

# Progressive Volume Rendering of Unstructured Grids

Steven P. Callahan<sup>1</sup>, Valerio Pascucci<sup>2</sup>, Cláudio T. Silva<sup>1</sup>

<sup>1</sup> Scientific Computing and Imaging Institute, University of Utah 

<sup>2</sup> Center for Applied Scientific Computing, Lawrence Livermore National Laboratory 

## Abstract

We describe a new progressive technique that allows real-time rendering of large tetrahedral meshes. Our approach incrementally refines the quality of the volume rendering until it converges on the final image. The results from the previous refinement steps are re-used in each subsequent refinement, thus leading to an efficient rendering. By operating on a simple set of triangular faces, our algorithm allows a robust and straightforward graphics hardware (GPU) implementation. Because the mesh is rendered in steps, interactive rendering is possible for a wide range of data sets and hardware configurations.

## Method

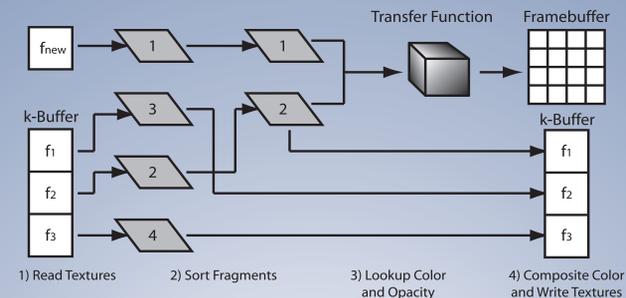
### Server:

- For each viewpoint
  - Stream boundary triangles
  - Stream remaining triangles in front-to-back order

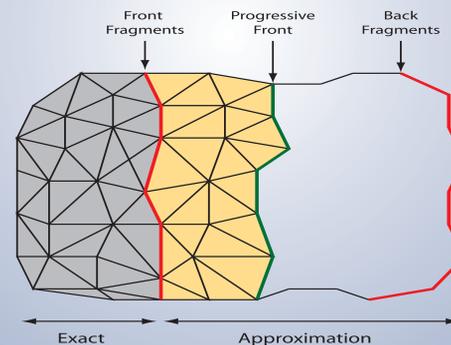
### Client:

- For every boundary triangle received from server
  - Rasterize triangle
- For every fragment rasterized
  - If front or back fragment, insert into *k*-buffer
- For every group of triangles received from server
  - Rasterize triangle
  - For every fragment rasterized
    - Insert fragment into *k*-buffer
    - Find closest two fragments, composite their contribution into *exact* buffer
- For every pixel in *k*-buffer
  - Find front and back fragments, composite their contribution into *approximate* buffer
- Composite *exact* buffer over *approximate* buffer and display to screen
- Clear *approximate* buffer

The *k*-buffer algorithm: A fixed number of fragments are stored as textures on the GPU. As a new fragment is rasterized, the closest two fragments are used to lookup color and opacity which are composited into the framebuffer.



Progressive volume rendering: The progressive front traverses the mesh in front-to-back order. A progressive image is created by combining the finished region with an approximation of the unprocessed space between the boundaries.



Results: Five progressive steps of the Langley Fighter data set consisting of 1.4 million tetrahedra. The first step contains only the boundary faces and the last step contains the whole mesh.

## Introduction

Interactive visualization of large unstructured grids has been a goal of the research community for many years. With the advent of programmable graphics processing units, volume rendering large grids has become a reality. However, the need still exists to render even larger meshes interactively.

Our progressive algorithm is based on the Hardware-Assisted Visibility Sorting algorithm [1], which operates on the triangles that compose the tetrahedra. In image-space, sorting is performed using a fragment stream sorter called the *k*-buffer. Our algorithm works by dividing the load between a client and a server. The server incrementally transmits a stream of triangles while the client renders these triangles using the *k*-buffer. At each incremental step, a progressive image is displayed using the results of all the previous steps with an approximation of the unprocessed portion of the mesh. The result is a system that can handle arbitrarily large meshes interactively while still allowing full-quality volume renderings.

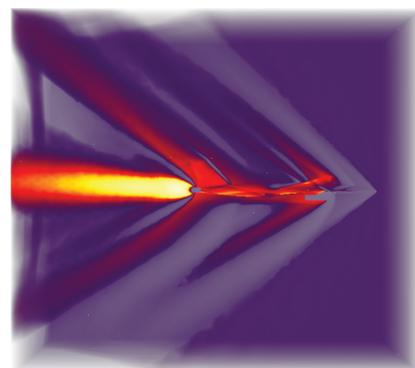
## Results

The images shown below contain the results of our progressive volume rendering using five refinement steps. Preliminary results show that our progressive algorithm is extremely useful for visualizing large data sets interactively. In fact, most of the important features in the volume can be detected very early in the refinement process.

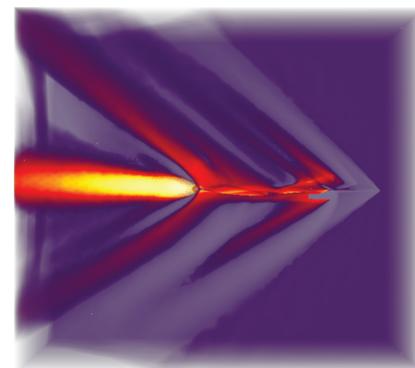
To further improve our technique, we need to explore the use of compression for the transmission of the triangle stream. In addition, through the use of cache-efficient layouts, we believe that data traversal can be optimized on the client side.

### References:

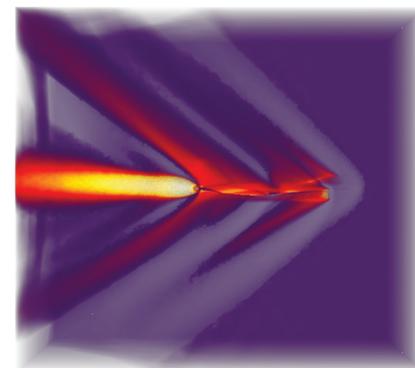
[1] S. P. Callahan, M. Ikits, J. L. D. Comba, and C. T. Silva. Hardware-Assisted Visibility Ordering for Unstructured Volume Rendering. In *IEEE Transactions on Visualization and Computer Graphics*, 11(3):285-295, 2005.



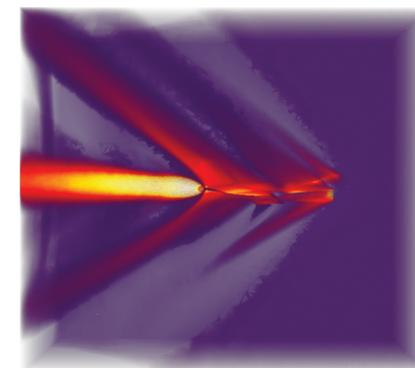
Step 1



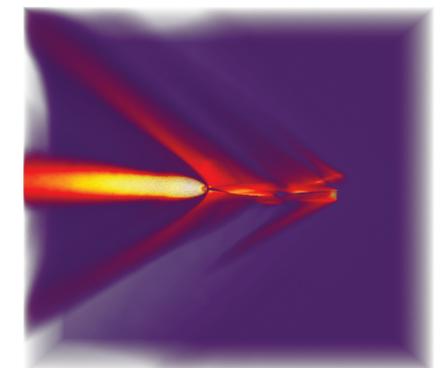
Step 2



Step 3



Step 4



Step 5