Building Social Virtual Reality with Open Source Software

// Liv Erickson
@ MissLiviRose

https://livi.link/WeRiseTerms

Hey everyone! I want to start off by saying a huge thanks to the We Rise organizers and for all of you for attending – this has been an incredible conference so far, and I'm so excited to share a topic that, quite frankly, I'm still a little bit obsessed with! So if you're looking to learn more about building social virtual reality experiences using open source software, you're in the right place! Before we dive into things, I want to point out this <link / QR code> - it's a page that I put together that has definitions for a lot of the terms that I'll be using in this talk. I'll try my best to cover them as I go, but if I get excited and forget, you can use that as a reference if anything is new to you! Now with that said, let's dive in! My name is Liv, but if you're looking for me around the internet or the metaverse, I go by the very secretive code name 'MissLiviRose'. Most of my friends know me as the Content Engineering Lead at High Fidelity, but I am, among many other things, an actual Jedi Knight. I have photographic evidence!

Okay, so, you can't actually tell that I'm a Jedi Knight in this photo. This is a picture that was taken the very first time that I ever tried virtual reality, back in 2014. I am a huge Star Wars fan, so being able to experience the world of one of my favorite fictional universes immediately had me sold on the power of virtual reality. It might not seem like it, but this was the turning point in my career that led me to turn that obsession for battling drones with lightsabers into the job that I have today, where I get to stand in front of you all today and talk about building for VR. I walked away from a meetup group knowing that I had just tried this mind-blowing experience, so I took to the internet when I got home that night, ordered my own Oculus Rift developer kit, and Googled 'How to be a VR developer'.

Google was a helpful tutor when I was starting out! I was introduced to the Unity game engine and found a few introductory tutorials on how to get started building a 3D environment that I could view in my Oculus Rift. I also discovered a few blog posts from engineers at Google about an experimental Chrome feature that they were working on with Mozilla called the ‘’WebVR API’, which would allow developers to create virtual reality experiences right in the browser. Every night after my full-time job ended, I would go home and piece together little snippets of code from sample tutorials that I found and those blog posts. A GitHub project that I found for the WebVR API helped me build my first site that supported my Rift.

About four months after starting my adventures into the world of VR development, I accepted a job at Microsoft as a developer evangelist, where my team was given the mission: ‘Build cool things, and then teach others how to do the same!’ I had an incredible opportunity to work full-time in exploring the rapidly-expanding virtual reality ecosystem, and building up a community around what I was discovering.

Resource Links:
https://github.com/borismus/webvr-boilerplate

https://unity3d.com/learn

My Early VR Projects

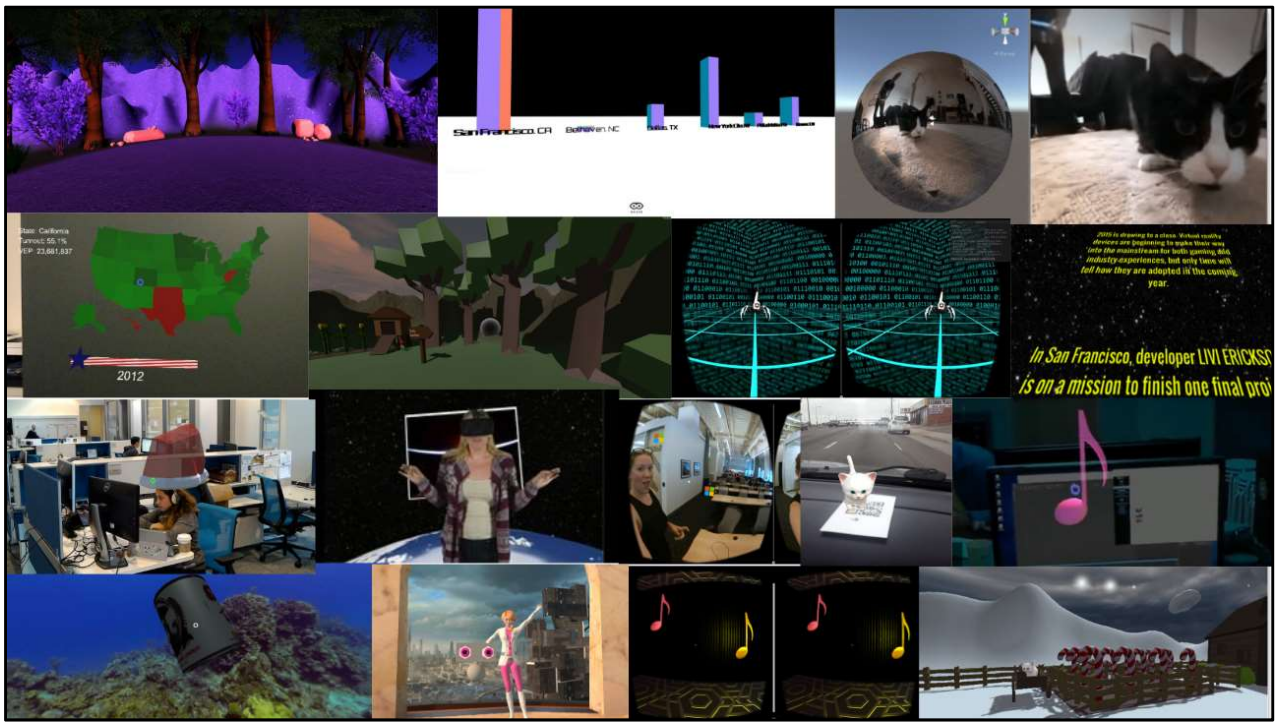
Open Source for Immersive Technologies

Using JavaScript to write Social VR experiences in High Fidelity

In This Talk

Today, I'm going to talk about building VR projects and give you an overview of what I've worked on and some of the tools that are available for building virtual reality experiences, specifically focusing on _social_ VR, then walk through some of the projects that I've worked on to

Early VR Projects



And I made a _lot_ of virtual and augmented projects. Many of them were prototypes for various platforms and built with a variety of technologies, but each of them taught me more and more about the way that these experiences could be developed, and the potential of the platforms.

Like this website, where you could float over the planet and view photos taken from the International Space Station. I built this using Three.js and the WebVR API, which I'll cover in more depth in just a few minutes.

Resource Links:
https://github.com/misslivirose/webvrspace

And this mobile VR game with robots for Google Cardboard, that used gaze tracking for aiming. I built this in Unity during a hackathon weekend.

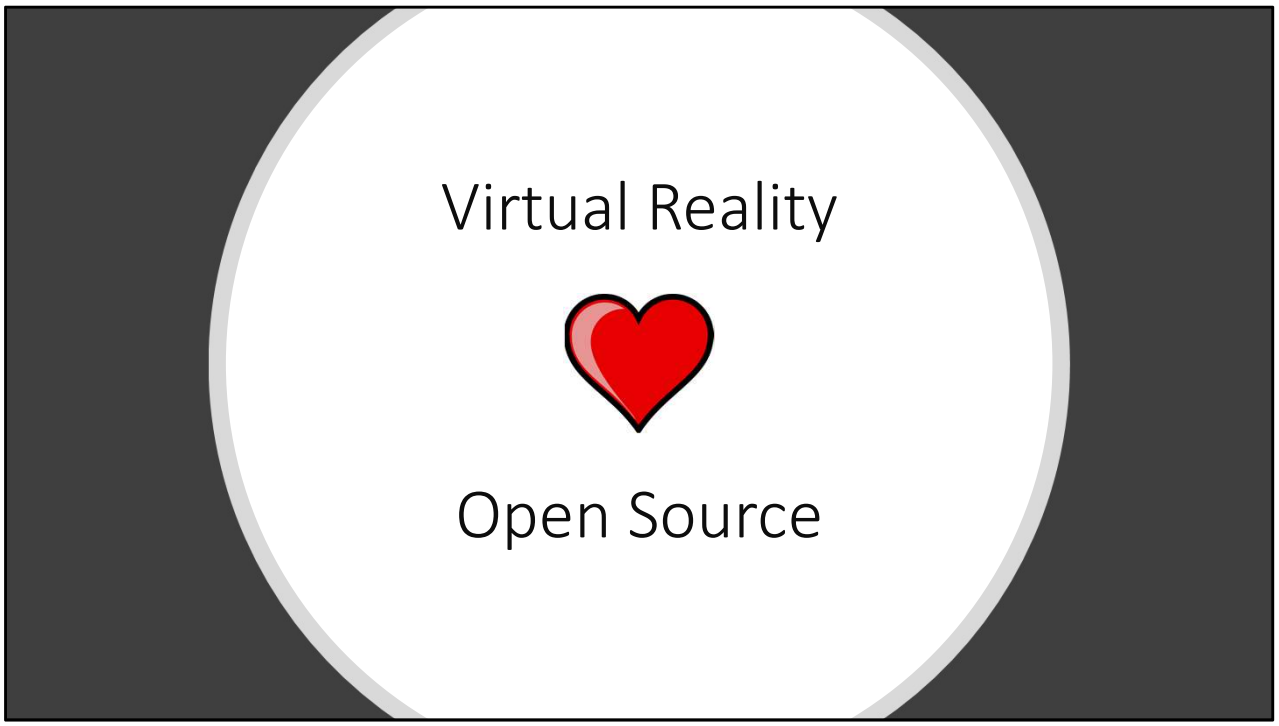
And this interactive HoloLens application for visualizing historic voter turnout data, which was also built using Unity. I learned a lot from these projects, but, as I was building all of these things, there was something missing: The other people! One of the most powerful aspects of immersive technologies is in how it brings people together, which was something that I knew I wanted to explore, and incorporate into my development. So…

Resource Links:
https://www.youtube.com/watch?v=hBxcGTYSvDw&feature=youtu.be&a=

A year and a half ago, I joined High Fidelity as the engineering lead for their social VR content team! High Fidelity is an open source social VR platform that enables developers and content creators to build and host their own 3D environments. Users are represented as their own avatar (you saw mine at the beginning of the talk!) and can travel around different worlds that are created by each other. If you decide to host your own server, you can immediately take advantage of a fully networked and connected metaverse full and focus solely on the content. And for the last year and a half, my job has been to design and implement new content and worlds that users can take advantage of.

Resource Link:
https://github.com/highfidelity/hifi
https://github.com/highfidelity/hifi-content
highfidelity.com

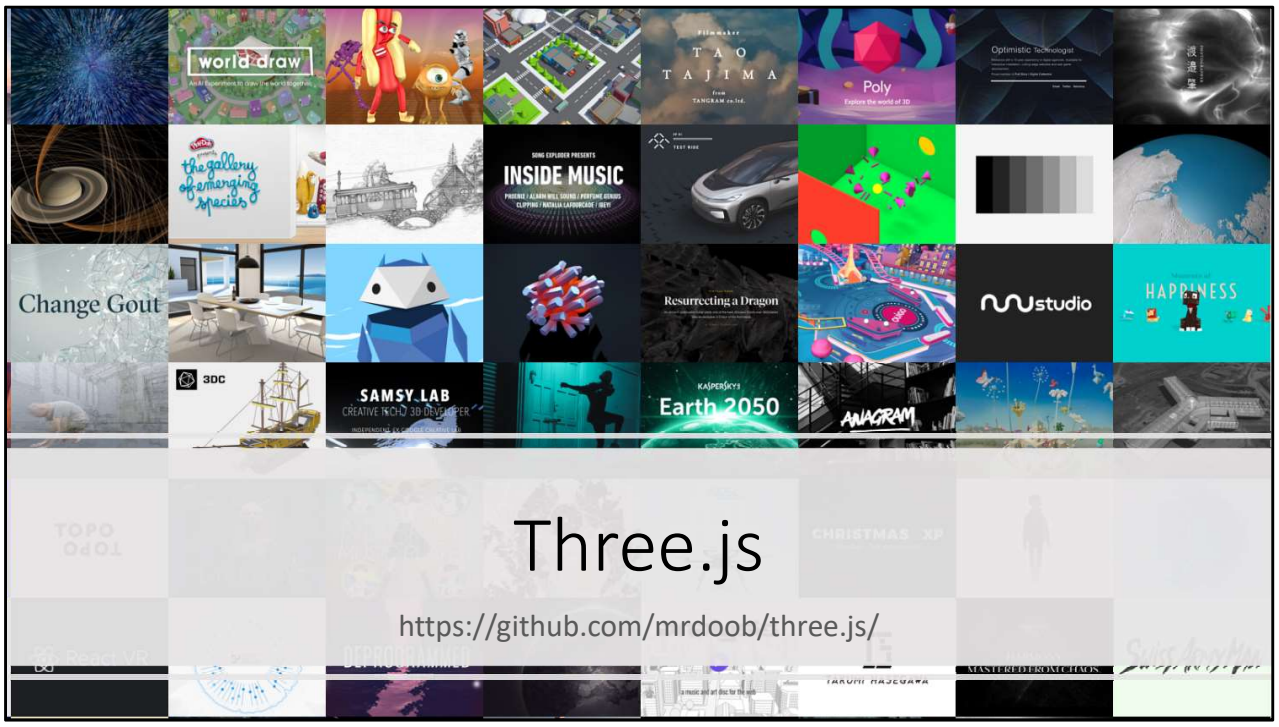
Open Source for Immersive Tech



The virtual reality open source community is awesome. Across the entire industry, creators are so generous with sharing their experiences as we work together to create this new computing paradigm, and that's reflected in the type of projects that exist for creating virtual and augmented reality experiences. Immersive computing is opening up all these new ways to think about technology and how we as humans interact with it, and it's incredible to be part of that. The types of open source projects across the virtual and augmented reality industries are truly pushing the boundaries about what is possible to do with this generation of HMDs.

There are a number of browser-based, open source web frameworks that are being developed to enable immersive content to be built and experienced directly within the browser. These take advantage of existing web frameworks and technologies, and allows creators to integrate those services and libraries into 3D worlds. Mobile browsers, Chrome, and Firefox all support virtual and augmented reality applications through the WebXR (formerly WebVR) API. On mobile, this uses your phone's accelerometer, cameras, and other sensors to understand the world around you and layer on digital content. On desktop computers, these APIs allow developers to create websites that can use VR hardware to display content in an immersive way.

Resource Links:
https://github.com/immersive-web/webxr-polyfill

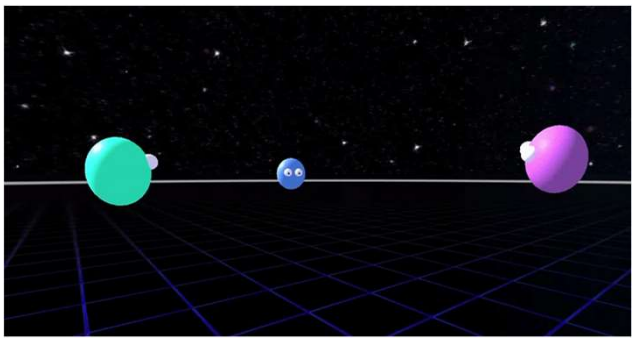


At the core of many of these WebXR libraries is the Three.JS library, which provides an extensive framework for developing three dimensional scenes for WebVR applications using a WebGL context on your browser. The general steps for setting up a scene in Three JS includes creating a camera position, adding models or other entities to your scene, and appending the renderer to your HTML document.

Resource Links:
http://threejs.org/
https://github.com/mrdoob/three.js/

One of the open source frameworks built on top of Three.js that enables rapid development of WebVR applications and experiences is A-Frame, a project spearheaded by Mozilla that enables developers to use a markup-style language for creating immersive content. A-Frame users an entity-component system to define elements within a scene, and uses their 'a-dash' tags to characterize items and give them various properties.

Resource Links:
https://aframe.io/
https://livierickson.com/blog/just-avr-show-building-the-vr-web-with-a-frame/
http://twitter.com/aframevr
https://github.com/aframevr/aframe

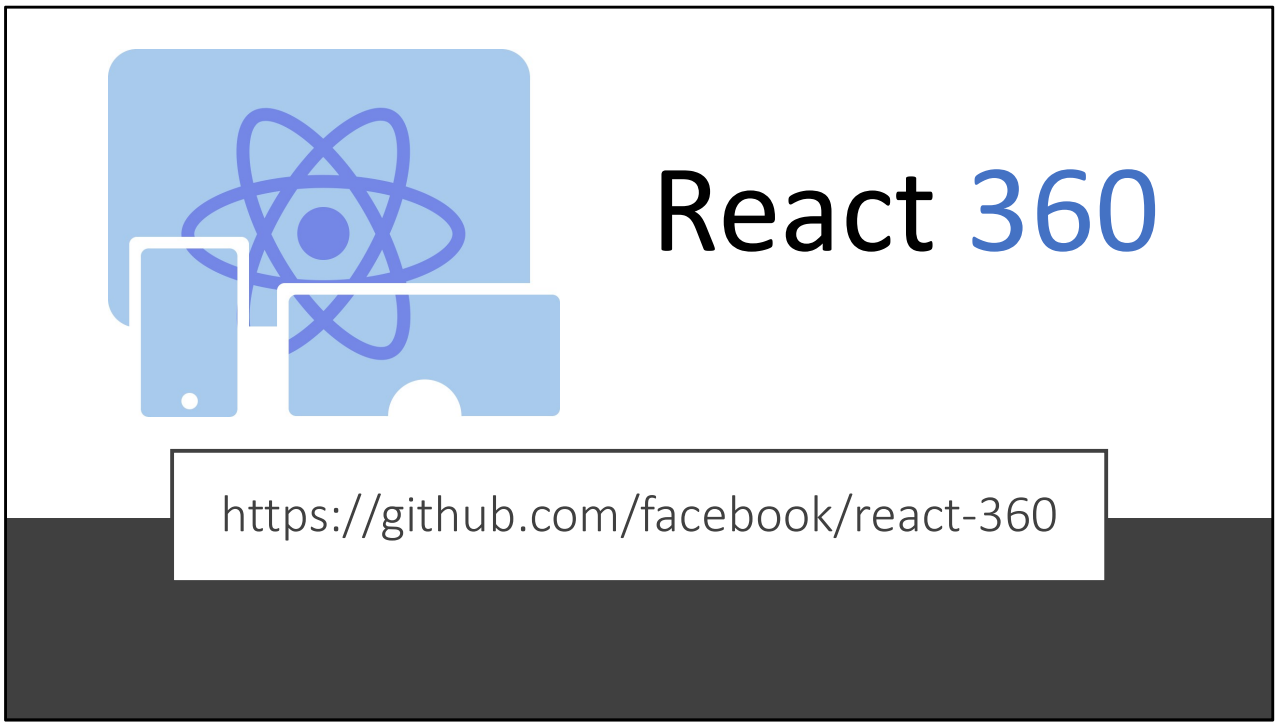


Networked A-Frame

https://github.com/networked-aframe/networked-aframe

The networked A-Frame project builds on top of A-Frame to synchronize component changes within a scene across different clients to enable multi-user apps and games.

Resource Links:
https://github.com/networked-aframe/networked-aframe



React 360 is a Node.js based open source framework from Facebook that provides a toolkit based off of their React libraries for creating a variety of different types of 360 and 3D content for the web.

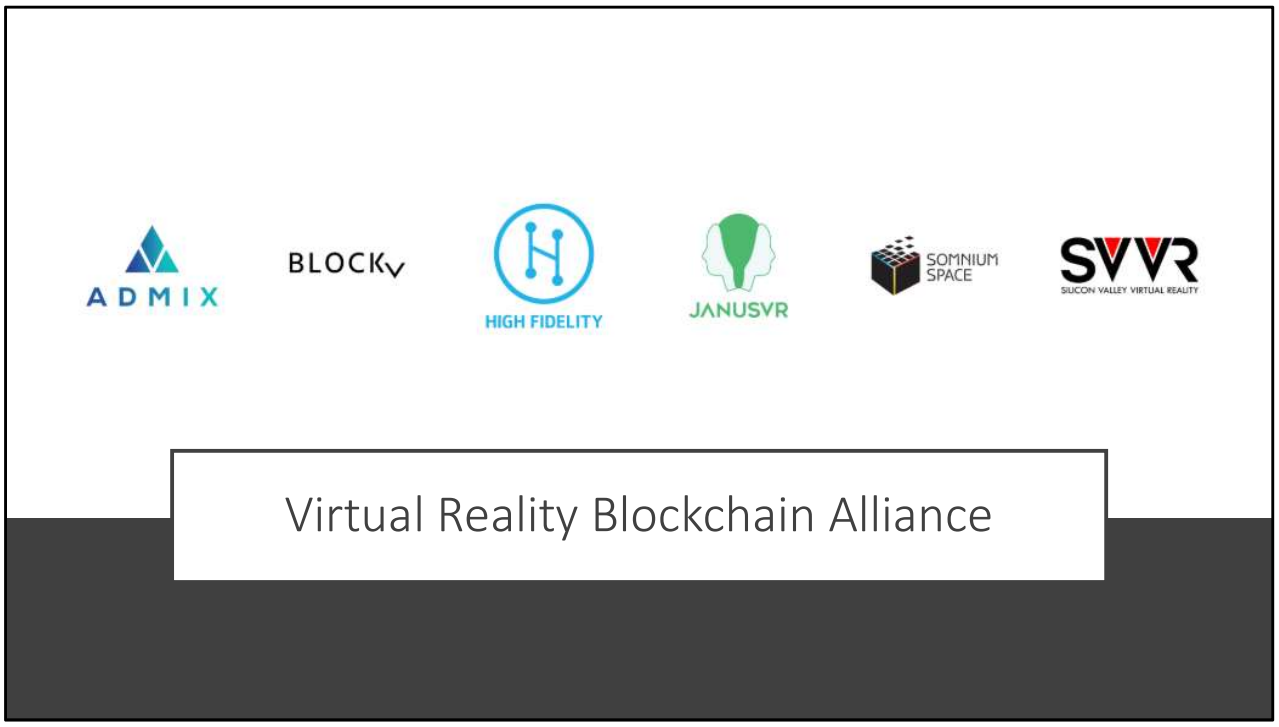
Resource Links:
https://github.com/facebook/react-360

There are also open source projects that build on top of game engines to allow developers to contribute to plugins that can be used in their applications and games. The Virtual Reality Tool Kit (VRTK) supports a number of different virtual reality headsets across desktop and mobile devices, and includes developer support for locomotion, spatial user interfaces, and game physics.

Resource Links
https://vrtoolkit.readme.io/
https://github.com/thestonefox/VRTK
https://assetstore.unity.com/packages/tools/vrtk-virtual-reality-toolkit-vr-toolkit-64131

Beyond the software, there are even virtual reality headsets that are open source! Partners within the Open Source VR movement, in addition to software, also includes schematics for hardware developer kits to break down the components of the electronics for headsets to enable hardware developers to build head-mounted displays.

Resource Links:
http://www.osvr.org/
https://github.com/OSVR/OSVR-HDK



Finally, there's a growing number of emerging platforms that have also embraced open source technologies to implement social world building software and ecosystems from the ground up. In addition to the source code for several of these platforms being open source, many companies are also aiming to create a collaborative framework around portable identity and global interoperability. These platforms enable users to form virtual identities, build and maintain their own digital worlds, and participate in blockchain-driven economies: even between different applications. One example of this is the Virtual Reality Blockchain Alliance, which looks to build a set of global standards for interoperability, security, and protection for users in virtual spaces.

Resource Links:
https://www.vrblockchainalliance.org/
http://highfidelity.com/
https://janusvr.com/
http://www.somniumspace.com/

Building Social VR Experiences

So now, I'm going to shift gears and talk a little bit more in depth about building out some of the content for these virtual worlds. These examples are written using our stack at High Fidelity for our platform, which uses basic JavaScript that runs on an EMCAScript engine built using Qt and C++. Before we get too far into the details, I'm going to talk just briefly on the client-server architecture that we use in creating content designed for use in social VR scenarios.

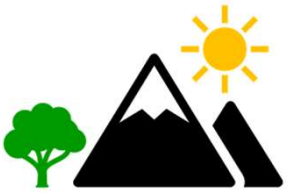
Resource Links
https://doc.qt.io/qt-5/qtscript-index.html

Clients

- Manage user's account
- Broadcast avatar and identity information to server
- Runs code to handle how a user interacts with items the virtual world
- Requests ownership to drive physics behaviors when manipulating an object
- Handle graphics simulation and rendering

Server

- Hosts the content that makes up a world environment via entities
- Mixes client audio and avatar data for a 'single truth' state for everyone connected
- Runs code to handle a synchronized state for an item in the virtual world

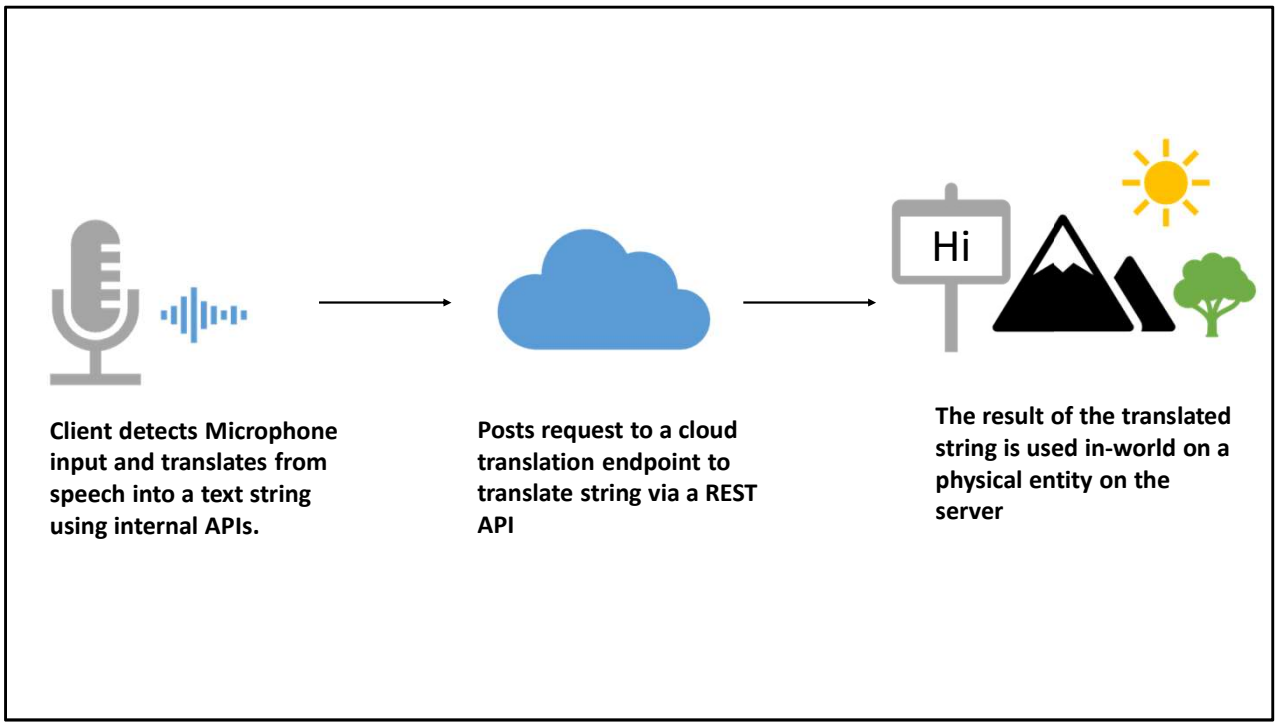


At a high level, the client-server architecture that we use for making social experiences is important to understand given that various parts of your system will be happening for each individual users, and rely on the server component to drive the shared space state across everyone who is connected. This means, with the content that you're building, you'll need to think about your 3D space (and the objects within that system) in the context of what will be run exclusively on a client, what in the world space should be handled by the objects in world, and at what times, and what components should run exclusively on the server. Social VR apps are much like today's existing web systems: and one challenge is considering how actions that correspond to web client-server applications map onto their virtual world counterparts in distributed environments.

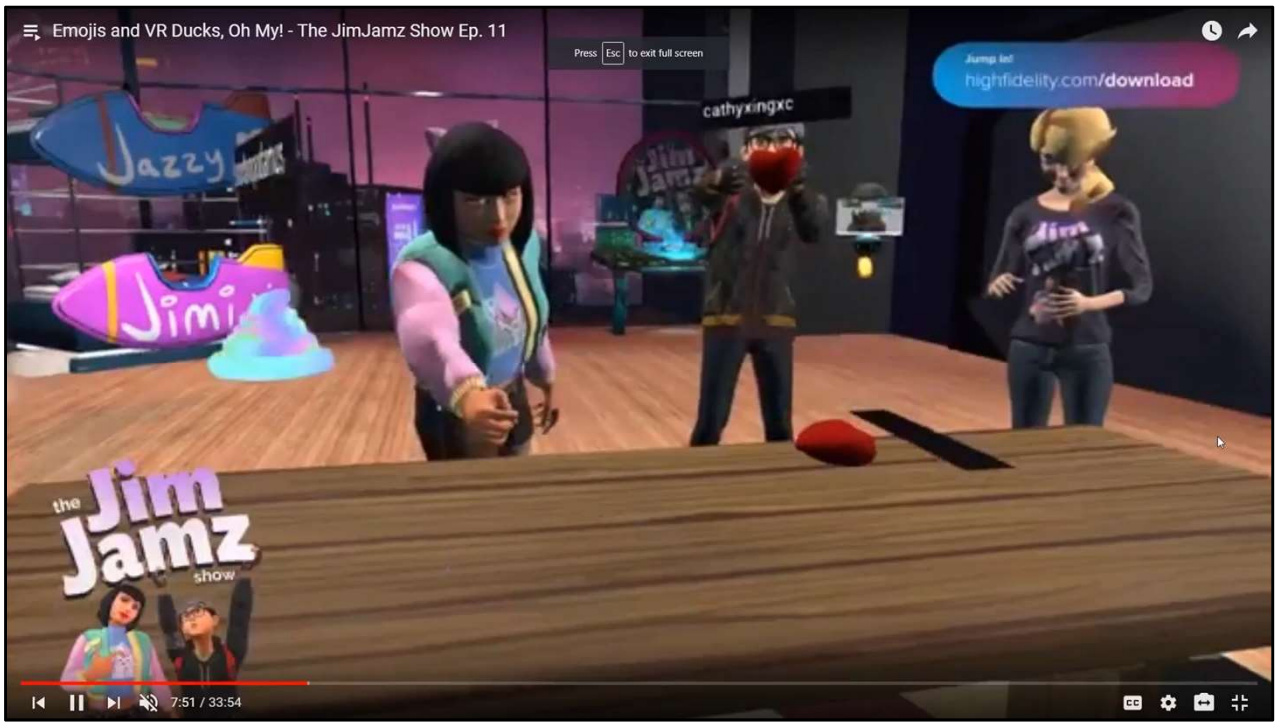
So what kind of stuff have we built?

Voice to Text Translation

One of my earliest project prototypes was an experiment in how we could create a solution for real-time speech to text translation, which is useful in a virtual world where users may not be communicating in the same languages. For this project, I used Microsoft's Cognitive Services API and Google Cloud Speech to Text to pick up my microphone stream while I was in-world and translate it into another language, which displayed on a text screen in front of me.

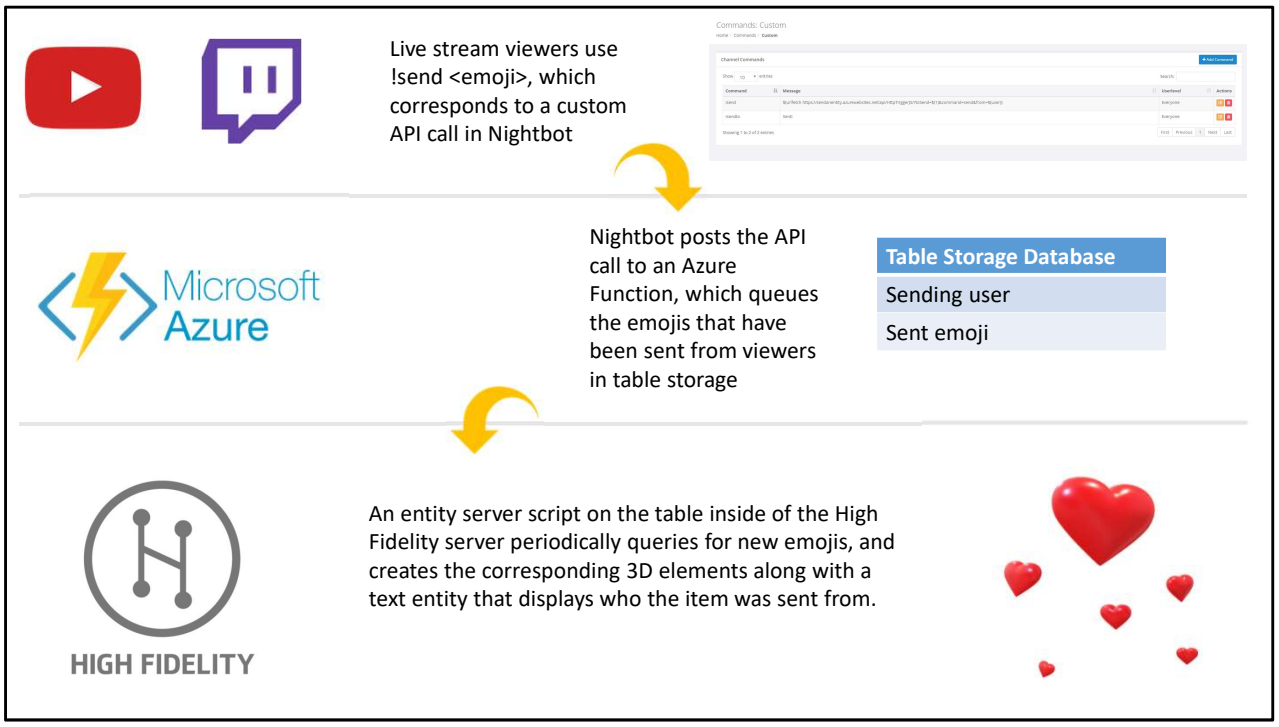

Hi
Client detects Microphone input and translates from speech into a text string using internal APIs.
Posts request to a cloud translation endpoint to translate string via a REST API
The result of the translated string is used in-world on a physical entity on the server

“Outside-In” Communication

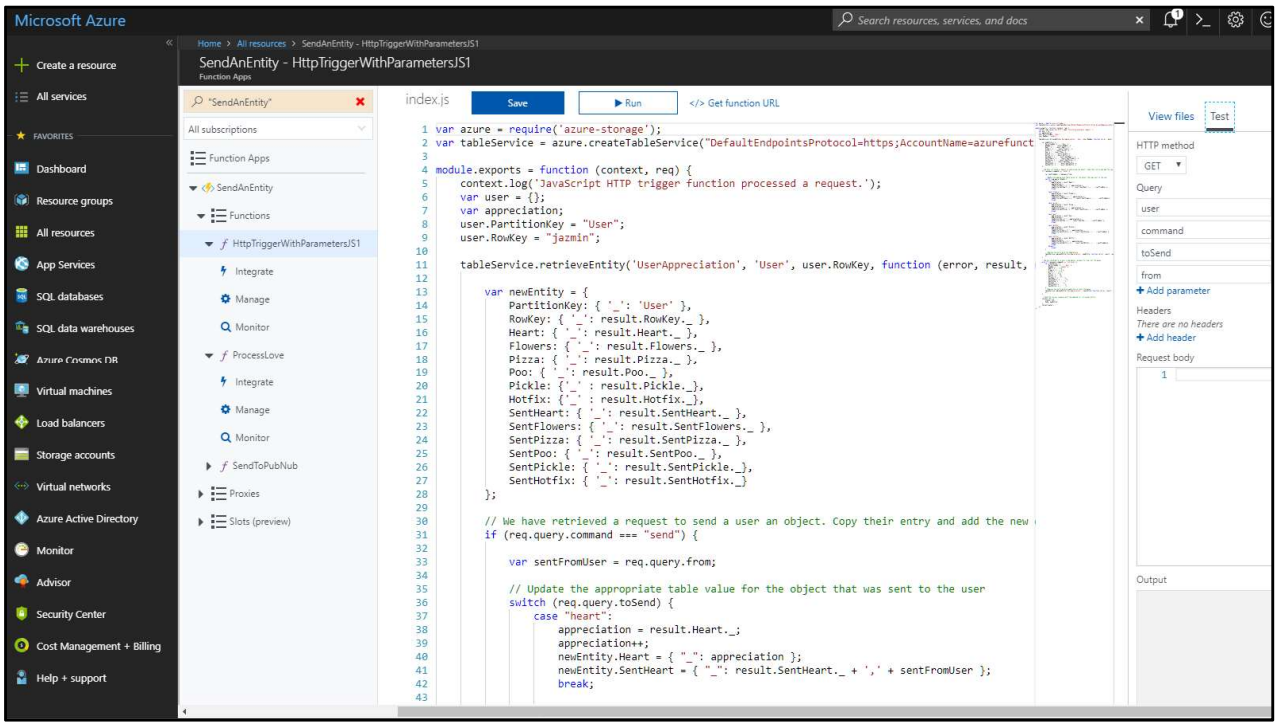


Another element to shared social VR is the ability to create asymmetric experiences that can span across not just devices, but completely different interaction models. One of the examples that I worked on last year was an “outside-in” method to support live performances in VR by giving viewers on a livestream the ability to send objects to people in-world. This is a clip from the JimJamz show, which was a weekly show where two of my coworkers did a live show in-VR to build an apartment together and showcase their 3D art processes.

Resource Links:
https://www.youtube.com/watch?v=iw99SlNwO-A&list=PLoe9GsfO1mjms6e8dtsWUc787QliYRfce



Resource Links:
https://beta.nightbot.tv/
https://azure.microsoft.com/en-us/services/functions/

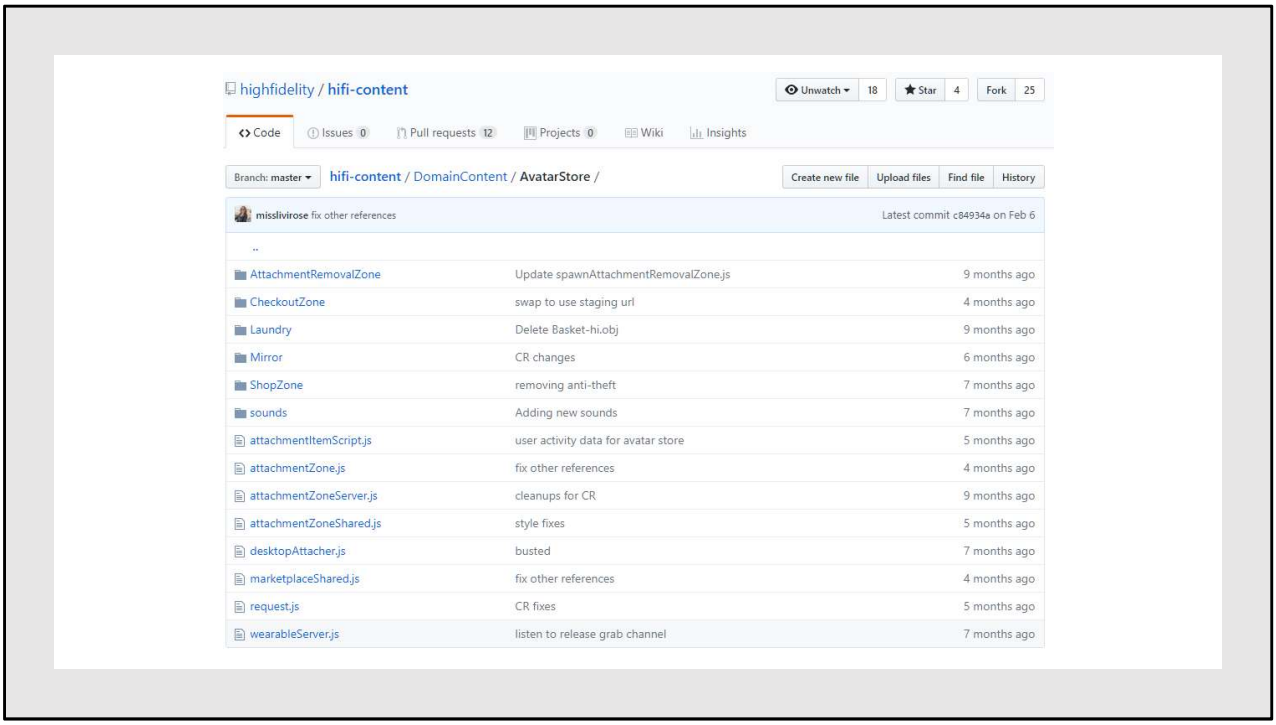


To build the JimJamz show emoji-system, I set up custom commands using the Nightbot API, which was a cross-platform scripting API for livestreaming broadcasts to implement their own custom functionality. I set up the Nightbot integration to post to an Azure Functions API that I wrote using Node.js, which parsed through the command syntax to store the name of the sender and type of emoji in table storage. From within High Fidelity, a server script set up on a table within the show environment would query the database for new emojis and use the information returned to create the specific objects in the environment.

Avatar Island

One of the biggest projects that I've worked on was the launch of our 'Avatar Island' shopping center last fall, which was created to showcase content made by artists that could be purchased when we launched our commerce platform. Avatar Island is a domain where users can choose avatars and accessories and shop socially before picking out and buying virtual goods.

Resource Links:
https://blog.highfidelity.com/building-a-social-shopping-vr-experience-in-high-fidelity-2164e37599c8



We designed Avatar Island as the commerce functionality was being implemented on the platform side, and developing the content involved a number of different prototypes as we figured out how to create a solution that was fun, engaging, and social for users would be connecting together in a number of different ways. We had to design for users that would be using a standard keyboard and mouse, viewing the content on their monitor, as well as for users who would be using hand-tracked controllers and head mounted displays. As it turns out, there are a lot of things about accessorizing that we take for granted – like being able to feel the back of our head when we put on a hat – and working within a virtual platform means contending with things like physics ownership and non-human personas. Ultimately, we designed a two part approach to how users could try on avatar accessories and check out. We also had to implement server protections prevent users from having the ability to create whatever they wanted in the public space or edit what we had in there, and implement mirrors.

Resource Links:
http://github.com/highfidelity/hifi-content/DomainContent/AvatarStore

```
releaseGrab: function(entityID, args) {
  var hand = args[0];
  var properties = Entities.getEntityProperties(entityID, ['parentID', 'userData', 'position']);
  ... // NOT SHOWN: inform parent that item has been removed
  var userData = properties.userData;
  var position = properties.position;
  var attachmentData = JSON.parse(userData).Attachment;
  isAttached = attachmentData.attached;
  if (!isAttached) {
    _supportedJoints.forEach(function(joint) {
    var jointPosition = MyAvatar.getJointPosition(joint);
    if (Vec3.distance(position, jointPosition) <= ATTACH_DISTANCE) {
      ... // NOT SHOWN: Avoid attaching to hands inadvertantly
      var newEntityProperties = Entities.getEntityProperties(_entityID, 'userData');
      touchJSONUserData(newEntityProperties, function(userData) {
        userData.Attachment.attached = true; });
        Entities.editEntity(_entityID, {
        parentID: MyAvatar.sessionUUID,
        parentJointIndex: MyAvatar.getJointIndex(joint),
        userData: newEntityProperties.userData,
        lifetime: -1
      });
      if (ATTACH_SOUND.downloaded) {Audio.playSound(ATTACH_SOUND, { position: MyAvatar.position,
                                    volume: AUDIO_VOLUME_LEVEL, localOnly: true });}
      Controller.triggerHapticPulse(TRIGGER_INTENSITY, TRIGGER_TIME, hand); }});
   } else if (isAttached) {
   ... // NOT SHOWN: Invert above logic to check for removal
}
```

At any given time in the store, there is one trial copy available on the shelf for a user to grab and try on. We made a few design decisions for the trial copies based on the available feature set for entities in High Fidelity:
The trial items are almost identical copies of their parent store copy with regards to their physical properties, but have different scripted behaviors that enable them to be tried on. An AttachmentItemScript.js defines all of the logic required to handle removal from the shelf, being tried on, and being removed from an avatar.

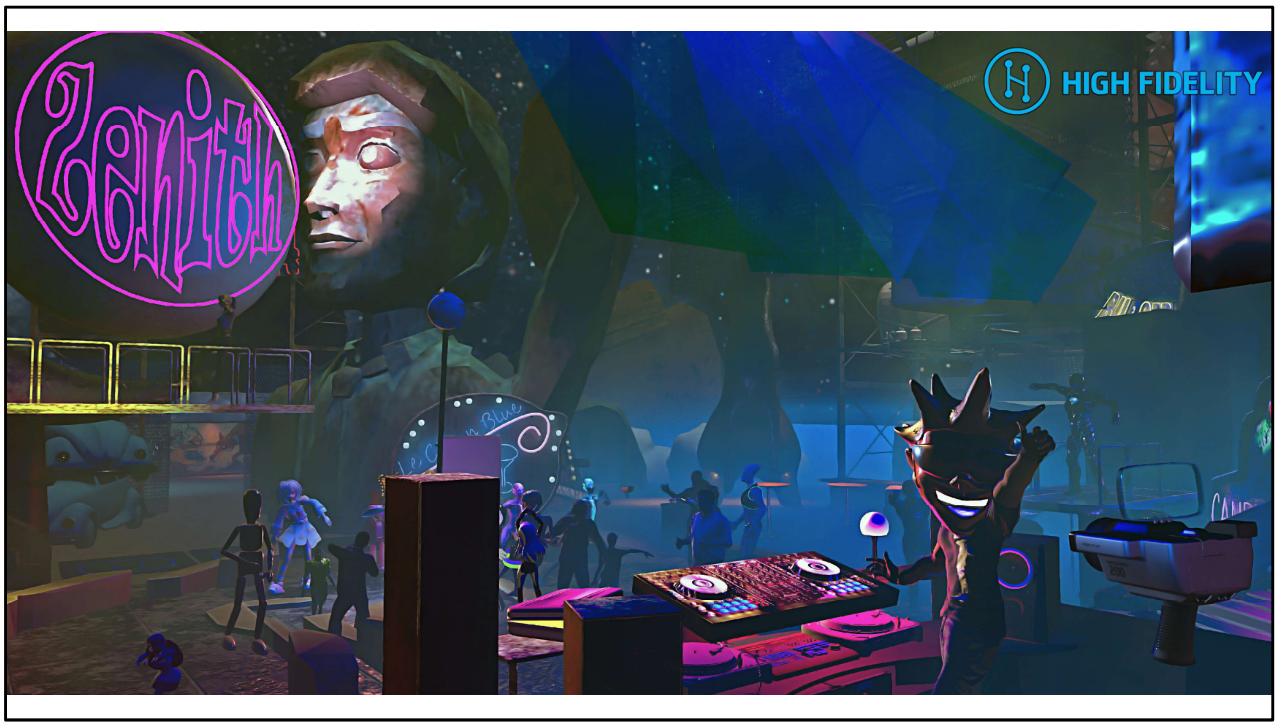
Trial copies are spawned in the exact same location as their parent, and are created as invisible copies to avoid z-index fighting with their parent model. Because entity server scripts do not have any understanding of the physics engine, the store copy does not respond to grab events from user hand controllers, so startNearGrab events in the engine are detected instead by the invisible trial entity. When the attachmentItemScript code detects that a user has grabbed the store copy, it makes the child visible and the user seamlessly experiences the ability to grab something they want to try on.

Each trial entity, when created, has the parentID property set to the store item entityID that spawned it. These entityIDs are unique identifiers that reference a

specific instantiation of an entity within a domain. Within the attachment item script on the child entity, we can reference the parentID to determine the state of a given item,and whether it's still "on the shelf", being worn by someone, or that it has been abandoned in the store. We use this state to know when to use the Messages API in High Fidelity to tell the store copy to create a new child to replace the one being tried on.

Trial copies have a lifetime property set as soon as they're spawned that defines how long they'll exist in the domain. When the trial copy is attached to an avatar and being worn, the lifetime property is set to -1, which prevents it from being removed. Once an entity is removed from the shelf, it has a lifetime of 60 seconds before it is automatically cleaned up if no one wears it within that time frame.

Live Events



One of our regular experiences that we run weekly is a live DJ set in VR. The DJs connect and perform in VR via body trackers, and we get the opportunity to experiment with different features for the performance to test and play with how users can interact with one another in ways that amplify the features that virtual reality can provide. In addition to creating new sets and stages for the DJs to perform in for users, we've also been able to implement features like applause, dynamic particle effects that the DJs can time to the music they play, graffiti walls, and a virtual bar. We also regularly host fireside chats in a virtual auditorium, where we get to test the limits of what's possible with 100 avatars or more in a single space, and develop ways to manage digital performances and even sit in chairs.

In a club setting, you aren't limited by physical lights or power requirements in the same way in VR that you would be on a stage within a traditional venue. You can move beyond controlling light colors and movements into dynamically changing the environment around you or manipulating the weather. Fire doesn't hurt in a virtual space, so you can push the limits of special effects for performances. In a crowd, you can show your appreciation for a performer or speaker with audible applause, but also with mixing in visual effects.

That isn't without trade-offs, though: We had to design a system that has a similar implementation to how we clap in a physical crowd, but with an affordance for the fact that, in VR, you've actually got two controllers in your hands that make striking them together impossible.

Each of these new projects brings its own set of challenges with it. A lot of our current behavior in social contexts and within a physical space is driven by factors that have yet to be reproduced in a digital context. In a virtual world, you can control some elements in ways that you can't in the physical world, but you also lose the physicality of being able to, say, clap your hands together to make a sound, give a tactile high five, or convey very nuanced facial expressions. And so, the content that we design and development needs to rethink some of that.

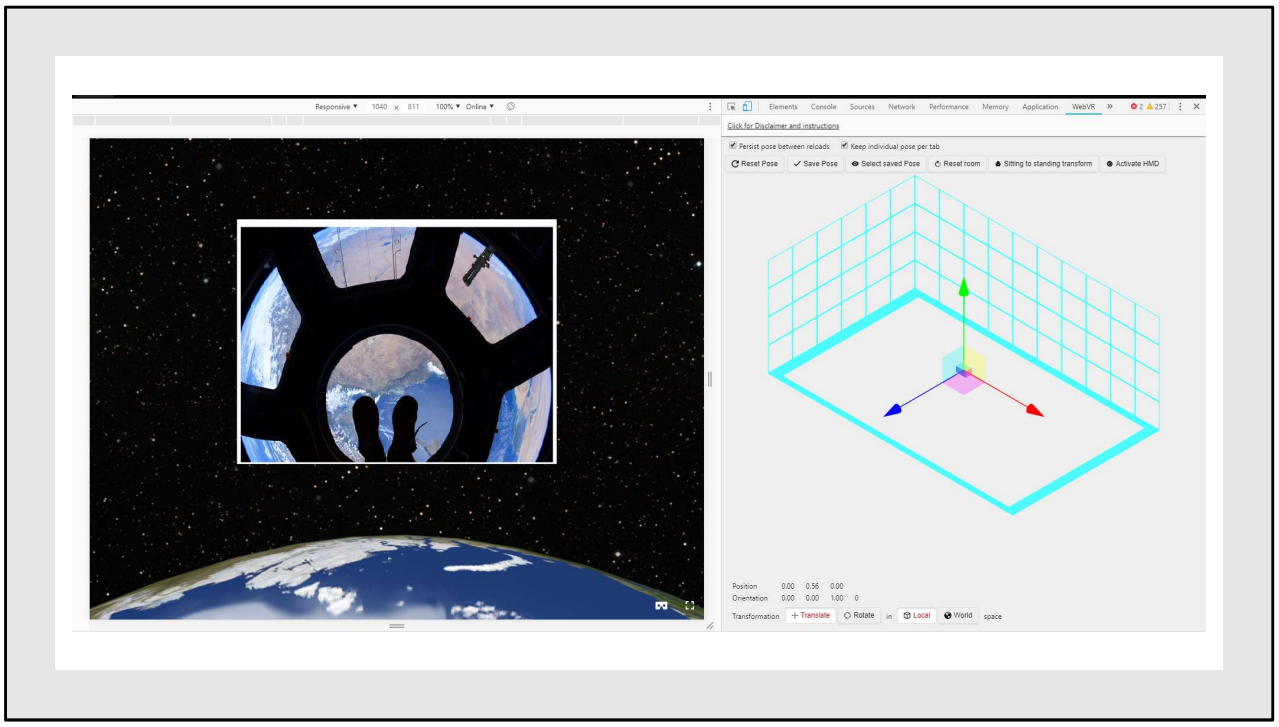
The cool thing is that advancements are coming that enable a lot of that.

With all of that said – this is just the beginning. In the coming months and years, we'll continue to see growth of community spaces that span virtual and desktop experiences. We are still in the earliest stages when it comes to thinking about how immersive computing and interaction paradigms will change how we interact with software and the world: both the one around us, and the ones that we create. We'll see more innovation around display technologies, reduced cost in hardware, increased bandwidth for streaming higher-resolution capture, and much more.

In fact, even by the time I'm done giving this talk, there might be exciting news that would have been perfect for this talk. The pace of this industry never ceases to amaze me – when I gave my very first talk in 2014, I walked off the stage and the first thing someone asked me was: "Did you see that Oculus just announced that they're deprecating support for Mac?" I had _just_ demoed and showed everything on my Macbook Pro! As I was writing these slides this week, Facebook open sourced a 2D to 3D image mapping project, DensePose, which uses deep learning to map 2D pictures of a human onto a 3d model of the human body in real time. And that's just one example of some of the cool stuff that will likely start making its way into social VR applications in the not-so-distant future.

Resource Links:
https://github.com/facebookresearch/DensePose

Looking at social VR, there are a lot of spaces that we'll continue to see increasing innovation. We'll see a growing economy for digital goods emerge as we continue to build out communities around work, entertainment, and education. Resources and tools are needed to enable users to have control over how they represent themselves in these spaces, and protect their information, and we'll see a larger investment in solutions that allow users to work together to create the spaces that will be required to support the distributed future of remote work and play.



Finally, I want to point out that you don't need an HMD to get started! The tools available to you today are enough to get you started. If you want to experiment with WebXR, you can use your phone or even your desktop browser with the WebVR Chrome extension, which simulates a room scale VR device and can help you with your implementation even if you don't have your own desktop hardware. Many of the engines that are available for virtual reality development include simulators and emulators, which has helped reduce the barrier to entry for creating immersive experiences and gives you the chance to start exploring.

Resource Links:
https://chrome.google.com/webstore/detail/webvr-api-emulation/gbdnpaebafagioggnhkacnaaahpiefil

And that's one of the things that makes this industry such a great one to dive into. There are so many opportunities, from development to art, production, design, storytelling, writing, and more. Virtual reality is more than gaming, and there's never been a better time to take steps to get involved. But you don't have to take my word for it – there's a Women in XR Atlanta meetup next Thursday, where you can meet local professionals also working in the space and hear from them.

Resource Links:
https://www.meetup.com/XRAtlanta/
https://gabrielaarp.eventbrite.com/

Resources

@MissLiviRose

Github.com/MissLiviRose

http://linkedin.com/in/misslivirose

livierickson.com/blog

https://livi.link/WeRiseVR

And with that, I'll leave you with this: It is an incredibly powerful thing, to be able to share an experience with someone. With the growth of social VR, we're able to break down geographic and physical boundaries like never before. Embracing the principles of open source software, in order to share and build upon the best practices learned by one another, allows us to democratize not just these new digital spaces, but the communities that grow within them.

Thank you everyone! If anyone has questions, please feel free to reach out – my contact info is on the slides above, I'd love to connect, and if you're ever in High Fidelity, let me know and I'll show you around!

Resource Links:
https://livi.link/werisevr