Introduction to Scientific Visualization

Aaron Birkland
Cornell Center for Advanced Computing

Data Analysis on Ranger
January 2012
A lab-intensive workshop

• Start off with basic concepts
  – Data, transformations, graphics, techniques
• Learn the tools
  – Hands on with ParaView and VisIt
• Learn the resources
  – Longhorn visualization cluster, large scale parallel visualization.
• Try it out!
From data to Insight

Data Representation → Visualization Operations → Graphics Primitives → Display

Iteration and Refinement
Points, Meshes & Coordinates

From *The Visualization Toolkit* by Schroeder et al.
Data

- Values at each point
- Type and nature will determine applicable techniques
  - Scalar, Vector, Tensor?
  - Discrete? Continuous?
  - Nominal, Ordinal, Interval, Ratio?
Data Example: Mummy (.vtk)

- **Mummy.vtk**
  - 128x128x128 regular grid (structured points)
  - Single scalar value at every point

```plaintext
# vtk DataFile Version 2.0
test volume
BINARY
DATASET STRUCTURED_POINTS
DIMENSIONS 128 128 128
ASPECT_RATIO 1.886 1.886 1.913
ORIGIN 0.0 0.0 0.0
POINT_DATA 2097152
SCALARS scalars unsigned_char
LOOKUP_TABLE default
```
Data Example: Simple unstructured grid (.vtu)

- Two Points: \{(1, 3, 5), (2, 4, 6)\}
  - Vector data: Force: \{(0, 2, 4), (1, 3, 5)\}
  - Scalar data: Radii: \{1, 3\}, Material: \{0, 1\}

```xml
<VTKFile byte_order="LittleEndian" type="UnstructuredGrid" version="0.1">
  <UnstructuredGrid>
    <Piece NumberOfCells="0" NumberOfPoints="2">
      <Points>
        <DataArray NumberOfComponents="3" format="ascii" type="Float32">
          1 3 5 2 4 6
        </DataArray>
      </Points>
      <Cells>
        <DataArray Name="connectivity" format="ascii" type="Int32">0</DataArray>
        <DataArray Name="offsets" format="ascii" type="Int32">0</DataArray>
        <DataArray Name="types" format="ascii" type="UInt8">1</DataArray>
      </Cells>
    </Piece>
  </UnstructuredGrid>
</VTKFile>
```
Data Example: Simple unstructured grid (.vtu)

- Two Points: \{(1, 3, 5), (2, 4, 6)\}
  - Vector data: Force: \{(0, 2, 4), (1, 3, 5)\}
  - Scalar data: Radii: \{1, 3\}, Material: \{0, 1\}

```xml
<PointData>
  <DataArray Name="Points" NumberOfComponents="3" format="ascii" type="Float32">
    1 3 5 2 4 6
  </DataArray>
  <DataArray Name="forces" NumberOfComponents="3" format="ascii" type="Float32">
    0 2 4 1 3 5
  </DataArray>
  <DataArray Name="radii" format="ascii" type="Float32">
    1 3
  </DataArray>
  <DataArray Name="material" format="ascii" type="UInt8">
    0 1
  </DataArray>
</PointData>
```
Visualization Operations

- Surface Shading (Pseudocolor)
- Isosurfacing (Contours)
- Volume Rendering
- Clipping Planes
- Streamlines
Surface Shading (Pseudocolor)

Given a scalar value at a point on the surface and a color map, find the corresponding color (and/or opacity) and apply it to the surface point.

Most common operation, often combined with other ops
Isosurfaces (Contours)

- Surface that represents points of constant value with a volume
- Plot the surface for a given scalar value.
- Good for showing known values of interest
- Good for sampling through a data range
Volume Rendering

Expresses how light travels through a volume
Color and opacity controlled by transfer function
Smoother transitions than isosurfaces
Volume Rendering

Transfer function (maps scalars to color, opacity) very important!
Clipping/Slicing planes

Extract a plane from the data to show features
Hide part of dataset to expose features
Particle Traces (Streamlines)

Given a vector field, extract a trace that follows that trajectory defined by the vector.

\[ P_{\text{new}} = P_{\text{current}} + V P \Delta t \]

Streamlines – trace in space
Pathlines – trace in time
Graphics Primitives

- Basic unit: Polygons, Colors, Textures, Opacity
  - Flat surface formed between points
  - This surface may have an associated color or texture, or opacity
- Complex surfaces composed of several polygons
Graphics Pipeline

- Modeling Transformation → Trivial Rejection → Illumination → Viewing Transformation
- Clipping → Projection → Rasterization → Display
Graphics pipeline in English

• Squeeze the world of your polygons into a normalized box.
• Rotate, translate, and scale them according to camera and model positions.
• Figure out what color they should be from lighting.
• Flatten them to a 2D world.
• Scan through the lines, turning them into pixels.
• (Along the way, cut out anything that won't be visible.)

Geometry, then Rasterization.
Graphics Pipeline

- Given polygons, show them on the screen.
  - Point 0: x, y, z
  - Point 1: x, y, z
  - Point 2: x, y, z
  - Color

- OpenGL does this for you

```c
glColor3f(0.0, 1.0, 0.0); // blue
glBegin(GL QUAD);
  glVertex2f(0.0, 0.0);
  glVertex2f(1.0, 0.0);
  glVertex2f(1.0, 1.0);
  glVertex2f(0.0, 1.0);
glEnd();
glTranslate(-1.5, 0.0, 0.0); // move object
```
From data to Insight

Data Representation → Visualization Operations → Graphics Primitives → Display

Iteration and Refinement