CSE 332

INTRODUCTION TO VISUALIZATION

SCIENTIFIC VISUALIZATION

KLAUS MUELLER

COMPUTER SCIENCE DEPARTMENT
STONY BROOK UNIVERSITY
Rendering Volumes as Surfaces

- Objects are explicitly defined by a surface or boundary representation (explicit inside vs outside)
- This boundary representation can be given by:
  - a mesh of polygons:
    200 polys
    1,000 polys
    15,000 polys
  - a mesh of spline patches:
    an “empty” foot
The Marching Cubes Polygonization Algorithm

- The *Marching Cubes (MC)* algorithm converts a volume into a polygonal model
  - this model *approximates* a chosen iso-surface by a mesh of polygons
  - the polygonal model can then be rendered, for example, using a fast z-buffer algorithm
  - if another iso-surface is desired, then MC has to be run again

- Steps:
  - imagine all voxels above the iso-value are set to 1, all others are set to 0
  - the goal is to find a polygonal surface that includes all 1-voxels and excludes all 0-voxels
  - look at one volume cell (a cube) at a time → hence the term *Marching Cubes*
  - here are 2 of 256 possible configurations:

![Diagram showing the Marching Cubes algorithm](image)

only 1 voxel > iso-value  the polygon that separates inside from outside  7 voxels > iso-value
the reverse case:  the same polygon results
Marching Cubes (2)

- One can identify 15 base cases
  - Use symmetry and reverses to get the other 241 cases

The 15 Cube Combinations

- The exact position of the polygon vertex on a cube edge is found by linear interpolation:

  \[ iso = v_1 \cdot (1-u) + v_2 \cdot u \quad \rightarrow \quad u = \frac{v_1 - iso}{v_1 - v_2} \]

- Now interpolate the vertex color by:

  \[ c_1 = uc_2 + (1-u)c_1 \]

- Interpolate the vertex normal by:

  \[ n_1 = ug_2 + (1-u)g_1 \]

  (the \( g_1 \) and \( g_2 \) are the gradient vectors at \( v_1 \) and \( v_2 \) obtained by central differencing)
Real-Time Marching Cubes
What is It?

10 petaFLOPS  Titan supercomputer (released in 2012)
  - $10^{15}$ floating point ops per second (1 PetaFlop)
18,688 AMD Opteron 6274 16-core CPUs
18,688 Nvidia Tesla K20X GPUs
Summit supercomputer (2018, #1 worldwide, Oak Ridge Nat’l Lab)

- 200 petaFLOPS (2x the top speed of TaihuLight, previous #1)
- 4,608 compute servers (each two 22-core IBM Power9 processors and six NVidia Tesla V100 GPIUs)
Frontier

- World’s first exascale supercomputer, at Oak Ridge Leadership Computing Facility in Tennessee,
- $10^{18}$ flops (1.102 quintillion operations per second)
- 9,472 AMD Epyc 7A53s "Trento" 64 core 2 GHz CPUs (606,208 cores)
- 37,888 Radeon Instinct MI250X GPUs (8,335,360 cores).
Compute, compute, compute

Examples:
- S3D, a project that models the molecular physics of combustion, aims to improve the efficiency of diesel and biofuel engines.
- Denovo simulates nuclear reactions with the aim of improving the efficiency and reducing the waste of nuclear reactors.
- WL-LSMS simulates the interactions between electrons and atoms in magnetic materials at temperatures other than absolute zero.
- Bonsai is simulating the Milky Way Galaxy on a star by star basis, with 200 billion stars.
- Non-Equilibrium Radiation Diffusion (NRDF) plots non-charged particles through supernovae with potential applications in laser fusion, fluid dynamics, medical imaging, nuclear reactors, energy storage and combustion.
Numbers, lots of them
  - Titan’s I/O subsystem is capable of pushing around 240 GB/s of data
  - that’s a lot to visualize

Example: a visualization of the Q Continuum simulation for cosmology
MORE EXAMPLES

Nuclear, Quantum, and Molecular Modeling

Structures, Fluids and Fields

Advanced Imaging and Data Management
Surface Rendering with vTK
(The Visualization Toolkit)

Volume Rendering
Where to Visualize All This?
Display Wall
The Stony Brook Immersive Cabin
CAVE
Microtomography
(BNL, soil sample)
The Stony Brook Immersive Cabin

Projector based system

- 5 walls, 12’ × 12’ footprint, 8’ tall
- difficult to scale up to Giga-pixel range
Can We Get Bigger?

(yes we can)
The Stony Brook University Reality Deck
Visualization

- 30’×40’×11’ environment
- 416 UQXGA LCD Displays
  - 2,560×1,440 resolution over 50’-100’ DisplayPort cables
  - fast response time, wide viewing angles, good dynamic range
- 20-node GPU cluster, each node equipped with:
  - 2× Six-core CPUs, 48 GB Ram
  - 4× AMD FirePro V9800 with 4GB Ram and 6 DisplayPort outputs each
  - AMD S400 hardware video synchronization card
  - 40Gb Infiniband adapter
  - 1TB storage
- In total:
  - 1,533,542,400 pixels (1.5 Gigapixel) over 6 miles of DisplayPort cables
  - 240 CPU cores: 2.3 TFLOPs peak performance, 20 TB distributed memory
  - 80 GPUs: 220 TFLOPs peak performance, 320 GB distributed memory
Automatic Door

3 × 5 section of displays
Visually indistinguishable from rest of display
- allows for a fully enclosed visualization environment
Reality Deck Tracking System

24-camera infrared optical system from OptiTrack
24.4 channel professional-grade system
Positional audio with real-time ambisonics
- using the Rapture3D OpenAL driver
User can make visual queries at an instant
- walk up to obtain more detail
- just like in real life – hence the Realty Deck
- 20/20 visual acuity at 1.5’-2’ away
Dubai dataset

- 45 Gigapixels, 180° field of view
Shuttle Radar Topography Mission dataset
Terrain Modeling
3D Relief Map
Sea level simulation
Say, you want to simulate the airflow around an airplane wing

- where is the flow most interesting?
- right, close to the surface
SIMULATION LATTICE

Make the simulation lattice densest along the surface

Regular $\rightarrow$ irregular grids
Structured grid
- more or less a bent regular grid

Unstructured grid
- collection of vertices, edges, faces and cells whose connectivity information must be explicitly stored
The Bluntfin Dataset

Mapping flow strength to color

Rendering by cell traversal
- go from cell to cell
- composite colors and opacities
Also called vector field visualization
Perform an integration through the vector field

- color maps to temperature
Connect two streamlines

- the center streamline gives direction, the other two indicate the twisting
Connect three or more streamlines
Sweep a line segment through the vector field
Smoke is injected into the flow field and compresses/expands due to the vector field.
Seek to give a more global view of the vector field

Hedgehogs
- oriented lines spread over the volume, indicating the orientation and magnitude of the flow
- do not show directional information

Glyphs, arrows
- icons that show directions, but tend to clutter the display
**Line Integral Convolution (LIC)**

- **Input:**
  - a 2D vector field
  - an image that will be “smeared” according to the stream lines described by the vector field

- **Output Image:** line-integrated white noise image

- **Stream Line**
  - For each output pixel \((x, y)\)
  - Follow the stream line forward for some distance \(\Delta s\)
  - Multiply each pixel value by a 1D filter kernel and add
  - Follow the stream line backward for some distance \(\Delta s\)
  - Multiply each pixel value by a 1D filter kernel and add

- **Salt+Pepper Noise**

- **Input Vector Field**

- **Filter Aligned with the Stream Line**
**Line Integral Convolution (LIC)**

- A flower image with different vector fields.
- A simple motion vector field over the hand.
- Using vector magnitude to determine $\Delta s$.
- Mapping LIC onto an object surface.
Textured Splats

- Embed flow field vector icons into a splat
  - this enables smooth blending of neighboring icons

- Create a table of texture splats with varying icon distribution (to prevent regular patterns)

- For a given location, select a random splat and rotate corresponding to the flow field direction

- Since the flow field is 3D, the component of the vectors that is parallel to the screen varies

- Need to provide splats that accommodate for vector foreshortening when the flow heads towards us

- Animated display
  - store a splat table with vector icons that are cyclically shifted from left to right
  - cycle through this table when picking splats to update the animated display
Textured Splats Examples
POPULAR SOFTWARE & LIBRARIES

VTK
- The Visualization Toolkit library
- developed by Kitware

Paraview
- built on top of VTK
- open-source
- multi-platform
- developed by Sandia & Los Alamos National Labs

VisIt
- open source
- developed by Lawrence Livermore National Lab