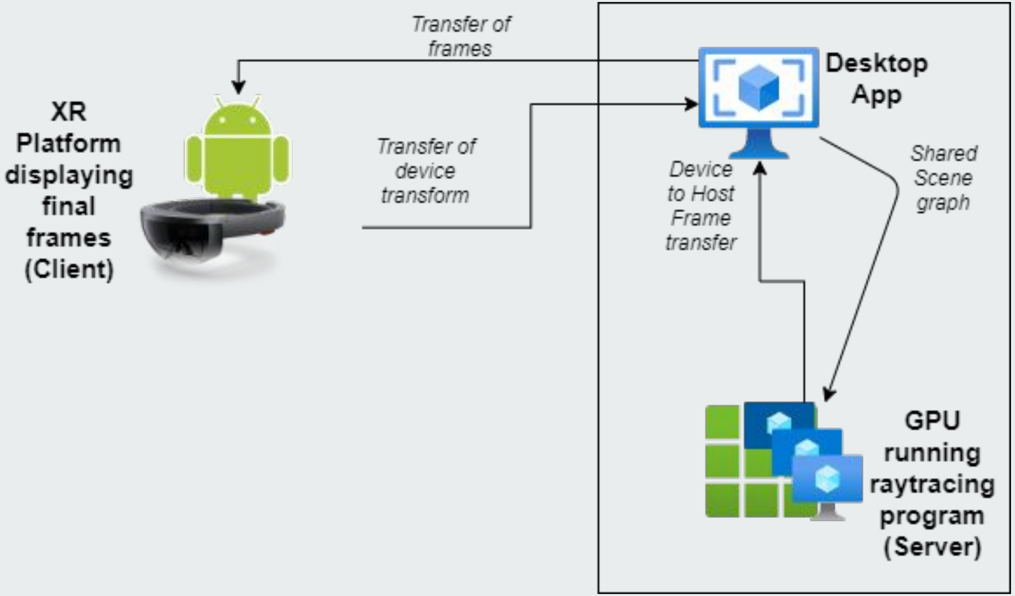
Remote Rendering for XR

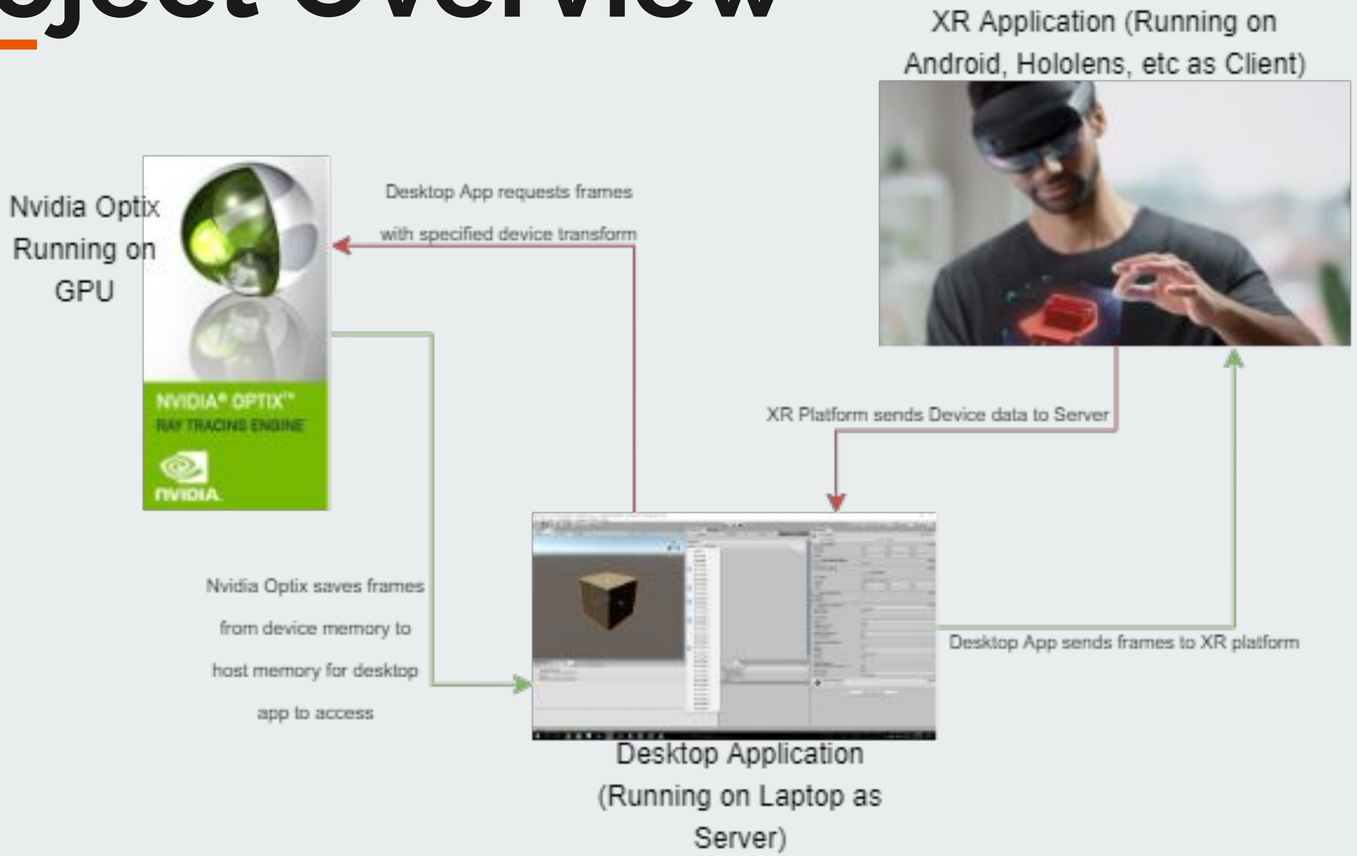
By- Gizem Dal, Dayu Li, Tushar Purang

```
Loading Model: 100.00 %
Render: 30 fps - 33.33 / 38.16 ms (avg / max)
Video frames: 32 / 1 / 60 skipped / reused / received
Video frames delta: 0.05 / 44.29 ms (min / max)
Latency: 72.40 / 16.97 / 89.64 ms (avg)
Video frames discarded: 0 / 0 fra
Frame time CPU/GPU
Utilization CPU/GPU: 1
Memory CPU/GP
```

Source: HoloLens 2 Azure Remote Rendering in-action


Transfer of frames
XR Platform displaying final frames (Client)
Transfer of device transform
Desktop App
Device to Host Frame transfer
Shared Scene graph
GPU running raytracing program (Server)

Project Overview



XR Application (Running on Android, Hololens, etc as Client)

Nvidia Optix Running on GPU

Desktop App requests frames with specified device transform

XR Platform sends Device data to Server

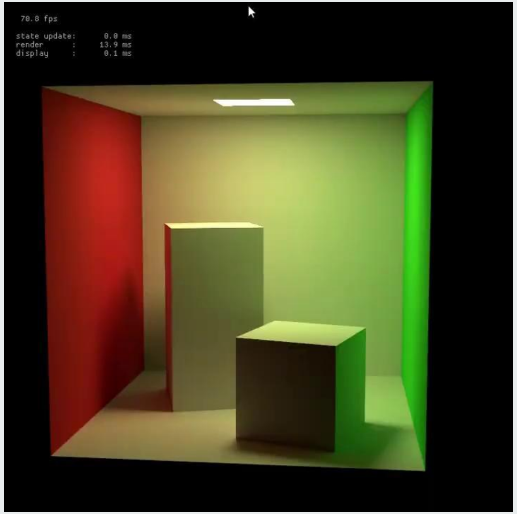
Nvidia Optix saves frames from device memory to host memory for desktop app to access

Desktop App sends frames to XR platform

Desktop Application (Running on Laptop as Server)

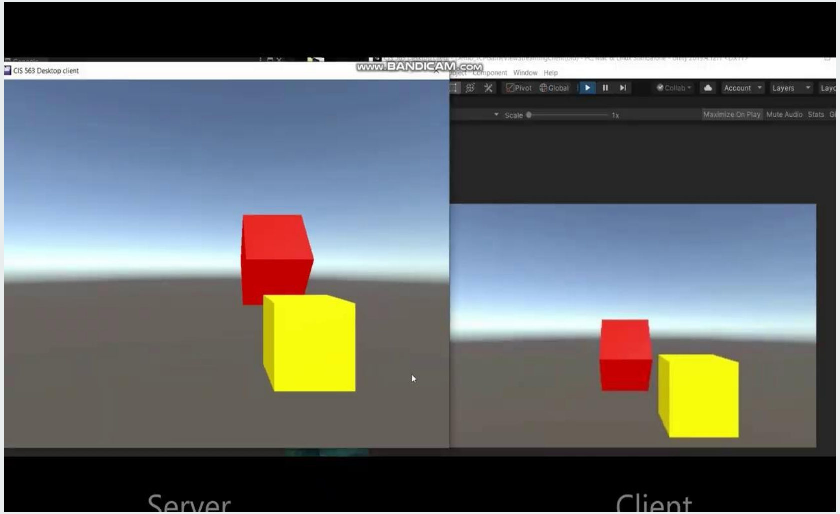
Review: progress in milestone 1

Basic client/server video streaming & Nvidia optix ray tracing base code.



```
 70.8 fps

state update:     0.0 ms
render      :    13.9 ms
display     :     0.1 ms
```



Overview: Milestone 2

Progress in project

- Obj & Mtl file parser
- OptiX Path Tracer sample enhancement
- Frame Streaming

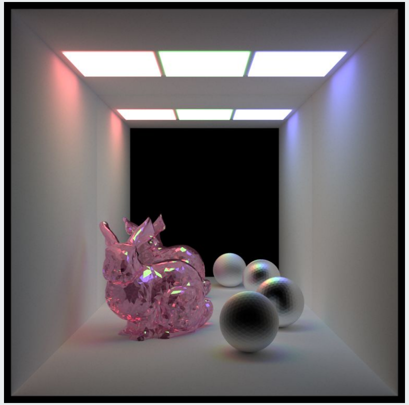
Researches and Studies

- Tiny Obj loader data structure & APIs.
- Mesh loading & Texture mapping in Optix.
- Hololens gesture & spatial mapping APIs.
- Physically Based Rendering: From Theory to Implementation

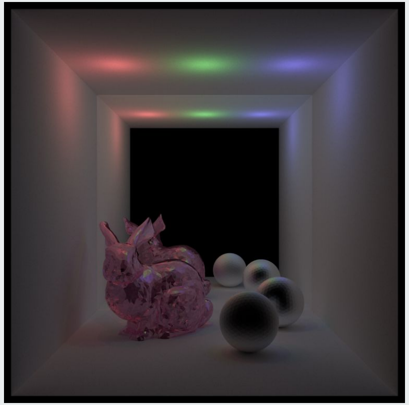
Tasks completed in milestone 2

- Support for area, point and spot lights
- Support for diffuse, perfect specular (mirror), imperfect specular (glossy) and fresnel material
- Support for cube, icosphere and arbitrary meshes
- Russian roulette termination for light paths with less contribution
- Select a random light in scene for MIS

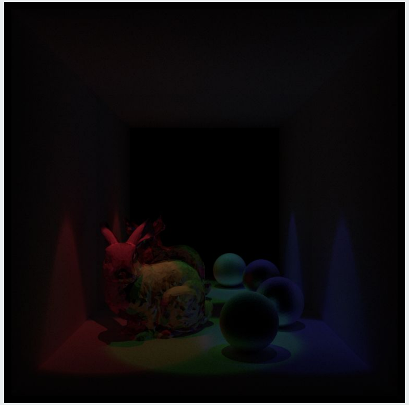
Stanford Bunny Mesh



Area lights



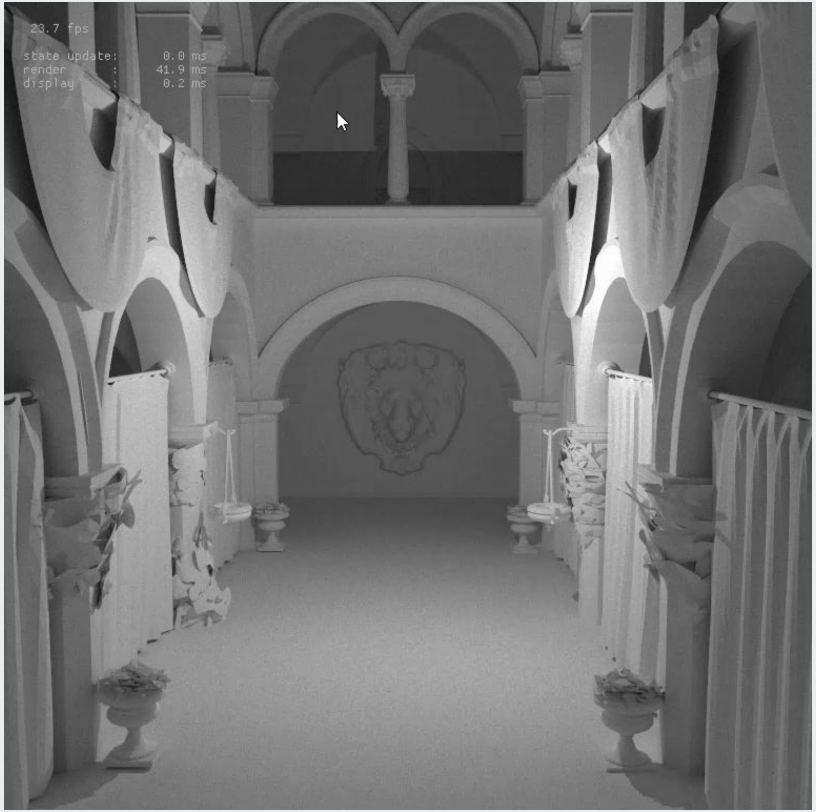
Point lights



Spot lights

Tasks completed in milestone 2

- Arbitrary mesh loader for .obj file format based on the tinyOBJ loader.
- The mesh will be rendered as a whole with a specified material.



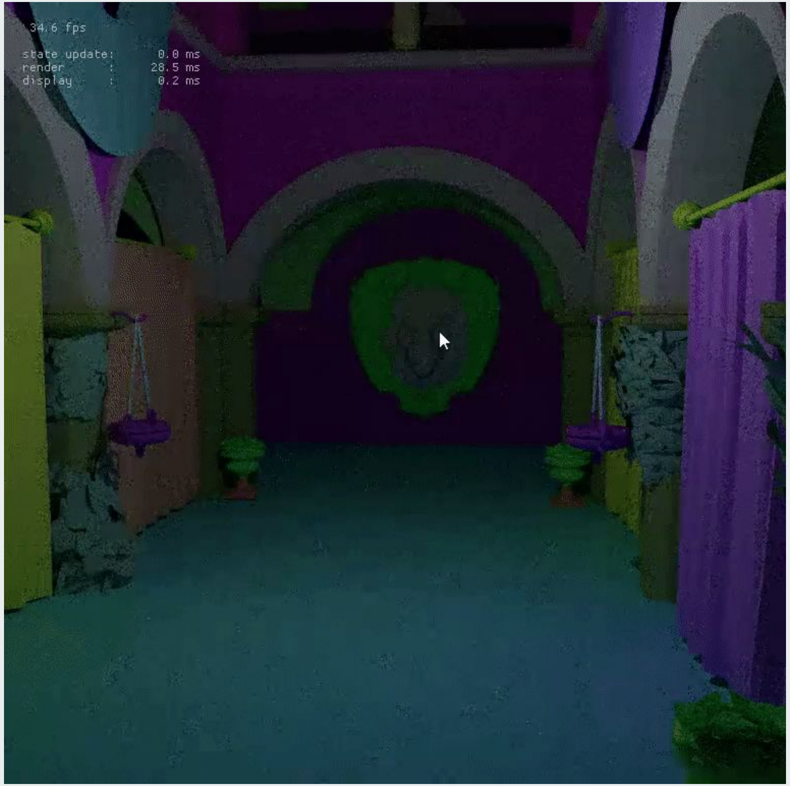
```
 23.7 fps

state update:      0.0 ms
render      :     41.9 ms
display     :      0.2 ms
```

Crytek Sponza Mesh

Tasks completed in milestone 2

- Material parser for .mtl format
- The Wavefront Material Template Library (MTL) file is a companion file for one or more Wavefront OBJ files. It is a "library" that contains one or more material definitions, color, texture, and reflection characteristics.
- Areas that are in the same material will be rendered as an individual piece with the mtl parser.



```
 34.6 fps

state update:      0.0 ms
render      :     28.5 ms
display     :      0.2 ms
```

Crytek Sponza Mesh

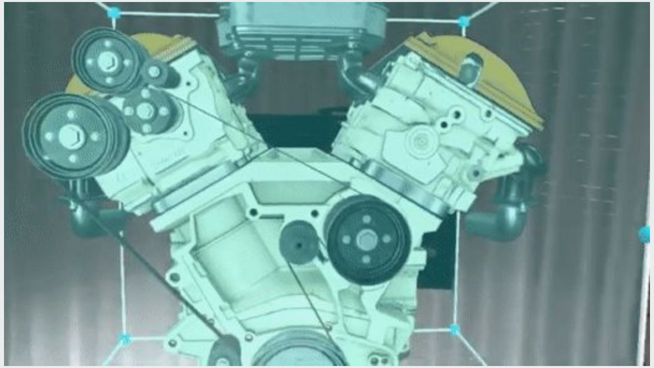
Tasks completed in milestone 2

- Streaming Raytracer frames via network
- Frames are stored in host memory
- Stored frames are sent from server (Laptop) to client platform (Android, Hololens etc.)
- Current latency (Raytracer to Android) :
  1.5 sec (Nvidia Optix to Unity Server)
   + 0.5 sec (Unity Server to Unity Client)

Tasks planned for milestone 3

- Finish up texture mapping
- Synchronizing the camera control in Raytracer with head movement of the Hololens
- Modifying the Raytracer result with hand interaction and gestures.
- Raytracing with the spatial mapping obj
- Optimizing streaming.

Source: *Ingo Wald OptiX7 course*



Source: Azure Remote Rendering + MRTK Demo Hololens 2

Schedule

Milestone 1 – Nov 18th:

- Basic Desktop app (Control Panel) + Hololens app + GPU networking + basic realtime raytracer

Milestone 2 – Nov 30th:

- (Still in progress) Hybrid Rendering (Scene + UI)
- Real time ray tracing + Material parser + Texture mapping

Milestone 3 – Dec 7th:

- Optimizing streaming frame rate
- Late Stage Reprojection for Hololens
- Real time ray tracing in XR

Final – Dec 13:

- Bug fixings and optimizations
- Performance analysis

References

1. About Azure Remote Rendering
   https://docs.microsoft.com/en-us/azure/remote-rendering/overview/about

2. CPU-GPU Algorithms for Triangular Surface Mesh Simplification
   https://imr.sandia.gov/papers/imr21/Shontz.pdf

3. A Positional Timewarp Accelerator for Mobile Virtual Reality Devices
   https://escholarship.org/content/qt96r870gs/qt96r870gs_noSplash_4abbeba6bd4266514b1d56cbdd9dc5d7.pdf

4. Differential Irradiance Caching for Fast High-Quality Light Transport Between Virtual and Real Worlds
   https://publik.tuwien.ac.at/files/PubDat_220665.pdf

5. Dynamic Diffuse Global Illumination with Ray-Traced Irradiance Fields
   http://jcgt.org/published/0008/02/01/paper-lowres.pdf

6. High-Quality Real-Time Global Illumination in Augmented Reality
   https://www.ims.tuwien.ac.at/projects/rayengine

7. Nvidia Optix SDK
   https://developer.nvidia.com/optix

8. A Streaming-Based Solution for Remote Visualization of 3D Graphics on Mobile Devices
   https://www.researchgate.net/publication/3411346_A_Streaming-Based_Solution_for_Remote_Visualization_of_3D_Graphics_on_Mobile_Devices