Sorting and cell compositing for irregular meshes

Nelson Max, Peter Williams, Richard Cook, Claudio Silva
Large LLNL simulations have

- Irregular meshes of mixed “zoo” cell types: hexahedra, tetrahedra, pyramids & prisms
- Millions of cells of different sizes
- Non-planar quadrilateral faces
- Non-convex data volumes
Polyhedron compositing

- Composite RGBA cell projections using software or hardware polygon rendering

- Requires global back-to-front visibility sort, so that if cell A hides cell B, B comes first
Topological sort for convex data volume of convex cells with planar faces

- Build a directed graph, with an edge for every interior face, directed at the cell on the same side of the face plane as the viewpoint
• Count dependency (incoming edges) for each cell
• Put cells with dependency count zero on queue
• While queue is not empty, move next cell A on queue to output list. Decrement dependency counts for each cell B pointed to by A. If B’s dependency count becomes zero put B on queue.
• Visibility cycle if queue empty and cells left over.
SXMPVO for non-convex volumes

- Sort exterior faces by centers of gravity
- For each exterior face in back to front order
  For each pixel in face, add face to depth sorted list of exterior faces for pixel

- For each pixel’s list
  Skip first entry
  For each successive pair (A, B) in pixel list
  add a new directed edge from A to B, if one is not yet present
Software cell projection algorithm

- Scan convert RGB$_\tau$z of front faces into front buffer
- Scan convert RGB$_\tau$z of back faces into back buffer
- For all pixels
  - Compute length of ray segment from frontz - backz.
  - Do analytic integrations for RGBA
  - Composite into final image
Hardware cell projection algorithm

- Divide image plane into polygons by adding projected edges one by one to a winged edge data structure.
- For each vertex, interpolate front and back RGB$\tau z$ ($\tau =$ extinction coefficient) and do RGBA integrals.
- Convert polygons to triangle fans for hardware to interpolate and composite.
Producing correct opacity per pixel

- If the extinction coefficient $\tau(s)$ varies linearly along the ray, with average $\tau$, and $d$ is the length of the ray segment, the opacity is

$$A = 1. - \exp(-\tau d)$$

- $d$ and $\tau$ vary linearly across each polygon so they can be interpolated as texture coordinates

- Load texture_table(u, v) with $1. - \exp(-uv)$
A cell with a non-planar face $F$

- Divide $F$ into two triangles, using diagonal from vertex of lowest index.
- A face $F$ or cell $A$ is a problem face or cell if a viewing ray can intersect it twice.
- Subdivide all problem cells into tetrahedra.
Parallelizing projection and integration

- Multiple projection threads take blocks of cells from sorting thread, and prepare blocks of triangle fans for OpenGL thread
Piecewise-linear transfer functions

• Exact color integration only possible for linear transfer functions
• Divide all cells with contour surfaces or breakpoints in the transfer functions into tetrahedra. Then divide the tetrahedra into slabs at the contour values.
• Sort tetrahedra
• Sort slabs in each tetrahedron
• Draw contour polygons
Quadratic tetrahedra (10 nodes)

- Position linear; function quadratic
- Divide into curved slabs
- Ray trace to quadric contours
- Gauss integration for color on ray segments within slabs