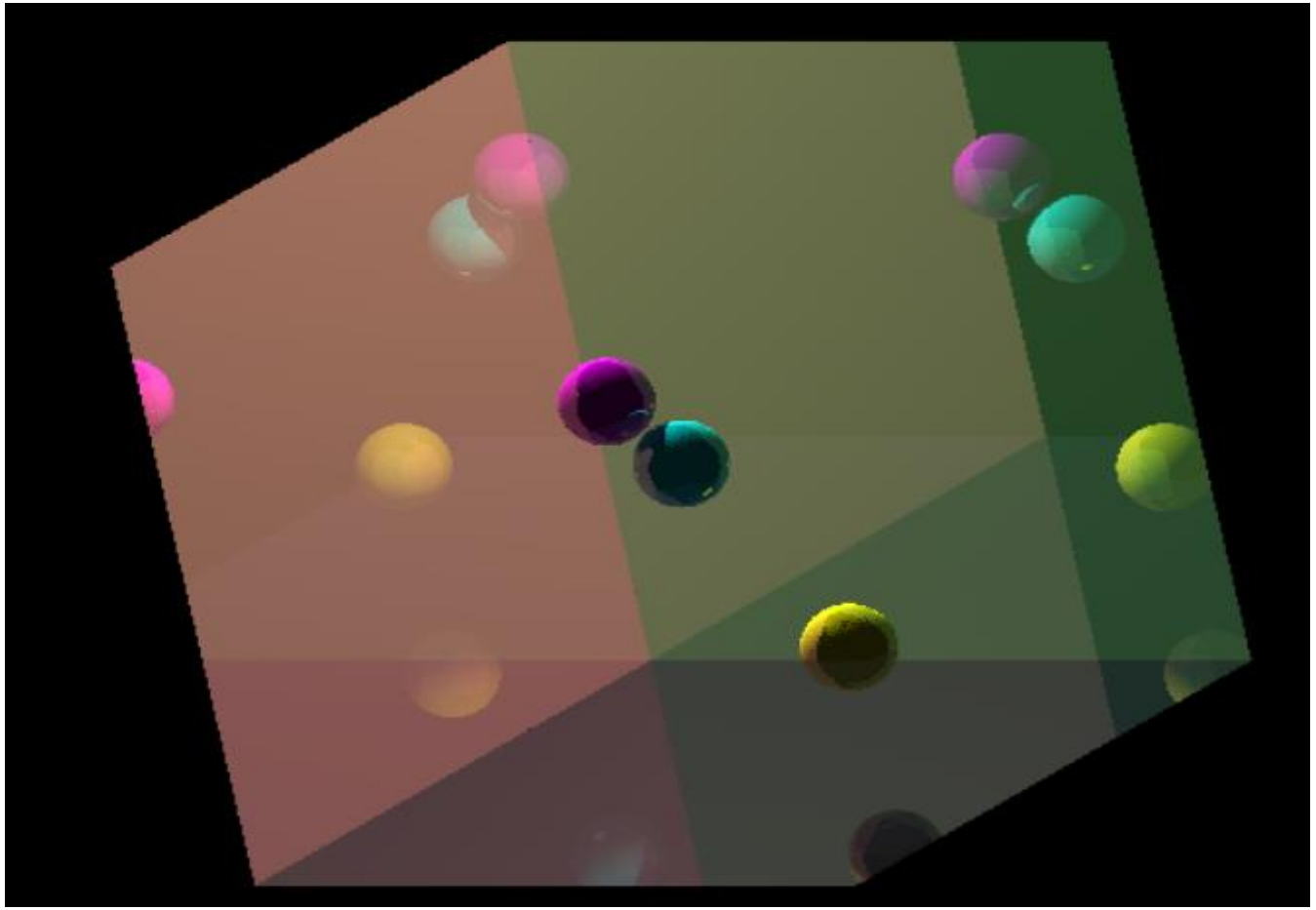
Real-Time RayTracing using OpenCL

Ajay Narasimha Mopidevi

CSCI 5229: Final Project

How many spheres did I create?

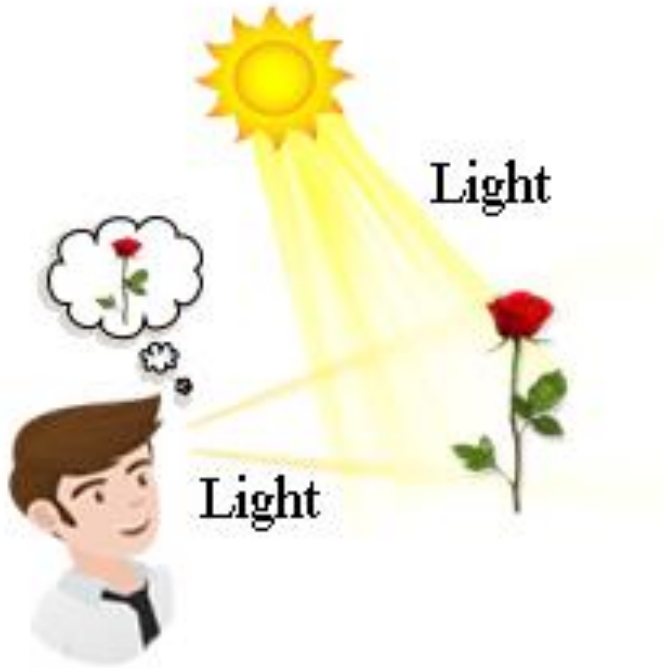
- 1 ?
- 3 ?
- 9?
- 10?
- 14?
- Other?



Differences?

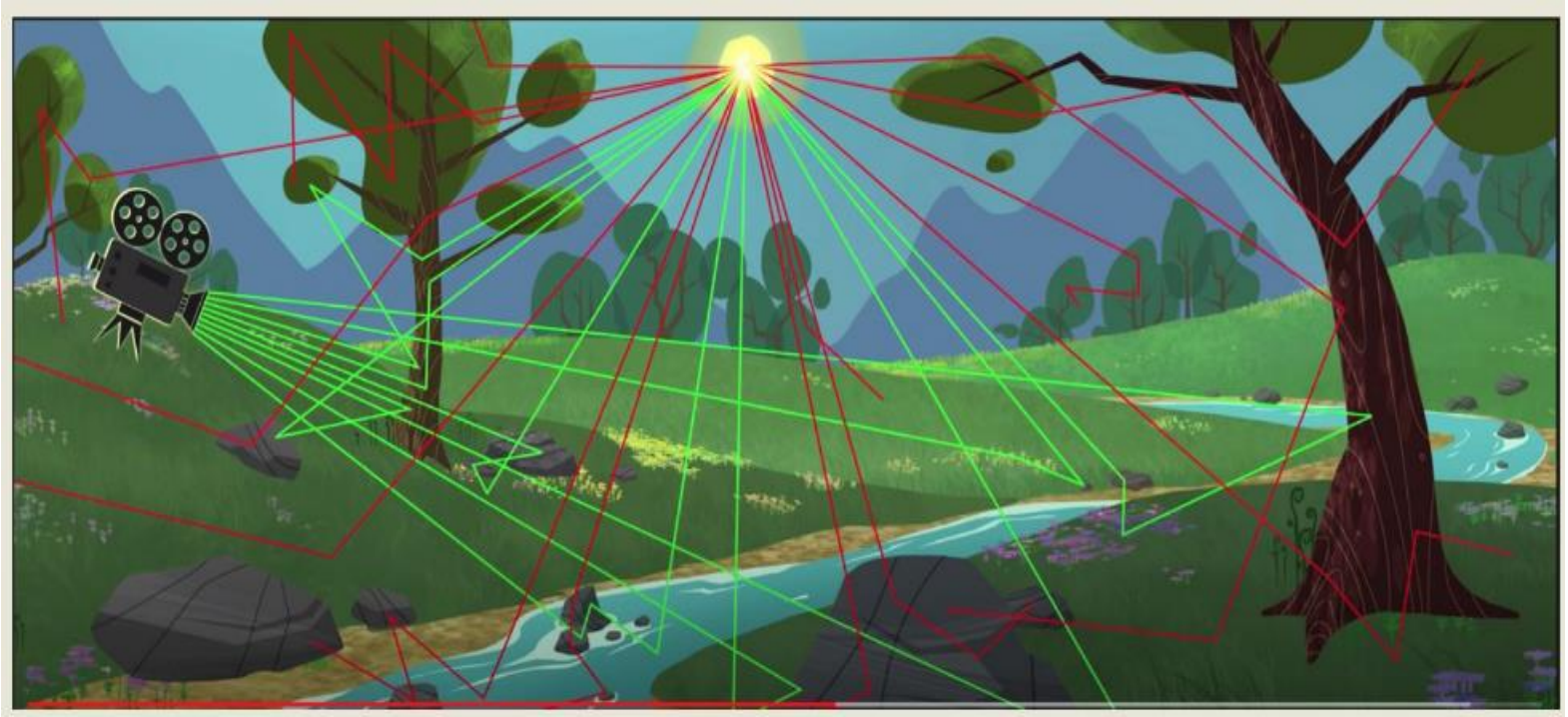
How do we see objects?

- Lights source – emits bunch of light rays
- Light rays hits an object and get reflected
- And finally the ray reaches our eye



Capturing the Scene

- Need to check for each ray
- Many rays doesn’t hit the eye/camera
- Reverse the process



Ray Tracing

- Reverse the light process
- Shoot a ray for each of the pixel and trace the path of the ray
  - Check if the ray hits an object
  - Find the nearest hit
  - Color of pixel based on the closest object hit and light sources
  - Calculate the reflected ray and repeat the process and accumulate the color

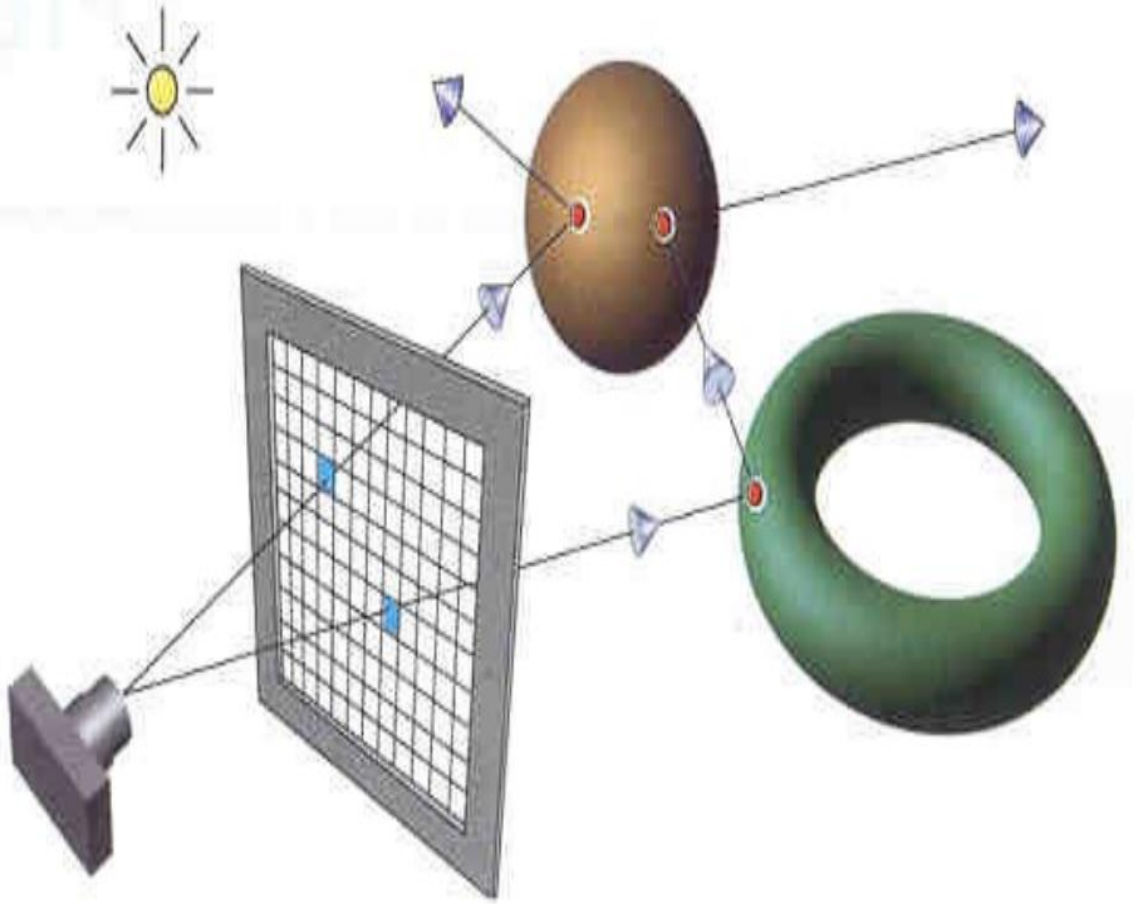
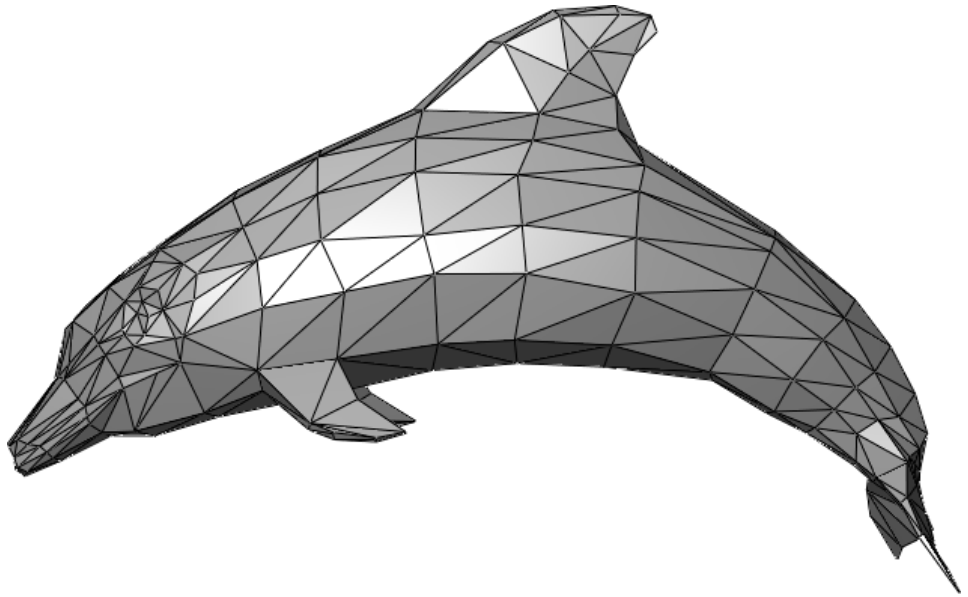


Figure 1. The ray-tracing process.

Scene

- Object – using set of triangles
- Triangle – all 3 vertices should be in same plane



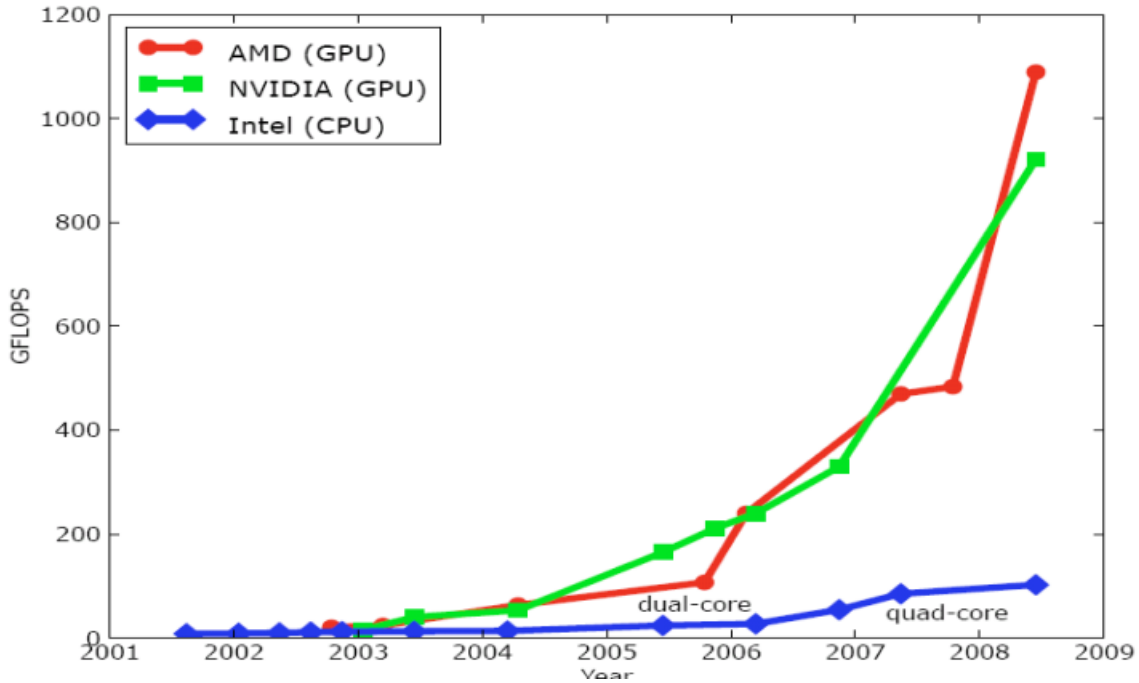
Too many computations

- Cube – 12 triangles
- Display window: 640x480 = 307,200
- Number of ray triangle tests = 307200x12 = 3,686,400
- Haven’t included
  - Time for each intersection test
  - Computation of color
  - Multiple levels (due to reflected rays)
- But, Each pixel computation is independent – PARALLEL

Parallel Processing

- NVIDIA GPU – CUDA
- Other GPU/CPU – OPENCL
- CPU – OpenMP

- Not every problem is parallel, but *some* part of it may be *made* parallel



Tasks Completed

- Base code : ex23 from CSCI 5239 (RayTrace Spheres)
- RayTracing a triangle
- Loaded objects as set of triangles
- Created complex scenes with spheres and triangles
- OpenCL support for real-time rendering

Challenges?

- Hardware Support only for OpenCL
- OpenCL – Need to use C (not C++)
- Rendering an .obj file
- Handling both spheres and triangles and finding the nearest hit object

Questions?