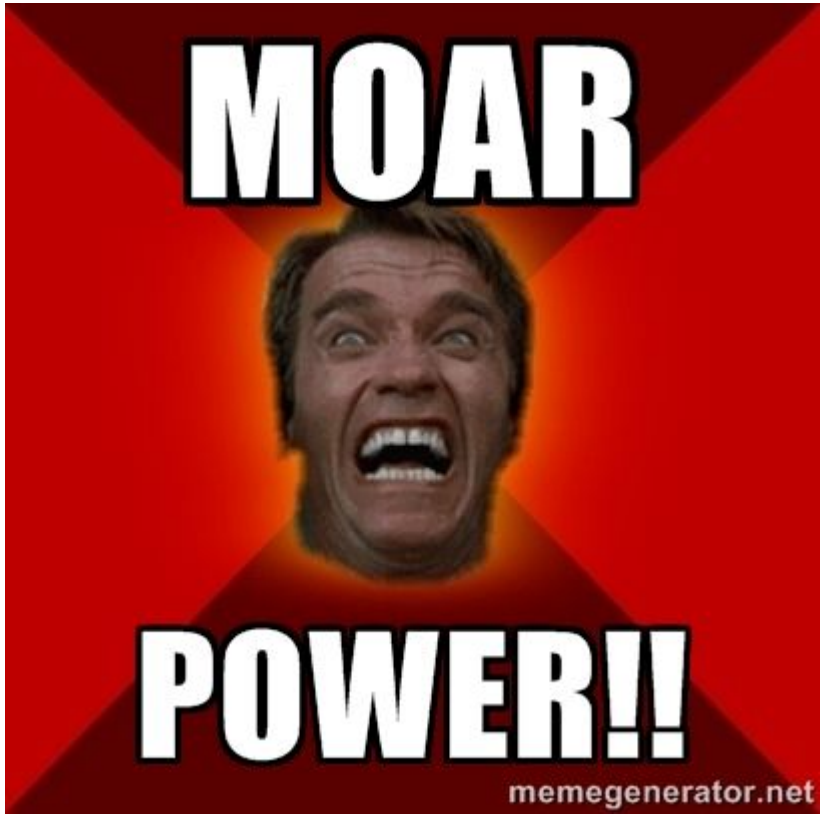
# Distributed Renderer

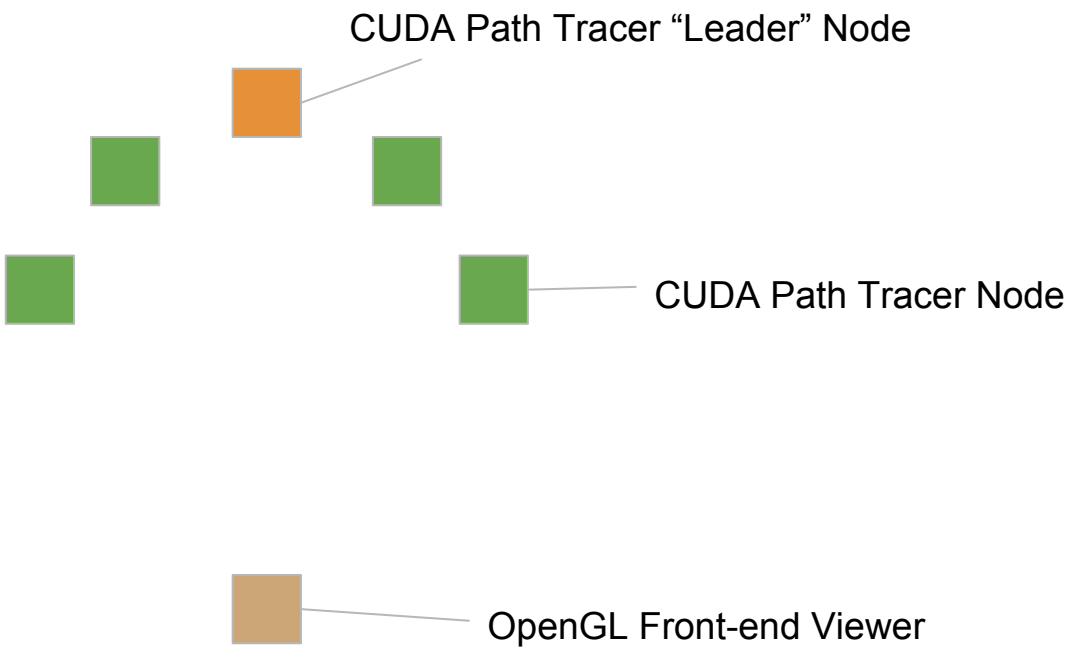
CIS565 Final Project

by Sanchit Garg & Dome Pongmongkol

# Motivation



# Design



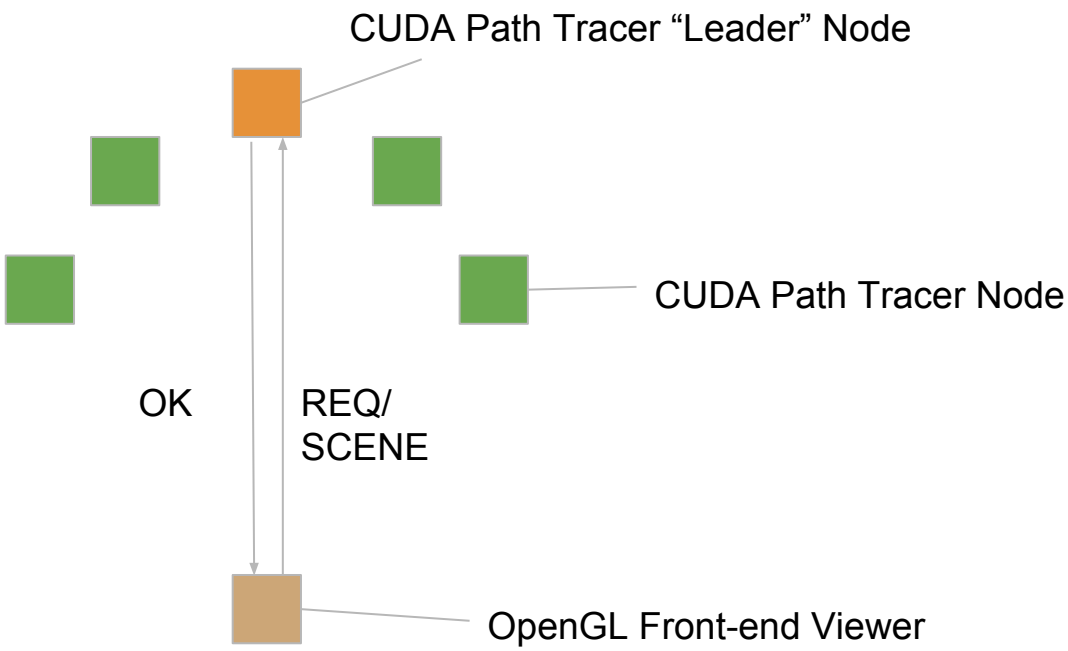

Each node has 3 Threads
- Renderer (CUDA or OpenGL)
- Receiver (UDP)
- Sender (UDP) + Networking Decision Making

use Boost Library for multithreading

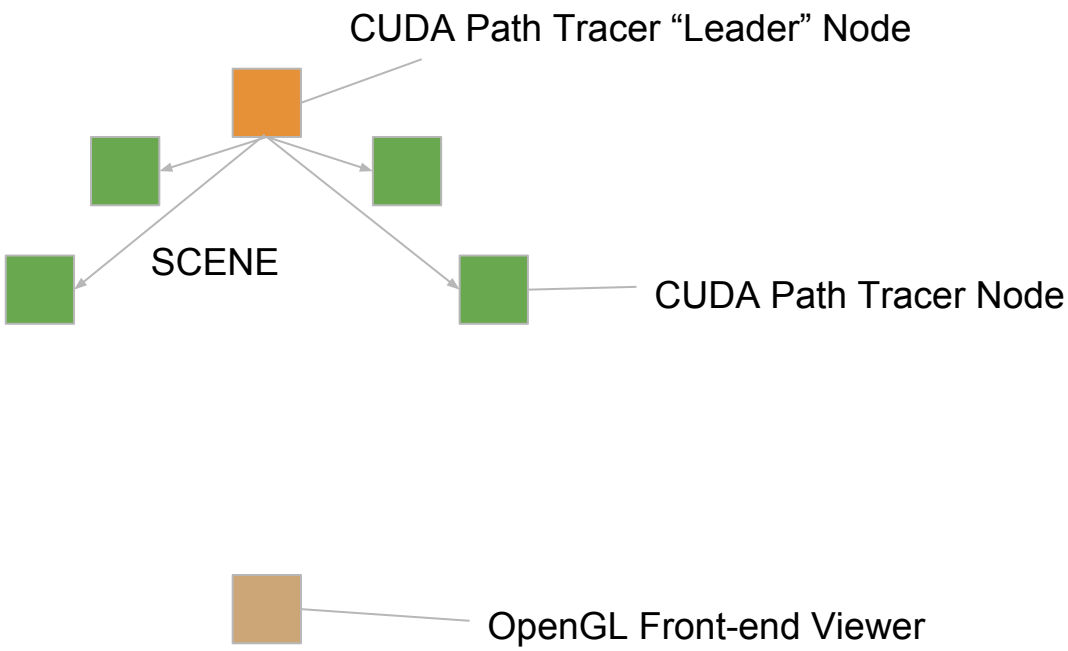
Networking are done through WinSock UDP
- Might need TCP for asset distribution

Packet are packed and unpacked with Google Protobuf API

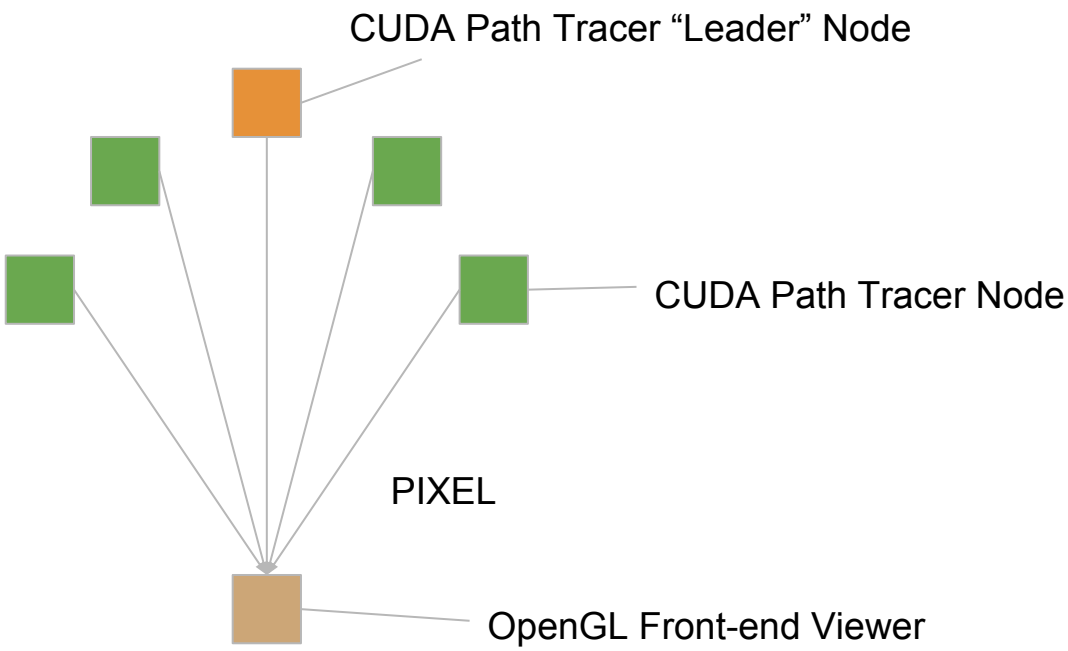
# Design


CUDA Path Tracer “Leader” Node
CUDA Path Tracer Node
OK
REQ/
SCENE
OpenGL Front-end Viewer

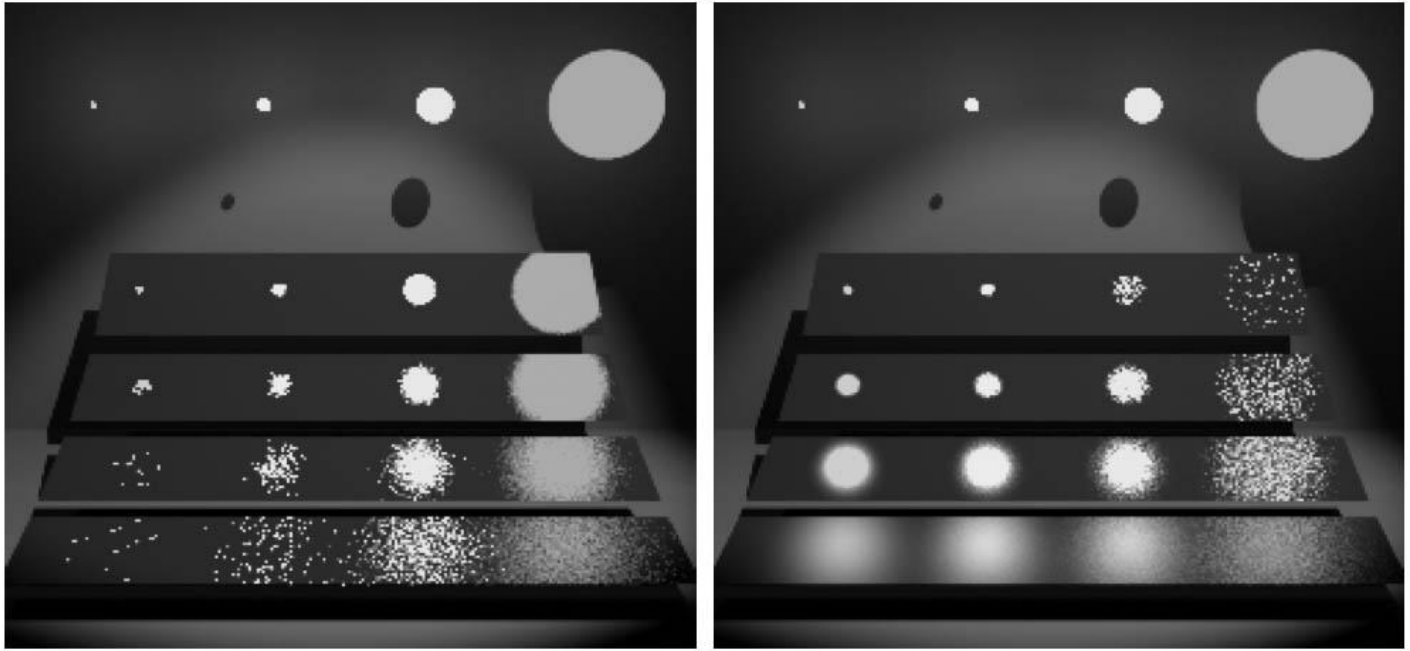
# Design


CUDA Path Tracer “Leader” Node
SCENE
CUDA Path Tracer Node
OpenGL Front-end Viewer

# Design


CUDA Path Tracer “Leader” Node
CUDA Path Tracer Node
PIXEL
OpenGL Front-end Viewer

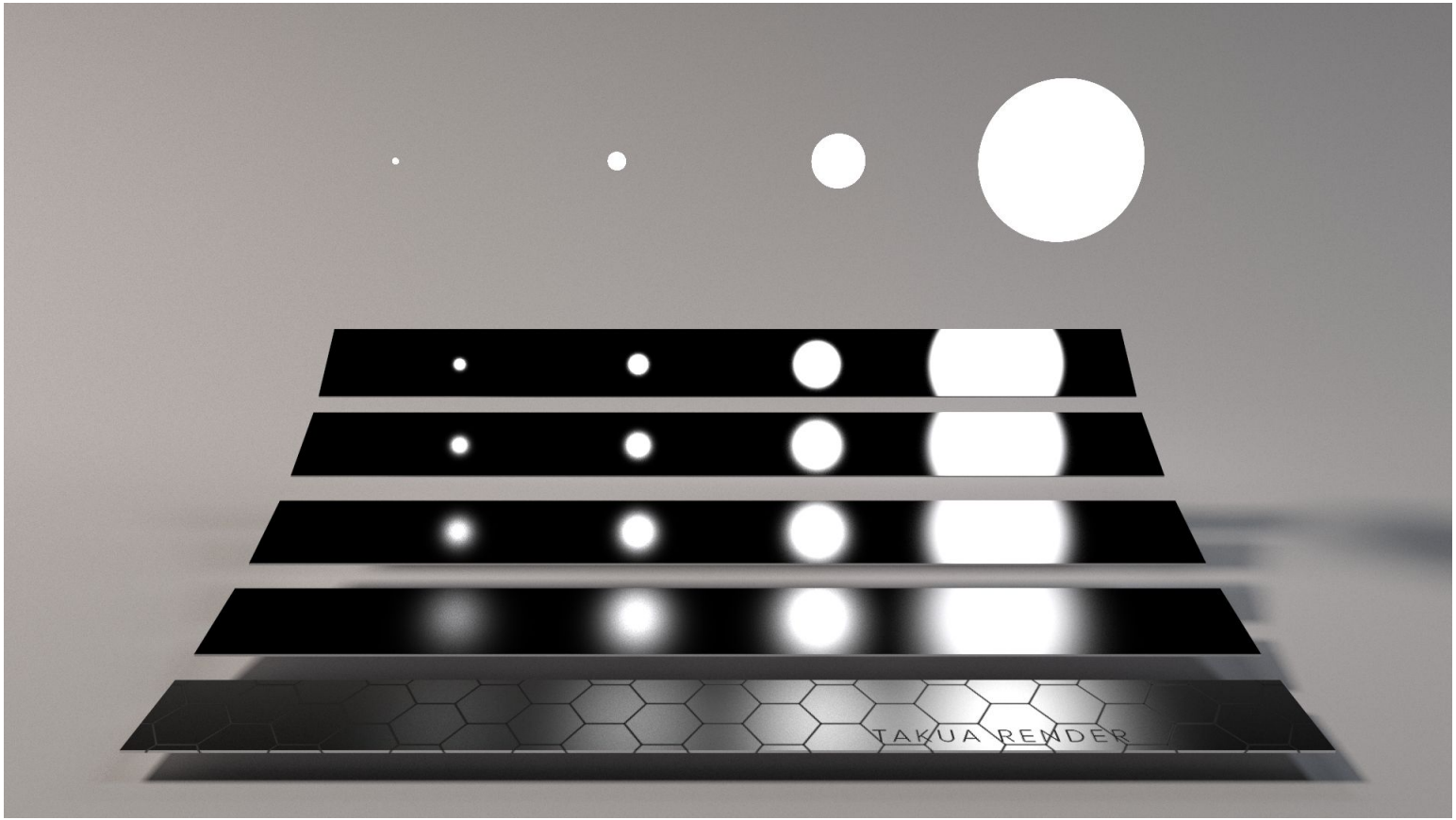
# Multiple Importance Sampling



**(a)** Sampling the BSDF

**(b)** Sampling the light sources

https://www.fxguide.com/featured/render-me-modo/




Thanks to Karl Yingling Li

# Progress

Research, install and test the required tools and libraries on Windows (Google Protobuf, Boost, WinSock)

Port our existing GPU Path Tracer & external libraries to Windows.

We have a mesh rendered.

# Roadmap

Milestone 1

- 1 CUDA Renderer & 1 Viewer
- Networking for 1-to-1 communication
  - Timestamp-based error handling (Packet dropping)
  - Test to determine the proper size of packet (i.e. how many pixels should be sent per packet)
- Implement BRDF Importance Sampling

# Roadmap

Milestone 2

- Multiple CUDA Renderer (Centralized) & 1 Viewer
  - Basic workload distribution (Leader)
  - Determine how often (or in what pattern) the renderer should send the pixel packets
  - viewer handling packets from multiple sources (This is why we use UDP!)
- Scene file distribution
- Implement Light Importance Sampling

# Roadmap

Milestone 3

- Distributed System error handling
  - i.e. If any node dies. Reset work distribution, Distributing the pixel informations that the leaving node was responsible for (so that they can continue from what's left behind)
- Asset files distribution
  - **STRETCH GOAL**: File distribution strategy (i.e. transfer between closest nodes instead of transferring from the leader only)
- Implement Multiple Importance Sampling
- **STRETCH GOAL**: Add Kd Tree acceleration Structure

# Roadmap

Milestone 4 (Final)

- **STRETCH GOAL:** *Decentralized* CUDA Distributed Renderer
  - Handle cases that the leader die/leave the ring.
    - Leader Election
- Testing & Optimization
  - bug fixing
  - perform tests on different workload distribution methods and compare the total time (i.e. tile-based VS iteration based)
  - **STRETCH GOAL**: work dividing criterias (i.e. network bandwidth, GPU specs of each node)
- Documentation