Project Proposal - EEC 277
Accelerating Ray Tracing on The Graphics Pipeline

Yuxin Chen and Ahmed H. Mahmoud

21 February 2017

1 Abstract

We propose to explore few challenges regarding implementation of ray tracing algorithm through the graphic pipeline using OpenGL fragment shader. Using efficient data structure is the natural choice to think of when implementing a CPU-based ray tracing due to well-developed implementation of these data structure on CPU and virtually unlimited amount of memory available. It is expected that that mapping such data structure into fragment shader will have an overhead due to the nature of the graphics pipeline. In this work, we would like to explore the challenges of implementing ray tracing on GPU in terms of which data structure is more efficient, how the necessary elimination of recursion calls would affect the performance and comparison with CUDA-based ray tracing where greater flexibility is available.

2 Motivation

3 Related Work

Ray tracing was first introduced in 1980 [1] in order to introduce global illumination effects in the rendered scenes and thus more realistic effects are attainable. Since then extensive research has been done in order simulate

more effects like reflection from glossy surfaces [2, 3], reflective [4], anti-aliasing [5, 6, 2], motion blur[2, 7], depth of field [8], etc.

In parallel, substantial amount of research has been devoted to accelerate the process of ray tracing in order to include ray tracing in interactive applications to, for example, introduce more realism in computer games. One way to accelerate the ray racing is by utilizing efficient data structure. Without such data structure, every ray would be tested against all objects in the scene making the complexity linear with the number of the primitives in the scene.

The accelerated data structure are used in order to subdivided the scene space. There has been two ways for subdivision: spatial subdivision and object subdivision [6]. The spatial subdivision relies on subdividing the space into sub-regions and storing primitives in each sub-region. If a ray passes through a sub-region (which is less expensive to compute), only primitives in this sub-region are checked against. One the simplest techniques for spatial subdivision is the uniform or adaptive grid where each primitives is registered into one of the grid cells [9]. Then only primitives in a certain cell are tested if a ray hits that cell. Better performance can be gained by using tree structure instead (BSP trees, kd-trees, octrees) especially for scenes with nonuniform distributions of geometry and/or with arbitrarily moving objects especially. Object subdivision breaks down the objects in a scene into smaller constituent objects which makes it easier to cull the parts that does not hit a ray. Objects here are made of primitives. Each major object contains minor objects. Thus, we can build a tree for the major objects of minor objects [**?**].

4 Target

The target of our project can be summed in the following milestone

1. Implementing of ray tracing on fragment shader. (1 Week)
2. Implementing uniform gird, kd-tree to accelerate the performance
3. Experimenting with complex scenes. (1 Week)
4. In parallel, implementation on CUDA for final comparison between OpenGL-based and CUDA-based ray tracing. (1 Week)
5. Report and presentation. (1 Week)

References

[1] T. Whitted, "An improved illumination model for shaded display," in *ACM SIGGRAPH Computer Graphics*, vol. 23. ACM, June 1980, pp. 343–349.

[2] R. L. Cook, T. Porter, and L. Carpenter, "Distributed ray tracing," in *ACM SIGGRAPH Computer Graphics*, vol. 18, no. 3. ACM, 1984, pp. 137–145.

[3] G. J. Ward, F. M. Rubinstein, and R. D. Clear, "A ray tracing solution for diffuse interreflection," *ACM SIGGRAPH Computer Graphics*, vol. 22, no. 4, pp. 85–92, August 1988.

[4] J. Amanatides, "Ray tracing with cones," in *ACM SIGGRAPH Computer Graphics*, vol. 18, no. 3. ACM, 1984, pp. 129–135.

[5] P. Shirley and R. K. Morley, *Realistic Ray Tracing*. AK Peters, Ltd., 2003.

[6] M. Pharr, W. Jakob, and G. Humphreys, *Physically Based Rendering: From Theory to Implementation*, 3rd ed. Morgan Kaufmann, 2016.

[7] R. L. Cook, "Stochastic sampling in computer graphics," *ACM Transactions on Graphics (TOG)*, vol. 5, no. 1, pp. 51–72, January.

[8] M. Shinya, "Post-filtering for depth of field simulation with ray distribution buffer," in *Graphics Interface*. Canadian Information Processing Society, 1994, pp. 59–59.

[9] J. M. Snyder and A. H. Barr, "Ray tracing complex models containing surface tessellations," *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, pp. 119–128, Aug. 1987. [Online]. Available: http://doi.acm.org/10.1145/37402.37417