Overview and Introduction to Scientific Visualization

Texas Advanced Computing Center
The University of Texas at Austin

http://portal.longhorn.tacc.utexas.edu/training
Scientific Visualization

“The purpose of computing is insight not numbers.”

-- R. W. Hamming (1961)
Visualization Process Summary

• The primary goal of visualization is *insight*

• A picture is worth not just 1000 words, but potentially tera- or peta-bytes of data

• Larger datasets demand not just visualization, but advanced visualization resources and techniques

• Visualization system technology leverages lots of advancing technologies: GPUs, high-speed networks, web technology….

• Visualization software takes time to adapt
Today

• Introduction to Visualization
  – What is scientific data?

• Scientific Data Visualization With Paraview
• Information Visualization
• Remote and Collaborative Visualization
• Parallel Visualization For Very Large Data
Visualization Allows Us to “See” the Science

Raw Data

01001101011001
11001010010101
00101010100110
11101101011011
00110010111010

Visualization Application
Getting from Data to Insight

Data Source → Data Representation → Visualization Algorithms → Rendering → Display

... And using insight to get more insight
I Think Of Two Kinds Of Data For Visualization...

• Data for ‘Scientific Visualization’
  – F(spatial dimensions[, time]) -> attributes
  – E.g. Weather data:
    F(latitude, longitude, altitude) -> temperature,
    wind velocity, direction
    humidity...

• Data for ‘Information Visualization’
  – List of facts, which have multiple attributes
  – E.g. A list of movies:
    Title, year, director, length, gate, male/female leads..
‘Scientific Data’

longitude

altitude

latitude

longitude

\( z(m) \)

450

350

300

250

200

150

100

50

0
‘Info Data’

Gene Expression Data
Any Number of Packages Do Viz

R (GenomeGraphs)

Python (matplotlib)
Probably The Most Common Viz Tool...
Geosciences
‘SciVis’ Data

• Data mapped onto a *computational domain*
  Heat distribution in a block of material
  \[ F(x,y,z) \rightarrow \text{temperature} \]
  for \((x,y,z)\) a point in the block of material

• Multiple variables (or properties)
  Weather
  \[ F(x,y,z) \rightarrow \text{pressure, temperature, wind-velocity} \]
  for \((x,y,z)\) a point in the atmosphere
SciVis Data Dimensionality

• Domain is generally 1, 2, 3 or more dimensions
  – Directly interpreted geometrically (see heat distribution)
  – Indirectly interpreted geometrically

\[ F(\text{lat}, \text{lon}) \rightarrow \text{temperature} \]
\[ X = (\text{earth radius}) \cdot \cos(\text{lat}) \cdot \cos(\text{lon}) \]
\[ Y = (\text{earth radius}) \cdot \cos(\text{lat}) \cdot \sin(\text{lon}) \]
\[ Z = (\text{earth radius}) \cdot \sin(\text{lat}) \]

• Multiple variables (or properties)
  Weather \[ F(x, y, z) \rightarrow \text{pressure, temperature, wind-velocity} \] at (x, y, z) a point in the atmosphere
SciVis Data and Time

• *Time* may vary also
  – Heat *transfer* in a block of material
    
    \[ F(x,y,z,t) \rightarrow \text{temperature} \]
    
    for \((x,y,z)\) a point in the block of material and \(t\) a point in time
Higher Dimensional SciVis

- A 7D space curve representing the amount of oil stored at a given point in time
- A 7D function representing the total amount of oil stored at the point \((l_0, l_1, l_2, l_3, l_4, l_5, l_6)\)
- A contour surface at C represents all the ways C barrels of oil can be contained in 7 tanks

An Oil Tank Farm
SciVis Data: Discrete vs. Continuous Data

- **Discrete** data is known at a finite set of points in the domain
  \[ F(x,y,z) \rightarrow \text{temperature} \]
  for \((x,y,z)\) from a finite set of points in the domain, unknown otherwise

- **Continuous** data is known throughout the domain
  \[ F(x,y,z) \rightarrow \text{temperature} \]
  for all points \((x,y,z)\) in domain
The **Grid**

- **Points** in the domain can be *regular* - specified by origin, delta vectors and counts, or explicitly listed.

- For *interpolated* grids:
  - **Topology**: how the points “connected” (implicit or explicitly listed).
  - **Interpolation Model**: How data values at an arbitrary point are derived from nearby points.
Types of data at a point/cell

- Scalar
- Vector
- Tensor/matrix
- Labels, identifiers
- Other tuples
Example Points
Example Connectivity
Example Data Visualized
Example Varying in Time
Types of Input Data

Point – scattered values with no defined structure
Types of Input Data

Grid – regular structure, all voxels (cells) are the same size and shape
Types of Input Data

Curvilinear – regularly grided mesh shaping function applied
Types of Input Data

Unstructured grid – irregular mesh typically composed of tetrahedra, prisms, pyramids, or hexahedra.
Types of Input Data

Non-mesh connected point data (molecular)
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 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
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
Visualization Techniques

• Surface Rendering is an *indirect* geometry based technique

• Direct Volume Rendering is a technique for the visualization of 3D scalar data sets *without a conversion to surface representations*
Volume Rendering

Expresses how light travels through a volume
Color and opacity controlled by transfer function
Smoother transitions than isosurfaces
Example Volume Rendered
Visualization Resources

• Personal machines
  – Most accessible, least powerful

• Projection systems
  – Seamless image, high purchase and maintenance costs

• Tiled-LCD displays
  – Lowest per-pixel costs, bezels divide image

• Remote visualization
  – Access to high-performance system, latency can affect user experience
XSEDE Visualization Resources

• Longhorn (TACC)
  – 256 Nodes, 2048 Total Cores, 512 Total GPUs
  – 13.5 TB Aggregate Memory, QDR InfiniBand interconnect
  – Longhorn Visualization Portal
    • [https://portal.longhorn.tacc.utexas.edu/](https://portal.longhorn.tacc.utexas.edu/)
    • Visualization job submission and monitoring
    • Remote, interactive, web-based visualization
    • Guided visualization using EnVision

• Stampede (TACC)
  – 160 racks, 6400 nodes (16-core Xeon + 62-core Xeon Phi),
  – 128 NVIDIA K20 GPU Nodes
  – 205 TB Aggregate Memory
XSEDE Visualization Resources

• Nautilus (NICS)
  – SMP, 1024 Total Cores, 16 GPUs
    4 TB Global Shared Memory, SGI NUMAlink 5 interconnect
  – Production date: August 1, 2010

• TeraDRE Condor Pool (Purdue)
  – 1750 Nodes, 14000 Total Cores, 48 Nodes with GPUs
    28 TB Aggregate Memory, no interconnect
Visualization Challenges
Visualization Allows Us to “See” the Science

Raw Data
01001101011001
11001010010101
00101010100110
11101101011011
00110010111010

Geometric Primitives

Application

Render

Pixels
But what about large, distributed data?
Or distributed rendering?

01001101011001
11001010010101
00101010100110
11101101011011
00110010111010
Or distributed displays?
Or all three?
Visualization Scaling Challenges

- Moving data to the visualization machine
- Most applications built for shared memory machines, not distributed clusters
- Image resolution limits in some software cannot capture feature details
- Displays cannot show entire high-resolution images at their native resolution
Visualization scales with HPC

Large data produced by large simulations require large visualization machines and produce large visualization results

- Terabytes of Data
- AT LEAST Terabytes of Vis
- Gigapixel Images

- Resampling, Application, ...
- Resolution to Capture Feature Detail
## Moving Data

- **How long can you wait?**

<table>
<thead>
<tr>
<th>File Size</th>
<th>10 Gbps</th>
<th>54 Mbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 GB</td>
<td>1 sec</td>
<td>2.5 min</td>
</tr>
<tr>
<td>1 TB</td>
<td>~17 min</td>
<td>~43 hours</td>
</tr>
<tr>
<td>1 PB</td>
<td>~12 days</td>
<td>~5 years</td>
</tr>
</tbody>
</table>
Analyzing Data

• Visualization programs only beginning to efficiently handle ultrascale data
  – 650 GB dataset -> 3 TB memory footprint
  – Allocate HPC nodes for RAM not cores
  – N-1 idle processors per node!

• Stability across many distributed nodes
  – Rendering clusters typically number N <= 64
  – Data must be dividable onto N cores

*Remember this when resampling!*
Displaying Data

Dell 30” flat-panel LCD

4 Megapixel display

2560 x 1600 resolution
Displaying Data

Stallion – world’s 2nd highest-resolution tiled display

328 Megapixels
40960 x 8000 pixel resolution

Dell 30” LCD
What’s the solution?

- It’s a Fan!
- It’s a Wall!
- It’s a Spear!
- It’s a Snake!
- It’s a Rope!
- It’s a Tree!
Solution by Partial Sums

- Moving data – integrate vis machine into simulation machine. **Move the machine to data!**
  - Ranger + Spur: shared file system and interconnect

- Analyzing data – create larger vis machines and develop more efficient vis apps
  - Smaller memory footprint
  - More stable across many distributed nodes

Until then, **the simulation machine is the vis machine!**
Solution by Partial Sums

• Imaging data – focus vis effort on interesting features parallelize image creation
  – Feature detection to determine visualization targets but can miss “unknown unknowns”
  – Distribute image rendering across cluster

• Displaying data – high resolution displays multi-resolution image navigation
  – Large displays need large spaces
  – Physical navigation of display provides better insights
Old Model
(No Remote Capability)
New Model
Remote Capability

- HPC System
- Data Archive
- Large-Scale Visualization Resource
- Display
- Pixels
- Mouse
- Remote Site
- Wide-Area Network
- Local Site
Using GPUs

• More than for just rendering!
  – HPC applications and Visualization algorithms

• Parallelism – kernel should be highly SIMD/SIMT
  – Switching kernels is expensive!
  – Fermi *hardware* supports multiple kernel execution

• Control Flow – avoid conditionals in kernels
  – Implemented with predication, harms utilization

• Job size – high workload per thread + many threads
  – amortize thread initialization and memory transfer costs
  – GPU is a throughput machine, must keep it busy!

• Memory footprint – task must decompose well
  – local store per GPU core is low (16 KB on Longhorn)
  – card-local RAM is limited (4GB on Longhorn)
  – access to system RAM is slow (treat like disk access)
Summary

• Challenges at every stage of visualization when operating on large data

• Partial solutions exist, though not integrated

• Problem sizes continue to grow at every stage

• Vis software community must keep pace with hardware innovations