Test Results

For testing our implementation we used spheres and a ground plane as the scene geometry which both can be checked for intersection in constant time. The geometry is stored in arrays on the GPU without any spatial data structure so the number of operations per pixel is $O(m + n + s(m+n))$ where *s* is the number of random samples, *m* is the number of spheres, and *n* is the number of planes. Each pixel ray must check for intersection with every scene object (m + n), and if so, send s sample rays that also check for intersection with every scene object.

The following results were from running our code on a single GPU node with the following specifications: *Dual Octocore Intel Xeon 2.4 GHz, 64GB Memory, Dual k20 NVIDIA*. For CPU comparison, we also ran a modified single threaded C++ version to run on an Intel Q6600 @ 2.6 Ghz.

# Samples, 15 Spheres	Constant Memory Run Time (ms)	Global Memory Run Time (ms)	Q6600 - single threaded (ms)
100	812	4,155	46,925
250	1,686	9,951	114,988
500	3,143	19,641	?
1,000	6,040	39,237	?

Table 1 - Average runtime in milliseconds of ambient occlusion on GPU and CPU, with different number of samples for ambient occlusion.

# Spheres, 250 Samples	Constant Memory Run Time (ms)	Global Memory Run Time (ms)	Q6600 - single threaded (ms)
5	725	5,183	58,766
15	1,685	9,950	114,987
50	4,792	28,747	?
100	9,155	52,935	?

Table 2 - Average runtime in milliseconds of ambient occlusion on GPU and CPU, with different number of spheres in the scene.

These results clearly demonstrate the performance benefit of using constant memory where possible. For handling large scenes and polygonal geometry, a spatial data structure such as a KD-Tree would greatly reduce the complexity of the algorithm.

Increasing the number of samples reduces the noise of the random algorithm. However, 100-200 samples is usually good enough for an ambient occlusion render pass, as the difference between 200 and 1000 samples is a small blur filter which can easily be done in post-production software. *Figures 4 - 6* show the outputs of different sample amounts.

Conclusion

Our code is not fully optimized, lacks a spatial data structure, and could benefit from experimenting with block/grid size. Nevertheless, these tests provide insight into the relative speeds of the different memory types and also the strong computing power of the GPU.

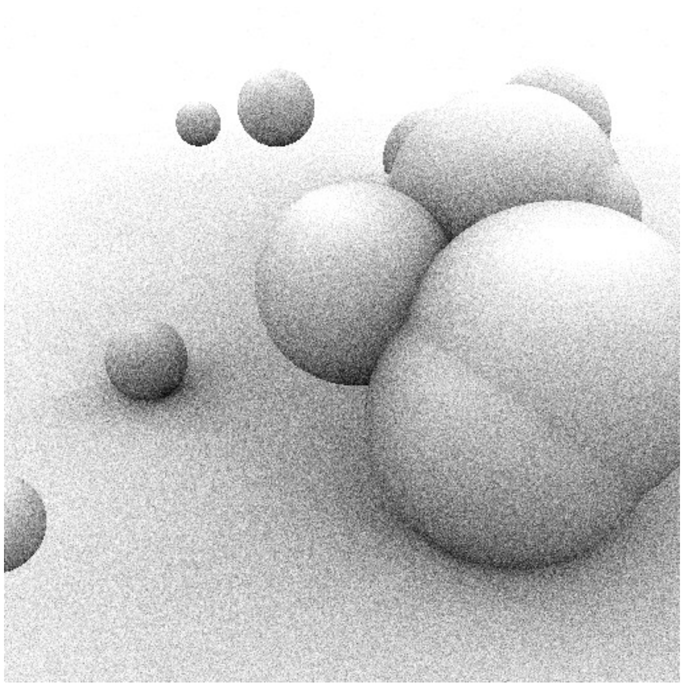


Fig. 4 - 20 samples per pixel

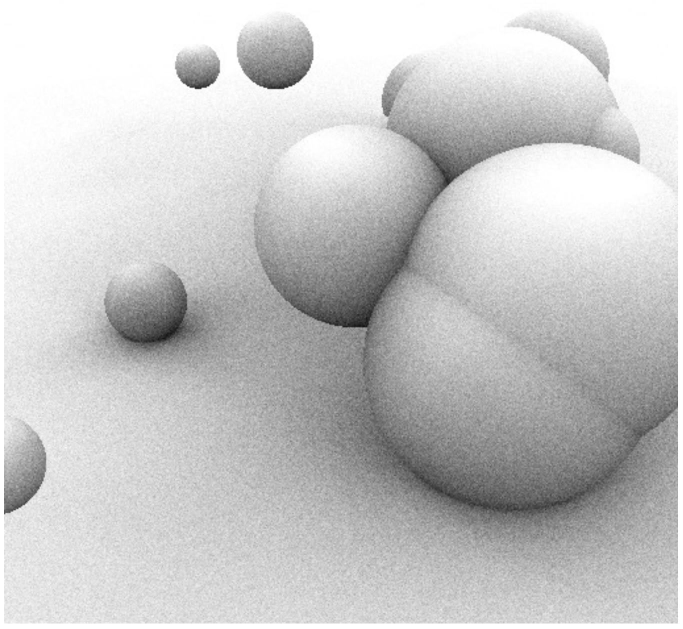


Fig. 5 - 100 samples per pixel

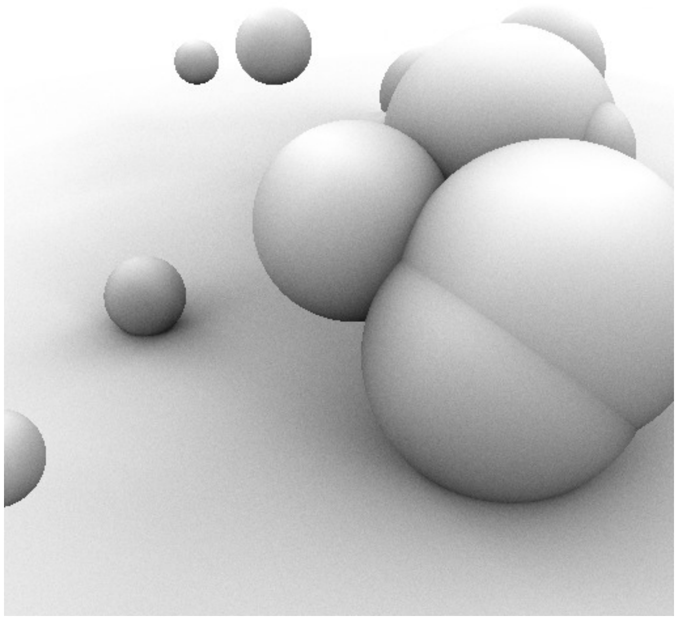
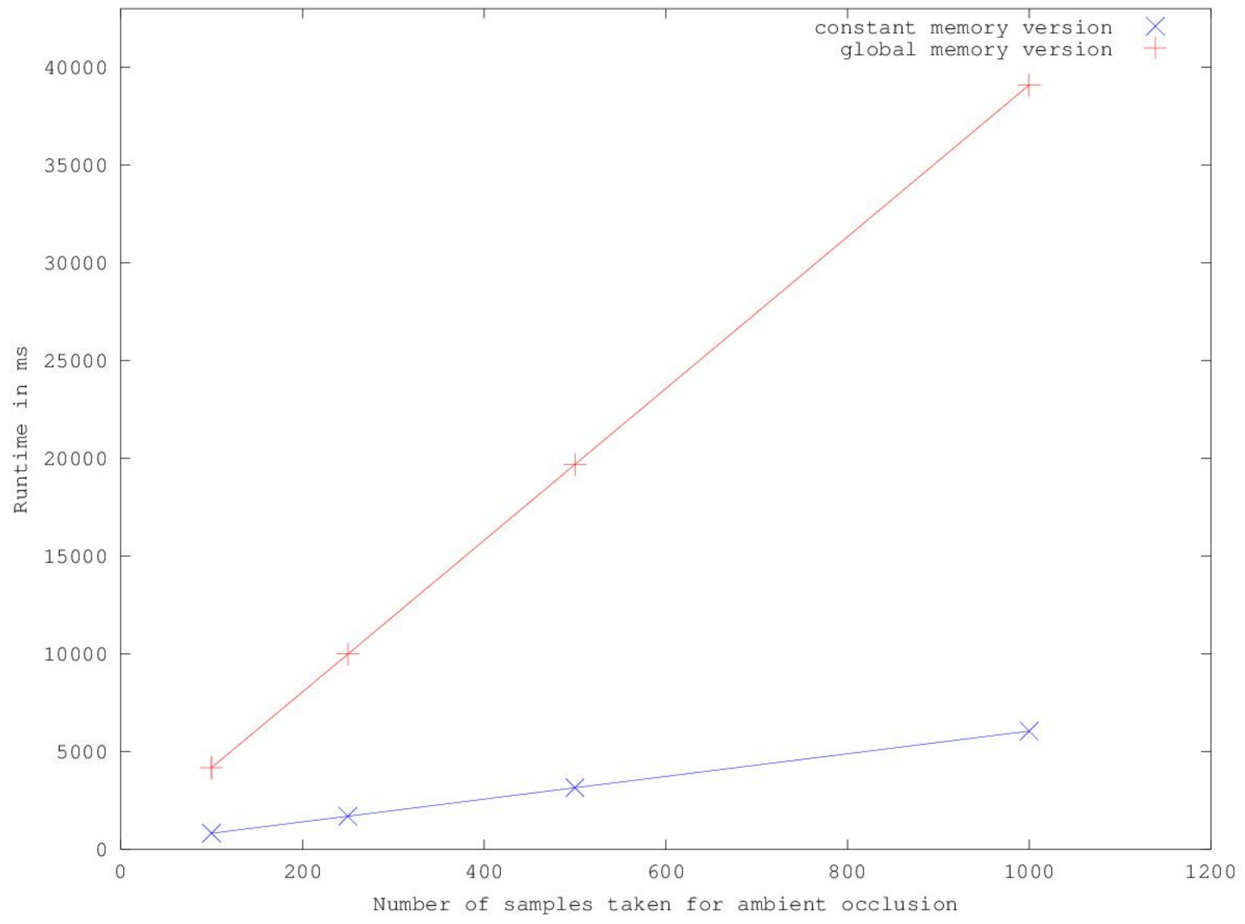
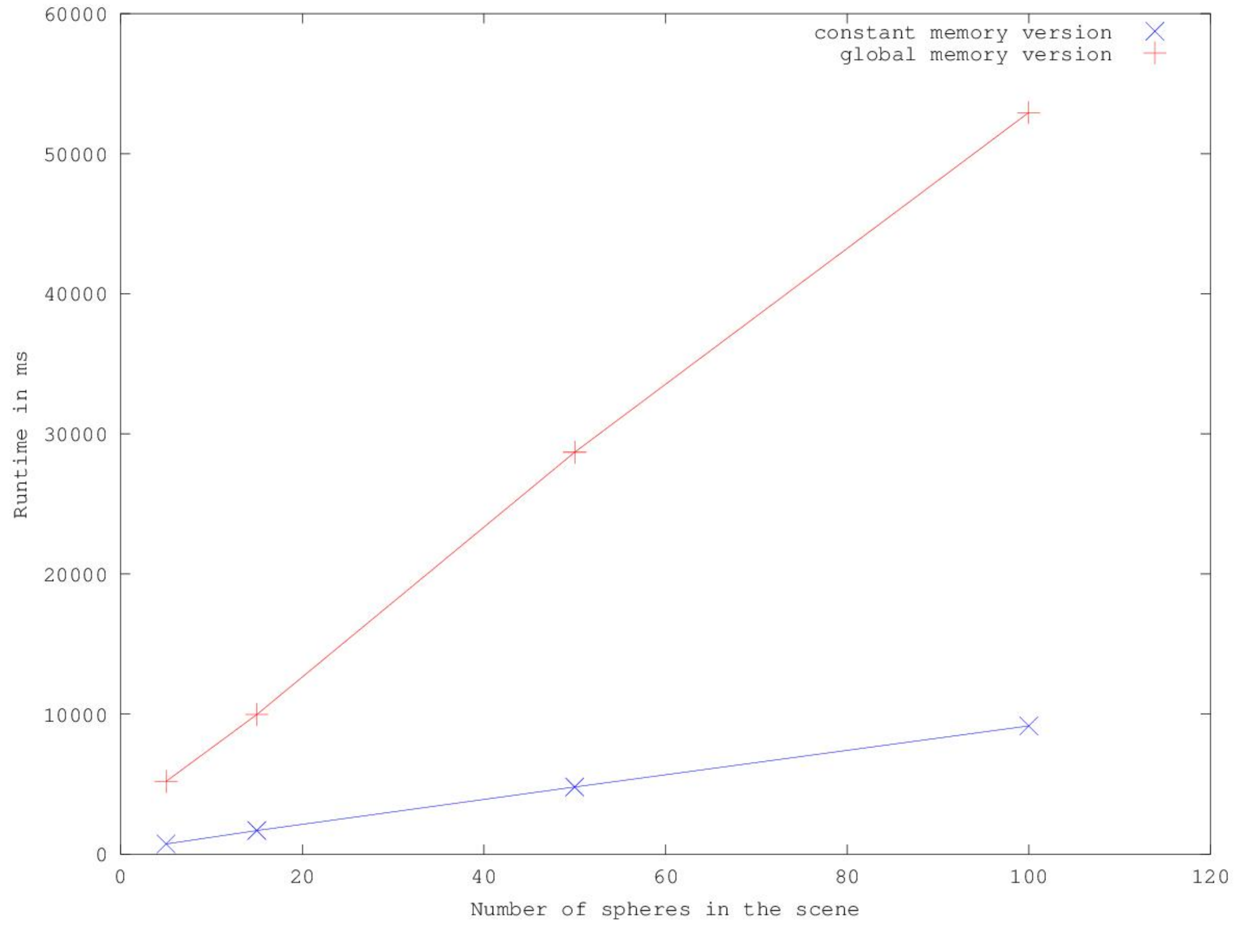


Fig. 6 - 1000 samples per pixel



Plot 1 - Runtime for fixed number of spheres in the scene and different number of samples.



Plot 2 - Runtime for fixed number of samples and different number of spheres in the scene.

Given fixed number of spheres, the run time of ambient occlusion with different numbers of samples taken.