Far Voxels Summary

Trenton Plager

Introduction & Related Work

In the introduction, the paper's authors detail why their approach for rendering is necessary by examining the shortcomings of other various rendering techniques. They state that in many of these techniques, memory is proportional to the number of pixels on screen rather than model complexity, but they are still limited because they do not integrate features such as LOD management. As a result, many of these models perform best with fairly smooth and regular geometry. The authors' approach integrates visibility culling and out-of-core data management with LOD construction and rendering. As a result, their model is more of a volumetric approach and it is applicable to a much wider range of model classes. As for related work, they cite a few different approaches that are closely related. The first is out-of-core view-dependent simplification for which they cite their method of using coarse multiresolution structures that compose at run-time pre-assembled optimized primitive groups. They also cite visibility culling, hybrid rendering approaches, and interactive ray-tracing approaches as related work.

Methods (Sections 3 – 5)

The authors begin their methods sections by explaining that their method changes the grain of the multiresolution surface model from points or triangles to small procedural volumetric clusters. This allows them to perform coarse-grained view-dependent refinement of the model rather than fine-grained. For this process, the model is first partitioned with an axis aligned BSP tree. Full resolution data is partitioned into fixed triangle count chunks, and inner nodes become a fixed number of cubical voxels arranged in a grid. The BSP tree is then used to construct the level-of-detail structure. Each leaf node of the tree contains a triangle strip that represents its geometry. These strips are then clipped to the node's bounding box. Ray casting is then used to determine visibility of each voxel. Simple parameterized shaders are then fit onto each voxel. Once this has been completed, view dependent rendering is conducted, in which each node is checked to see if it falls within the viewport. If it doesn't, it and its entire subtree are not rendered. If it does, it is rendered.

Results

In the results section, the authors reveal that they tested their method extensively using 3 different models. Each of these 3 models: the St. Matthew, the Richtmyer-Meshkov Isosurface, and the Boeing 777, represent highly complex benchmarks that they say may serve as the best benchmarks for each of their respective domains. In the preprocessing subsection, they state that their method apparently performed slower than other methods, even though is seemed to scale better. Regarding adaptive rendering, their approach seems to have performed extremely well. Their system maintains interactivity, which is favored by their model rather than accuracy. They also attempted some tests using network streaming. These tests showed that their system is in

fact usable for very large models on standard networks because its rendering rate remains the same as a local version, even though latency progressively increases.

Conclusion

In conclusion, the authors state that their method is an efficient technique for view-dependent rendering of complex models on today's graphics platforms. They state that the main benefit is its performance and wide applicability compared to the very specific classes of models that are required by some other rendering techniques. They also state that their future work is focused on compression, exploring alternate view-dependent voxel representations, implementing multi-resolution sampling methods, exploring higher quality texture-based volume rendering, and prefetching and predicting occlusion/visibility events to reduce artifacts found in their current method.