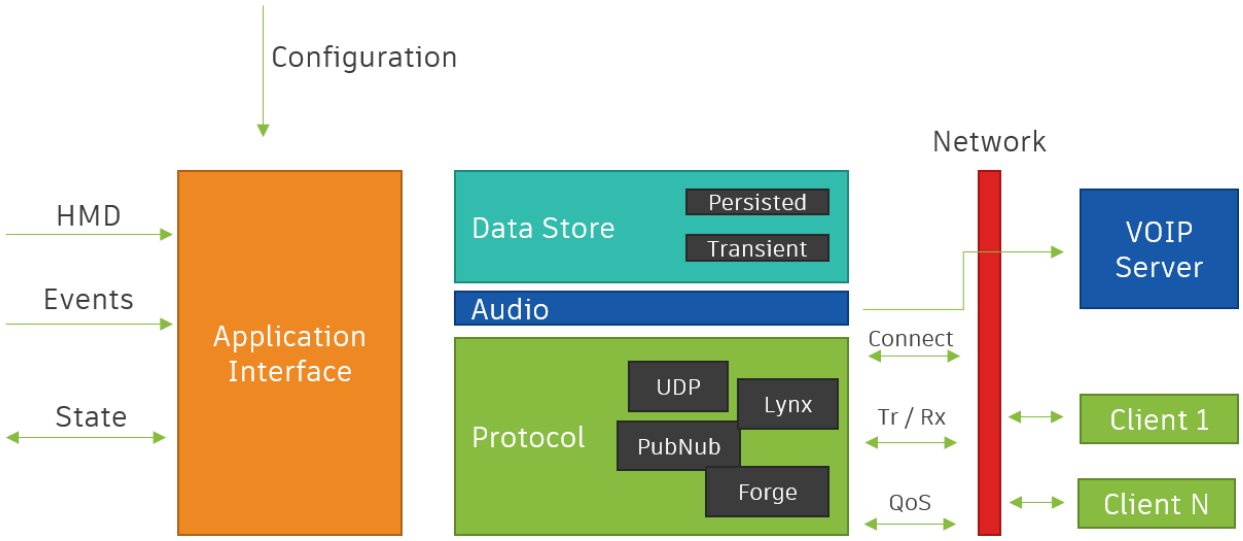

Figure 4: Network communication stack

We choose this communication design for the following reasons:

1. Abstracts the communication so that the app does not need to worry about the data transport.
2. Allows us to optimize the transport of data between apps.
3. Provides a way for different apps to communicate.
4. Flexibility in choice over data protocol (e.g. UDP, TCP, HTTP, etc.), as well as allow different communication channels simultaneously.

Each layer had a designated purpose.

1. **Application Interface –** This is how the application
2. **Data Storage** – We acknowledge two types of data in our data model. These two types of data is based on the fact that network transport is unreliable and there is a cost to make sure that data is reliably sent. Protocols like TCP guarantee delivery but they are

very expensive in terms of performance cost. Protocols like UDP are unreliable but have less cost in terms of transmission. [2] So, to support protocols like UDP or any other unreliable transport we need to know which messages are important.

a. **Transient –** Transient data is anything that is "not important". This is controller, HMD, tool values. This is data that is sent every data frame and if a piece of data is not received/sent it doesn't matter because eventually we will send another frame of data. This is similar to how some real-time stream protocols work.[3] The key point is that this data is constantly changing and a missed datagram is not the "end-of-the-world".
b. **Persisted –** This is data that we can't lose and must be shipped along the transport lines and must have some guarantee that it will arrive to other users. This is used for any state changes. For example, the environment or model properties change.

3. **Audio Layer** – We used mumble[4] for audio transport. This layer of the communication stack is responsible for sending positional information. This layer's responsibility is to attach the collaborator to the user in a mumble chat client, as well as, send positional data.
4. **Protocol Layer –** This determines how the data is transported. We support multiple transports simultaneously because we assume the data layer is heterogenous. This also allows the app to not worry about how the data gets to the other end. The application interface provides a way for the user to send the data and the importance of the data and get updates about any collaborators. But it doesn't need to know about how to send the data to the collaborators.
5. **Network Transport –** this is the physical transport of the data. Once of the app sends the data over the wire it is left to the packet switch network. Because of the unreliable nature of the transport anything that is important to send (or sits on the persisted data queue) is only sent using a reliable data protocol (TCP, HTTP, etc). The exception is the UDP data channel running on the same subnet. In controlled situations, the router will not drop UDP packets and can be treated as reliable in controlled environments. There are certain careful

Providing an abstraction between the network communication and the actual application allowed us to support other applications. Beyond the future of making things demo we used the communication app to link Stingray sessions as well as link VRED applications. This was mainly driven by limitations in the Python interface in VRED but ultimately allowed multi-user collaboration.

We used Autodesk Forge as a notification API for collaboration. This is an internal API that is not publicly available. For general message passing across geos, we used PubNub which is used by game and non-game environments for reliable and fast message passing.

[2] https://en.wikipedia.org/wiki/User_Datagram_Protocol
[3] https://en.wikipedia.org/wiki/Real_Time_Streaming_Protocol
[4] https://wiki.mumble.info/wiki/Main_Page

Data Messages

In VRED and Stingray, we used JSON objects as a way to pack and unpack data. JSON data was freeform and each application decided on how they would implement their messages. In VRED, we have a simple message scheme like the following:

```
{
   "data": {
      "c0": {rotation & transform}
      "c1": {rotation & transform}
      "head": {rotation & transform}
      "state": [
         {left ToolState},
         {right toolstate}
      ]
   },
   “id”: “unique id”
   "type":"collab"
}
```

The type of message indicates if the message is persistent or transient. In this case, “collab” is a transient state know by the system. The data section is application specific. Therefore, the collaboration app has very little knowledge about what data is being sent. All packets would be tagged with the id that identifies the collaborator uniquely on the network. This way we can know who initiated the message and attach that unique id with a specific avatar in the 3d scene.

For persistent the data it easy even more open:

```
{
   "data": {
      message specific
   }
   “id”: “unique id”
   "type": "event"
}
```

The event type “event” is persistent data storage and must be received by the participants.

Note that each protocol must hold its own copy of the persistent and transient data queues because each protocol can be reliable and unreliable. Therefore, for each protocol that we store separate data queues that are consumed at the rate specified by the protocol.

The following is an example of a post-it note event that is sent. Post-it notes are like annotations that can be added to the scene, and they are only understood by VRED implementation of the collaboration app.

```
{
   "data": {
      "name": "postit",
      "id" : positId,
      "pos": position,
      "rot": rotation,
      "type": ptype
   }
   "type": "event"
}
```

In a post-it note; the data contains the information that the app can use to unpack. In the data section of the JSON you have the name a name that is identify it with the application, an id that uniquely identifies the post-it note, the position, and the rotation of the note, and the type of note that we are sending. We tell the collaboration app that this is important data by classifying it as “event”

AUTODESK UNIVERSITY

Collaboration Configuration

The collaboration tools were configured by a JSON configuration script that contained all of the application specific information. Since this was developed as an in-house prototype, a UI was never developed.

```
{
    "serverPort"            : 8771,
    "avatar"                : "luigi",
    "application"           : {
        "name": "vred",
        "initVRMenus" : false,
        "args":{
            "port" : 8888
        },
        "samplingRate"      : 0.02
    },
    "protocols"             : [
        { "name" : "udp",
          "port" : 9772,
          "peers" : [
              { "host" : "192.168.1.17",
                "port" : 9772 }
          ],
          "publishRate" : 0.02 }
    ],
    "channelName"           : "vrCollabRefactor"
}
~
~
```

Here you can see that users could specify their avatar and the communication ports. The protocols section indicates which type of communication to use and the channel name would use to differentiate different channels of communication when relayed over the PubNub network.