Real-Time Ray Tracing

Ankit Agarwal - 2011020
Prateek Malhotra - 2011077

Abstract

•Ray Tracing is a well-established technique for the generation of high-quality images from geometric representation of scenes.

•Ray Tracing is known to be a computationally expensive operation.

•Modern Graphics hardware capabilities can be exploited to significantly improve the performance of Ray Tracing.

Objective

- To build a CUDA-based ray tracer capable of rendering large scenes.

Methods

- Use CUDA threads to trace rays.
- Every thread traces the path of one ray.
- Find the closest hit point for the ray and perform shading computations using Phong Illumination at that point.
- Two separate implementations to find the closest hit point
  1. Brute-force Search
  2. K-d Trees

Findings

- Highly uneven load distribution in the case of brute-force search
- Doing the ray-intersection tests is the costliest operation in ray tracing
- Using k-d trees, the number of ray-intersection tests required can be brought down by a factor of 10-20.

Results

- ~150 seconds to render a scene with 17k triangles in full HD resolution with secondary rays and a depth limit of 3
- ~90 seconds to render a 800x600 image of the same scene with no secondary rays
- A speed up of 3x when rendering the same scene using K-d trees.

Conclusions

- K-d trees can significantly reduce rendering time in ray tracing
- K-d tree implementations on the GPU suffer from divergence due to different paths taken by rays in the same warp
- K-d tree memory access patterns are random as a result, and memory requests are not coalesced which impacts their performance