

# Cache Scheduling for a Ray Architecture

Susan Frank, Kevin Kreeger

Department of Computer Science, SUNY at Stony Brook, NY 11794-4400

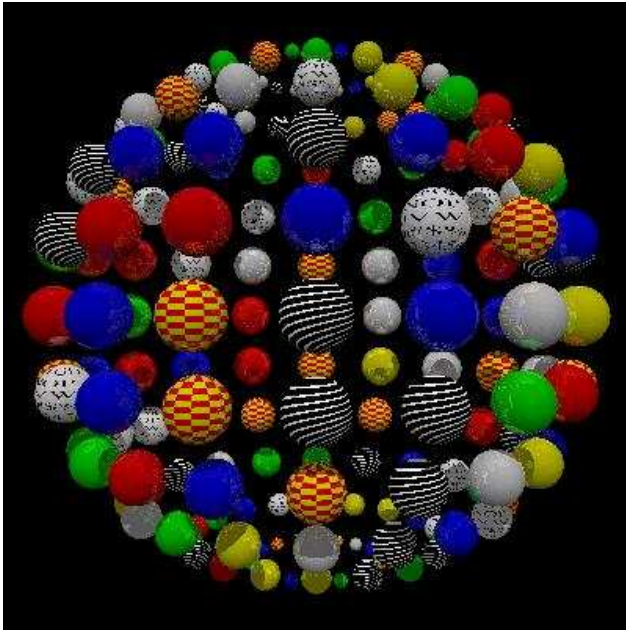
## Extended Abstract

Direct volume rendering has become an important part of many applications such as visualization of 3D sampled medical data (CT, MRI). Various researchers have proposed special purpose hardware for volume rendering. The Cube family of architectures has been developed at SUNY Stony Brook University [2]. Cube-5 is the first programmable special-purpose scalable architecture supporting real-time, high-quality volume rendering of high-resolution datasets. It is a universal 3D rendering and processing engine. Ray tracing has been one of the widely utilized global illumination techniques to generate realistic images in volume visualization. The goal of this research is to develop an efficient cache scheduling of algorithm for the ray-tracing paradigm in order to help achieve interactive volume processing and rendering.

This work is an extension of the RAYA project at Mitsubishi Electronic Research Labs (MERL) [3]. This design was inspired by [4]. The paper presents a framework for rendering large scenes with ray tracing without holding the entire scene in memory. The authors divide the scene into a 3D grid of scheduling cells. Due to the rendering equation decomposition described in [1] rays can be partially rendered as they intersect the geometry in a cell. Hence a subset of a large geometry and texture database can be cached in memory, with the partial result being propagated with the ray. Ray queues are used to determine which rays to render (partially) in each cell. The RAYA project extends this idea to include volumes as well as geometry data.

The volume is initially store in main memory. The initial rays are generated and each placed on a *ray queue* for the cell of the first volume that it intersects. For the first frame of a scene, the Max Work algorithm is used. This algorithm schedules the cell with the highest number of rays on its ray queue. This algorithm is easy to implement but doesn't always yield the optimal schedule.

We are developing an algorithm based on the dependency graph to take advantage of frame to frame coherence of the rays. Each ray can recursively spawn multiple rays. These rays can revisit the same cells as an ancestor on the same ray. Thus, the dependency graph is cyclic. As the rays are created and propagated through the volume, we build a cell dependency tree. We add virtual links between nodes with the same cell, and between each node and it's parent, thus turning the tree into a graph with cycles. By assigning the proper weights to the edges of the graph the problem is reduced to the well-known NP-Complete Traveling Salesman optimization problem. We plan to implement TSP heuristics to determine the schedule to follow for subsequent frames. In the event that the schedule is no better than the Max Work schedule, this will be followed for the next frame. We will test various combinations of volume, cache, cell, and main memory sizes, as well as different memory hierarchy levels. Figure 1. Shows a typical image used for our tests. This type of image is used because it has many reflections.



**Figure 1** Ball of spheres

## References

- [1] J. Kajiya, "The Rendering Equation", *Proceedings of SIGGRAPH 1986*, pages 143-150, August 1986.
- [2] H.Pfister and A. Kaufman, "Cube-4 A Scalable Architecture for Real-Time Volume Rendering", *Proceedings of 1996 Symposium on Volume Visualization*, pages 47-54, October 1996
- [3] H.Pfister and K. Kreeger, "RAYA: A Ray Tracing Architecture for Volumes and Polygons", Whitepaper, 1999
- [4] M.Pharr, C. Kolb, R. Gershbein and P. Hanrahan, "Rendering Complex Scenes with Memory-Coherent Ray Tracing", *Proceedings of SIGGRAPH 1997*, pages 101-108, August 1997.