# GPU FINAL PROJECT PROPOSAL

*Name: Leiya Ma*
*UID:2018042*

## Summary

I am going to implement a real-time GPU Ray Tracer and profile its performance and compare the performance compared with the original CPU version.

## Background

Ray tracing is a hot topic in computer graphics capable of producing much higher fidelity images than traditional rendering methods. Besides, it is also a typical parallel GPU implementation. The basic idea is that we can determine how a scene looks by simulating how the light bright scenes up and reflex different color. Generally, this is simplified slightly by backtracking from the human eye to find which photons could possibly reach the human eye.

## The Challenge

Parallelizing ray tracing is rather straightforward (we can simply parallelize across the pixels in scene). What's more, doing so efficiently on a GPU is more interesting and challenging. I hope to optimize the GPU performance by manipulating thread-block assignment during the computation. For example, I wish to minimize the costs of ray divergence by performing basic hill climbing algorithm on the initial dimensions at the start of a trace. Besides, I also wish to leverage the spatial data structure to reallocate rays among CUDA threads in order to optimize memory access.

The idea is that by ensuring that our blocks can efficiently use shared memory and at the meantime avoid problems like bank conflict when utilizing shared memory. The object of the project can be concluded in three points:

- Efficiently updating rays in the spatial data structure after a trace step.
- Minimizing discrepancy between the times each block takes to execute.
- Minimizing the latency between trace steps.

## Goals and Deliverables

- Functional Real-time GPU Ray Tracer
- Report on analysis of different optimizations

## *Schedule*

| | |
|---|---|
| **May 12, 2016** | Project Proposal |
| **May 20, 2016** | 2016 CPU Version of Ray Tracer Implementation |
| **May 24, 2016** | 2016 Hill climbing |
| **May 28, 2016** | 2016 Optimized data structure + Dynamic ray assignment |
| **June 3, 2016** | 2016 GPU Version + Final Project Report |