EEEC5698
Networked XR Systems

Recap

- Rendering Basics
- Edge/Cloud/Remote Rendering

Lecture Outline for Today

- Edge/Cloud/Hybrid/Remote Rendering

Rendering Performance

- Rendering computation is expensive
  - Offload rendering computation elsewhere for high-quality

- Remote rendering
- Cloud rendering
- Edge rendering
- Distributed rendering

Local vs. Remote Rendering

- **Local Rendering**: The traditional approach where rendering is done on the same device that is being used for display and interaction.
- **Remote Rendering**: Offloading the rendering process to a remote server or dedicated hardware and streaming the output back to the local device.
- Advantages and Disadvantages:
  - Local rendering leverages direct access to the GPU, minimizing latency but can be limited by the device's hardware capabilities.
  - Remote rendering allows for more powerful processing and potentially better graphics quality but can introduce network latency and require stable connectivity.

Cloud Rendering

- Using cloud computing resources to perform rendering tasks, with the rendered content streamed back to the user's device.
  - Scalability, access to high-performance hardware, and the ability to offload intensive computational tasks from local devices.
- Considerations: Requires reliable and fast internet connection, and there can be concerns about data security and latency.

Cloud Rendering

- Two-way latency
  - Need to wait until the user's pose is sent to the Cloud, render the content, and receive the rendered video

Edge Server Rendering

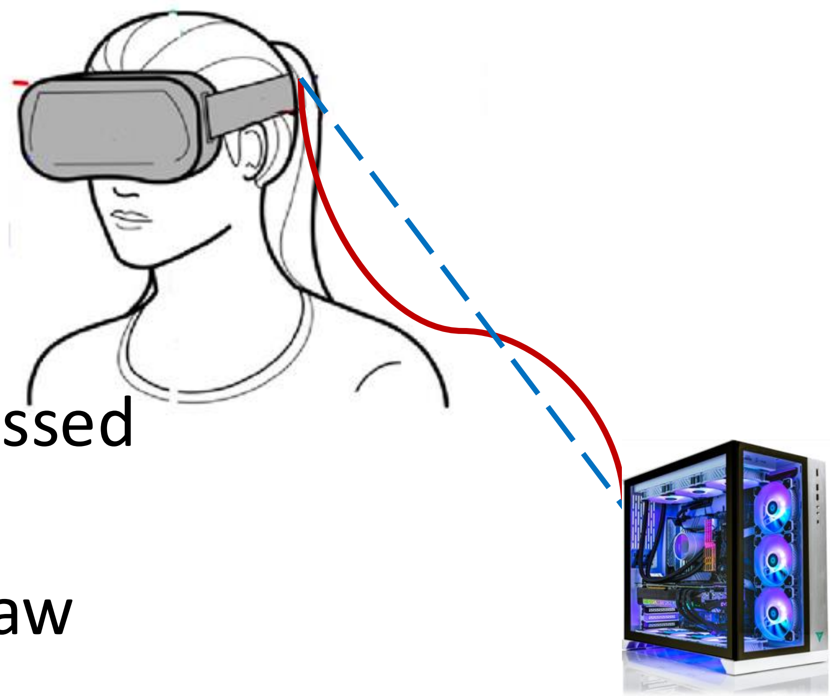
- Edge rendering is done at the edge of the network, near the user, rather than on centralized data centers or the user's device.
  - The purpose is to reduce latency, decrease the bandwidth needed for high-quality graphics, and alleviate the computational load on user devices.
- Key Benefits:
  - Faster content delivery due to proximity to the user.
  - Improved performance for real-time applications.

Edge Server Rendering

- Cellular Networks
  - Rendering is placed at the Base station
- Need to stream rendered video from Base station
  - Base stations are placed at a few miles away
  - High frequencies provide high bandwidth but LOS problem
  - Lower frequencies are okay but low bandwidth

Edge Server Rendering

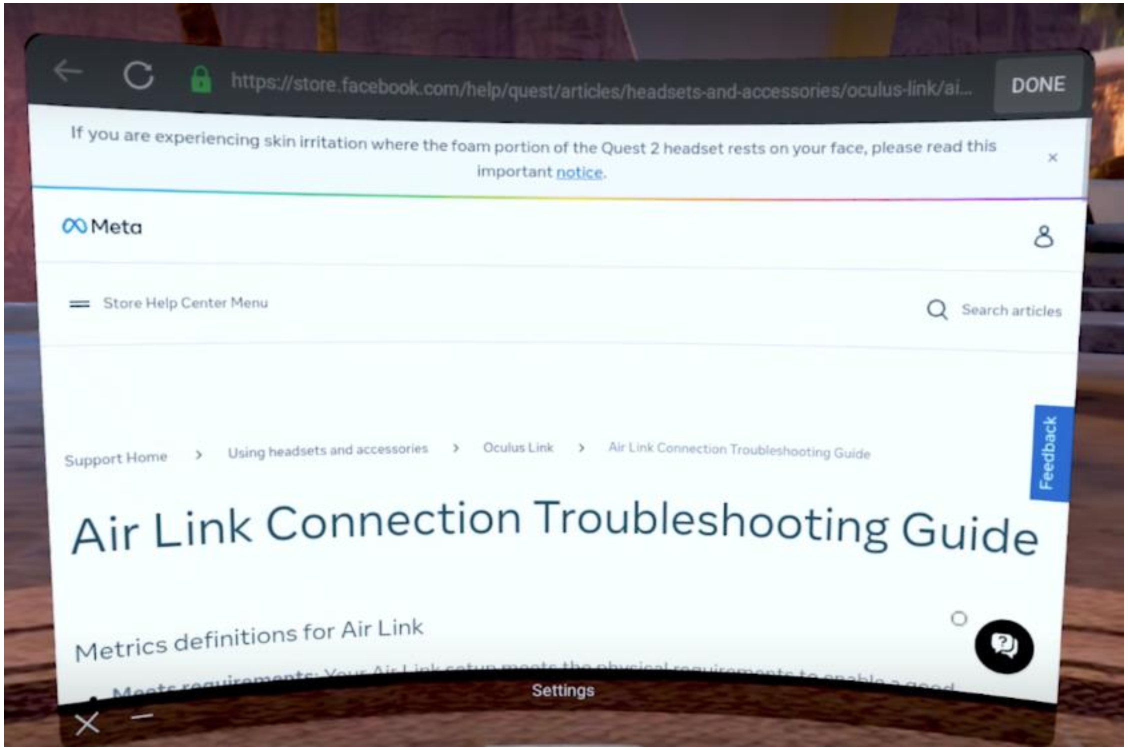
- WiFi
  - Rendering is placed a computer within the same WIFi LAN
  - Closer to users
  - Low latency



- Works for streaming compressed rendered content
- What if we want to stream raw video?

Edge Server Rendering

- WiFi
  - Connect Meta Quest to your PC over Wi-Fi with Air Link



Edge Server Rendering

- Why do we want to stream raw video to XR devices?
  - Eliminate the computation demands of compression and decompression
  - Also saves latency

- mmWaves, THz or Optical links for higher bandwidths

Edge Server Rendering

- Problem with higher frequency wireless links
  - Links are not reliable – narrow wavelength
  - Environmental impact
  - Line of sight

- Problem with XR devices
  - Users move around
  - Mobility impact

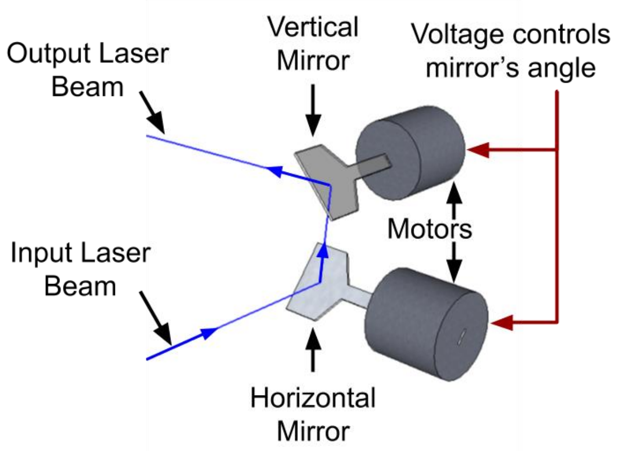
Edge Server Rendering

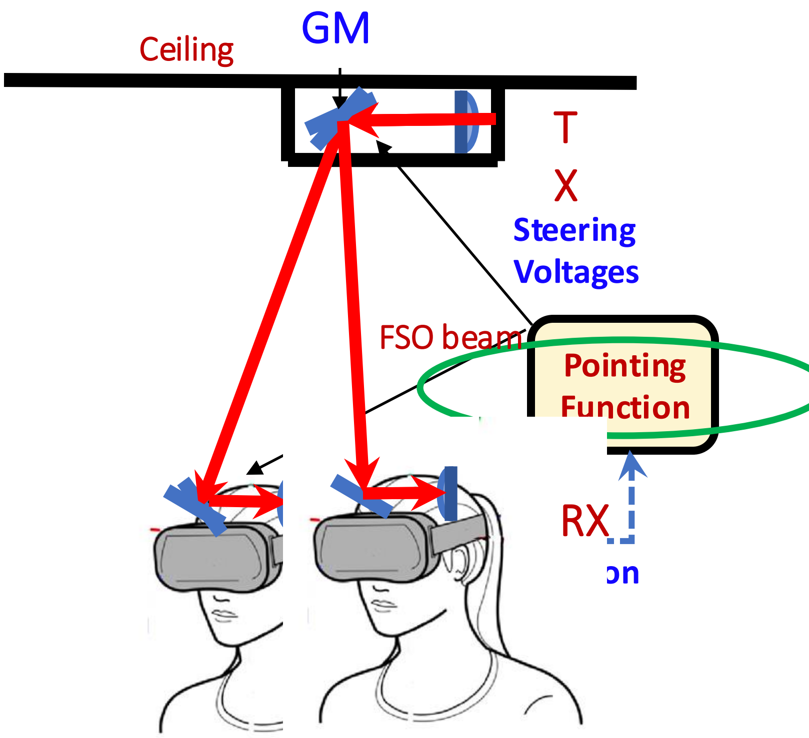
- Let’s take an example scenario with Free space optics (FSO)
  - Narrow laser links, collimated beams

FSO-based VR Wireless Link

- TX (renderer) fixed on ceiling.
- RX (VRH) moves
- To realign the beam:
  a. Localize RX [mm accuracy; via VRH's in-built localization]
  b. Steer TX and RX [using Galvo Mirrors (GMs)]

(2-axis) Galvo Mirror





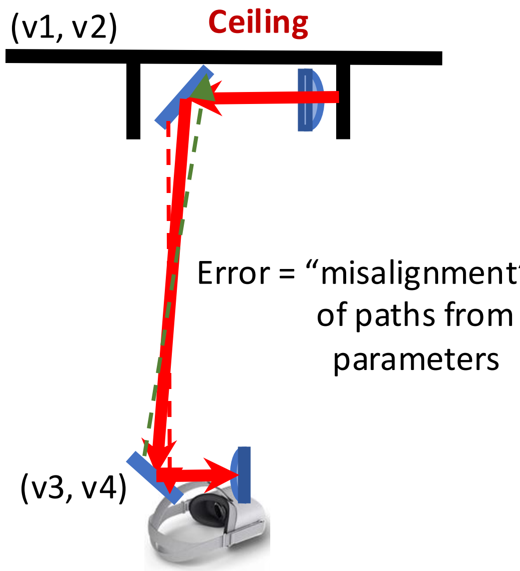
Pointing Function:

- Pointing function P:
  - Input: VRH/RX location [In the unknown VRH coordinate system]
  - Output: 4 GM Voltages [To steer TX and RX to realign beam]
- Learning P directly from (input, output) samples is infeasible
- Our approach:
  1. Learn GM models (two functions G and G’)
     [Offline]
     a) In the GM’s coordinate system (a known space).
     b) Map to the VRH coordinate system.
  2. Use GM functions to compute P.
     [Real-time]

1b. Map GM Functions to VRH Space

- Tantamount to estimating GMs' positions in VRH space.
- Need to estimate 12 parameters (6 for each GM).

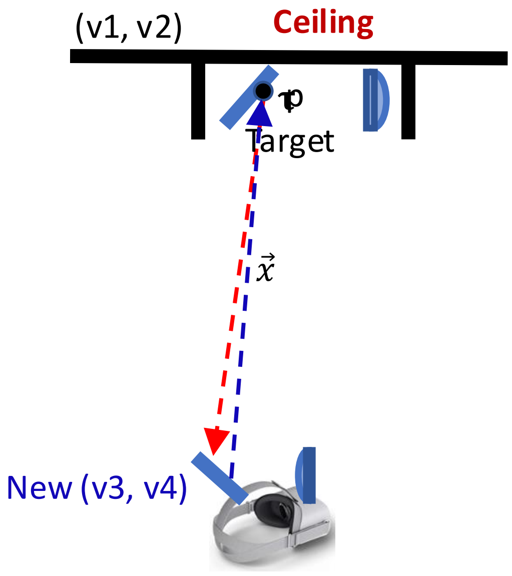
1. Gather training samples (aligned beam state).
   - (VRH Position, 4 voltages) for each sample.
2. Define an <u>error function</u> for given parameter values.
3. Determine parameter values that minimize the total loss over samples.



2. Pointing Function P from GM Functions

Pointing Function P:

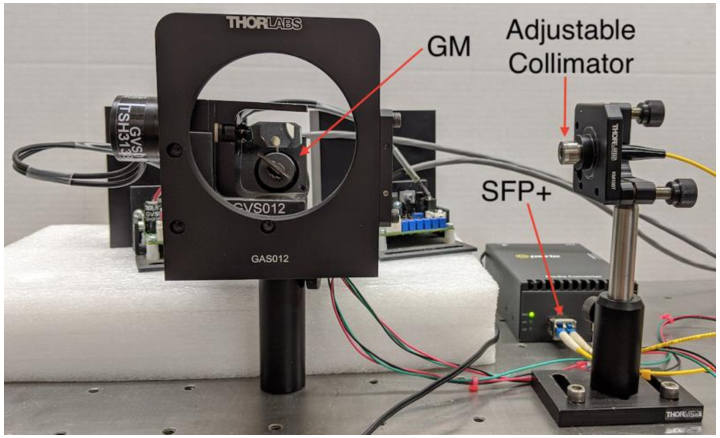
- Input: VRH position.
- Output: 4 Voltages.
- Approach (Real-Time):
    - Initialize voltages v_1, v_2, v_3, v_4
    - $(p, \vec{x}) = G(v_1, v_2)$ TX-beam output specs
    - New $(v_3, v_4) = G'(\boldsymbol{\tau} = p)$ RX-beam should hit p.
    - Similarly, compute new (v_1, v_2).
    - Iterate.



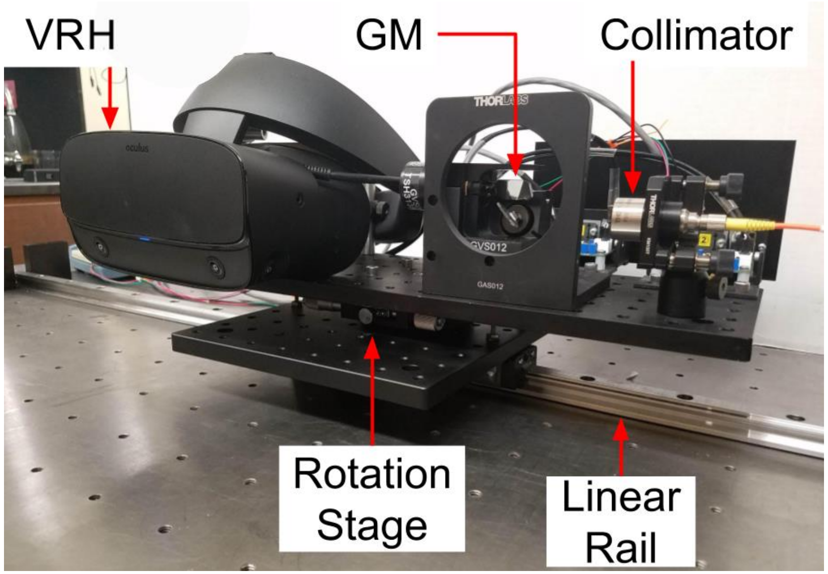
FSO-VR Prototype Design

- Link Design
  - Divergent beam offered higher movement tolerance.
  - 10 and 25 Gbps links.
- Prototype:

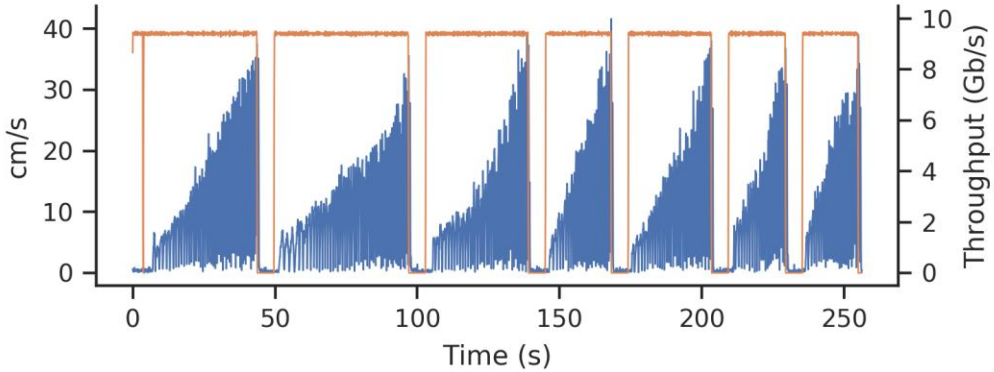
TX

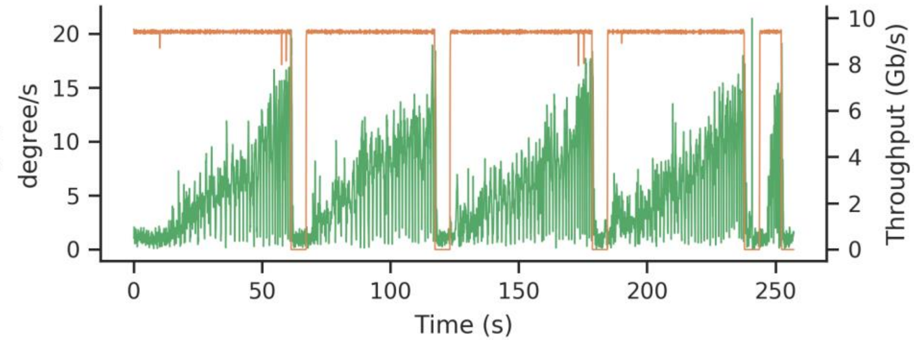


RX



FSO-based VR Link Performance

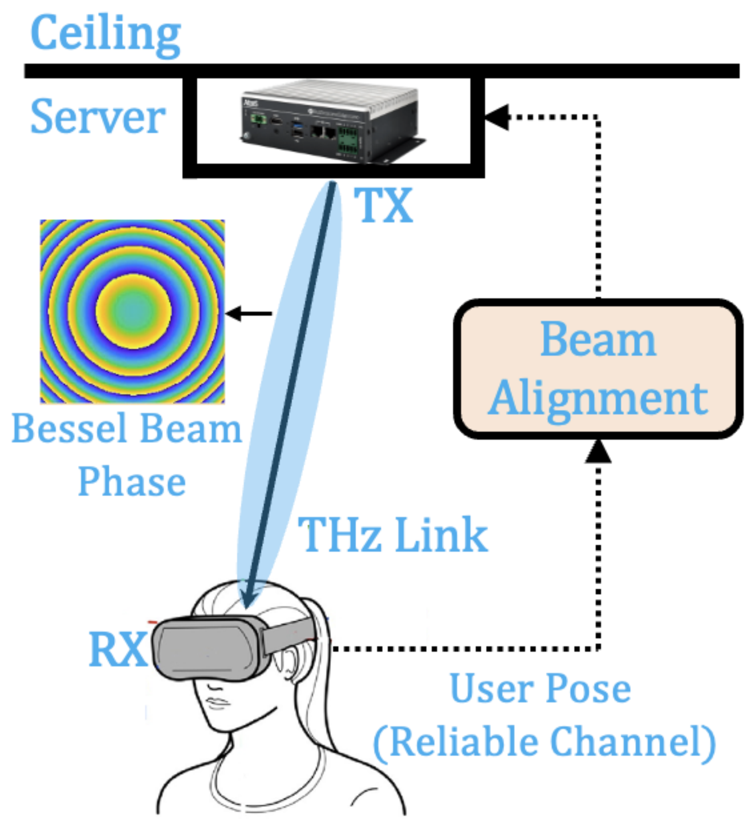




- Performance could be much improved, with customized components.
    - E.g., higher tracking frequency, customized optical components.

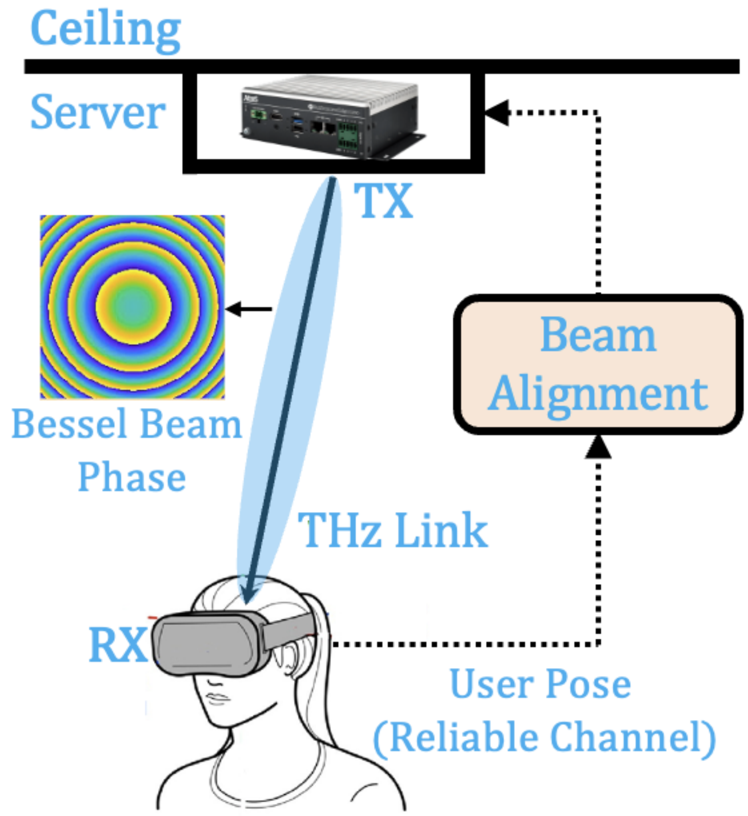
THz Band based VR Link

- Above 100GHz Radio frequencies
  - Affected smaller obstacles e.g., raindrops or atmospheric effects, in additional regular blockage issues

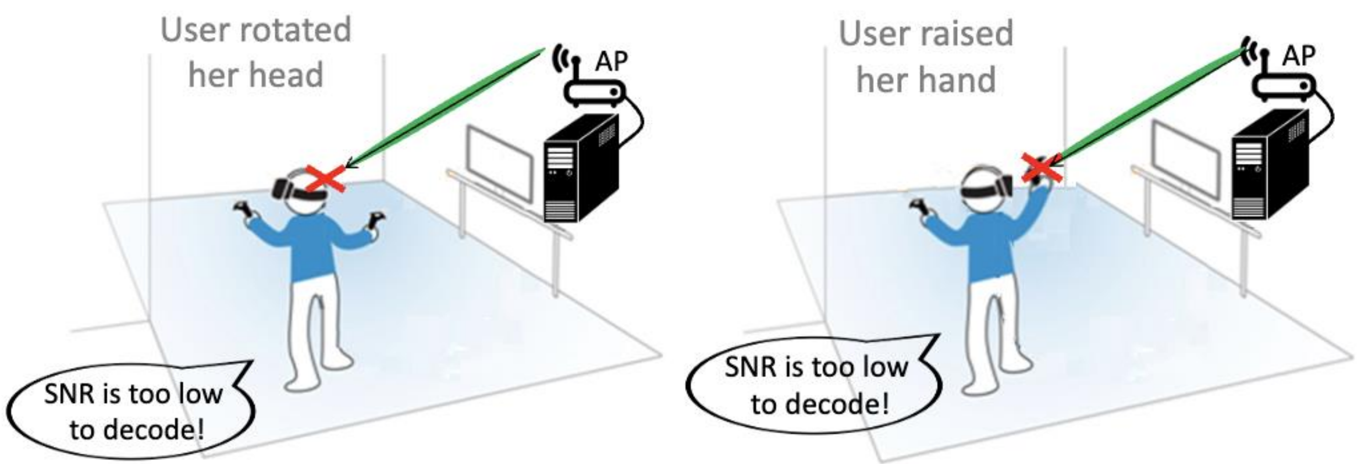


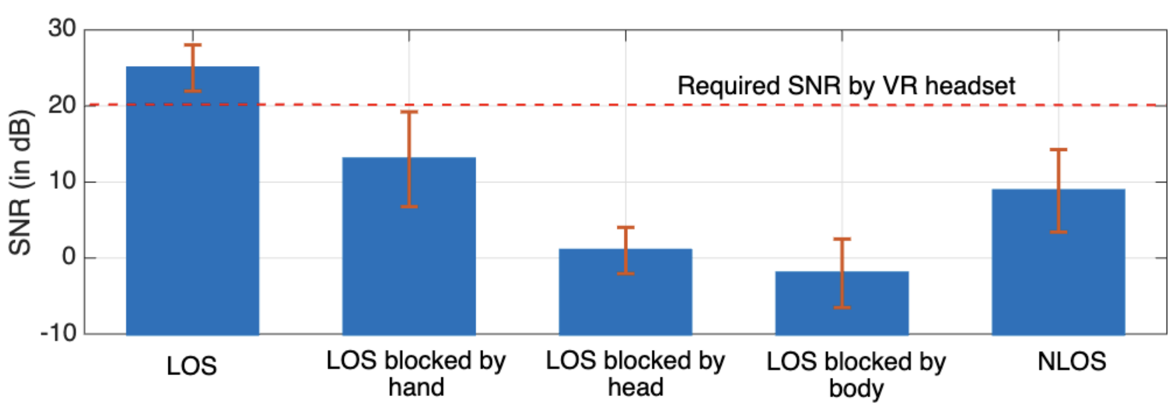
THZ Band based VR Link

- Need Beam alignment algorithms
  - RF anchors can be placed in the environment for absolute location estimate
- Predict, track and point beams based on mobility models



mmWave based VR Links





Abari et.al

mmWave based VR Links

- Build a highly directional antenna by packing multiple antenna elements into an array, and controlling the phase of each element.

mmWaves based VR Links

- HTC Vive

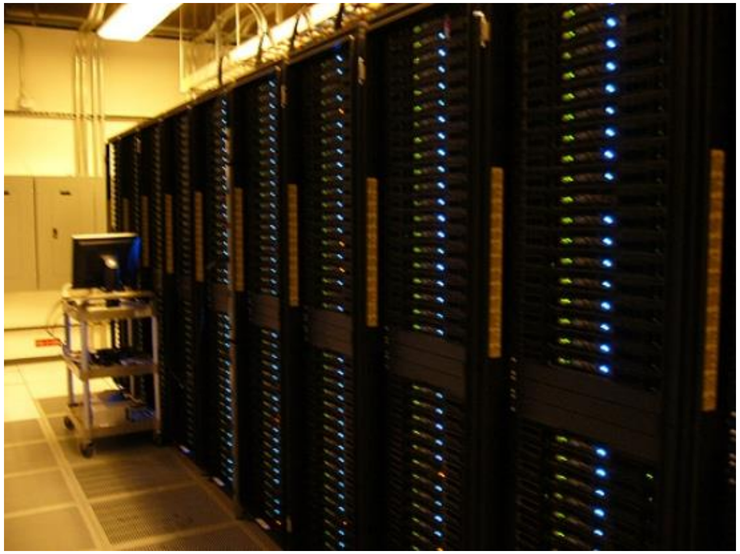
Distributed or Parallel Rendering

- Splitting rendering tasks across multiple machines or nodes, often used in high-end graphics production and complex simulations.
  - Each node processes a portion of the rendering task, and the results are combined to produce the final image or animation.

Distributed or Parallel Rendering

- Pixar’s RenderFarm

Render their
big-screen 3d
animated
films

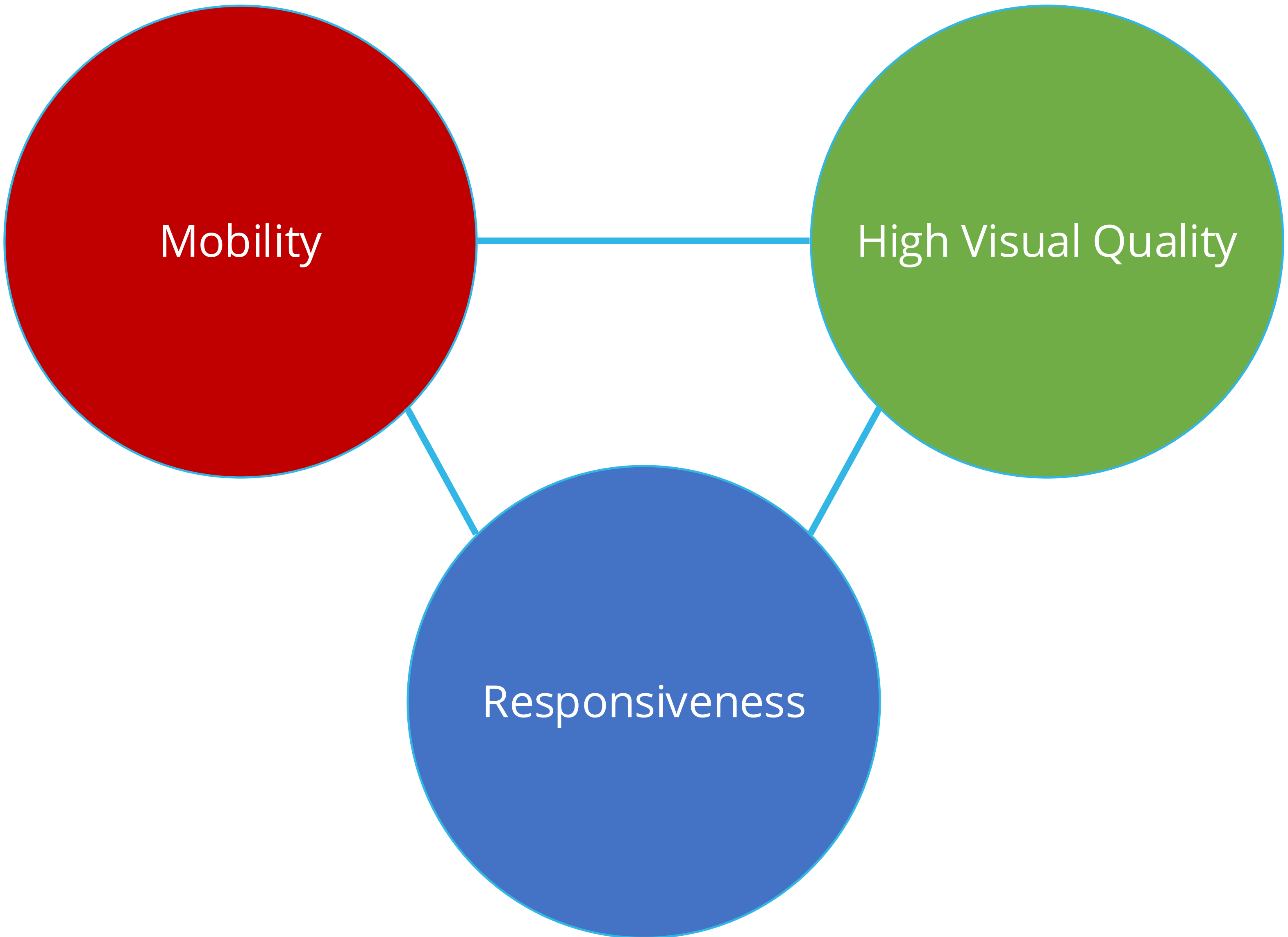


Types of Rendering

- Remote Rendering
  - Edge Rendering
  - Cloud Rendering
  - Distributed Rendering
- Local Rendering

- How about Hybrid Rendering?

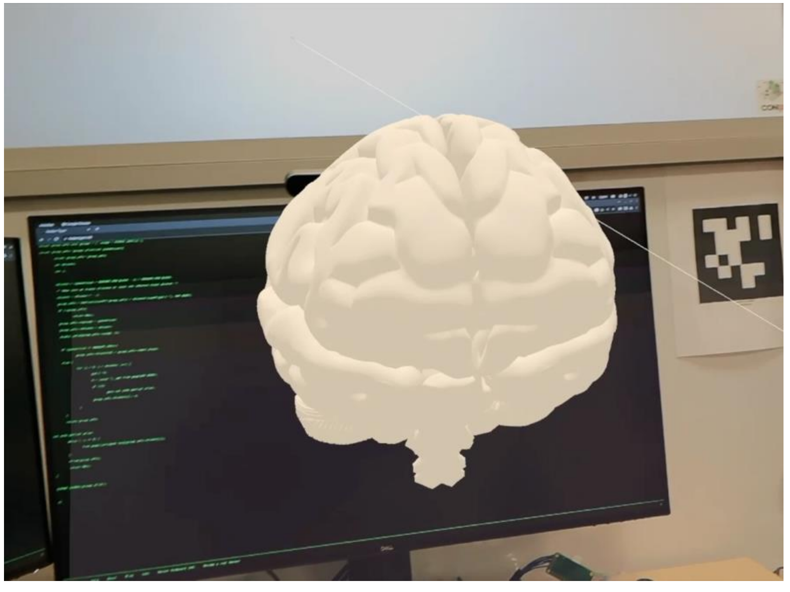
Real-Time Rendering for XR



Local Rendering

✓ Low latency interactions
X Low object complexity

Recorded on Magic Leap 2



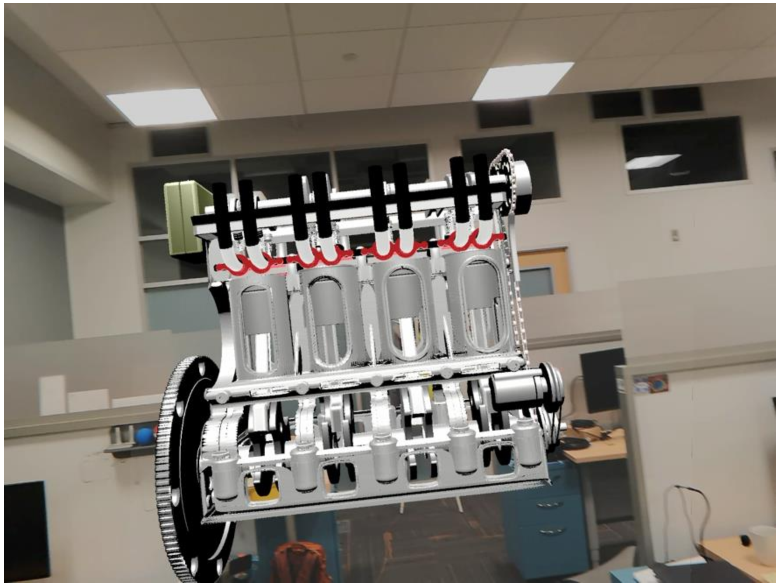
50K triangles

Local Rendering

✓ Low latency interactions

X Low object complexity

Recorded on Magic Leap 2



11M triangles

50K triangles

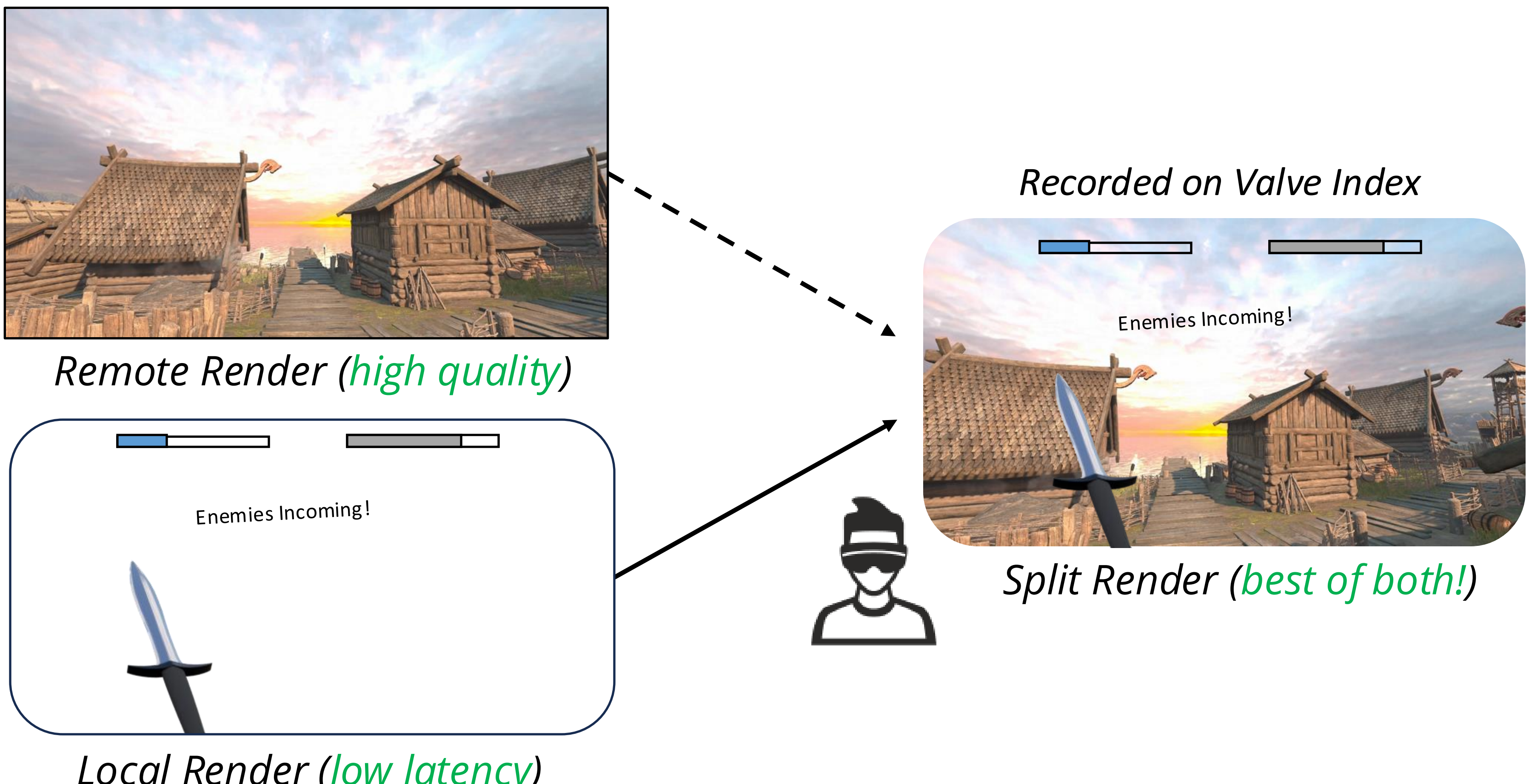
Remote Rendering

Recorded on Valve Index

- ✓ High quality
- X High latency (Device→Network→Server→Network→Device)
- X Reprojection needed to mask latency
- X Networks can be unreliable

Split Rendering

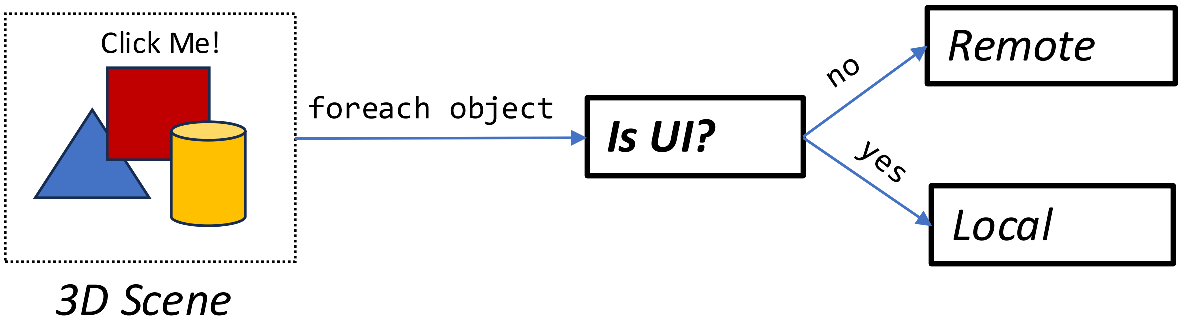
Split scene into **local** and **remote** portions



State-of-the-art Split Rendering

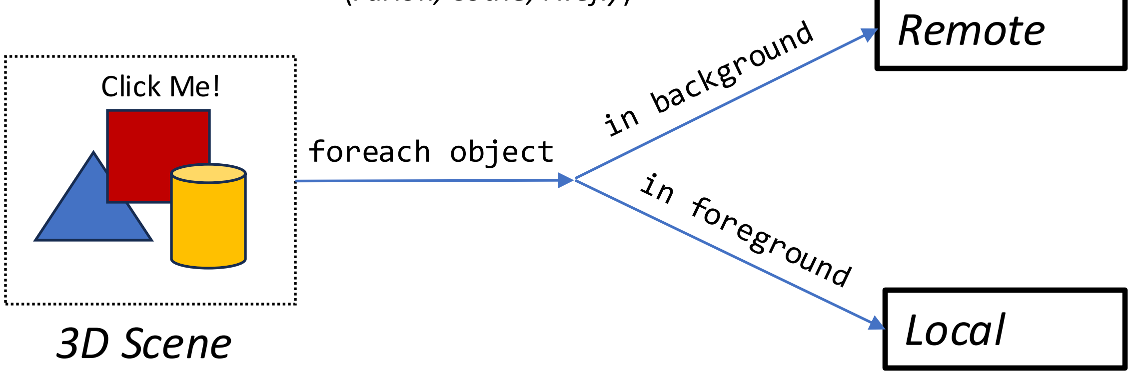
(1) Static determination of what gets rendered where:

(*Azure Hybrid Rendering*)



(2) Distance-based determination of what gets rendered where:

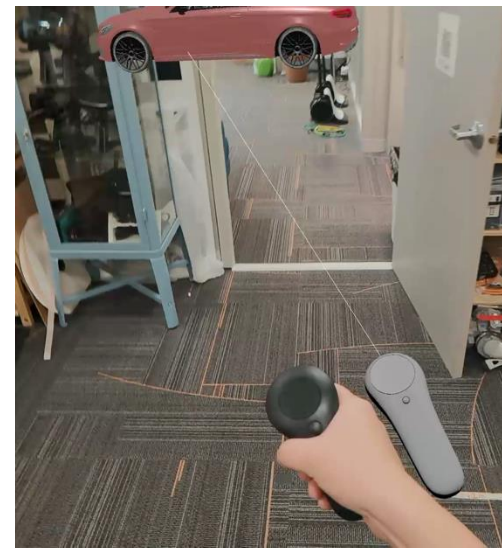
(*Furion, Cotrie, Firefly*)



Recorded on Magic Leap 2

State-of-the-art Split Rendering

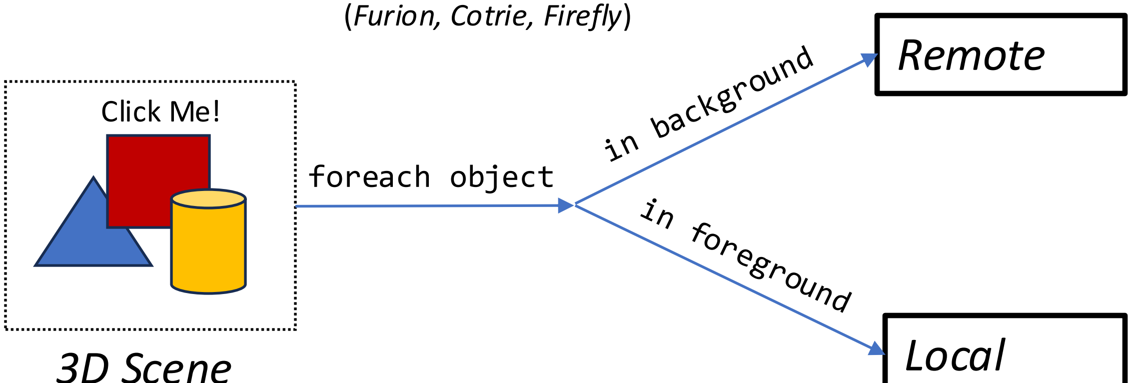
Coterie, Asplos'20



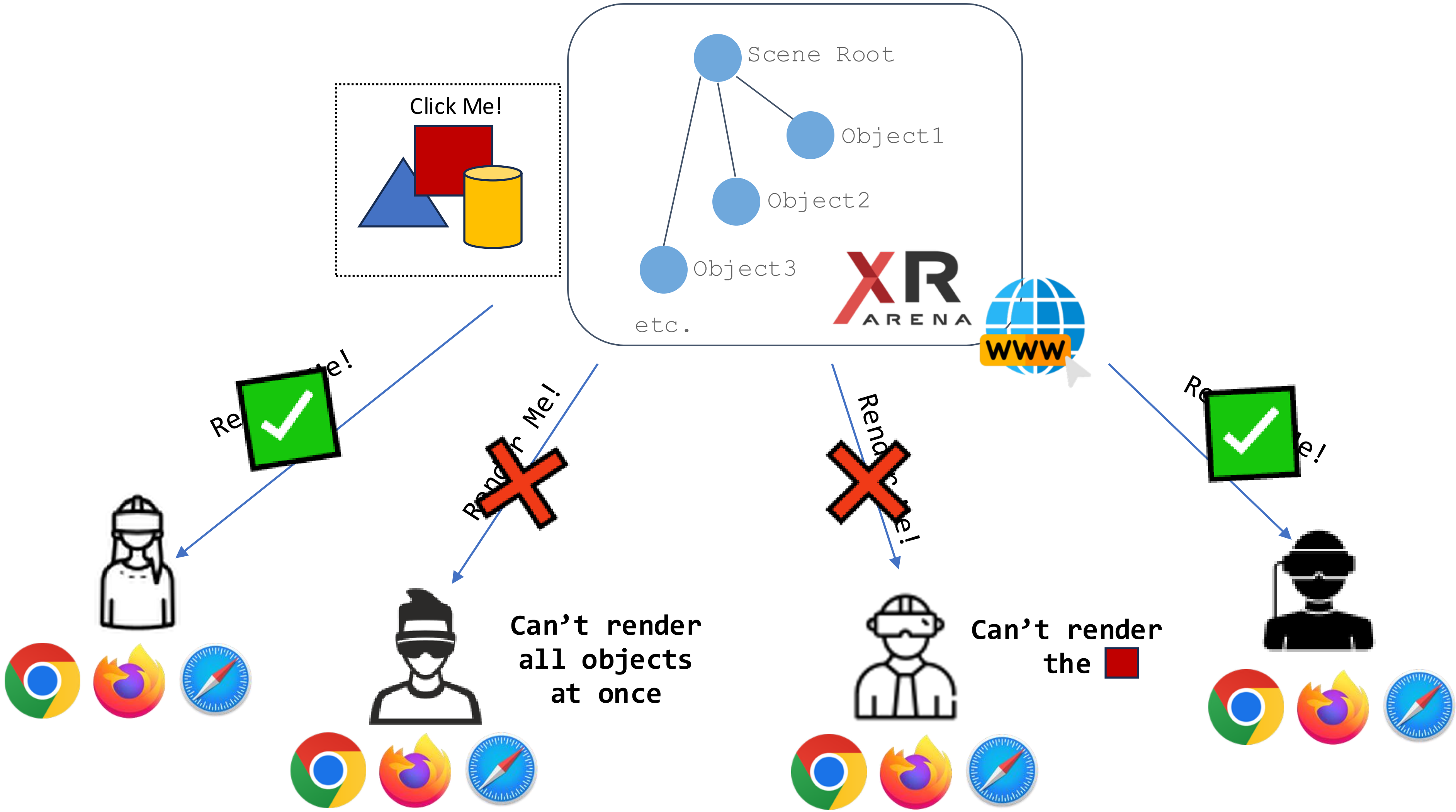
Recorded on Magic Leap 2

(2) Distance-based determination of what gets rendered where:

(*Furion, Cotrie, Firefly*)

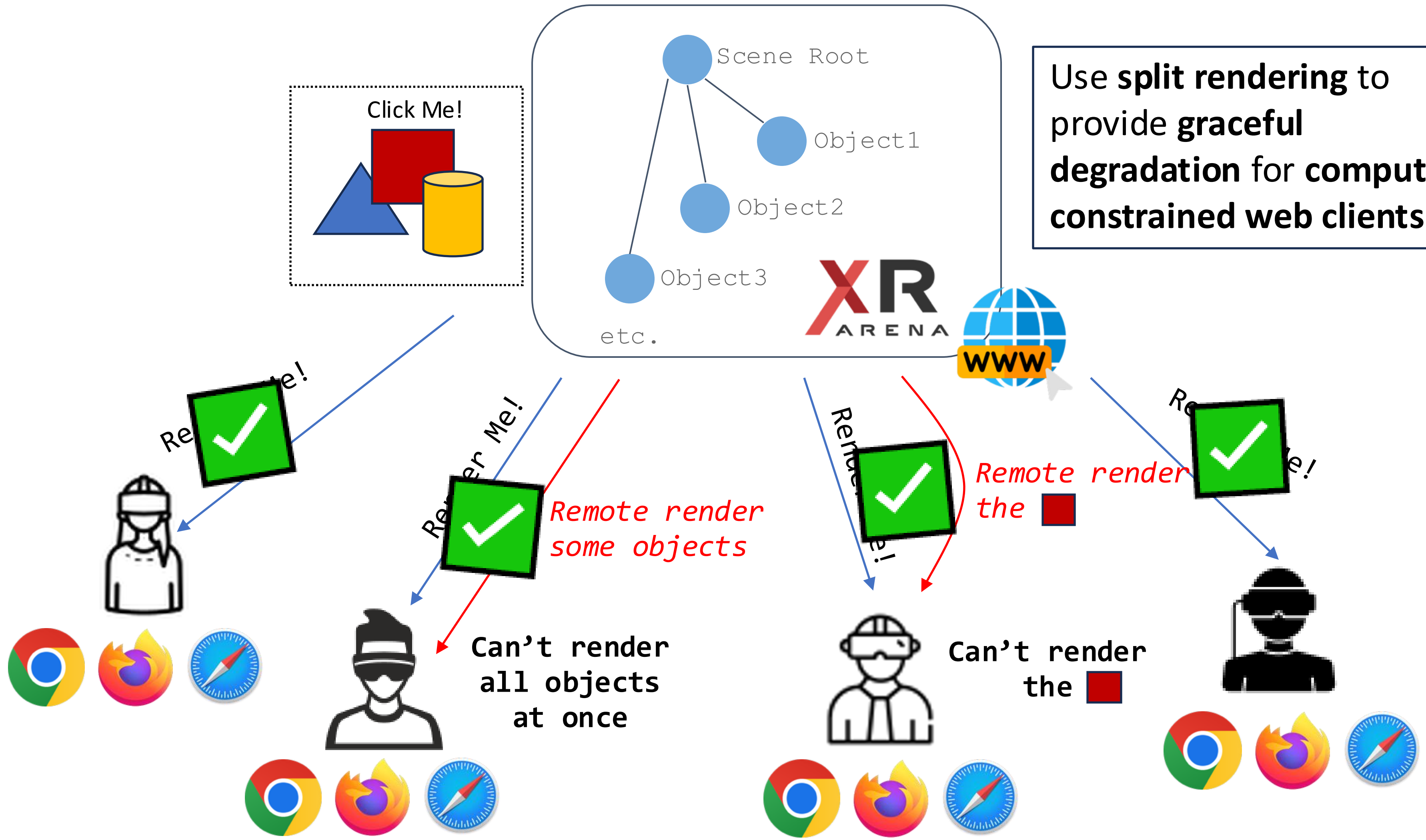


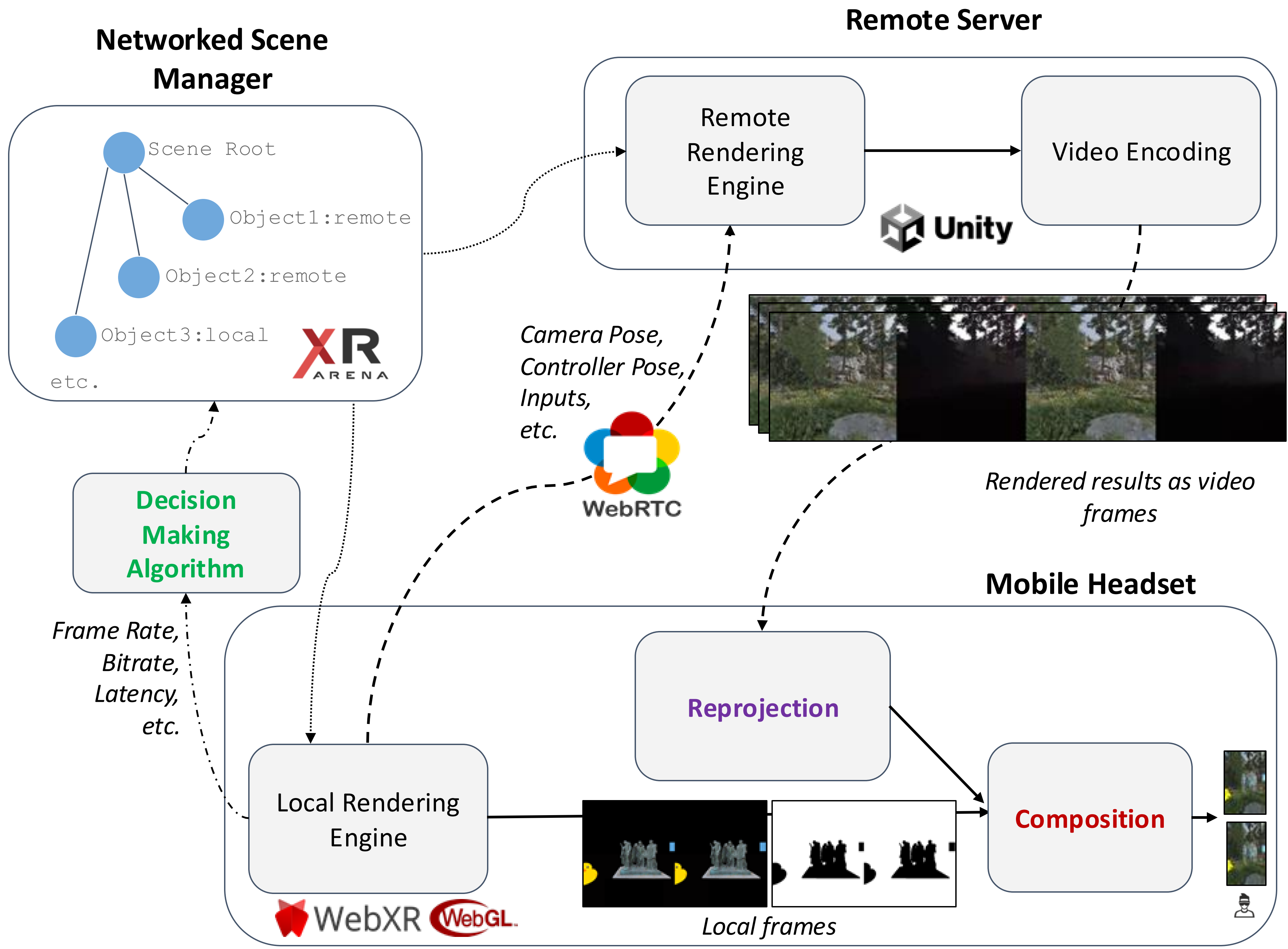
Split Rendering for ARENA



Split Rendering for ARENA

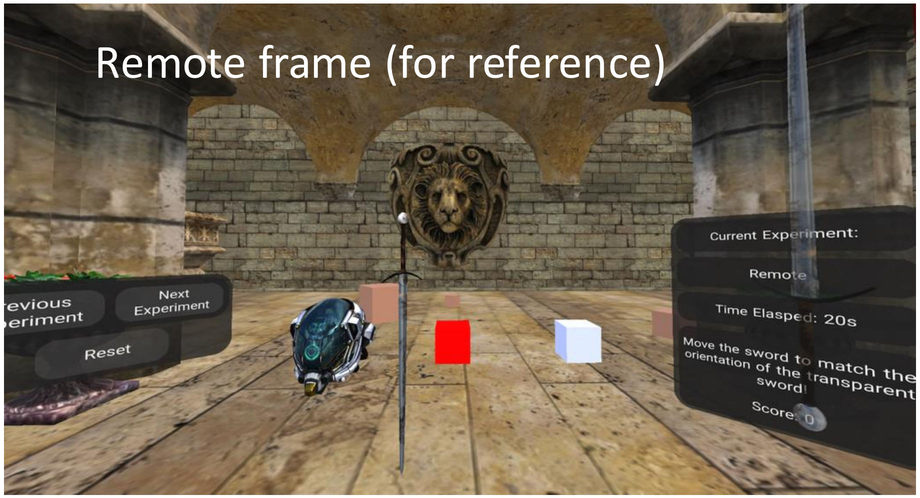
Use **split rendering** to provide **graceful degradation** for **compute-constrained web clients**!

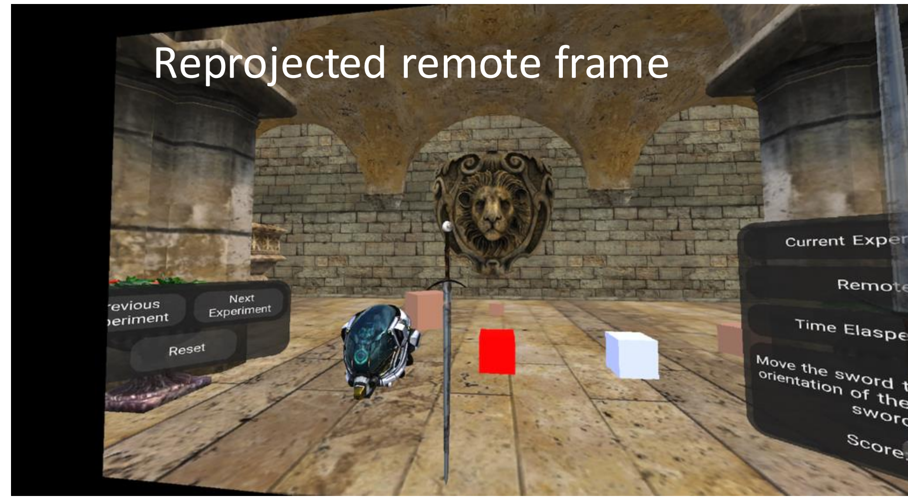


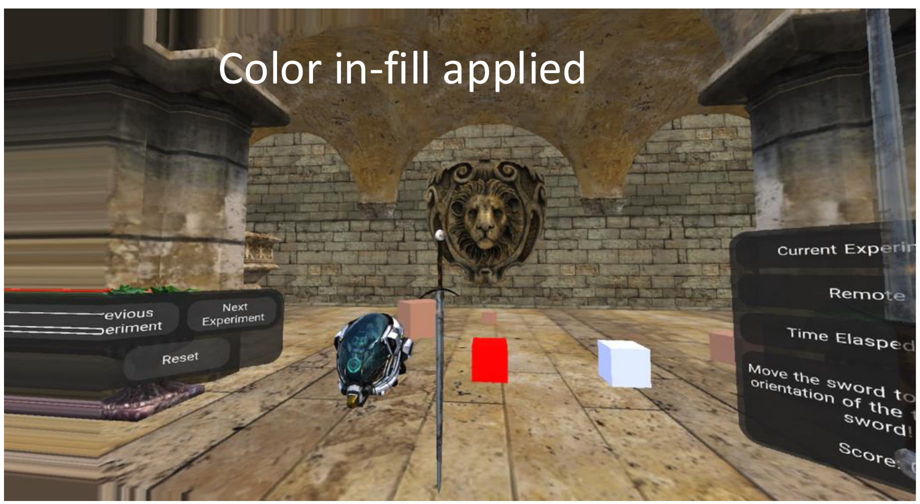


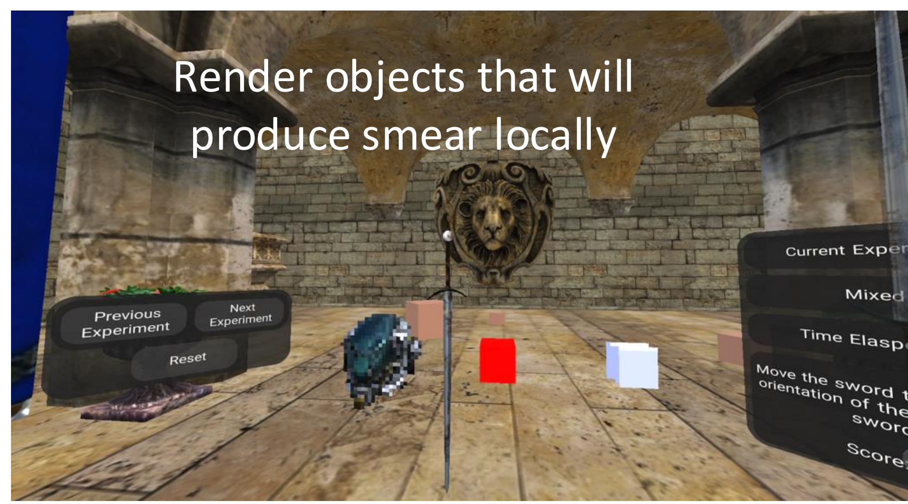
Networked Scene Manager
Scene Root
Object1:remote
Object2:remote
Object3:local
etc.
XR ARENA
Remote Server
Remote Rendering Engine
Video Encoding
Unity
Camera Pose, Controller Pose, Inputs, etc.
WebRTC
Rendered results as video frames
Decision Making Algorithm
Frame Rate, Bitrate, Latency, etc.
Mobile Headset
Reprojection
Local Rendering Engine
Composition
WebXR
WebGL
Local frames

Reprojection with ATW









Object Rendering Mediums

(a) r = *HL* (b) r = *LL* (c) r = *R* (under low bitrate)

In RenderFusion, an object could be one of three representations, *r*:

Representation	Local Resource Usage	Visual Quality	Response Latency
Highpoly Locally Rendered (HL)	High	High	Low
Lowpoly Locally Rendered (LL)	Low	Low	Low
Remotely Rendered (R)	Very Low	High (under good bitrates)	High

Decision Making Algorithm

Which representation (*r*) is best?

For all objects in the scene

$$\begin{aligned} \max \quad & \sum_{o \in O} A(o) B(o, r) \\ \text{s.t.} \quad & \sum_{o \in O_L} \text{Polycount}(o) \leq \text{MaxLocalPolycount} \end{aligned}$$

Goal: **Find *r* for each object to maximize sum of total *benefit*, *B(o,r)*, scaled by object size relative to viewport, *A(o)***

Decision Making Algorithm

<u>Which representation (*r*) is best?</u>

is less than the max polygons that can be rendered locally within a target frame rate

$$\begin{array}{ll} \max & \displaystyle\sum_{o \in O} A(o)B(o,r) \\ \text{s.t.} & \displaystyle\sum_{o \in O_L} \text{Polycount}(o) \leq \text{MaxLocalPolycount} \end{array}$$

Ensuring that total polycount of all locally rendered objects…

Perceptual Study: Quality

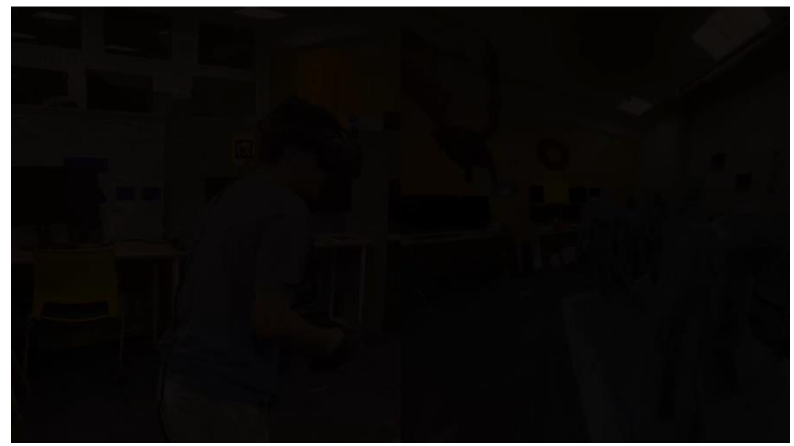
Pure Local

Perceptual Study: Quality

Pure Remote

Perceptual Study: Latency

Pure Local



Pure Remote

Perceptual Study: Latency

RenderFusion

Summary of the Lecture

- Different types of rendering
- Rendered video streaming over wireless
- Rendering Performance
- Hybrid or Split Rendering
- Scheduling Objects for Local and Remote Rendering
- Compute and Network Adaptation

Next up: Tracking algorithms