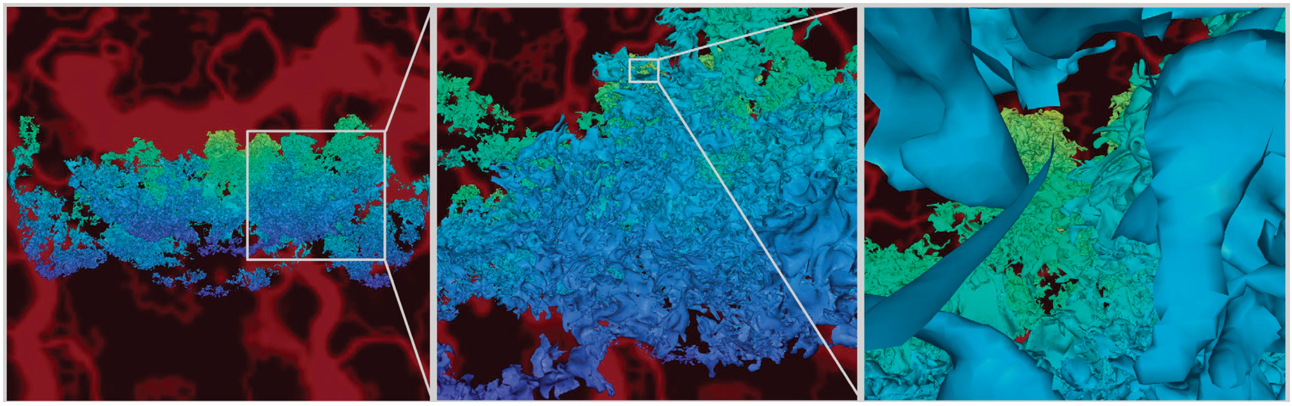
Quick-VDR: Interactive View-Dependent Rendering of Massive Models in Commodity GPUs

Department of Computer Science University of North Carolina at Chapel Hill November 2004


These images show the application of Quick-VDR to a complex isosurface (100M triangles) generated from a simulation of Richtmyer-Meshkov instability and turbulence mixing. The middle and right images show zoomed views. The isosurface has high depth complexity, holes, and a very high genus. Quick-VDR can render it at 11 – 21 frames per second on a PC with NVIDIA GeForce FX5950 card and uses a memory footprint of 600MB. *Image courtesy of the LLNL ASCI VIEWS Visualization project.*

The Challenge

We present a novel approach for interactive view-dependent rendering of massive models on commodity GPUs. Our algorithm combines view-dependent simplification, occlusion culling, and out-of-core rendering. In general, view-dependent simplification is not well suited for current graphics hardware because it requires high bandwidth between the CPU and GPU and works well only if all the geometry resides in GPU memory. To overcome these challenges, we present a novel scene representation, a clustered hierarchy of progressive meshes (CHPM). Also, our system incorporates a frame latency system to allow extra time for loading newly visible clusters after occlusion culling.

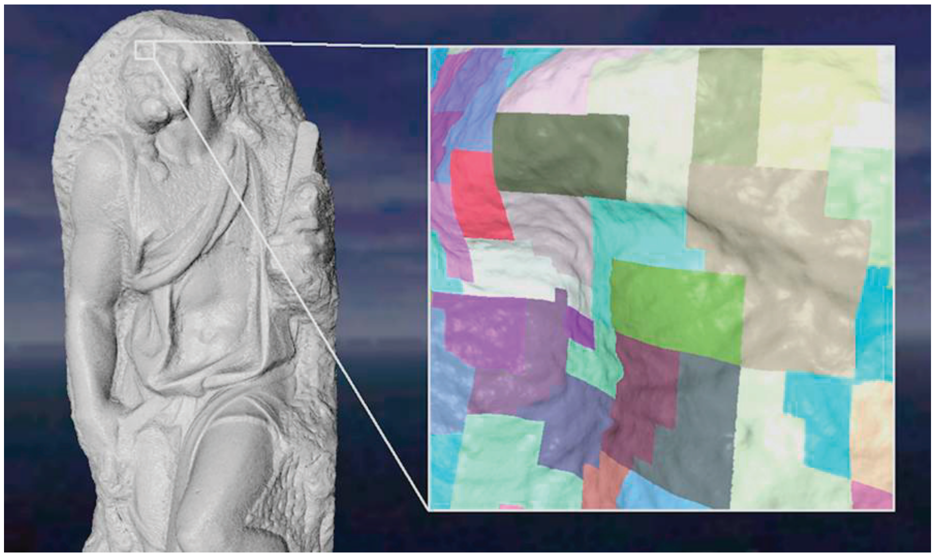
Highlights

- **Lower refinement cost: Very low overhead of view-dependent refinement in the CHPM.**
- **Massive Models: computing drastic simplifications of massive models necessary for interactive rendering.**
- **Runtime performance: exploiting the features of GPU to obtain a high frame rate.**
- **Rendering quality: high image quality and alleviate popping artifacts.**
- **Generality: applicable to all types of polygonal models.**

Approach

To overcome these challenges, we present a novel scene representation, a clustered hierarchy of progressive meshes (CHPM). We use the cluster hierarchy for coarse-grained selective refinement and progressive meshes for fine-grained local refinement. The CHPM allows a high refinement rate and also reduces the visual popping between successive frames. The clusters are used for occlusion culling and out-of-core rendering. We perform out-of-core management of GPU memory resources at the cluster level and send only the vertices and indices of refined clusters between successive frames. Using the vertex buffer object (VBO) extension, we can achieve a throughput of over 20 million triangles per second rendering massive models. Our rendering algorithm uses temporal coherence and hardware accelerated occlusion queries at the cluster level. The set of previously visible clusters is used as a potentially visible set (PVS) for rendering an occlusion representation in the current frame. Then, we check if each active cluster is visible with respect to the occlusion representation using occlusion queries.

Newly visible clusters are then rendered to complete the final image. For out-of-core rendering we prefetch clusters for LOD changes. Occlusion events are difficult to predict so we add a frame of latency in the overall pipeline for fetching newly visible clusters. To efficiently implement this latency approach, we use a pair of off-screen buffers (pbuffers) and render each pair of consecutive frames into different buffers (see Fig. 3). We have implemented and tested Quick-VDR on a commodity PC with an NVIDIA GeForce FX 5950 Ultra GPU with 128MB RAM. To illustrate the generality of


Scan of Michelangelo's St. Matthew. This 9.6GB scanned model consists of 372M triangles. The right inset image shows clusters in color from a 64K cluster decomposition of the model. Quick-VDR is able to render this model at 13-23 frames per second using a memory footprint of 600MB. *Image courtesy of the Digital Michelangelo Project at Stanford University.*

Project Leader
Dinesh Manocha, professor

Graduate Research Assistants
Sung-Eui Yoon, Brian Salomon, Russell Gayle

Research Sponsors
U.S. Army Research Office
National Science Foundation
Office of Naval Research
Defense Advanced Research Projects Agency
Intel Corporation

Key Words
Interactive display, view-dependent rendering, occlusion culling, external-memory algorithm, levels-of-detail

our approach we have tested the algorithm's performance on several classes of models: a complex power plant CAD environment (12M triangles), the scanned St. Matthew sculpture (372M triangles), and an isosurface from a fluid simulation (100M triangles). We can render these models at $10 - 35$ frames per second using a limited GPU memory footprint of $50 - 100$MB.

Rendering Pipeline
Our rendering algorithm uses temporal coherence and occlusion queries at the cluster level. We check if each active cluster is visible with respect to the occlusion representation using occlusion queries. Newly visible clusters are then rendered to complete the final image. Occlusion events are difficult to predict so we add a frame of latency in the overall pipeline for fetching newly visible clusters.

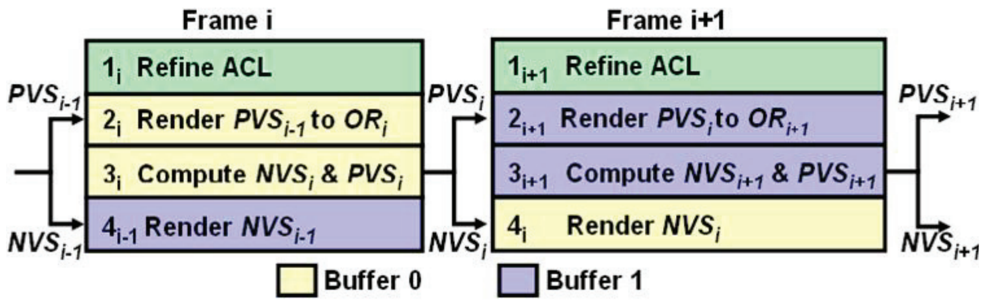
Clustered Hierarchy of Progressive Meshes (CHPM)
The CHPM consists of two parts: the cluster hierarchy and progressive meshes. We use the cluster hierarchy for coarse-grained selective refinement and progressive meshes for fine-grained local refinement. The CHPM allows a high refinement rate and also reduces the visual popping between successive frames. These two levels of refinement make a CHPM a middle ground between pure view-dependent and static simplifications. We also introduce cluster dependencies to simplify boundary triangles between clusters.

Selected Publications
Yoon, S-E., B. Salomon, R. Gayle, and D. Manocha. Quick-VDR: Interactive View-Dependent Rendering of Massive Models, *IEEE Visualization 2004*, Austin, USA.

Yoon, S-E., B. Salomon, M. Lin, and D. Manocha. Fast Collision Detection between Massive Models using Dynamic Simplification, *ACM SIGGRAPH/Eurographics Symposium on Geometry Processing 2004*, Nice, France.

Yoon, S-E., B. Salomon, and D. Manocha. Interactive View-dependent rendering with Conservative Occlusion Culling in Complex Environments, *IEEE Visualization 2003*, Seattle, USA.

For More Information
http://gamma.cs.unc.edu/QVDR



Frame i
Frame i+1
1_i Refine ACL
2_i Render PVS_{i-1} to OR_i
3_i Compute NVS_i & PVS_i
4_{i-1} Render NVS_{i-1}
1_{i+1} Refine ACL
2_{i+1} Render PVS_i to OR_{i+1}
3_{i+1} Compute NVS_{i+1} & PVS_{i+1}
4_i Render NVS_i
PVS_{i-1}
NVS_{i-1}
PVS_i
NVS_i
PVS_{i+1}
NVS_{i+1}
Buffer 0
Buffer 1

Our Rendering Pipeline. In frame i occlusion culling is performed for frame i but the final image for frame i-1 is displayed. This allows extra time for loading the PMs of newly visible clusters. Two off-screen buffers facilitate this interleaving of successive frames. The partial rendering of frame i is stored in one buffer while occlusion culling for frame i + 1 occurs in the other buffer.