15418 Milestone Report

Parallel Monte Carlo Path Tracing

Cassidy Diamond (cdiamond) and Akintayo Salu (asalu)

URL: https://dialmond.github.io/15418-project/

Work Completed

We initially began this project with a pretty superficial knowledge of raytracing, but have conducted significant research on the subject, modifying our goals. We have written the sequential version of our code, created multiple scenes to test on, and bespoke system to perform nuanced tests based on numerous parameters. Our testing system produces animated images as well, which we have a few to share.

Initially we wanted to use CUDA and OpenMP, and we're still on that. However, we've ran into somewhat significant challenges with implementing these parallel programming methods (outlined later).

I – Cassidy, after developing this testing suite and modifying the raytracing code from "Raytracing in One Weekend" as our sequential version – have also started implementing a parallel version of our code using pthreads because I believe this will help us diagnose the issues we've been facing with artifactual communication. I've put significant effort into parallelization using Pthreads, which I will outline in our final report, but effectively, I've only been able to achieve minimal speedup with 2 cores, and we actually slow down using more than that. I believe we will need to somewhat significantly modify our sequential version in order to avoid these issues – which through testing I've determined could be caused due to resource sharing between threads.

Goals and Deliverables

We're a bit behind our schedule but I think we can still get both CUDA and OpenMP versions of our code completed. I feel at least like we're close on the OpenMP front.

We've also changed some of our original goals based on more research on raytracing:

- We will still implement a CUDA and OpenMP version of the path tracing

- We have successfully created our benchmark scenes and images using our sequential implementation – goal completed, and then some, as we originally did not expect to support animated scenes.
- I now also run an experiment to augment the OpenMP version of the code based on how recursion is rendered in `ray_color`. In our project proposal we identified how ray tracing worked and noted one difficulty was the recursive nature of spawning and scattering rays. I believe that we may be able to achieve moderate speedup by modifying how often we produce scattered rays, and with what probabilities
    - Originally we wanted to introduce a notion of "correctness" to measure the effectiveness for other algorithms like Metropolis Light Transport. I still want to define "correctness" – now based on deviance from our sequential images – and use this as an additional metric in our ray scattering experiment.
- Originally we wanted to explore different algorithms like Metropolis Light Transport. However, after more research we have learned that these algorithms are more complex than we initially realized with our starter code, and furthermore, even if we did implement them they would not offer unique differences in parallelization. This may still remain a "nice to have" goal, but is less of a priority now.
- If we have more time than anticipated, I think we could even try using OpenMPI to parallelize, which might actually be faster to implement after solving our challenges with CUDA and OpenMP.
- Another possible nice to have goal would be implementing an interactive demo, since we already have the code for animations (i.e, the user moves their cursor around a screen and we use our rotation code to move their perspective of our scene live).

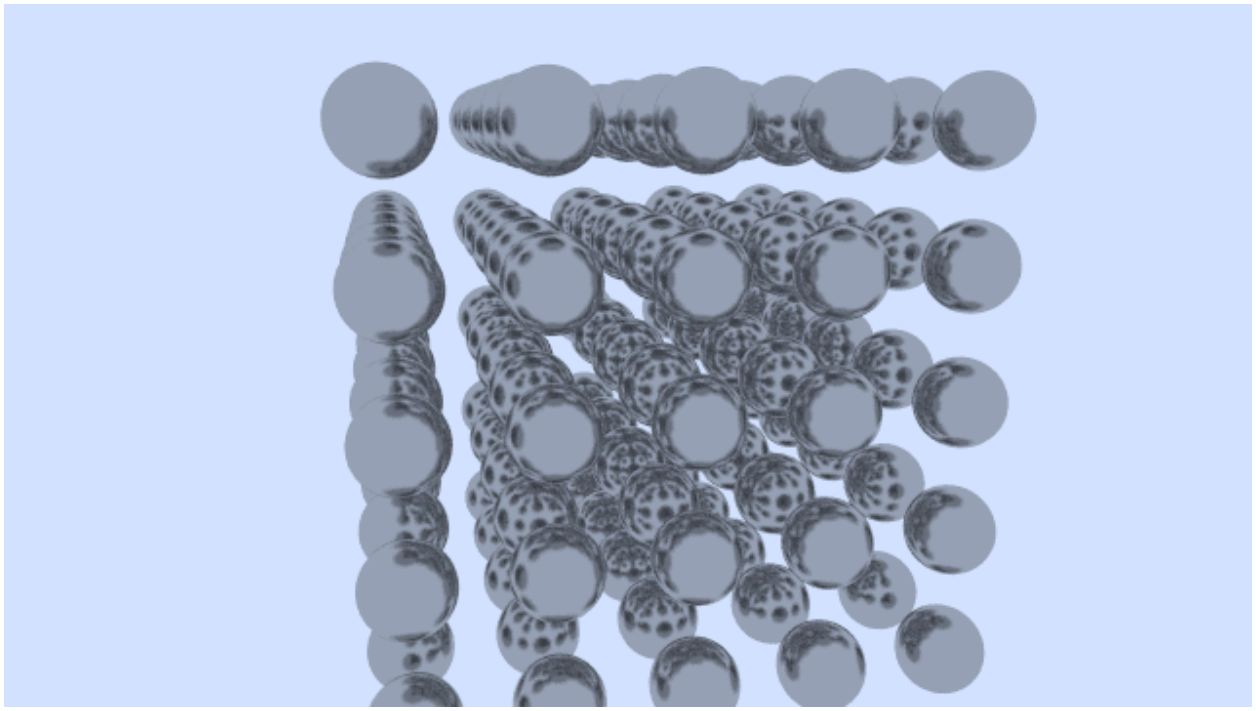
Remaining Schedule

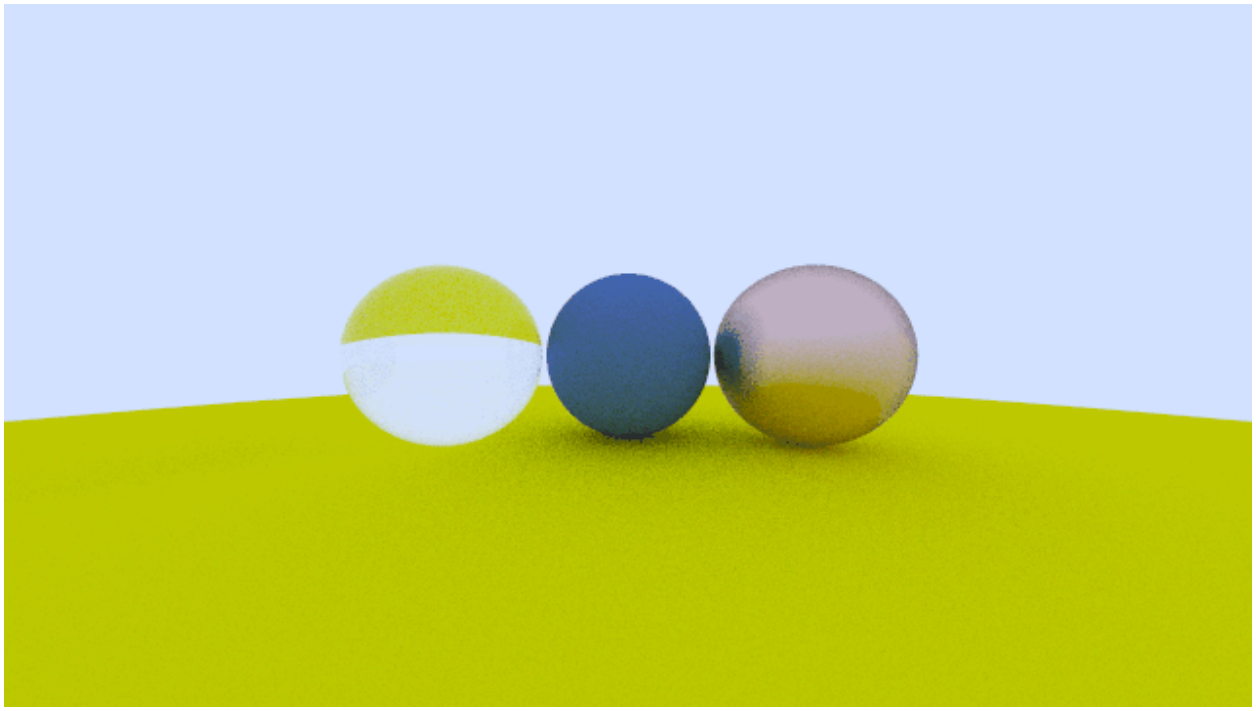
- Cassidy: By Saturday I want to finish OpenMP parallelization as well as additional experiments with how often we recurse on our rays, with analysis on this. Since I also added the `threads` implementation as a goal, I also want to finish this by then as well.
- Akintayo: CUDA rendering by Sunday, then further nice to have goals and final report
- Cassidy: By next Friday I want to finish the OpenMPI implementation if we decide to do that. I think benchmarking and creating performance results for our poster session should be pretty easy and quick to do . After that, work on other nice to haves and our final report.

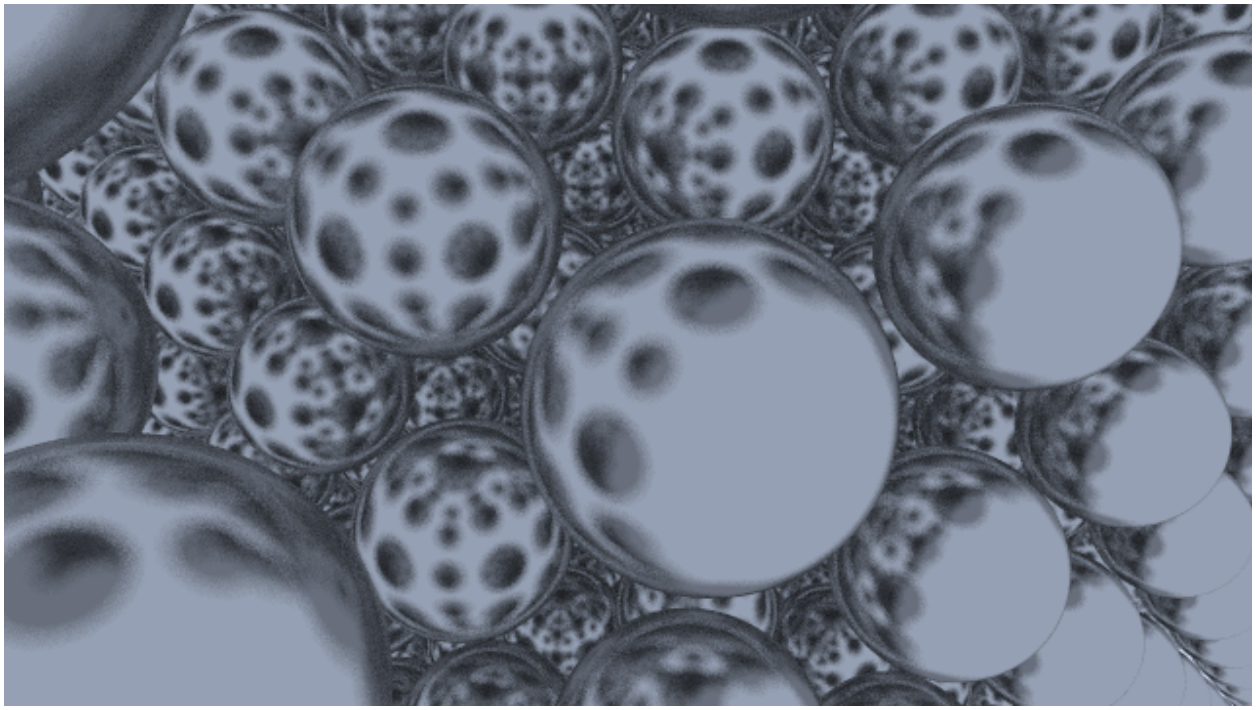
Issues and Concerns

OpenMP was supposed to be one of the easier implementations but it's been frustratingly challenging so far. CUDA has also been difficult too, but we think we can finish our required parallel components by this weekend.

Preliminary Results







Some debugging on our cache problems with pthreads, with a few flags in our renderer:

```
Performance counter stats for './render simple --num-threads 1 --samples-per-pixel 256':
624,418          cache-misses
14.244104454 seconds time elapsed

Performance counter stats for './render simple --num-threads 8 --samples-per-pixel 256'
4,036,457        cache-misses
38.989374637 seconds time elapsed
```