NeRF Environment Creation System for VR

Cyrus Vachha

9/6/22

Objectives

1. Capture environments

2. View environments

3. Edit/Create environments

Questions

- What is the best way to capture, view, and construct/generate VR nerf environments? (this is the overall question)
- What is the best way to capture an environment and objects for nerfs?
- How do we guide the user to capture nerfs of environments or objects in an intuitive way?
- What is the best way to manipulate an environment in VR?
- What is the best way to manipulate an environment made of NeRFs? (we can’t directly treat as just meshes or use unity regularly)

Room Capture

- Guide users to capture a designated volume of an environment
- Prompt user to scan object or more of environment
- Send the video of the camera from the HMD as well as the camera poses (head locations/rotations) to the server to generate a nerf
- Also capture objects by showing all sides up to the camera
- Output the nerf as a mesh with neural texture

Environment Generation

- Creating NeRF environments based on captured or existing NeRF
- Editing the environment entirely in VR
- Using hand tracking/controllers, eye tracking, possibly face tracking
- Add AI generated NeRF synthetic objects, environments, or variants

Synthetic NeRFs

- Create NeRF on the spot (speech/text to nerf)
- Create nerfs and place them into the environment
- Inpainting/outpainting nerfs like in Dall-e
- NeRF neural completion (fill in all gaps and add artificial detail)
- Create variants of objects
- Example: “Add a door over here”

How are the NeRFs being represented

- NeRFs will act as meshes with a neural texture like from MobileNerf (Google or Luma Labs AI)
- Within Unity they act as meshes so we can position them around the scene
- Mesh + Neural texture is not a true way to render NeRFs, but this component can be modular and added later

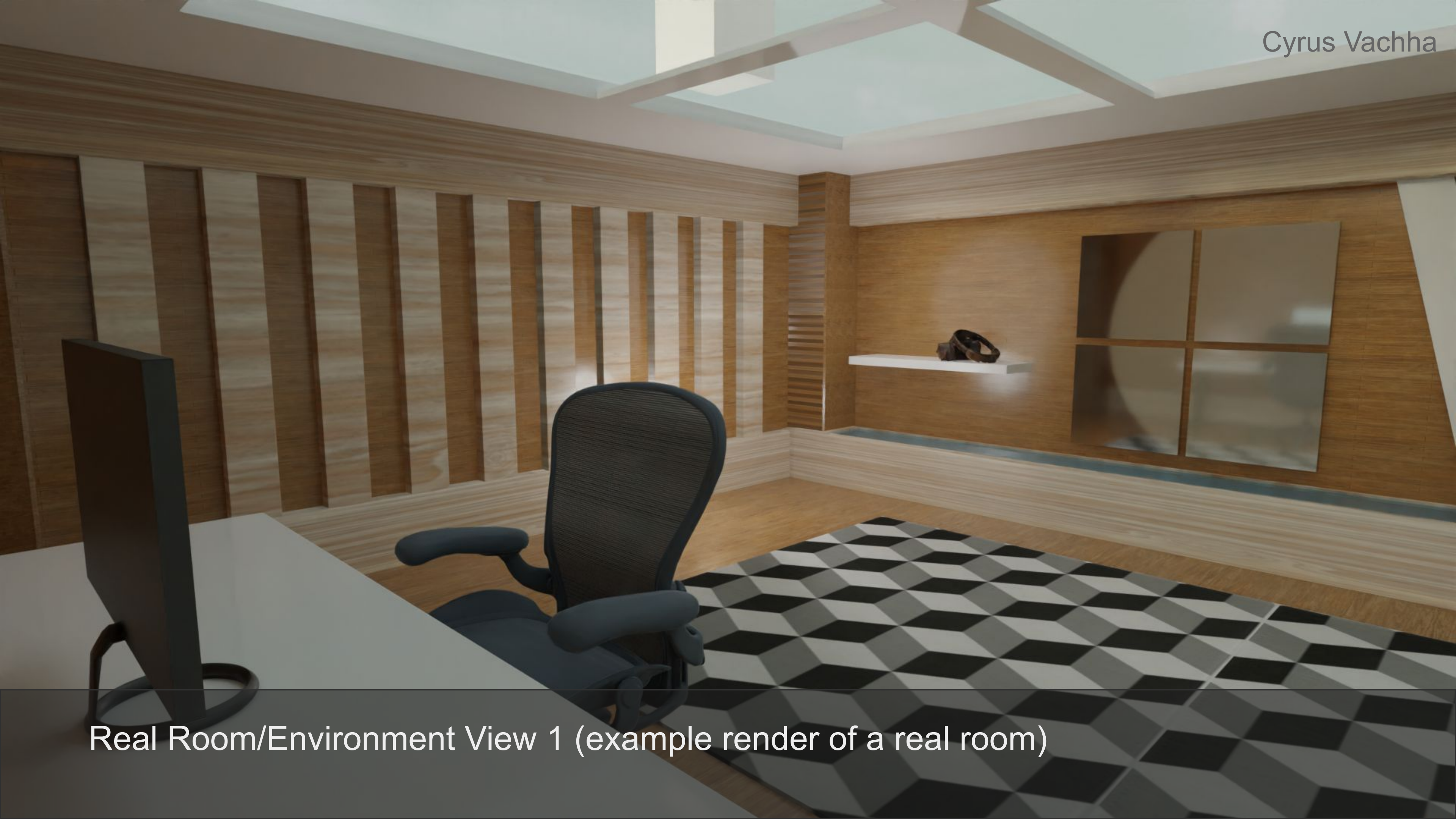
Selection

- What is a good way to select an area/volume?
- Select a volume or 3D space
- Hand tracking
  - Guesstures, grabbing objects, translating, volume selection
- Eye tracking/gaze
  - Paired with head and hand tracking for selection
- Possibly additional tracking methods (facial, other)
- Controller support, but emphasis on hand/eye tracking

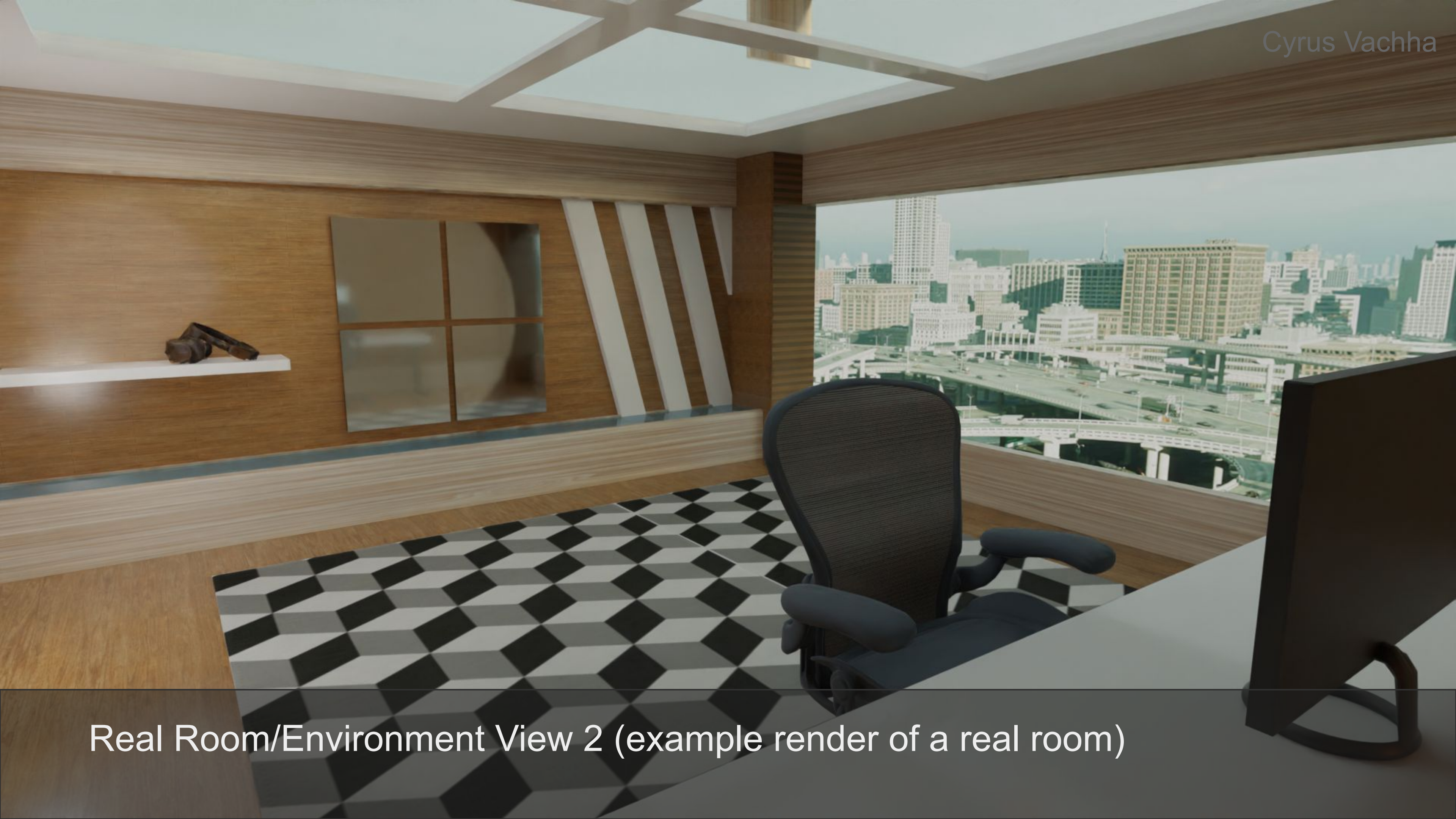
AI Tools

- Creation of NeRFs like Diffusion models like Dall-E
- Neural completion of captured NeRFs (complete object/environment and every perspective of it)
- Editing parts of it (inpainting)
- Extending it (outpainting)
- Creating Variants (alternate versions in a similar style)
- Editing attributes intelligently (color, size, materials, style)

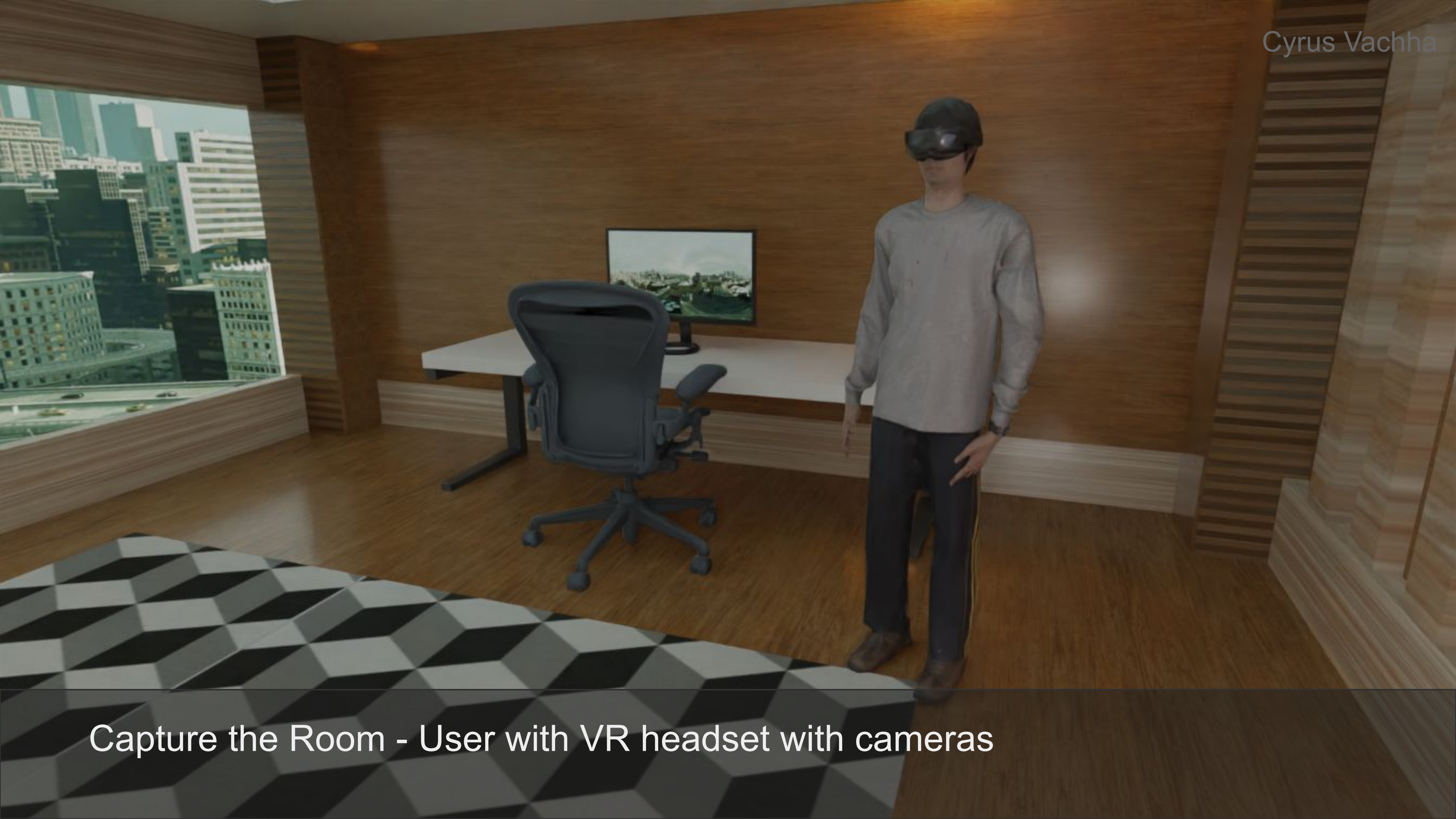
Capture the Environment



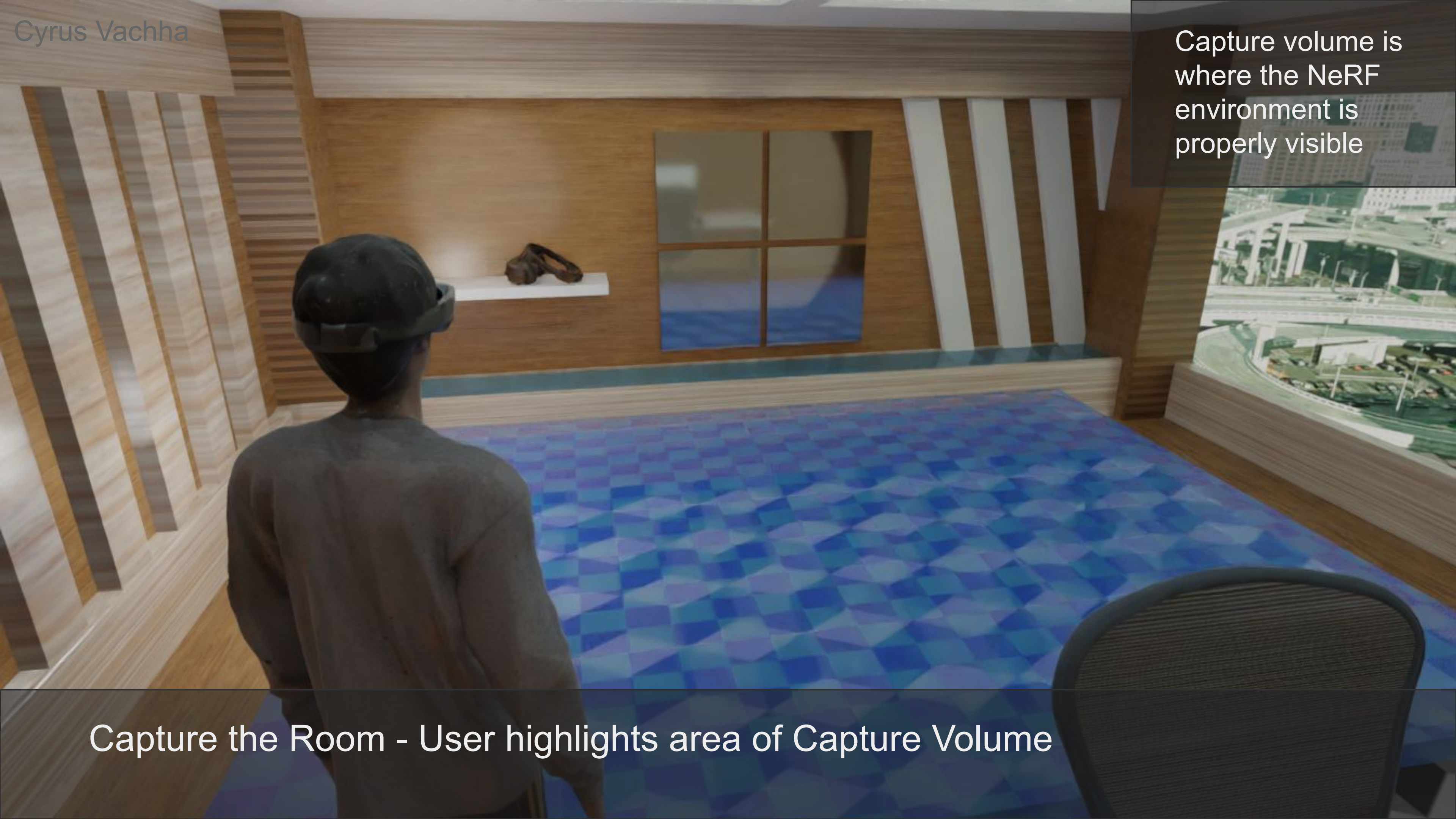
Real Room/Environment View 1 (example render of a real room)



Real Room/Environment View 2 (example render of a real room)

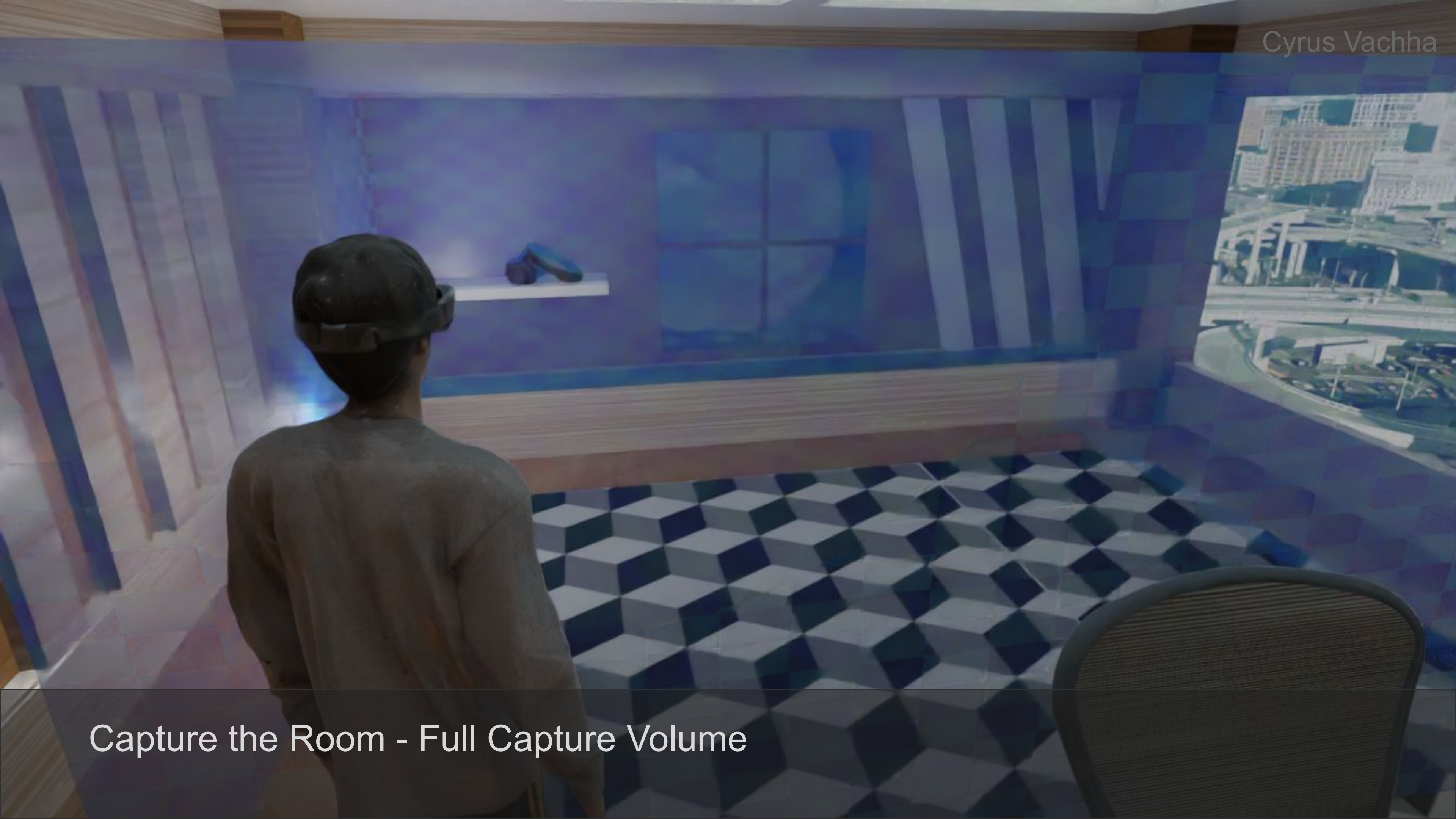


Capture the Room - User with VR headset with cameras

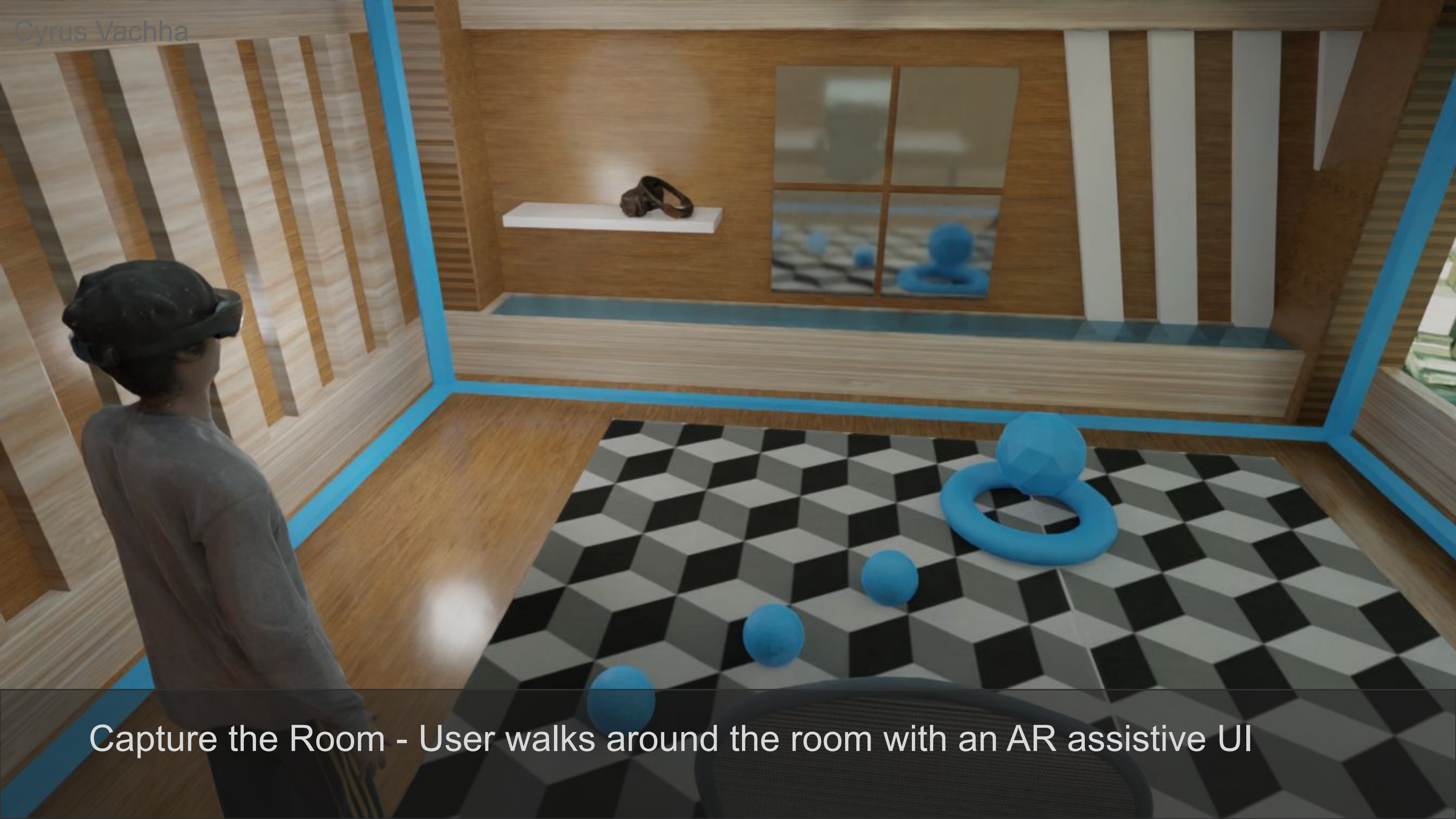


Capture volume is where the NeRF environment is properly visible

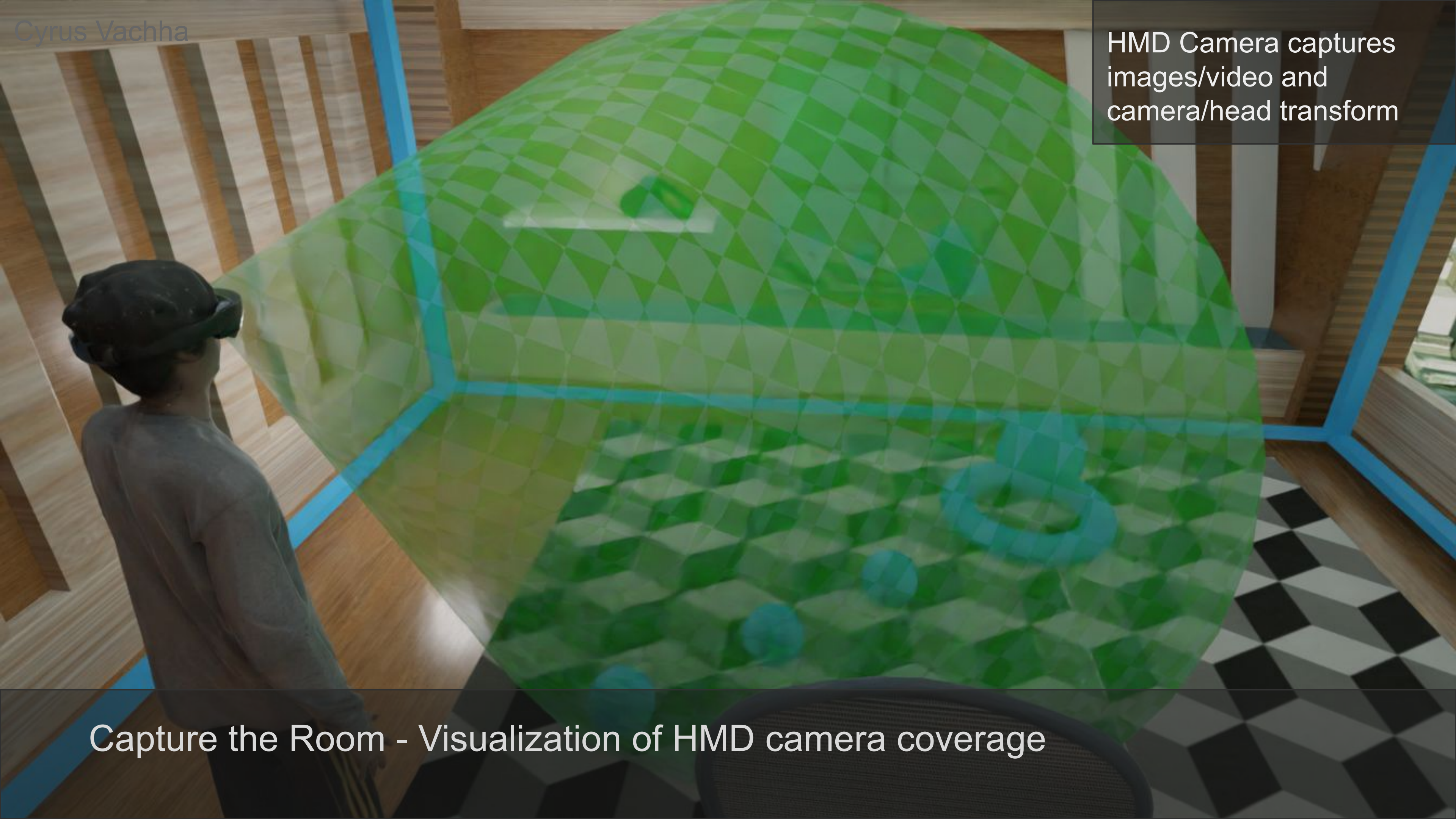
Capture the Room - User highlights area of Capture Volume



Capture the Room - Full Capture Volume



Capture the Room - User walks around the room with an AR assistive UI

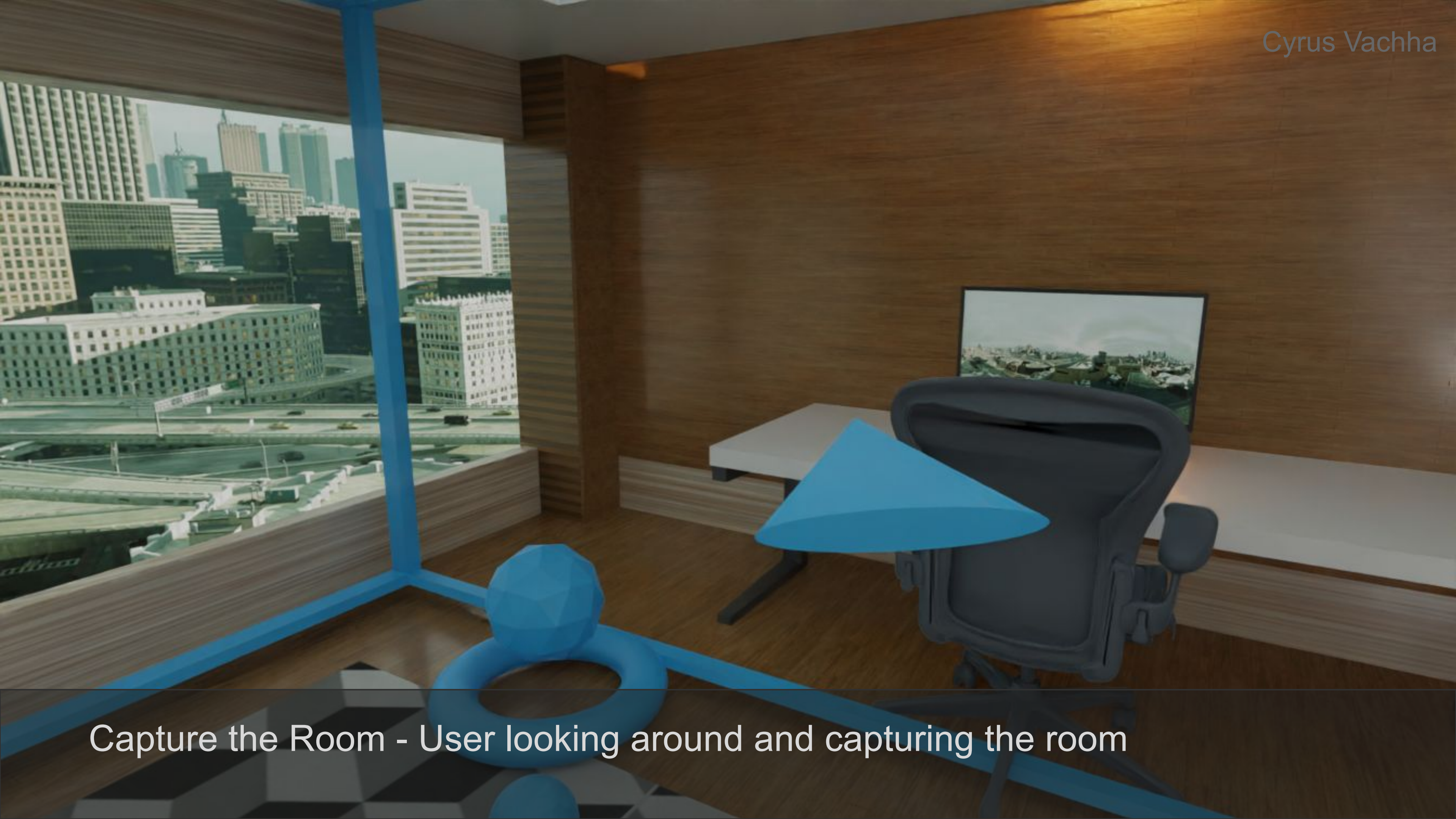


Cyrus Vachha

HMD Camera captures images/video and camera/head transform

Capture the Room - Visualization of HMD camera coverage

Capture the Room - AR interface



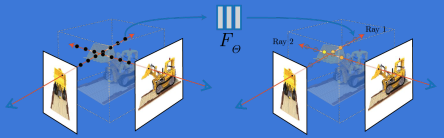
Capture the Room - User looking around and capturing the room

Capture Overview

rus Vachha

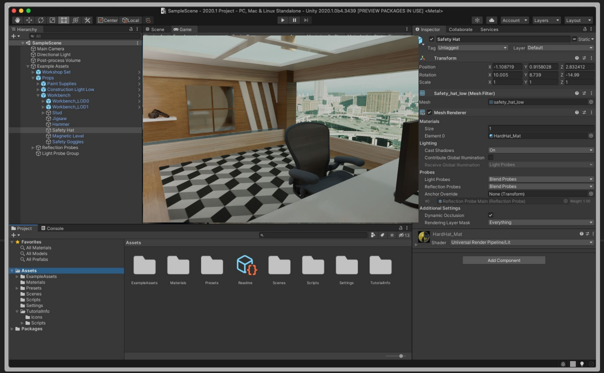
Captured images (50+) or video and camera poses/ transforms (position and rotation of HMD)

NeRF Creation System



Generates NeRF of room/environment

(For now) Output mesh and neural texture



☑ Safety Hat Static

Tag Untagged Layer Default

Transform

Position X -1.108719 Y 0.9158028 Z 2.832412

Rotation X 10.005 Y 8.739 Z -14.99

Scale X 1 Y 1 Z 1

Safety_hat_low (Mesh Filter)

Mesh safety_hat_low

☑ Mesh Renderer

Materials

Size 1

Element 0 HardHat_Mat

Lighting

Cast Shadows On

Contribute Global Illumination □

Receive Global Illumination Light Probes

Probes

Light Probes Blend Probes

Reflection Probes Blend Probes

Anchor Override None (Transform)

#0 Reflection Probe Main (Reflection Probe) Weight 1.00

Additional Settings

Dynamic Occlusion ☑

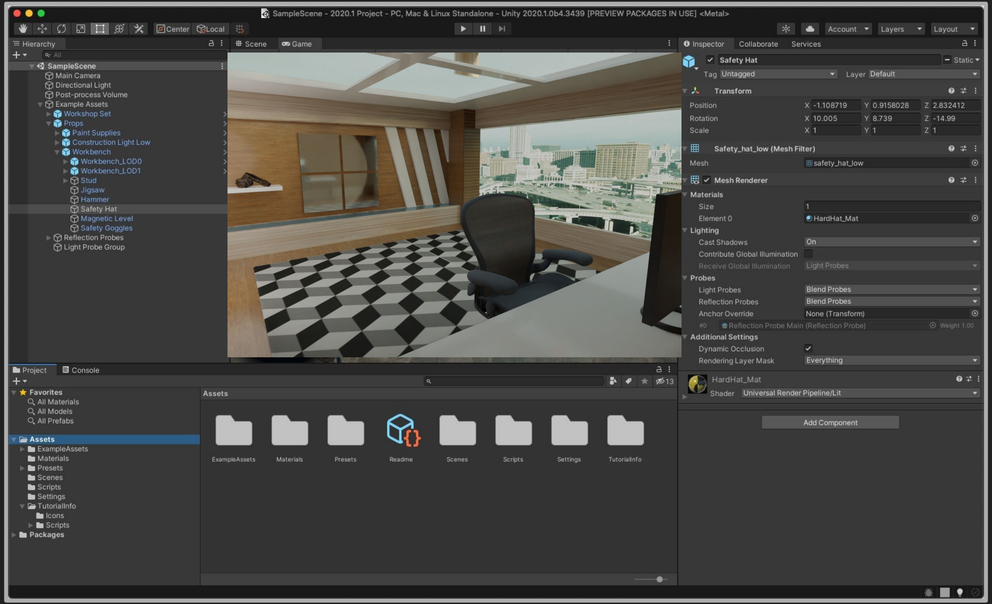
Rendering Layer Mask Everything

HardHat_Mat

Shader Universal Render Pipeline/Lit

Cyrus Vachha

Visualizing NeRFs in Unity



SampleScene - 2020.1 Project - PC, Mac & Linux Standalone - Unity 2020.1.0b4.3439 [PREVIEW PACKAGES IN USE] <Metal>

Safety Hat — Static
Tag Untagged — Layer Default

Transform
Position X -1.108719 Y 0.9158028 Z 2.832412
Rotation X 10.005 Y 8.739 Z -14.99
Scale X 1 Y 1 Z 1

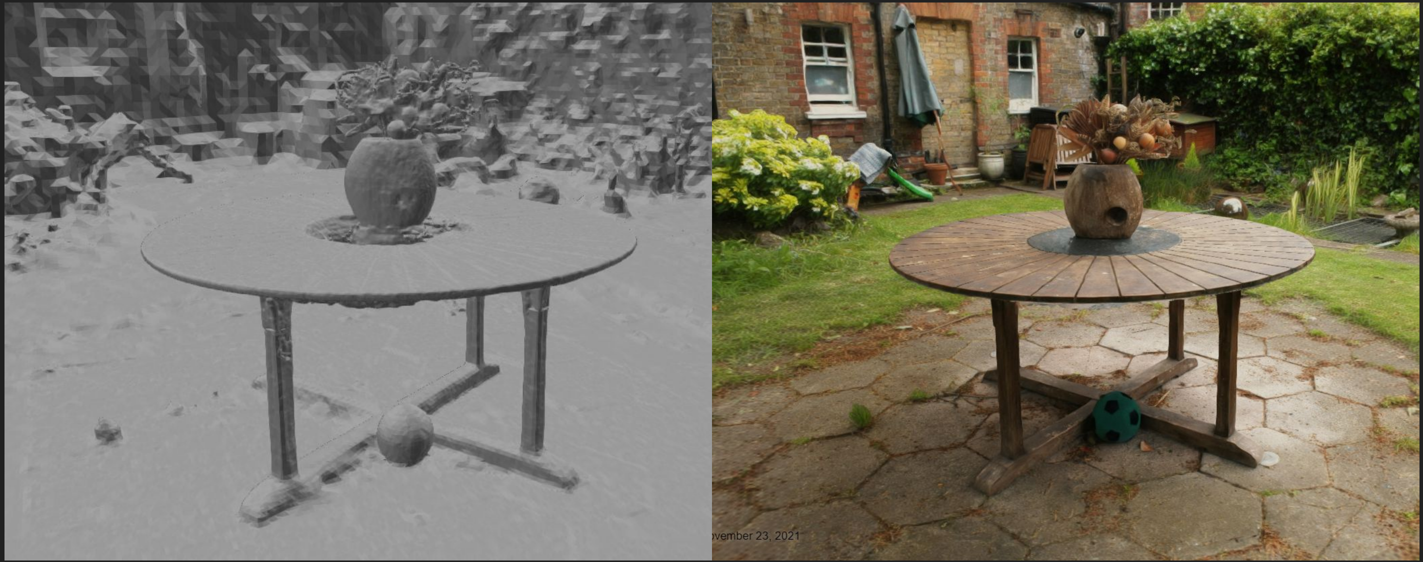
Safety_hat_low (Mesh Filter)
Mesh safety_hat_low

Mesh Renderer
Materials
Size 1
Element 0 HardHat_Mat
Lighting
Cast Shadows On
Contribute Global Illumination
Receive Global Illumination Light Probes
Probes
Light Probes Blend Probes
Reflection Probes Blend Probes
Anchor Override None (Transform)
#0 Reflection Probe Main (Reflection Probe) Weight 1.00
Additional Settings
Dynamic Occlusion ✓
Rendering Layer Mask Everything

HardHat_Mat
Shader Universal Render Pipeline/Lit

- View NeRF in Unity/app as a mesh with a neural texture
- Neural texture is streamed from the cloud
- This is still a work around (not rendering a true NeRF
- Goal is to eventually stream nerf volumes (re-lighting included)

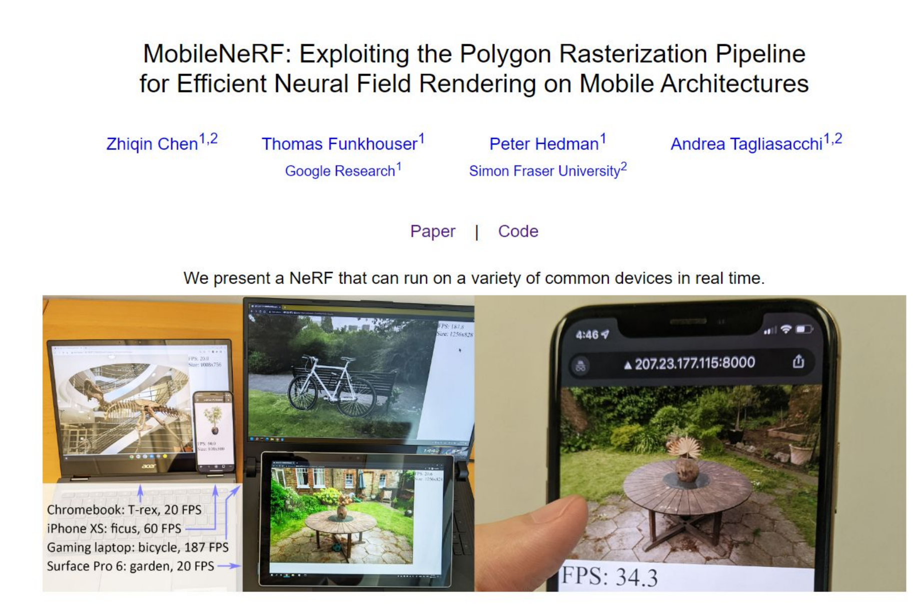
Luma Labs AI NeRF Mesh and Neural Texture



- View NeRFs from browser (https://captures.lumalabs.ai/unbounded)

- Mesh with a neural texture (single layer MLP)

Google MobileNeRF (https://mobile-nerf.github.io/)



MobileNeRF: Exploiting the Polygon Rasterization Pipeline for Efficient Neural Field Rendering on Mobile Architectures

Zhiqin Chen[1,2] Thomas Funkhouser[1] Peter Hedman[1] Andrea Tagliasacchi[1,2]

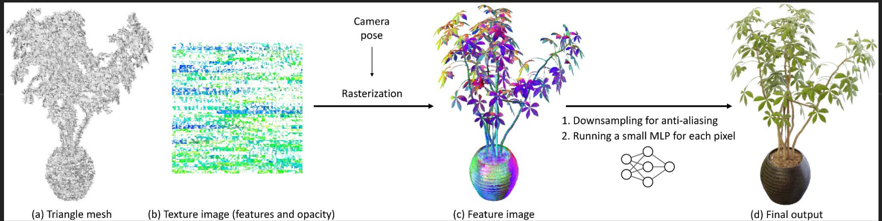
Google Research[1] Simon Fraser University[2]

Paper | Code

We present a NeRF that can run on a variety of common devices in real time.

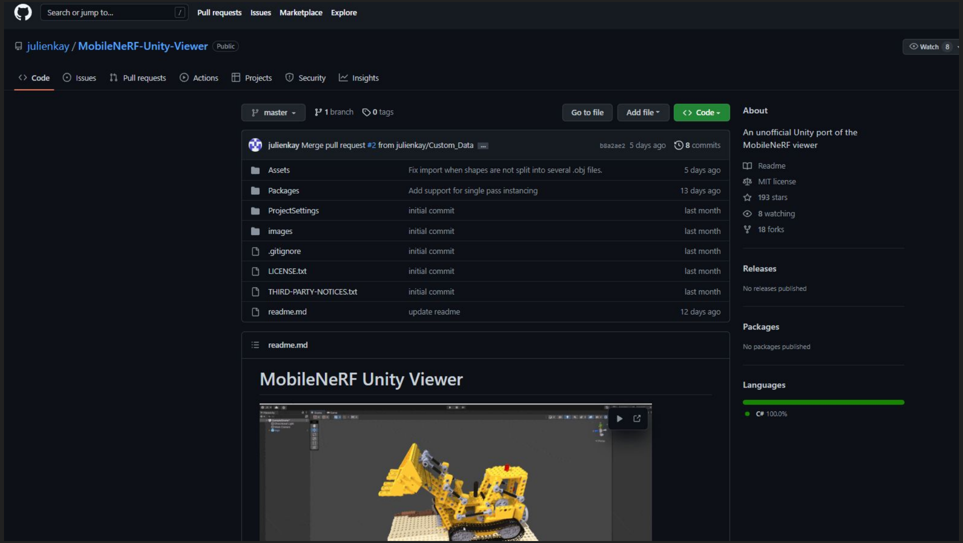
Chromebook: T-rex, 20 FPS
iPhone XS: ficus, 60 FPS
Gaming laptop: bicycle, 187 FPS
Surface Pro 6: garden, 20 FPS

- View NeRFs from browser
- Efficient enough for with VR
- Mesh with a neural texture (single layer MLP)



Camera pose
Rasterization
1. Downsampling for anti-aliasing
2. Running a small MLP for each pixel
(a) Triangle mesh
(b) Texture image (features and opacity)
(c) Feature image
(d) Final output

Unity implementation of MobileNeRF (https://github.com/julienkay/MobileNeRF-Unity-Viewer)



julienkay Merge pull request #2 from julienkay/Custom_Data … b8a2ae2 5 days ago 8 commits

Assets	Fix import when shapes are not split into several .obj files.	5 days ago
Packages	Add support for single pass instancing	13 days ago
ProjectSettings	initial commit	last month
images	initial commit	last month
.gitignore	initial commit	last month
LICENSE.txt	initial commit	last month
THIRD-PARTY-NOTICES.txt	initial commit	last month
readme.md	update readme	12 days ago

readme.md

MobileNeRF Unity Viewer

About

An unofficial Unity port of the MobileNeRF viewer

Readme

MIT license

193 stars

8 watching

18 forks

Releases

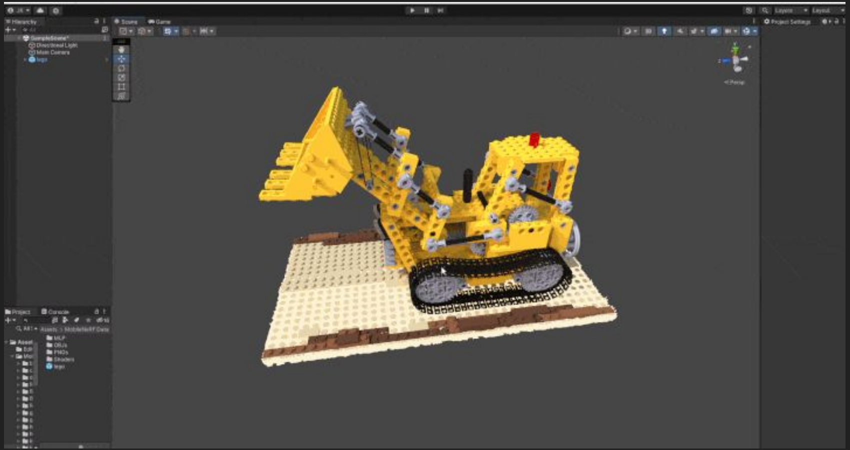
No releases published

Packages

No packages published

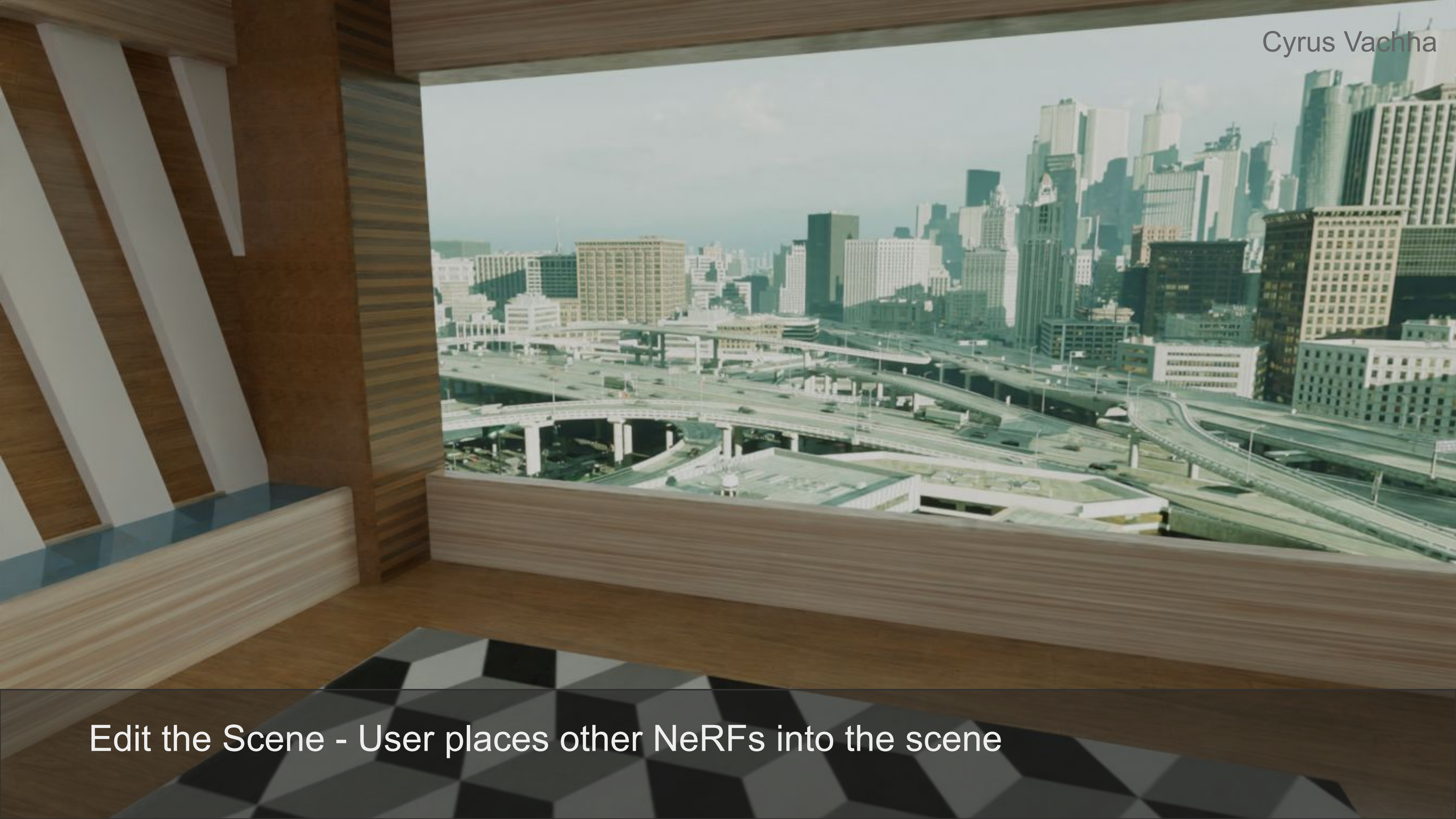
Languages

C# 100.0%



- View NeRFs in Unity as a mesh with texture
- Efficient enough for with VR and AR
- Downside is limited volume for environment NeRFs

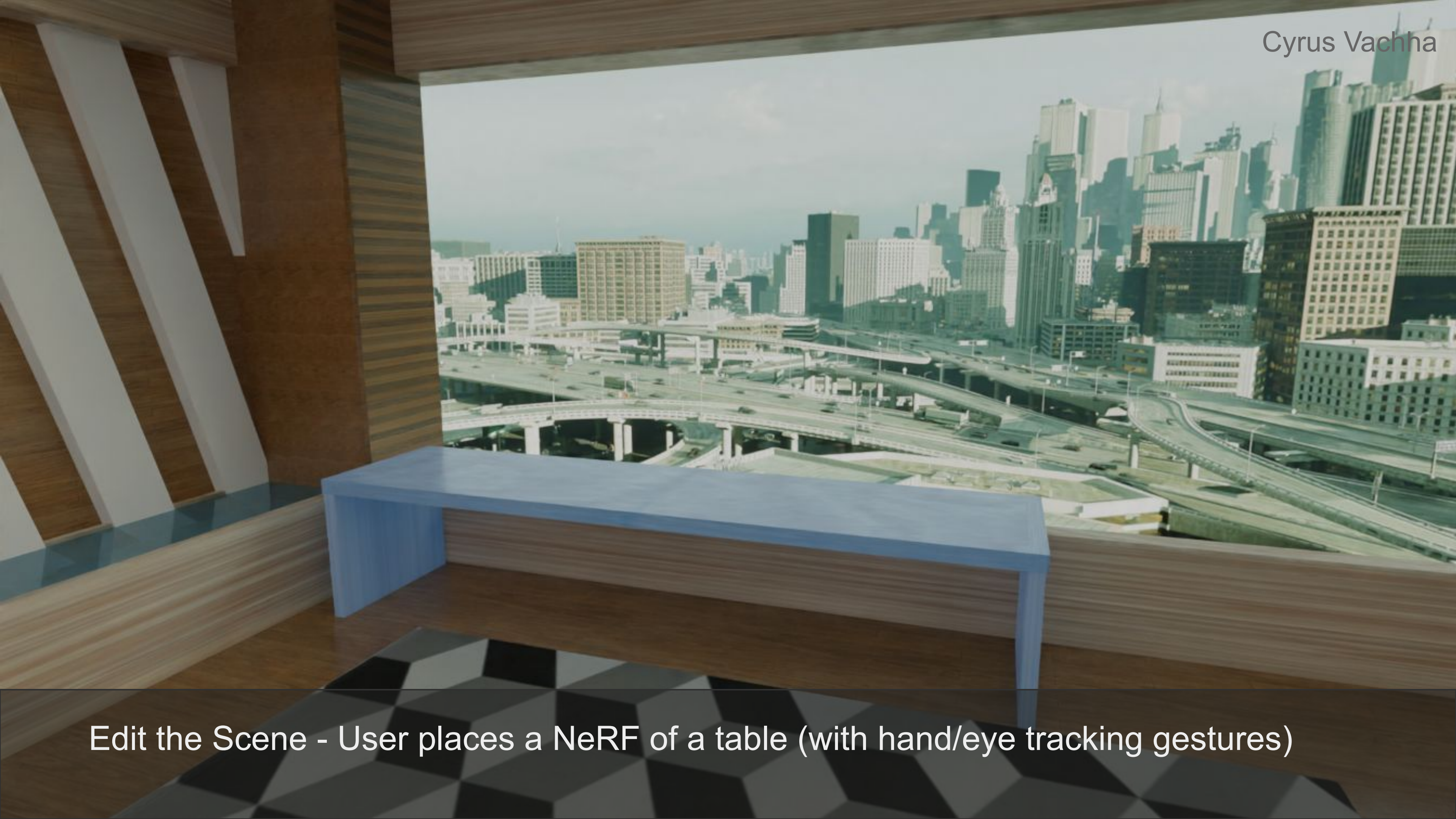
Editing the Scene



Cyrus Vachha

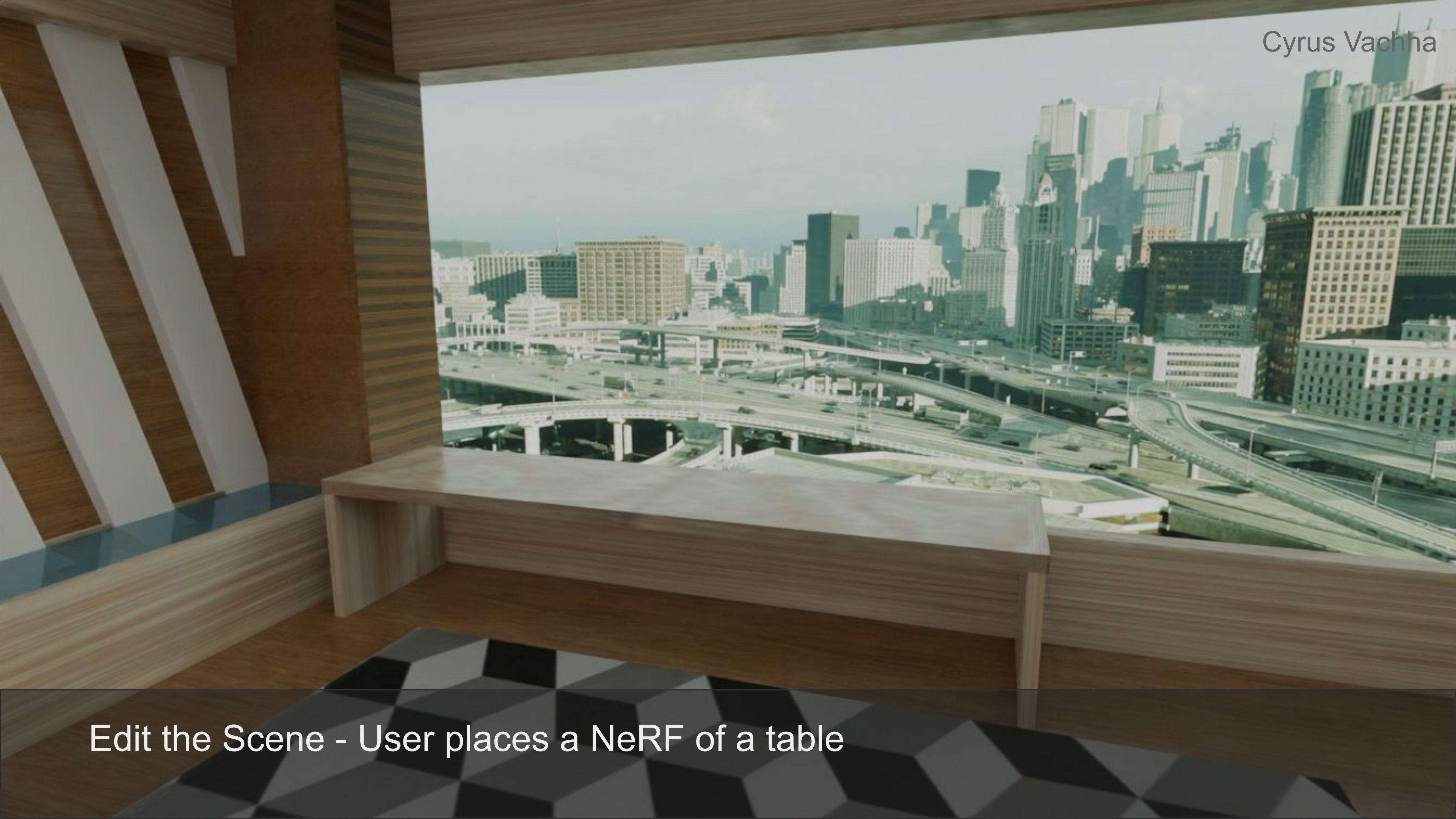
Edit the Scene - User places other NeRFs into the scene

Cyrus Vachha

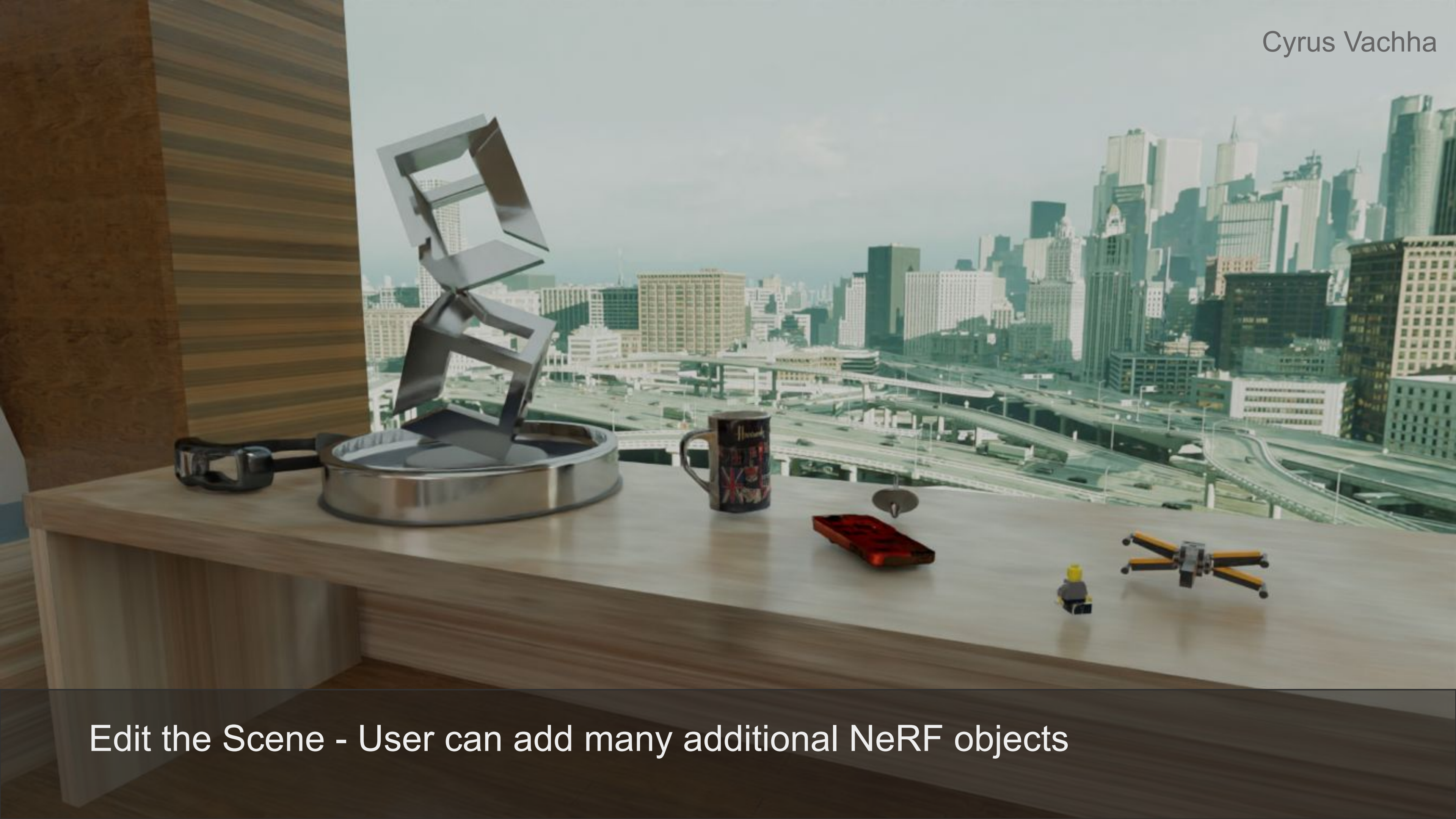


Edit the Scene - User places a NeRF of a table (with hand/eye tracking gestures)

Cyrus Vachha

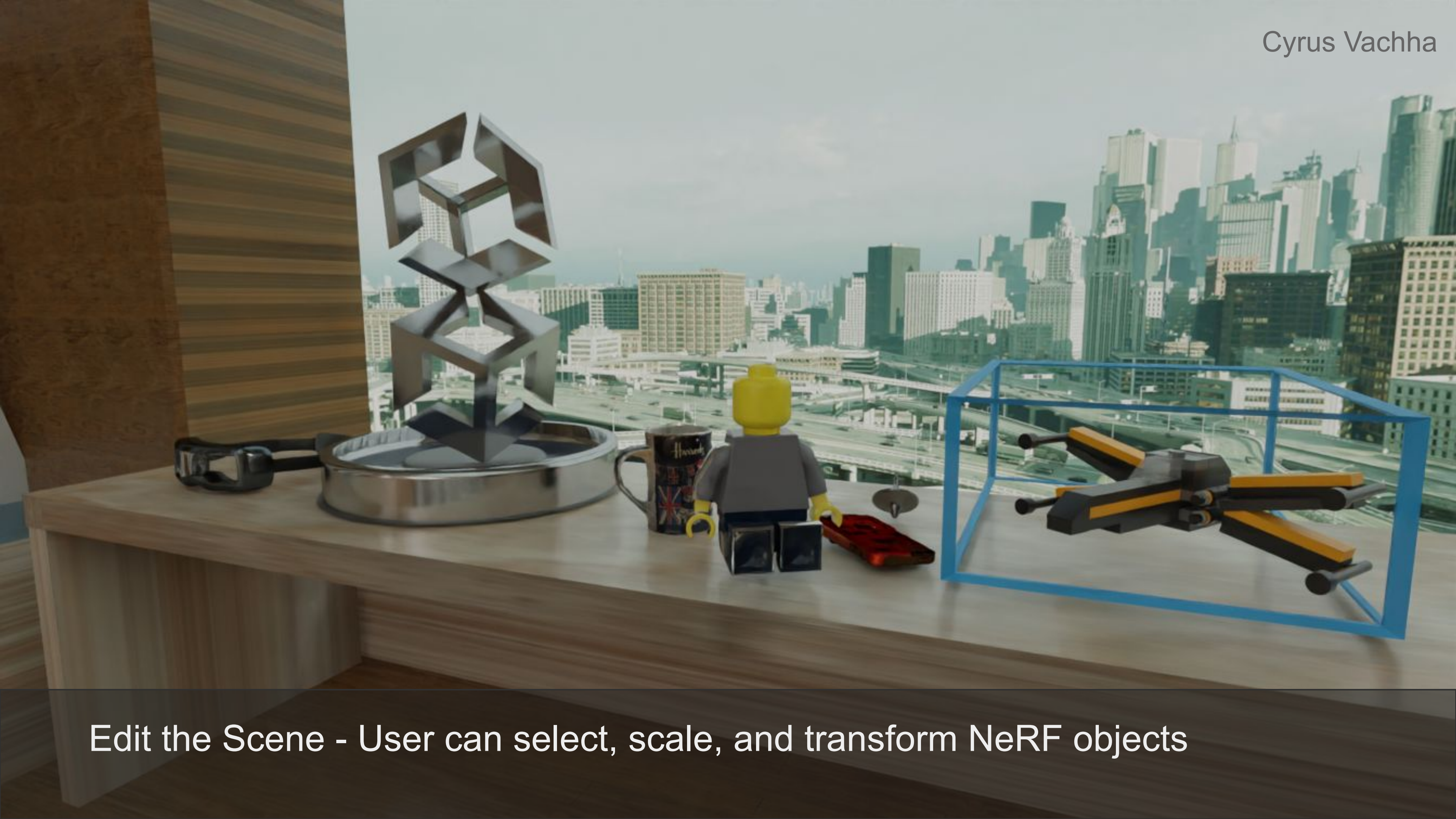


Edit the Scene - User places a NeRF of a table

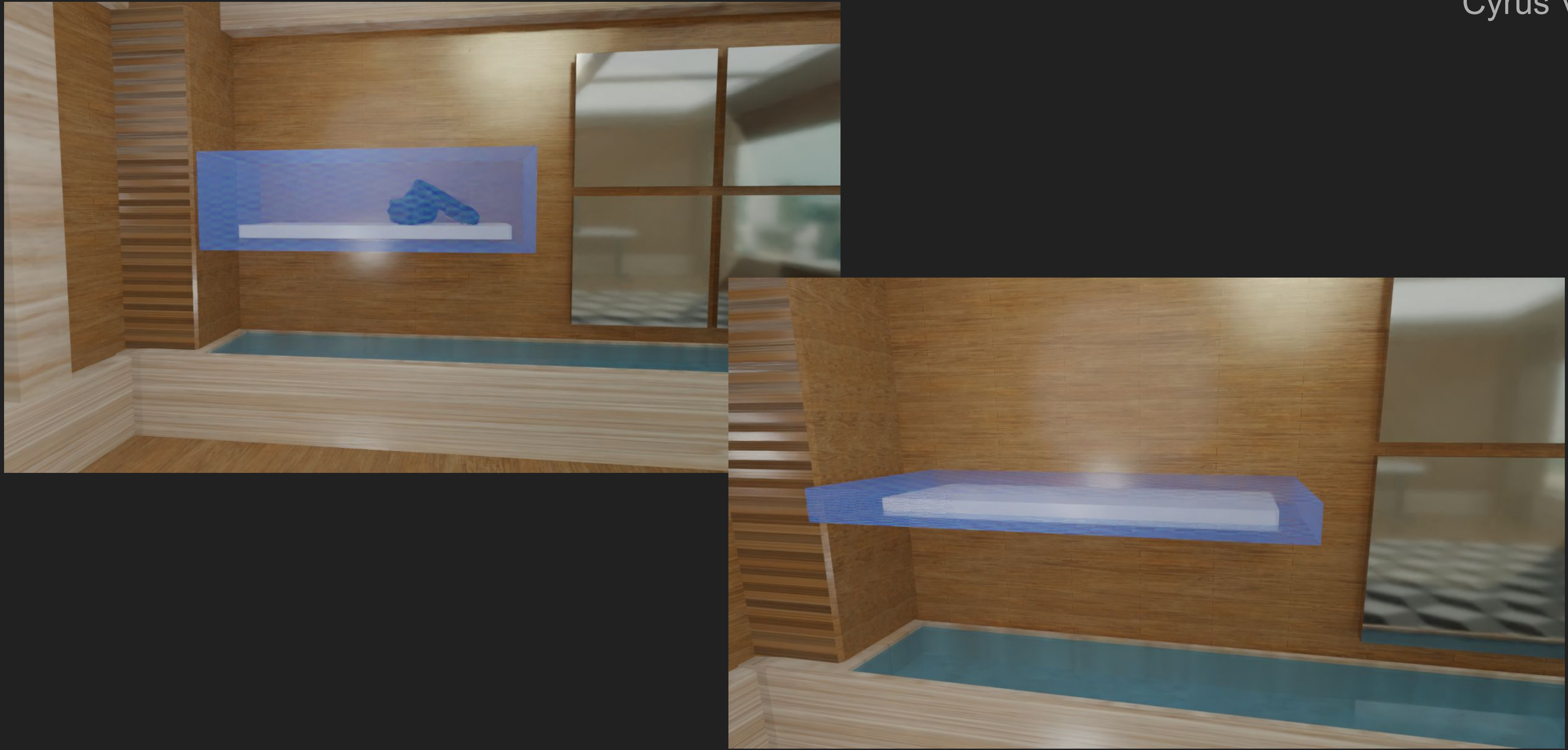


Edit the Scene - User can add many additional NeRF objects

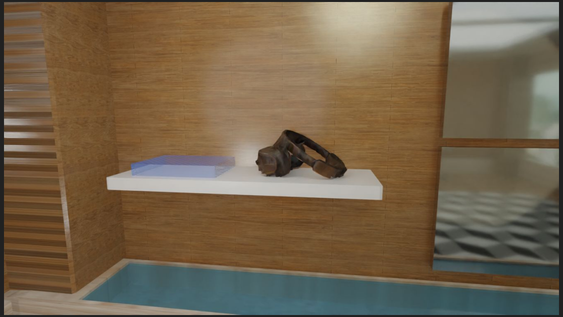
Cyrus Vachha



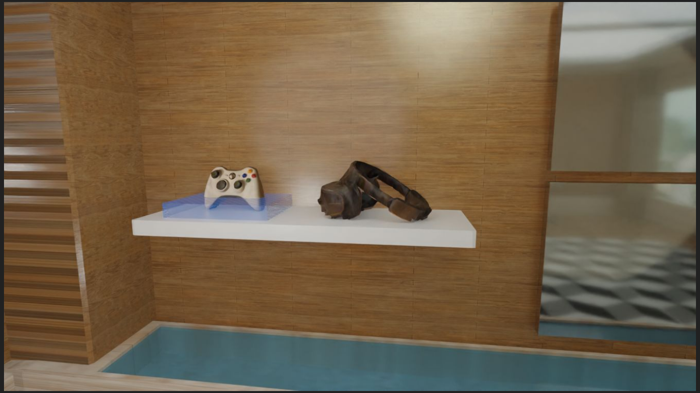
Edit the Scene - User can select, scale, and transform NeRF objects



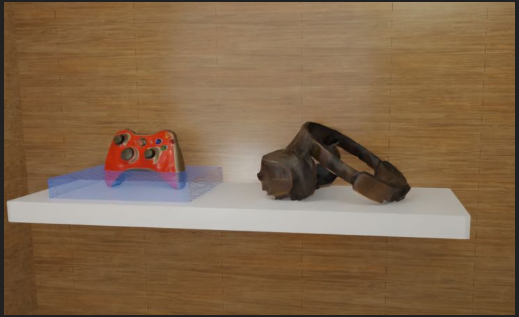
Edit the Scene - User can edit existing NeRF environment or objects (volumetrically slicing)



User selects the volume to place or edit



Adds NeRF through text input or speech to text: “Add an Xbox 360 controller”



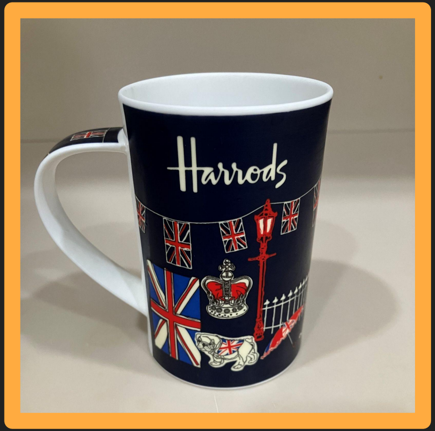
“Change the color to red”

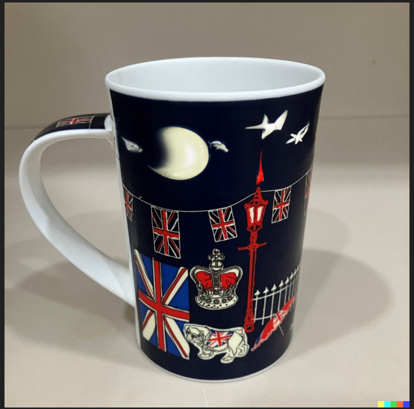
Edit the Scene - Users can add NeRFs into the scene from a library or AI Generated

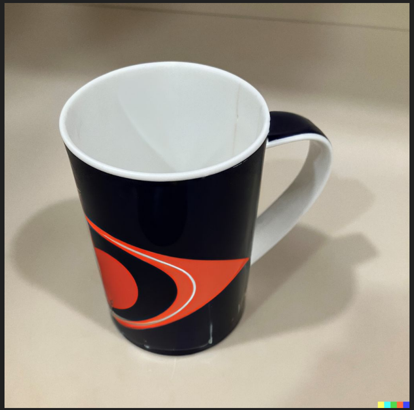
Dall-e Experimentation (all 2D)

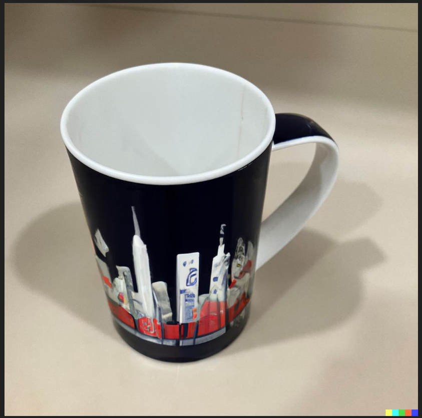
- Creating objects
- Editing existing objects
- Creating variants of existing objects
- Completing blurry objects/scenes
- Editing an environment

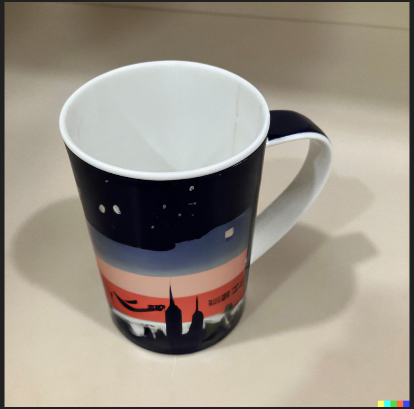
Cyrus Vachha

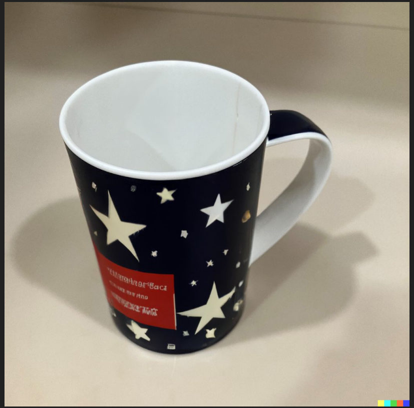






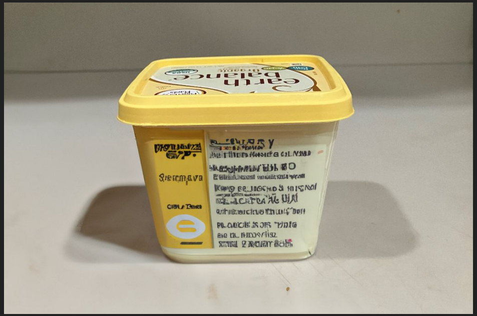


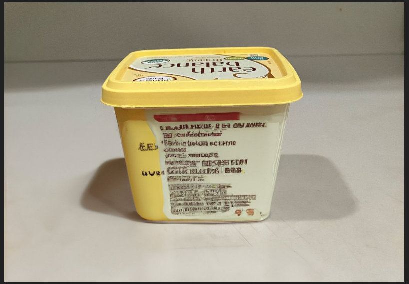


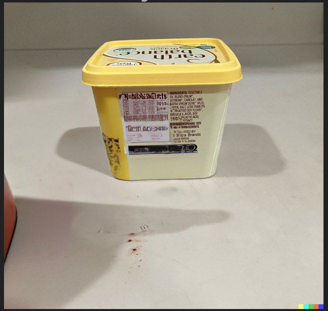


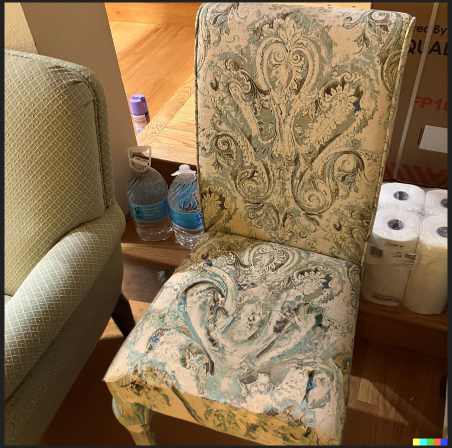
Dall-e Object In-painting with text prompts

Cyrus Vachha





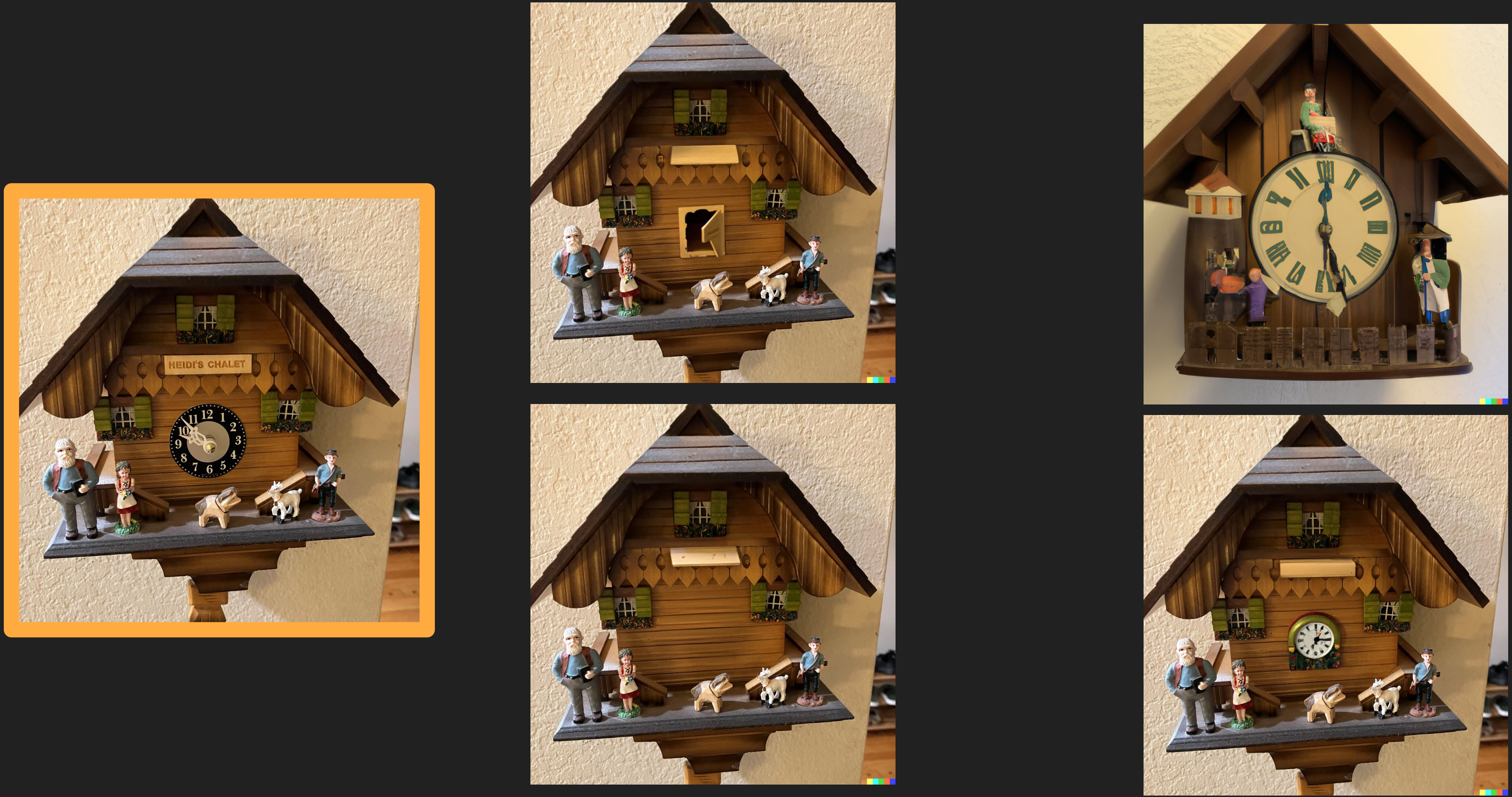




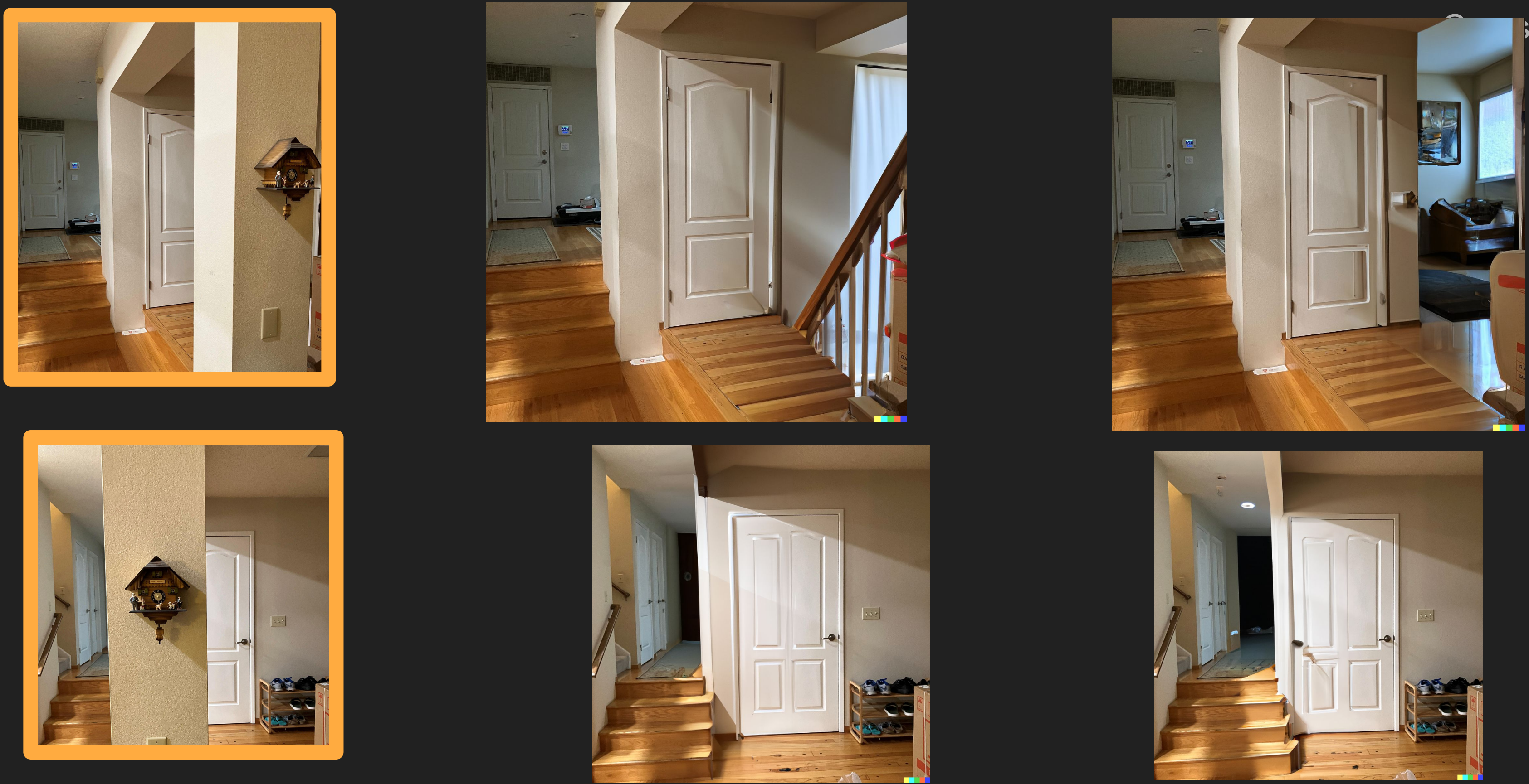
We can add relevant synthetic detail

Object In-painting/neural completion with text prompts and highlighting

Cyrus Vachha

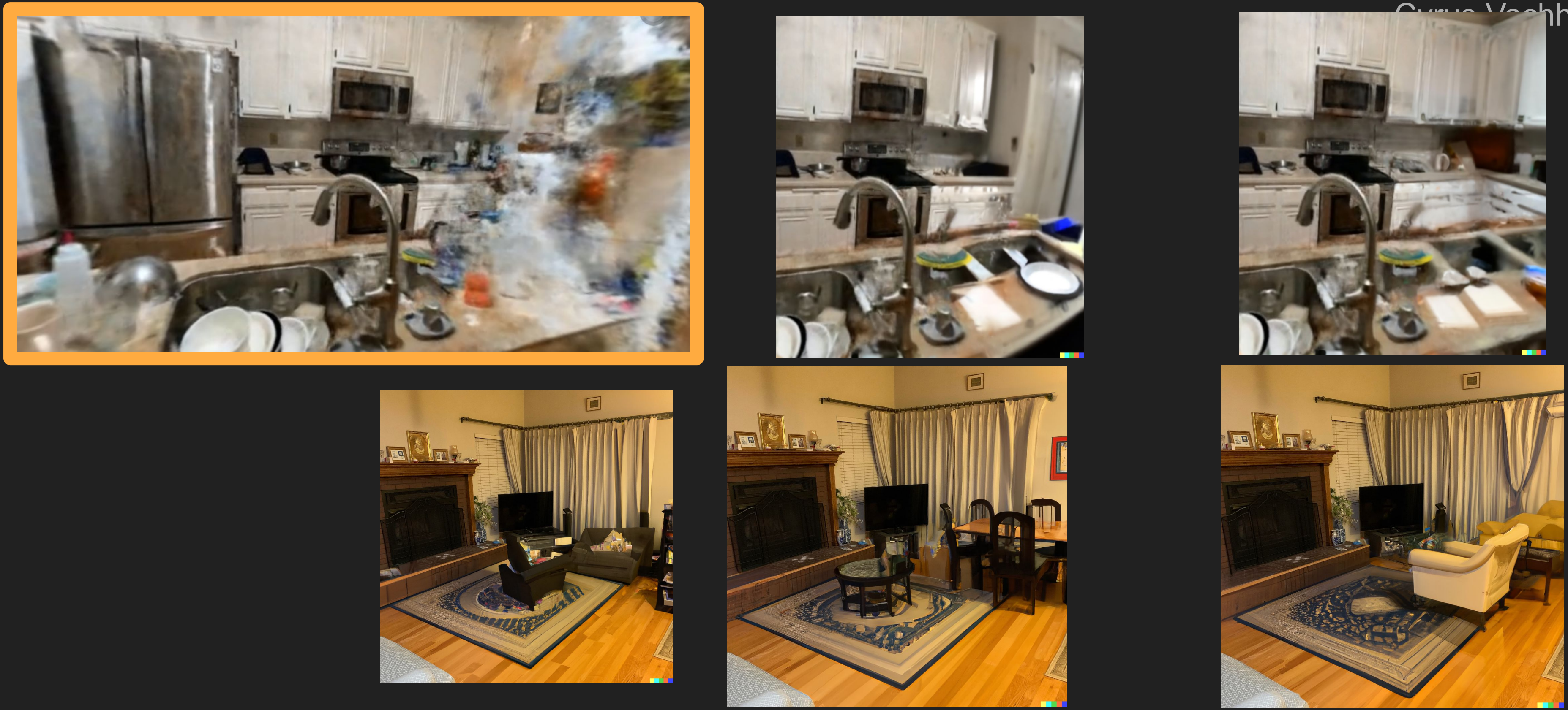


Object In-painting variants with text prompts and highlighting



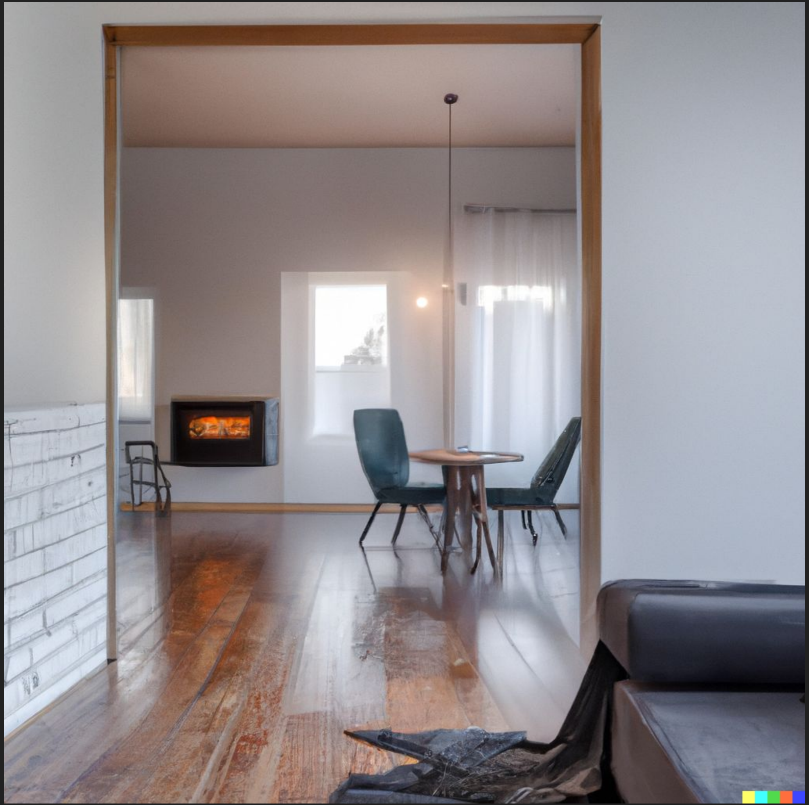
Object In-painting variants with text prompts and highlighting

Cyrus Vachha



Object In-painting variants with text prompts and highlighting

"A living room with a wooden floor, a patio door and a sofa. There is a wooden coffee table in front of the grey cloth sofa and a small chair next to it"



Environment Generation (can we generalize this to NeRF environments)

Additional

- Automated system to capture environments with 360 camera
  - Automated remote control tripod with a 360 camera
  - Head mounted 360 camera
- Synthetically create very large scale environments (capture many NeRF environments first as a dataset)
- Mixed Reality/pass-through environments instead of just NeRF environments
  - Create reflection map from the HMD cameras
- Modular system so we can update how NeRFs are represented (streaming volumes or render rays within app)
- Add artificial detail to the environment or object NeRFs
- Emphasis on using new MR headsets with color HMD passthrough

Cyrus Vachha

NeRF Studio Integration

11/16/22

Compositing 3D Objects/Meshes

- Ability to place 3D objects
  - Custom import mesh
  - Text to 3D (DreamFusion)
- Relight 3D objects
- NeRF Studio Web UI uses 3js, so 3D objects can be placed in the scene. Objects might have to be rendered separately from the NeRF and composited over.
- Goal is to composite other NeRFs. This may have to be done through MobileNeRF and later compositing with final NeRFs rendered separately and then composited.

Compositing People

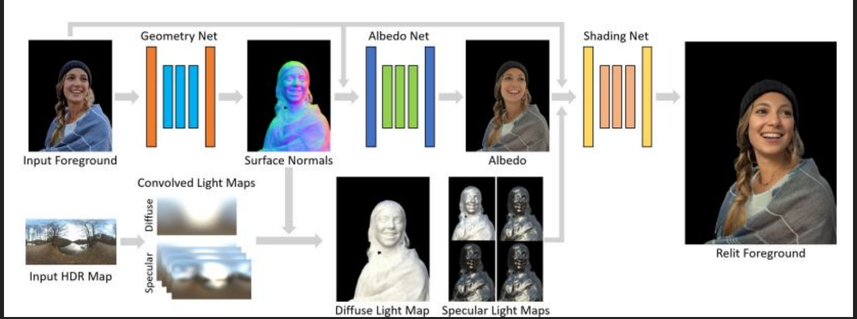
- 360 HDRI image rendering and relighting people recorded with green screen
- Track the person and get a 360 HDRI video of the environment at every frame and relight the personal at those positions
- Possibility of better relighting with each point and luminance

Pipeline Compositing People

- Take NeRF of environment (sunset at Campanile)
- Record video of person with a green screen or camera tracked with easy to composite the background
- Render the NeRF background layer based on camera movements
- Render a 360 HDRI video/images based on the person positional movement
- Relight person at every frame based on the HDRI maps
- Composite the person with the NeRF background
- If the Nerf is not a 360 full environment, we can use a diffusion model to fill in the gaps so the 360 image is complete

Relighting People

- Similar method as Nvidia and Google Total portrait relighting.
- Follow the Total Relighting Pipeline including Neural Relighting
- Google Reference: https://augmentedperception.github.io/total_relighting/total_relighting_paper.pdf



VR UI

- Ability to view NeRF as mobile NeRF in VR
- Ability to place objects and create camera path
- Basic filmmaking tools and camera controls
- Also ability to capture NeRF scenes from VR and export the images/camera poses in the format compatible with NeRF Studio (VR/AR headsets with color passthrough)

Intelligent Worldbuilding

- Ability to make implicit references in the scene (change the color of “that” wall to green)
- “Place a door over here”
- Edit objects similar to diffusion model based programs