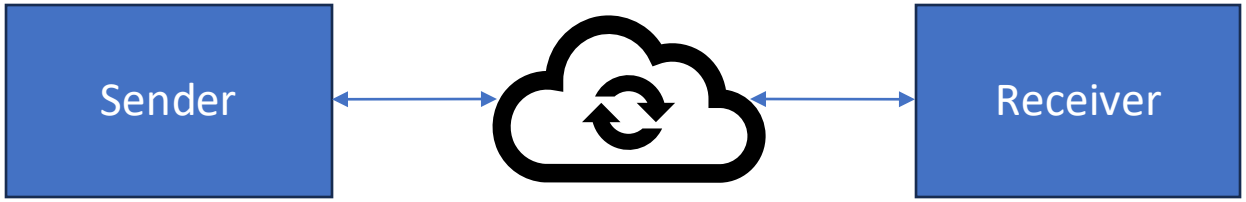
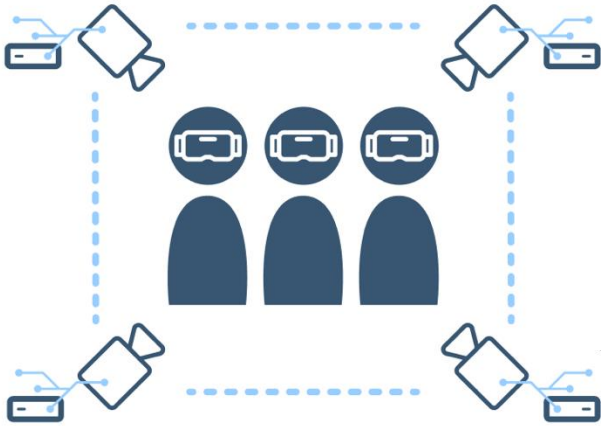
EECE5698
Networked XR Systems

Some of the slide credits: Anthony Rowe

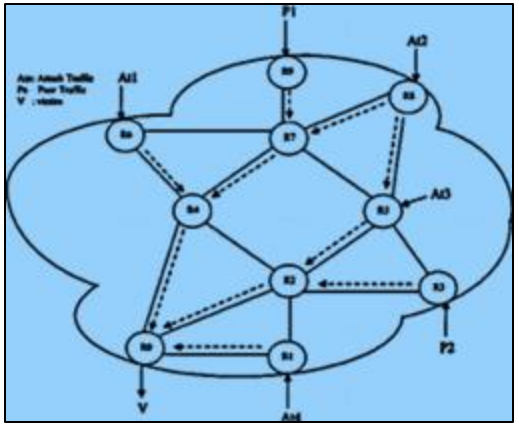
Networked XR System



Classical networked system pipeline



Digitize 3D spaces



Network Transport

Display Interfaces

Lecture Outline for Today

- Rendering Basics
- Edge/Cloud/Remote Rendering

Rendering Basics

- “Rendering or image synthesis is the process of generating a photorealistic or non-photorealistic image from a 2D or 3D model by means of a computer program. ”

Rendering Basics

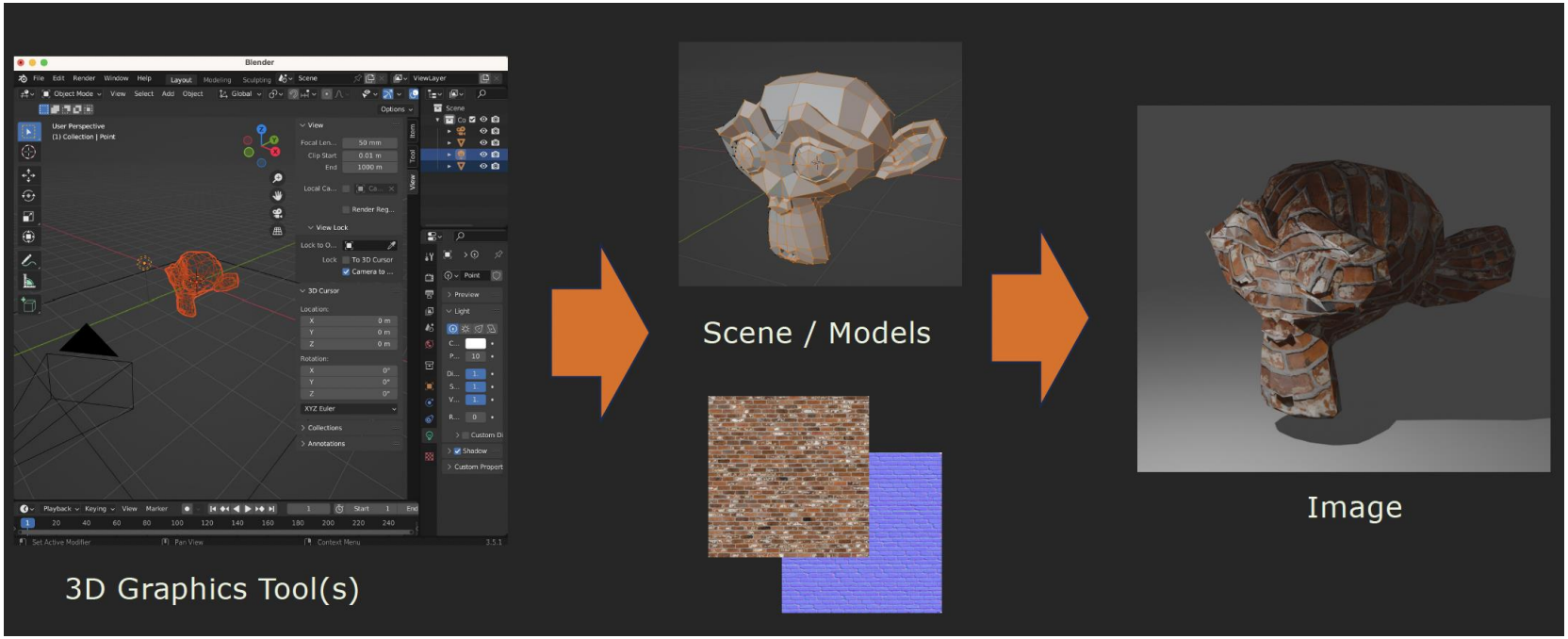
- Rendering is crucial in various fields such as video games, simulations, movie production, and virtual reality, providing the final appearance of models and scenes with textures, colors, and lighting.
- Key Components:
  - **Models:** The geometric data representing 3D objects.
  - **Textures:** The surface details that give materials their appearance.
  - **Lighting:** The simulation of light to create shadows, highlights, and color variations.

Rendering Basics

- **Real-time Rendering:**
  - Used in video games and interactive graphics where images must be generated at a rapid pace, typically 30 to 60 frames per second.
  - Prioritizes speed over image quality, employing various optimizations to achieve smooth performance.
- **Offline (Pre-rendered) Rendering:**
  - Used in situations where image quality is paramount, such as in feature films and high-quality animations.
  - Takes more time to produce a single frame but achieves higher levels of detail and lighting accuracy.
- **Ray Tracing vs. Rasterization:**
  - **Ray Tracing:** Simulates the physical behavior of light to produce more realistic images, calculating reflections, refractions, and shadows.
  - **Rasterization:** Converts 3D models into 2D images quickly, often used in real-time rendering, but less capable of complex light interactions compared to ray tracing.

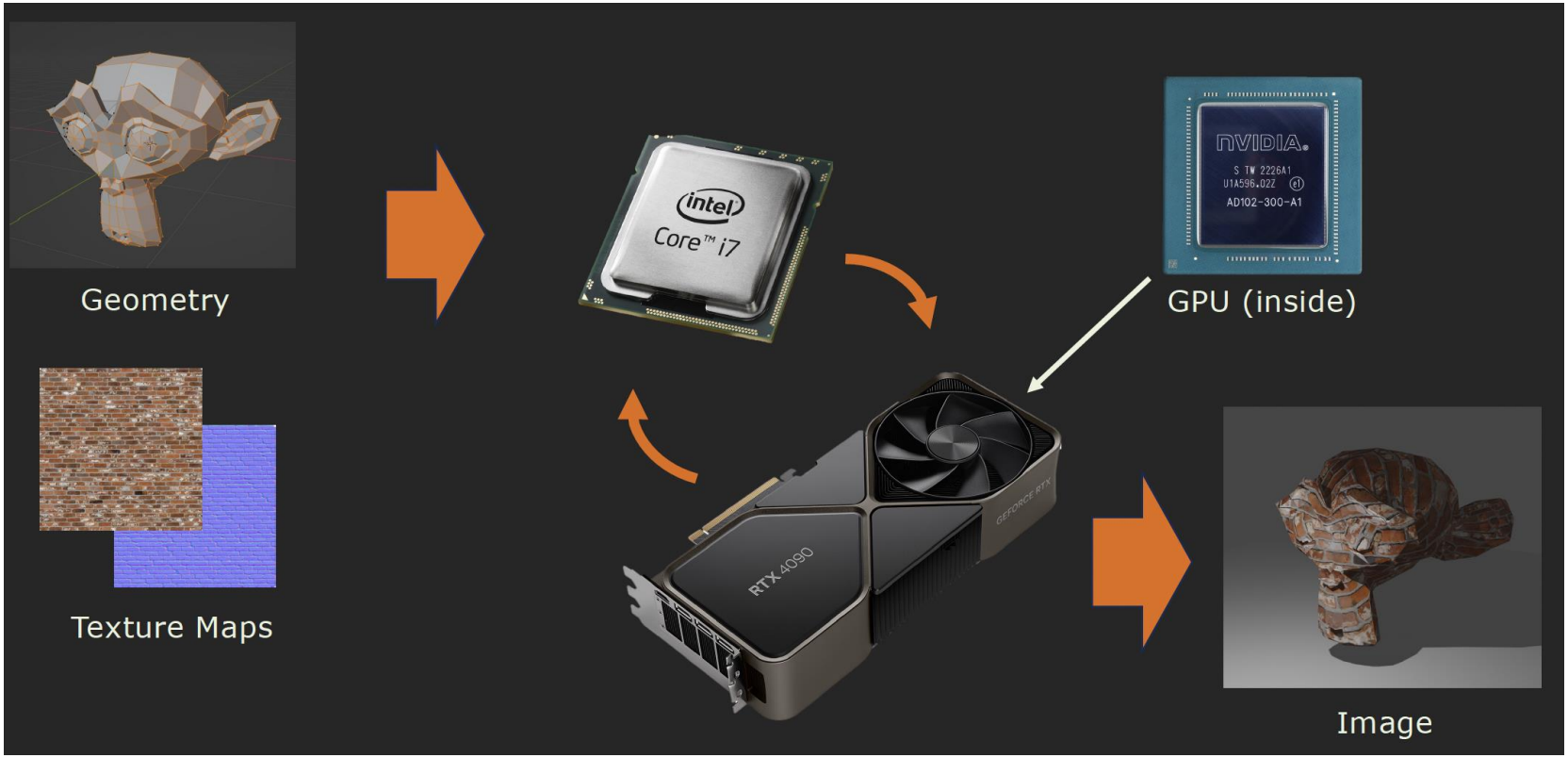
Rendering Basics

High level



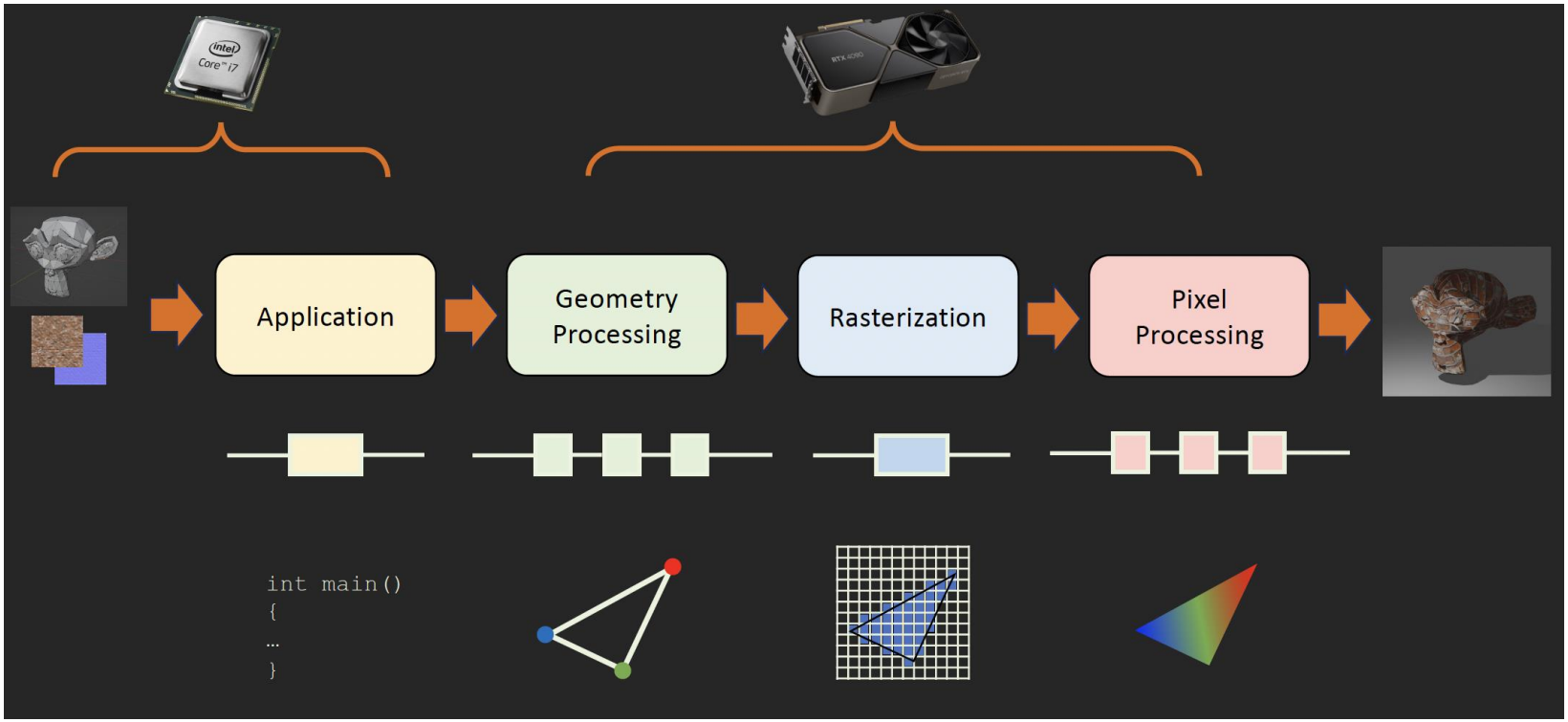
Rendering Basics

Hardware view



Rendering Basics

- Rendering pipeline

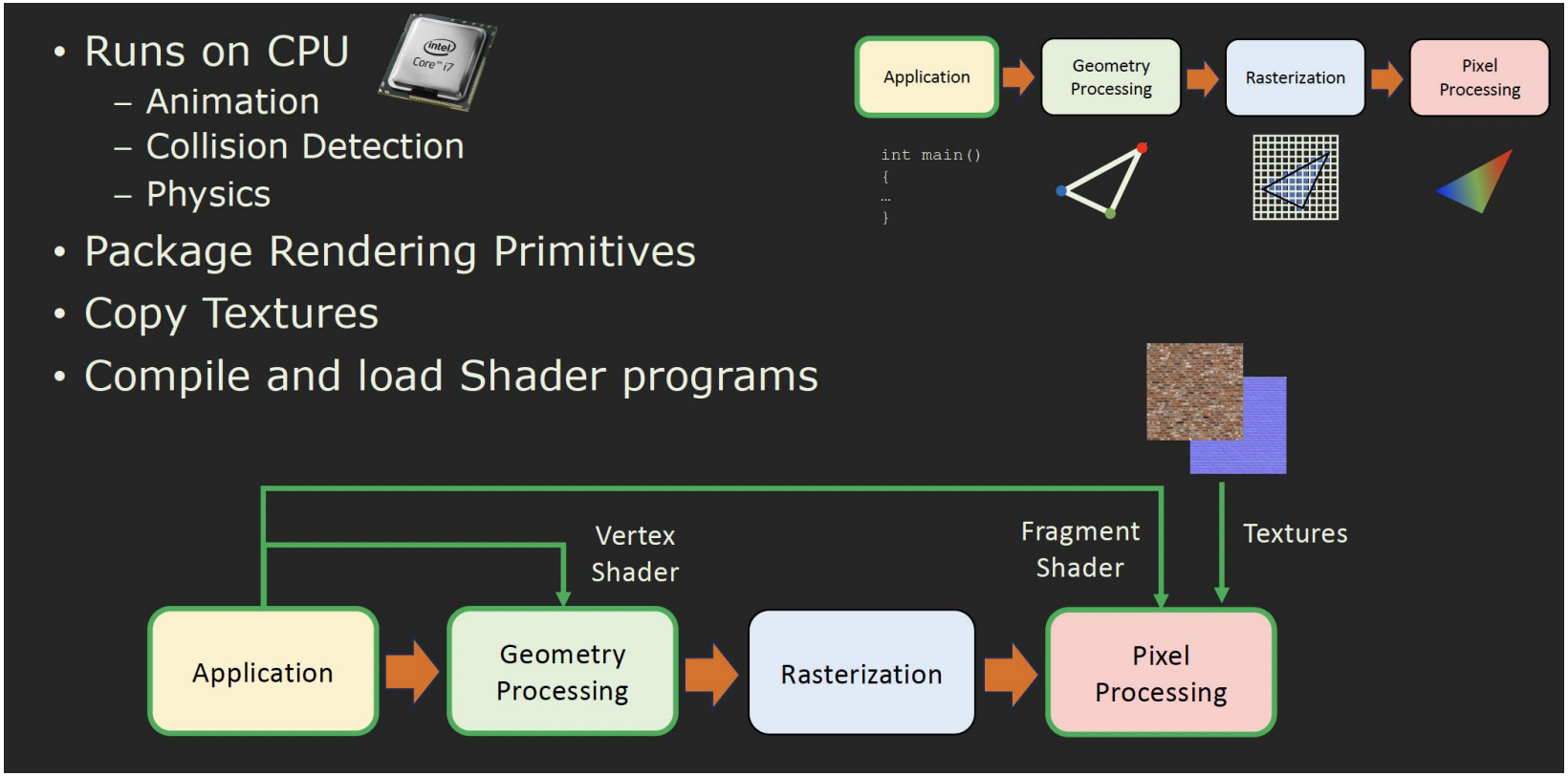


Rendering Basics – Key Steps

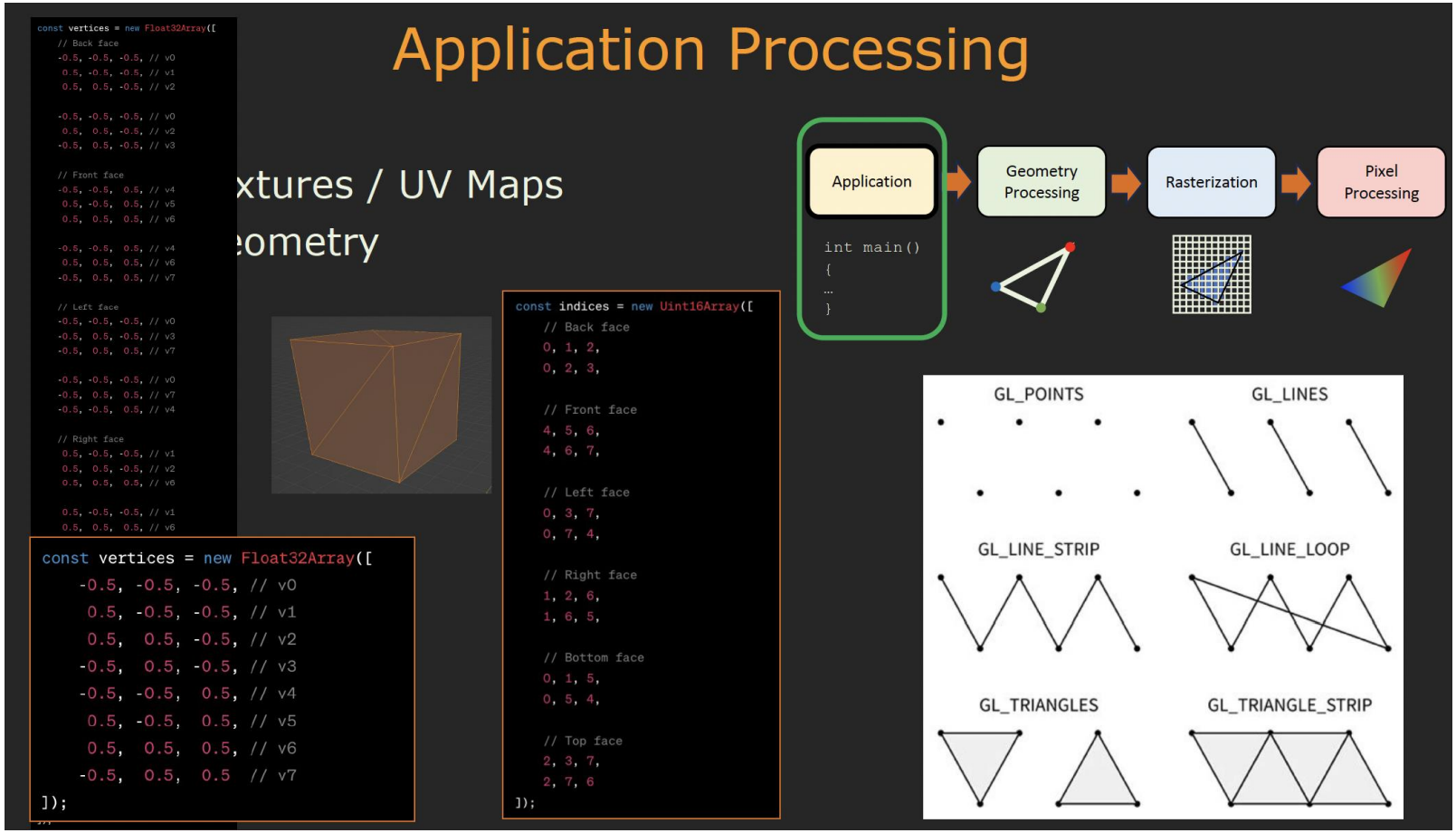
1. **Model Loading:** Importing 3D models into the rendering engine.
2. **Scene Setup:** Positioning models, lights, and cameras within the scene.
3. **Geometry Processing:** Transforming 3D coordinates to 2D screen space.
4. **Rasterization:** Converting 3D models into pixels on a 2D surface.
5. **Shading:** Applying textures, colors, and lighting effects.
6. **Output:** Rendering the final image for display or storage.

Rendering Basics

- Application processing



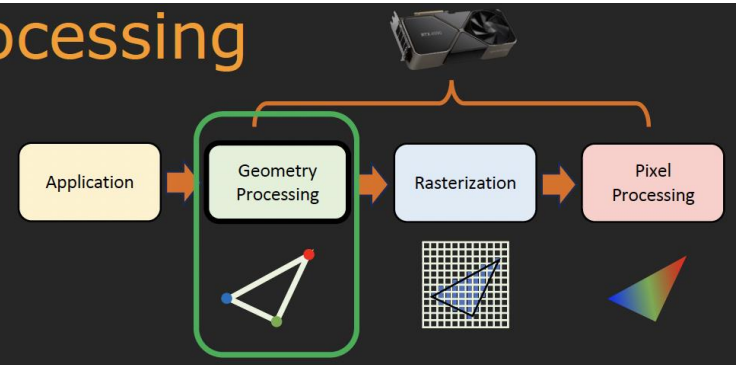
Rendering Basics

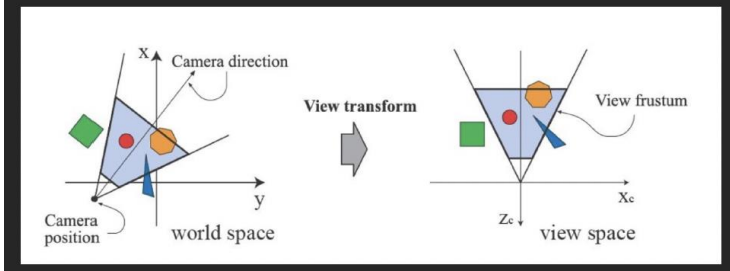


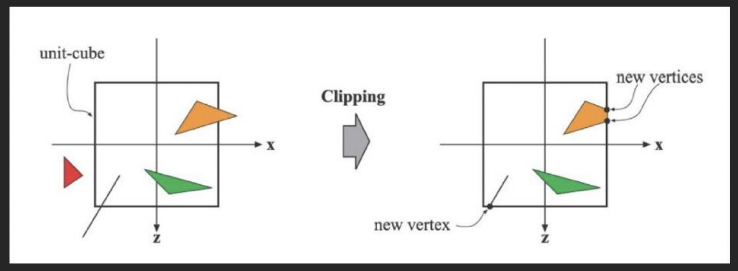
Rendering Basics

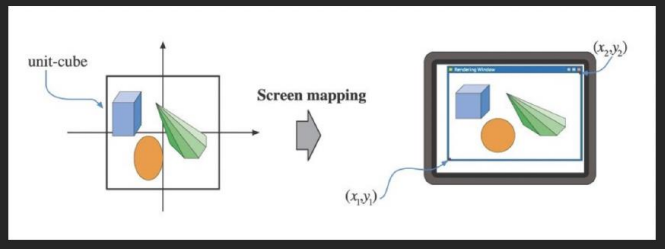
Geometry Processing

- Per-Triangle and Per-Vertex Ops
  - Vertex Shader (and others)
  - Coordinate Transform
  - Clipping
  - Screen Mapping

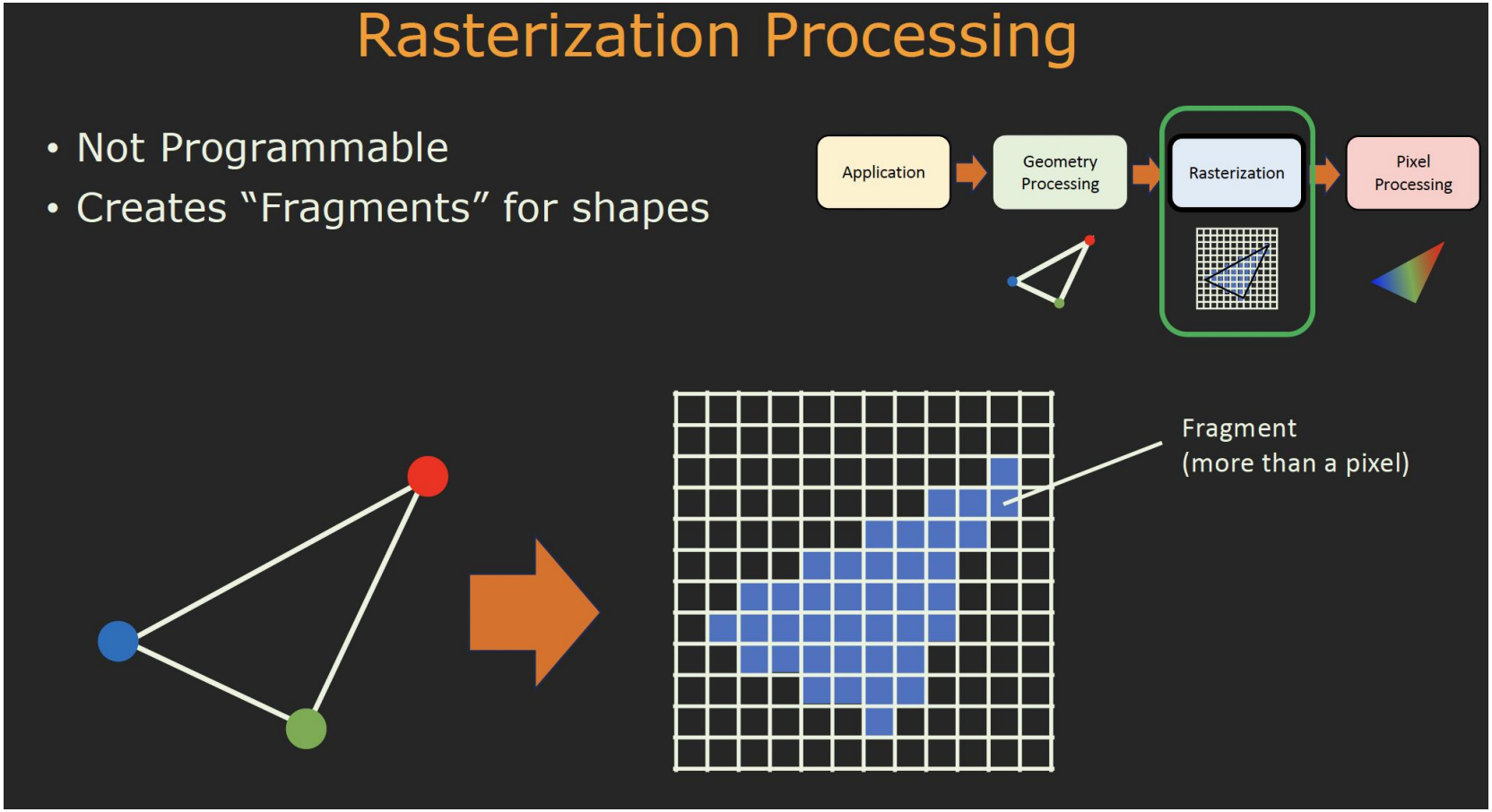








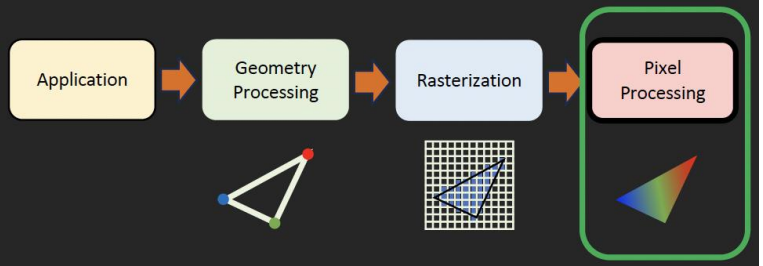
Rendering Basics

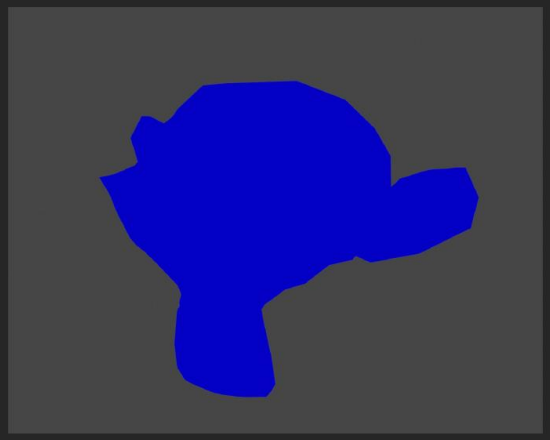


Rendering Basics

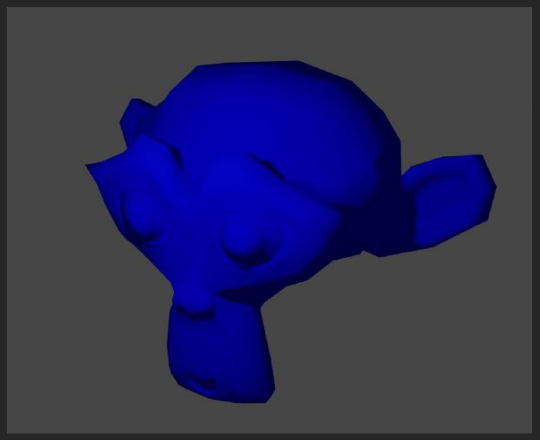
Pixel Processing

- Responsible for coloring pixels





No Shading



Diffuse Shading

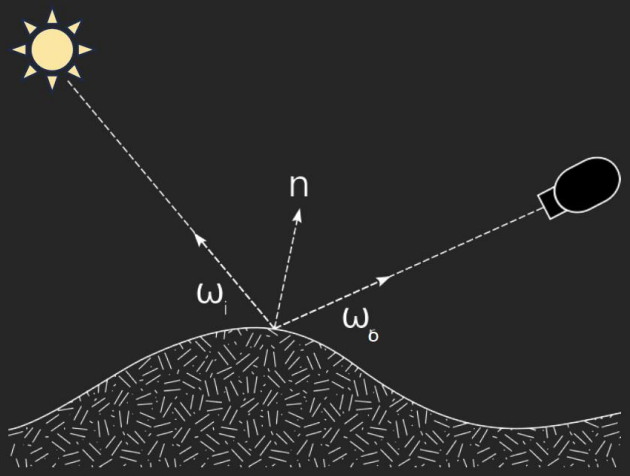
Rendering Basics

The Rendering Equation (simple)

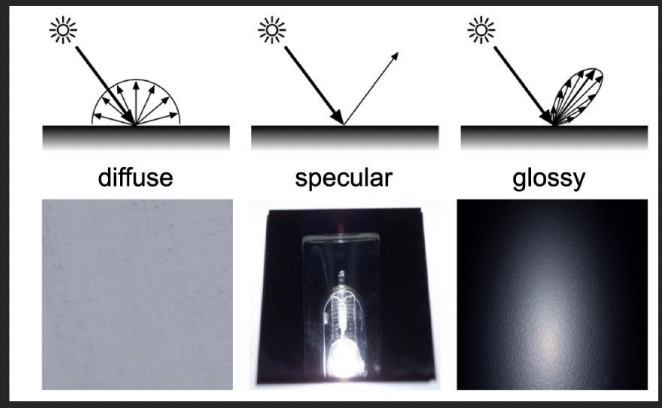
Bidirectional Reflectance Distribution Function (BRDF)

$$L_o(\omega_o) = \int_\Omega L_i(\omega_i) \cos\theta_i \; f_r(\omega_i, \omega_o) \; d\omega_i$$

Geometry Term



Cem Yuksel, Utah: https://www.youtube.com/watch?v=GOfzX7kRwys



Rendering Basics

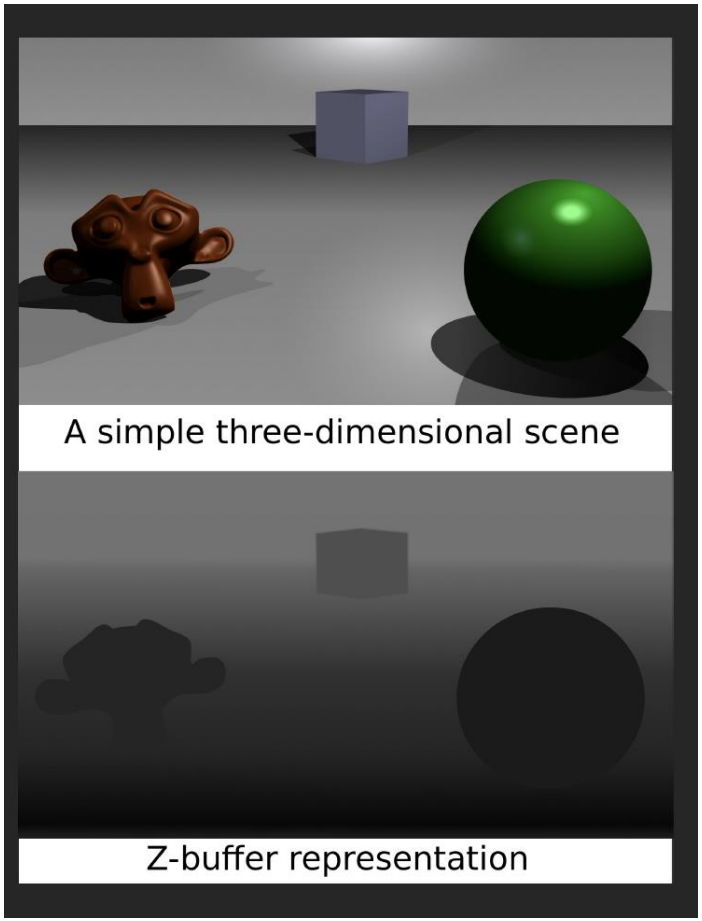
- Lighting and Shadows
  - Lighting Types:
    - Ambient: Soft, directionless light that simulates indirect lighting.
    - Directional: Parallel light rays, mimicking sunlight.
    - Point: Light emitted from a single point, radiating in all directions.
    - Spot: Light emitted in a cone shape, like a flashlight.
  - Shadows: Essential for depth and realism, with techniques like shadow mapping and ray-traced shadows to simulate how objects block light.

Rendering Basics

Z-Buffer – Composite depth map after rasterization

Z-Culling – Early z-test to avoid pixel processing

Z-Test – Depth test in pixel processing stage



Rendering Basics

- Shaders and the GPU
  - Shaders: Small programs that run on the GPU to perform custom rendering effects, such as lighting calculations, texturing, and color adjustments.
  - Types of Shaders: Vertex shaders, Fragment (or Pixel) shaders, Geometry shaders, etc.
  - GPU’s Role: Executes shaders to render images quickly, efficiently handling complex calculations required for realistic rendering.

Rendering Basics

Vertex Shaders

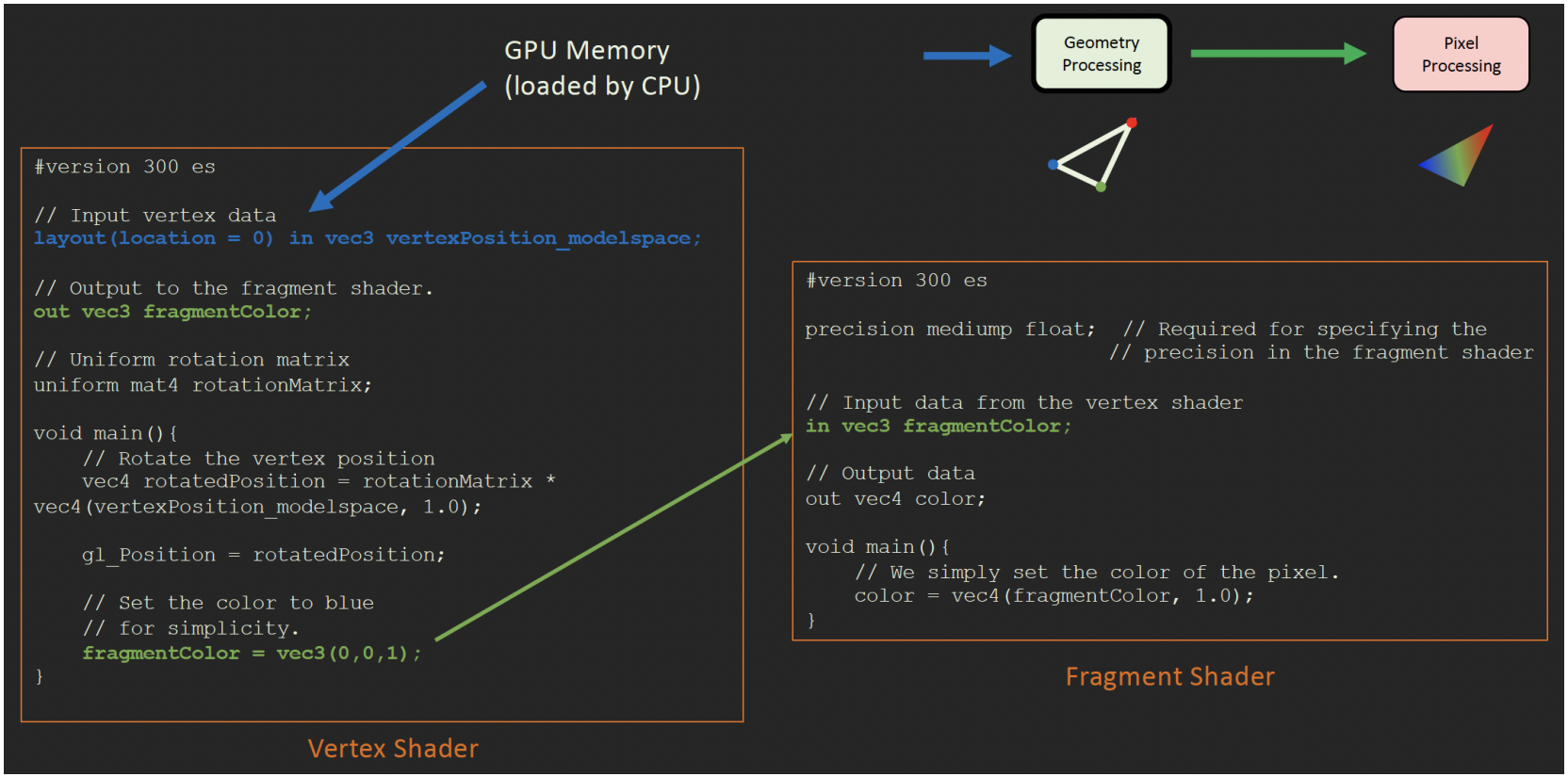
- **Purpose**: Vertex shaders process individual vertices of a 3D model. They handle the transformation and lighting calculations needed to project the 3D coordinates of vertices onto the 2D screen space.
- **Functionality**:
  - Apply transformations like translation, rotation, and scaling to vertices.
  - Adjust lighting properties based on the vertex position in the scene.
  - Pass per-vertex data (like position, normal, and texture coordinates) to the next stages in the pipeline, often including the fragment shader.
- **Operation Level**: Operate on each vertex in the model's geometry, executing once per vertex.

Rendering Basics

Fragment shaders

- **Purpose**: also known as pixel shaders, operate on fragments, which are potential pixels on the screen. They determine the final color and other attributes of each pixel, including texturing and lighting effects.
- **Functionality**:
  - Calculate the color of each pixel based on textures, lighting, and the material properties.
  - Implement detailed surface effects like glossiness, roughness, and ambient occlusion.
  - Often used for complex visual effects like bump mapping, specular highlights, and shadow computation.
- **Operation Level**: Execute once per fragment, which can be more frequent than per-vertex due to the rasterization process producing multiple fragments for each vertex, especially when rendering detailed or close-up views.

Rendering Basics



Rendering Basics

- Post-processing effects: Techniques applied to the rendered image before final output to enhance visual quality or achieve specific artistic styles.
- Common Effects:
  - Bloom: Simulates light bleeding around bright areas.
  - Motion Blur: Blurs objects based on movement, adding realism or speed sensation.
  - Depth of Field: Blurs parts of the scene not in focus, mimicking camera lens effects.
  - Color Grading: Adjusts the color palette to convey mood or time of day.

Rendering Basics

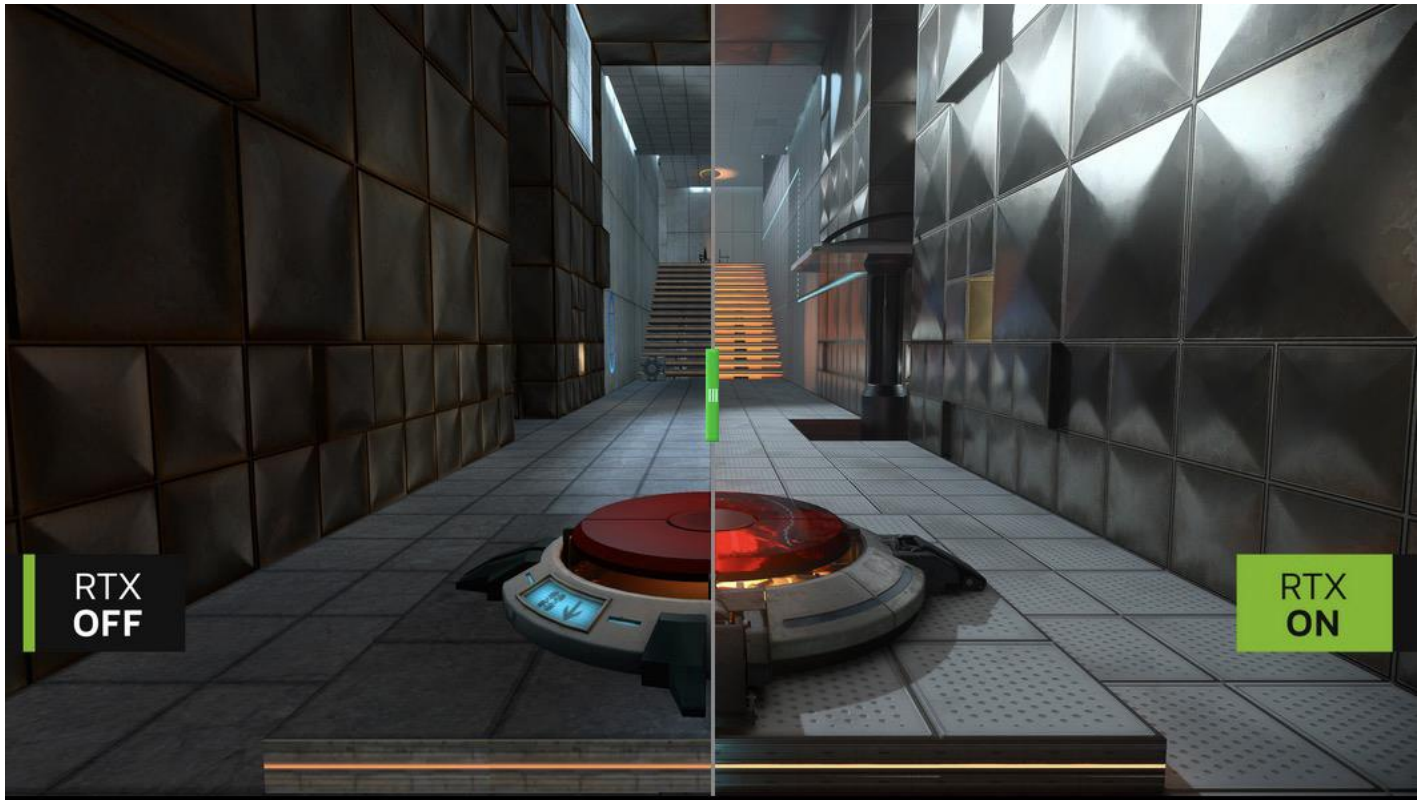
- Post-processing effects

Real-time Rendering

- RTX 4090

Real-time Rendering

- RTX 4090



Real-time Rendering

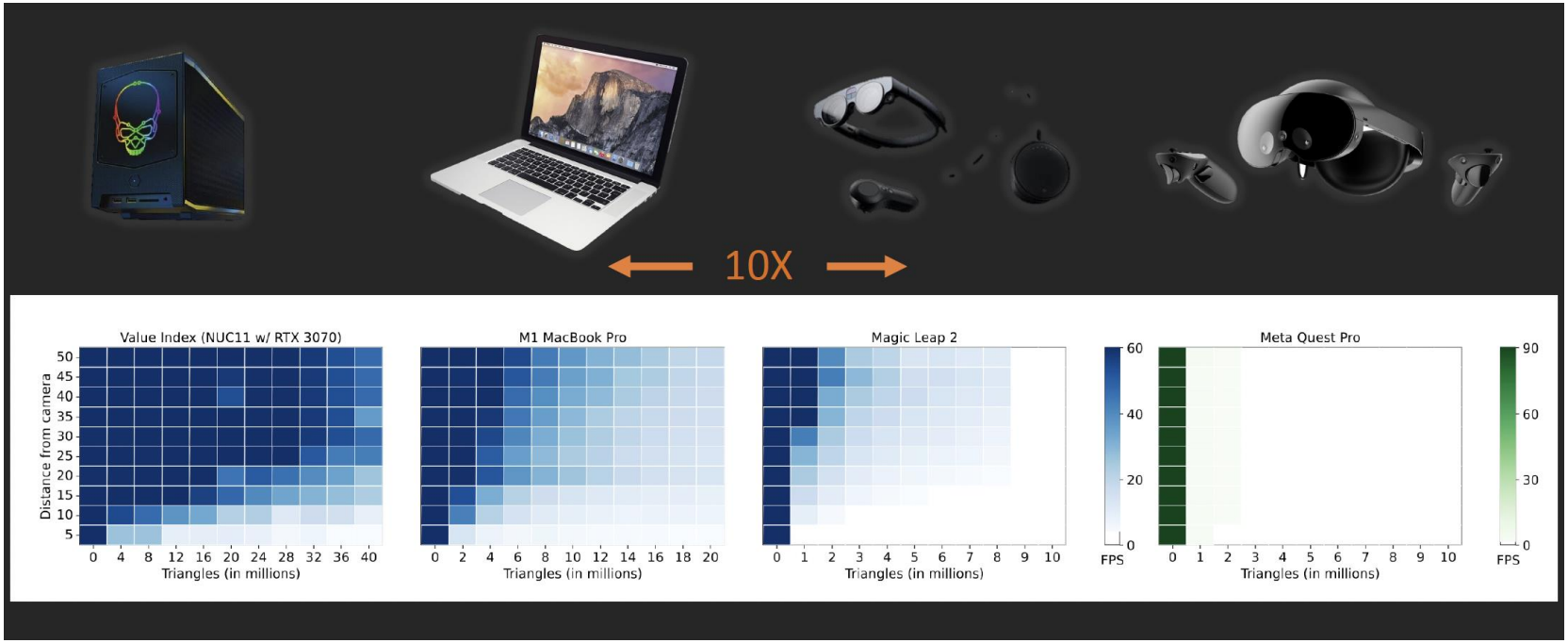
- RTX 4090

Rendering Performance

- *Frames per second*
  - *Speed*
- *Polygons per frame*
  - *Related to detail*
- *Latency*
  - *How long before system input to updated frame*
- *Power*
  - *Computation and data transfer*

Rendering Performance

Across different XR devices



Rendering Performance

- How about real-time rendering on ultra-thin wearable XR devices like glasses?

Rendering Performance

- How about real-time rendering on ultra-thin wearable XR devices like glasses?

Brilliant Labs Frame

Took 10 seconds to render an image

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

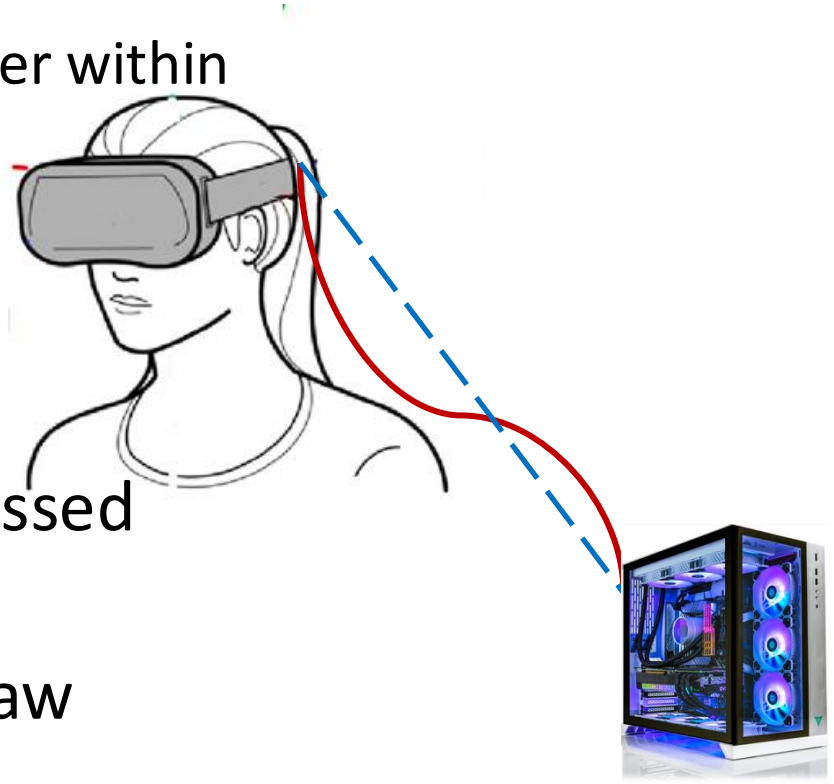
- Edge rendering is done at the edge of the network, near the user, rather than on centralized data centers or the user's device.
  - The purpose is to reduce latency, decrease the bandwidth needed for high-quality graphics, and alleviate the computational load on user devices.
- Key Benefits:
  - Faster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

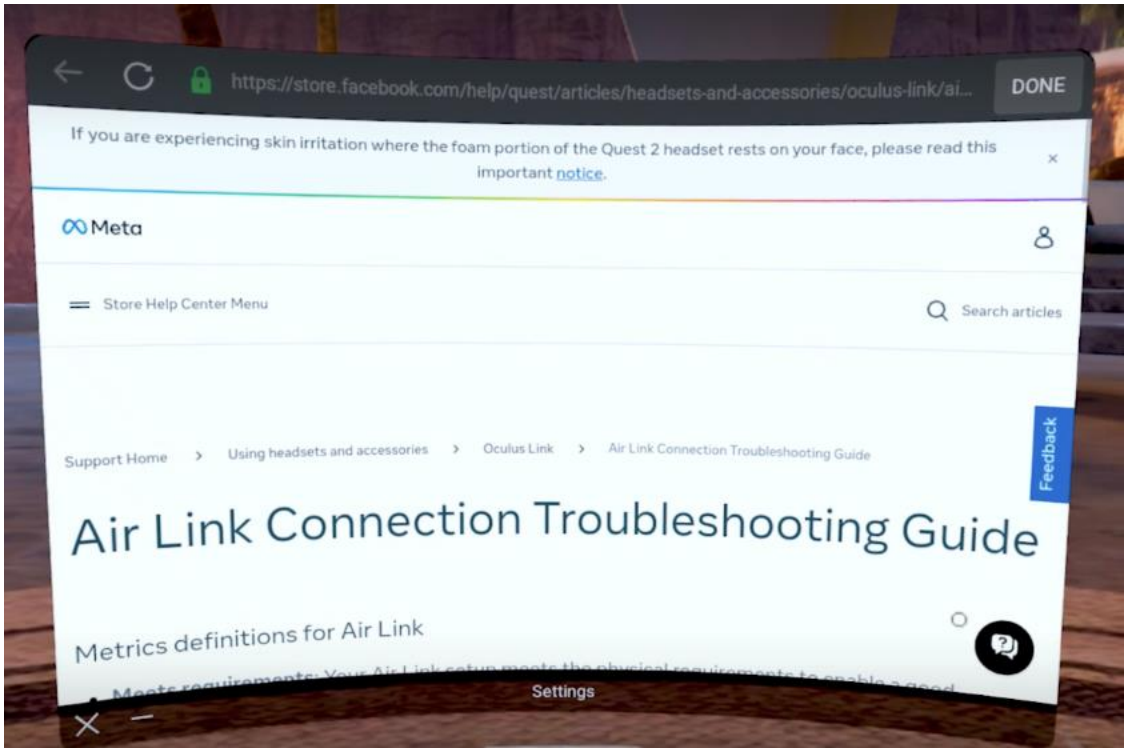
- WiFi
  - Rendering is placed a computer within the same WIFi LAN
  - Closer to users
  - Low latency



- 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

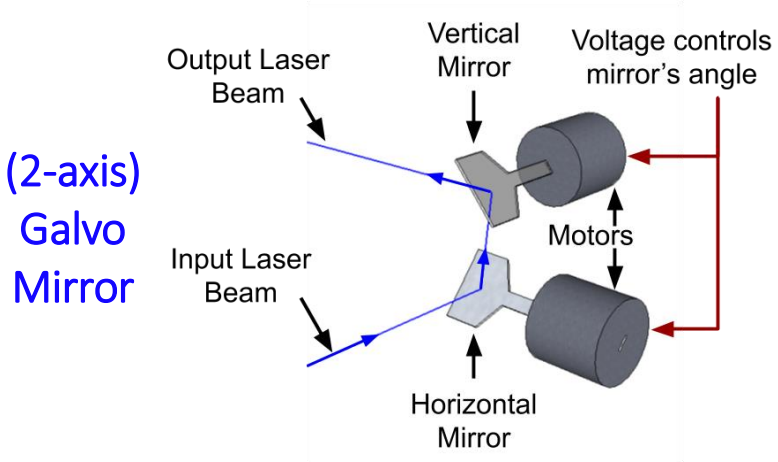
- Problem with higher frequency wireless links
  - Links are not reliable – narrow wavelength
  - Environmental impact
  - Line of sight
- Problem with XR devices
  - Users move around
  - Mobility impact

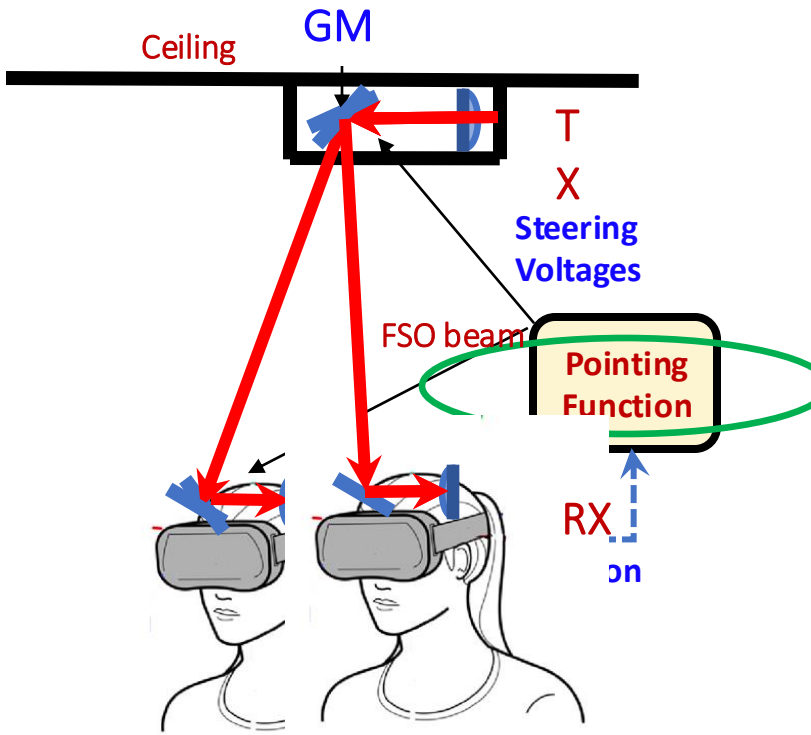
Edge Server Rendering

- 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)]





Pointing Function:

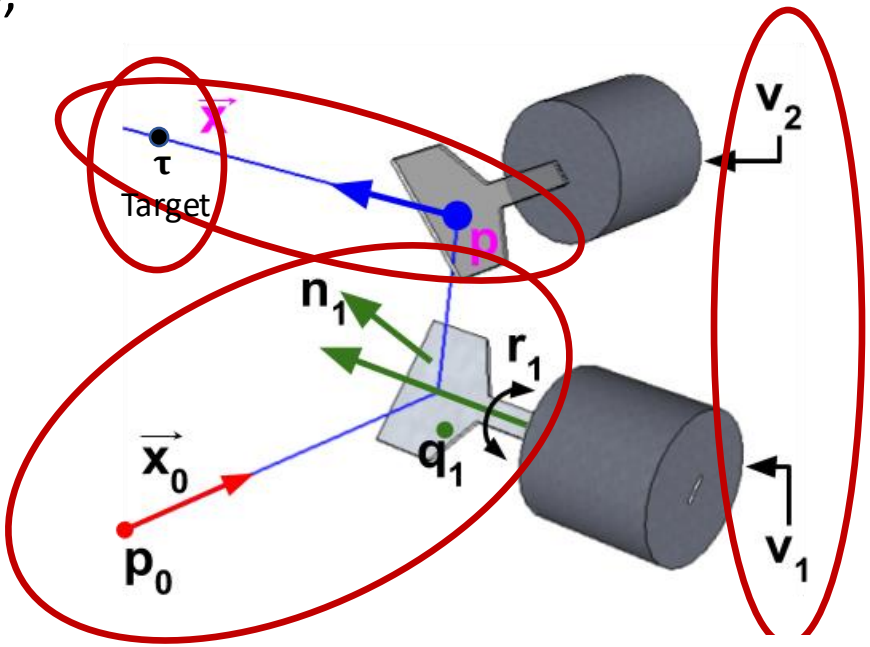
- Pointing function P:
  - Input: VRH/RX location [In the unknown VRH coordinate system]
  - Output: 4 GM Voltages [To steer TX and RX to realign beam]
- Learning P directly from (input, output) samples is infeasible
- Our approach:
  1. Learn GM models (two functions G and G') [Offline]
     a) In the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

1a. Learn GM Model (in GM space)

Function G: $(v_1, v_2) \longrightarrow$ Output beam (p, $\vec{x}$).

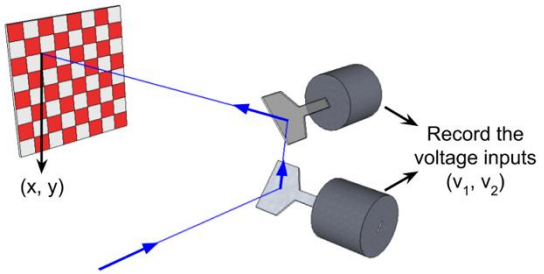
i. Derive an expression for G from its physical configuration.

ii. Learn the parameter values, using training data.



Function G': (target point τ) $\longrightarrow$ (v_1, v_2)

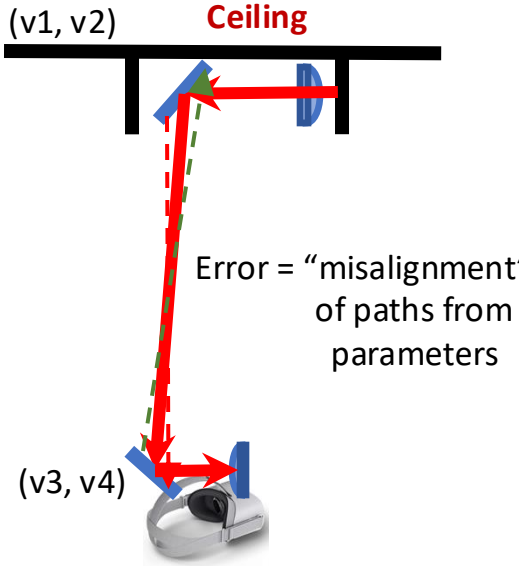
- Use G iteratively to estimate G'.



1b. Map GM Functions to VRH Space

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

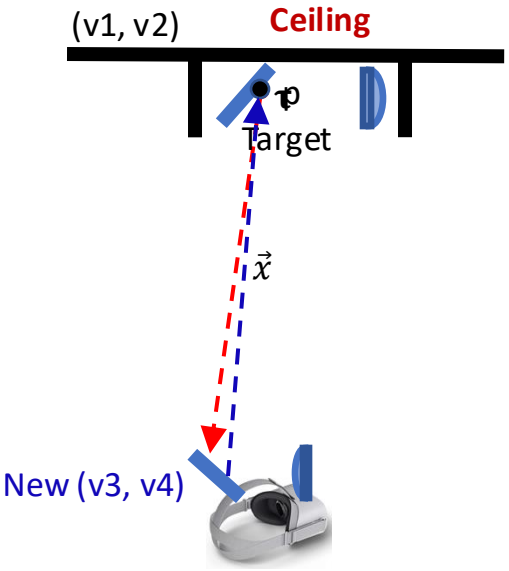
1. Gather training samples (aligned beam state).
   - (VRH Position, 4 voltages) for each sample.
2. Define an <u>error function</u> for given parameter values.
3. Determine parameter values that minimize the total loss over samples.



2. Pointing Function P from GM Functions

Pointing Function P:

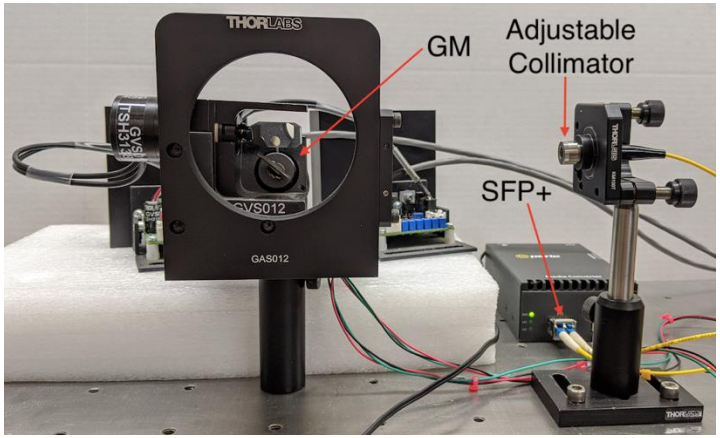
- Input: VRH position.
- Output: 4 Voltages.
- Approach (Real-Time):
  - Initialize voltages v_1, v_2, v_3, v_4
  - (p, $\vec{x}$) = G (v_1, v_2) TX-beam output specs
  - New (v_3, v_4) = G'($\boldsymbol{\tau}$ = p) RX-beam should hit p.
  - Similarly, compute new (v_1, v_2).
  - Iterate.



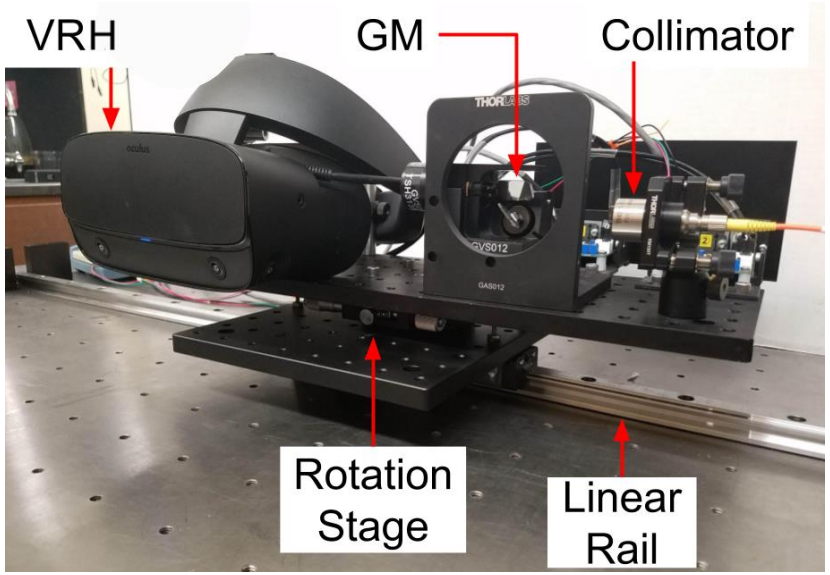
FSO-VR Prototype Design

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

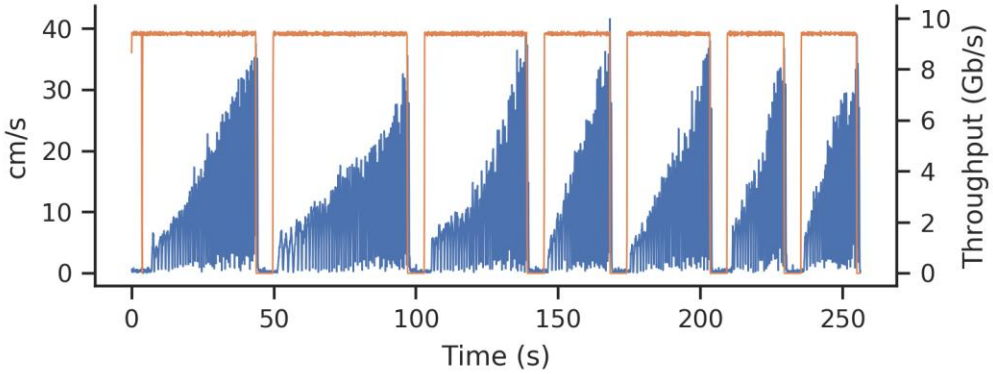
TX

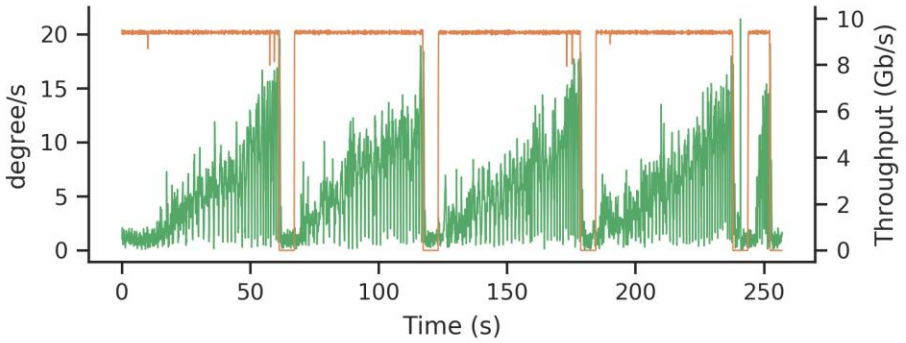


RX



FSO-based VR Link Performance

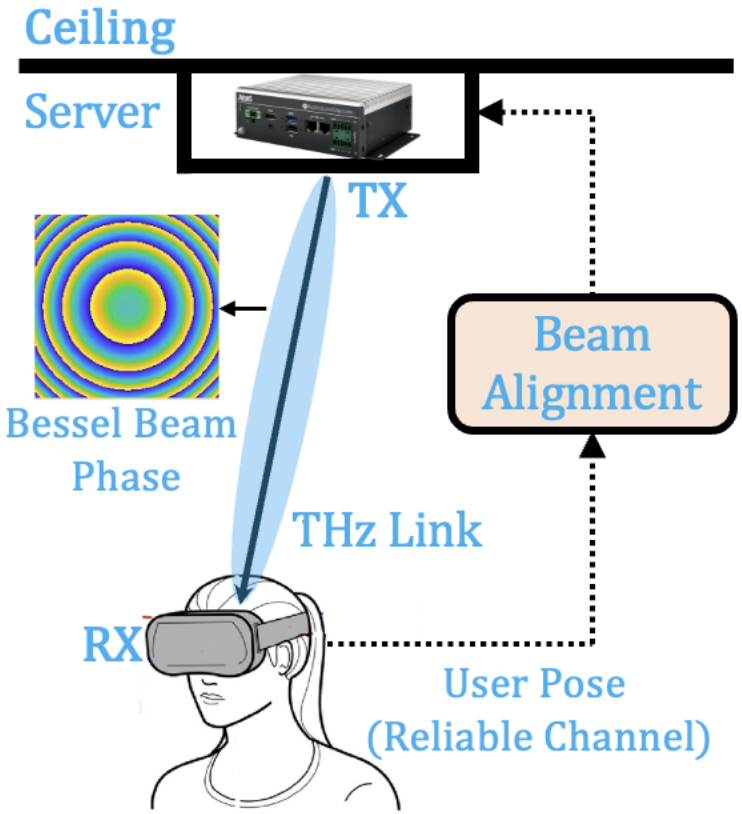




- Performance could be much improved, with customized components.
  - E.g., higher tracking frequency, customized optical components.

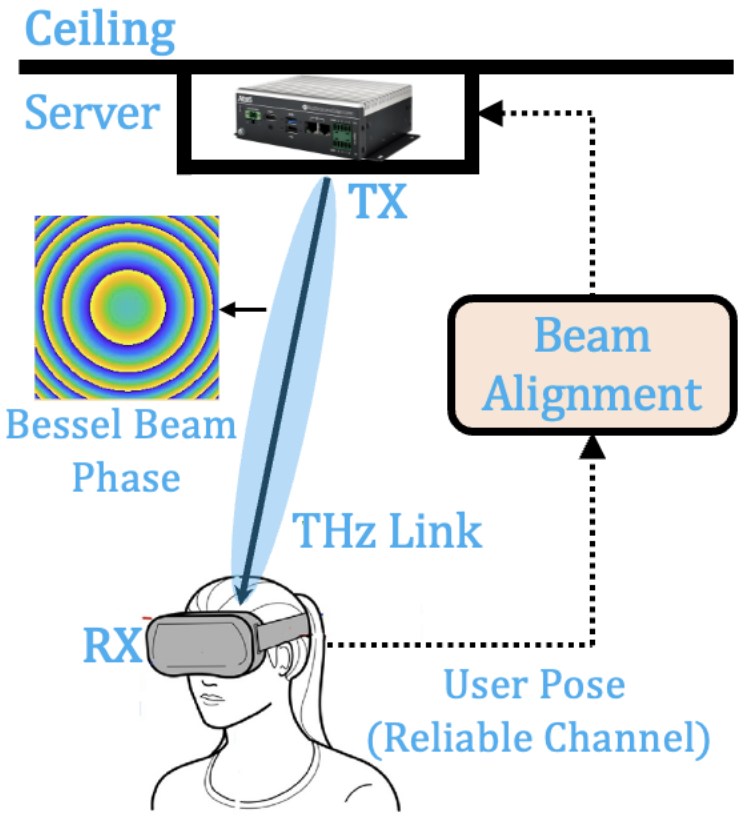
THz Band based VR Link

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

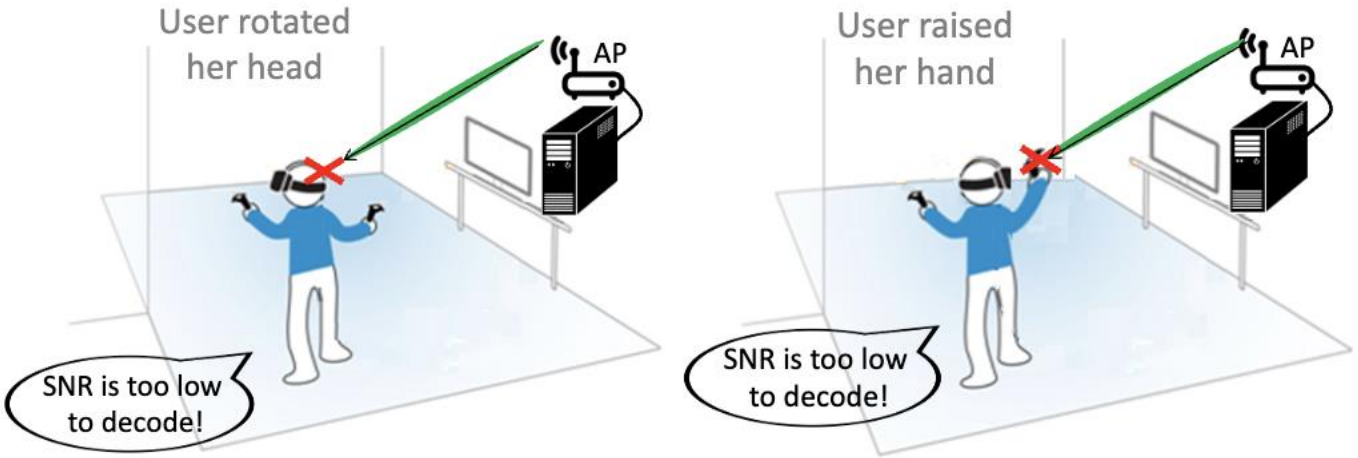


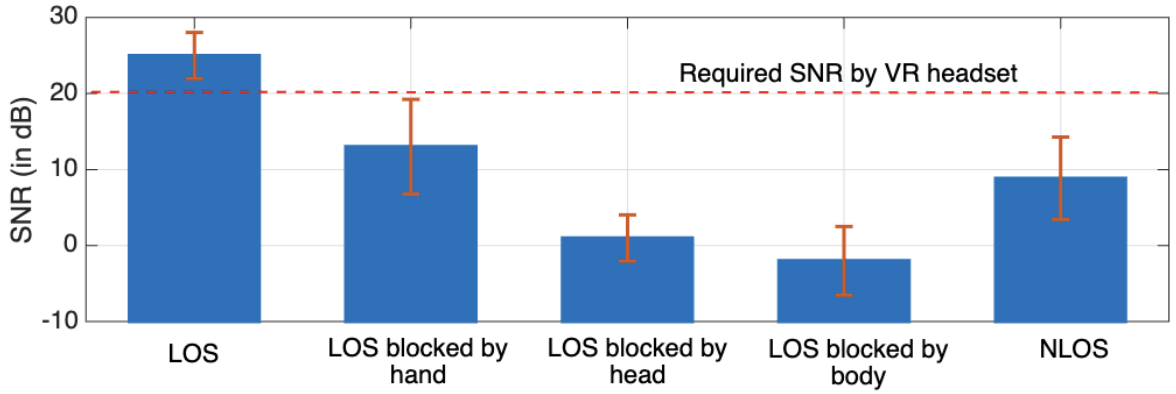
THz Band based VR Link

- 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

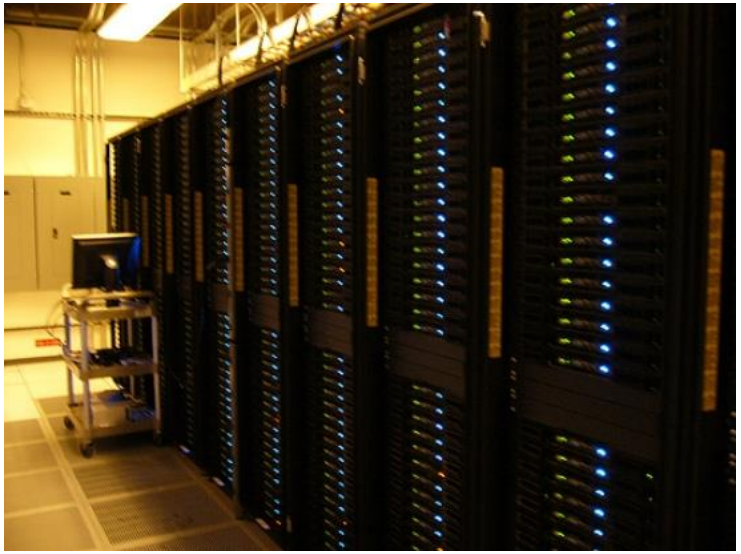
Distributed or Parallel Rendering

- 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



Summary of the Lecture

- Rendering Basics
  - Types of rendering
  - Rendering pipeline
  - Real-time rendering

Next up: Hybrid Rendering