ECE408 Final Project Milestone 2

Group 08

Up until Milestone 2, we successfully implemented a workable GPU version of the Ray Tracing algorithm. The code is uploaded on GitLab.

Current Parallelization Approaches

There are a couple of modifications we made in order to have our code running properly. We may keep adjusting the details further later one. The changes we made includes but are not limited to:

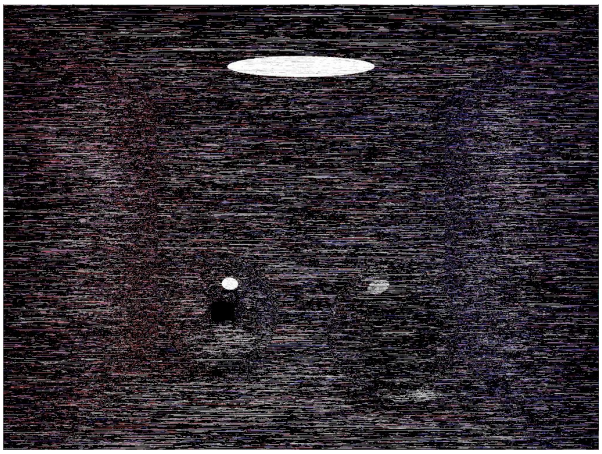
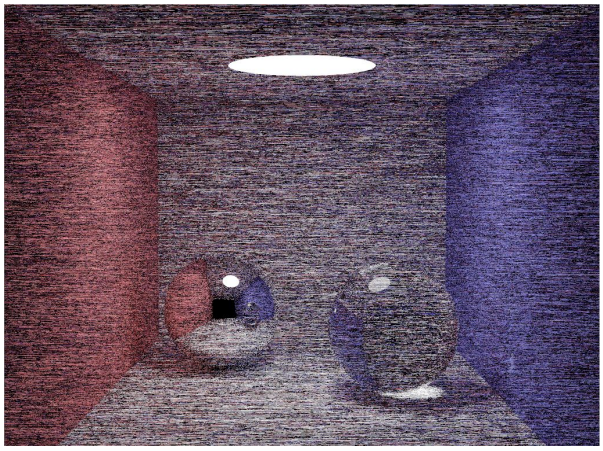
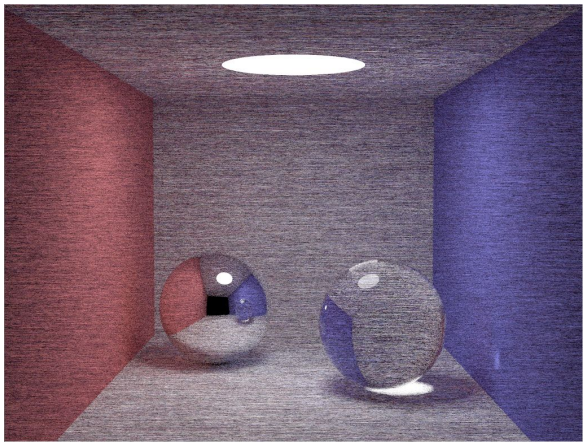
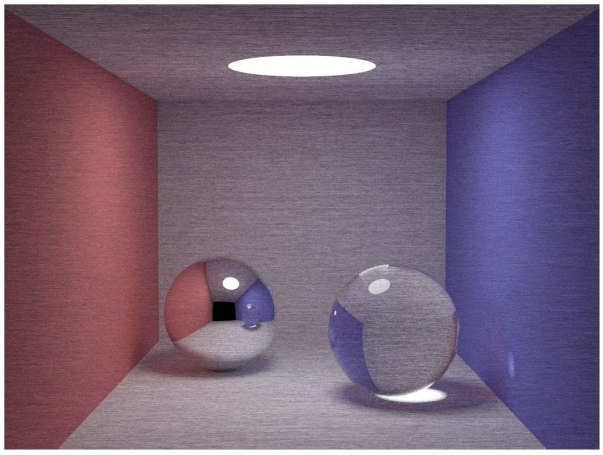
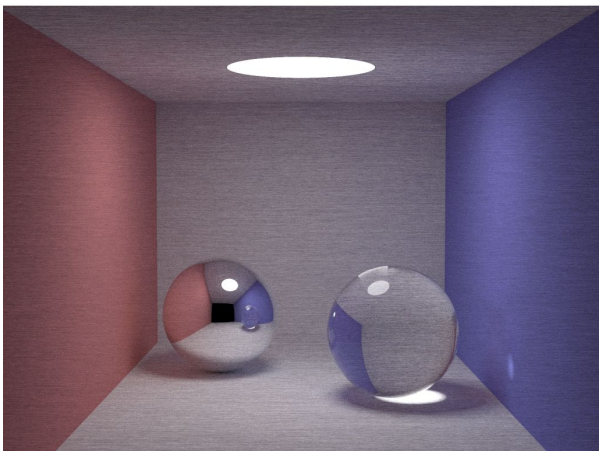
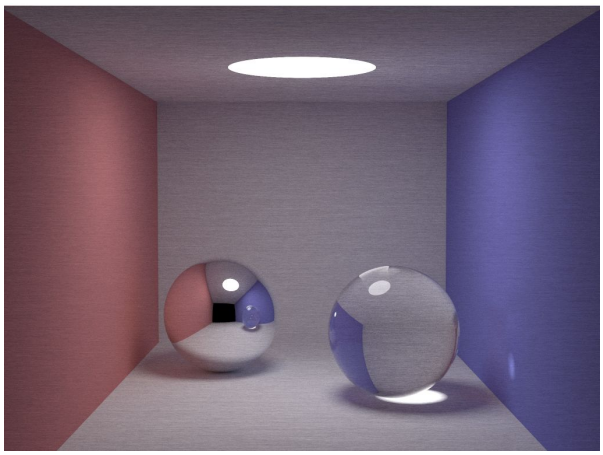
1. Using 2-dimensional thread blocks to cover the whole image, each thread does its own calculation for one pixel
2. Switch from recursion to iteration for function radiance() and limit the iteration depth to 10.
3. Use curand() instead of erand48() as the random number generator.

GPU Activities of Execution

Here we run the GPU version for 80000 samples per pixel, with a BLOCK_WIDTH of 16. NVprof is used for profiling. The execution time shows as follows.

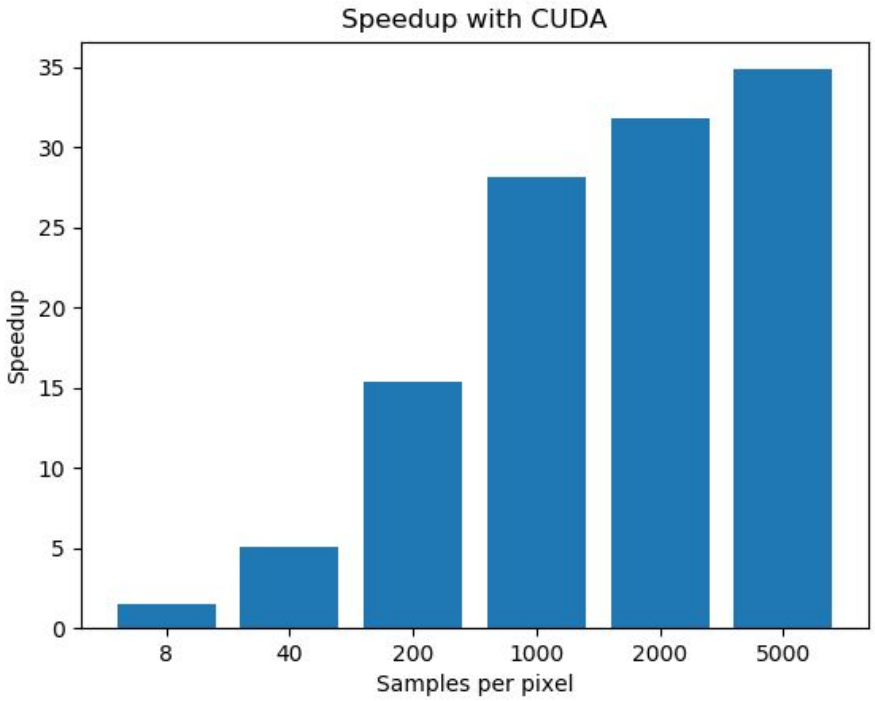
Name	Time (%)	Time
render(int, Sphere const *, Vec*, int)	99.99%	266.845s
[CUDA memcpy DtoH]	0.01%	2.0481ms
[CUDA memset]	0.00%	1.3120us
[CUDA memcpy HtoD]	0.00%	1.1200us

Resulting Images

	
spp = 8	spp = 40
	
spp = 200	spp = 1000
	
spp = 2000	spp = 5000

CPU (with OpenMP) vs. GPU Program Execution Time

Time (s)						
spp	8	40	200	1000	2000	5000
CPU	0.99	4.24	20.00	98.58	201.15	510.22
GPU	0.68	0.83	1.30	3.50	6.32	14.63
Speedup	1.46	5.11	15.38	28.17	31.83	34.87



Future Plans

After our initial implementation, we believe there is quite some room for optimization. Our approach will be the following subcategories:

1. Optimize intersection by eliminating unnecessary operations
2. Load sphere objects into constant memory instead of global memory
3. Pre-calculate/pre-frame constants
4. Optimize block size to maximize effective warps