Volume Rendering Primer / Intro to VisIt

Hank Childs, University of Oregon
Build YOUR IT Career

FREE FOOD & DRAWING FOR A JAMBOX*

*Please bring your resumé to enter the drawing.

JOIN US AT OUR
Info Seminar

COMPUTER SCIENCE, ENGINEERING & MATH STUDENTS
Explore your career opportunities at Fast Enterprises

Wednesday, November 11, 2015
4:00 - 4:50
Deschutes - Room 220
Announcements

• Volume rendering lectures: preview today, more next week
• VisIt lectures: today, Friday
• Project proposal (whoops):
  – WEBSITE SAID NOV 15th
  – WEBSITE SAID IT WOULD BE GRADED
  – 510: due Weds Nov 11th
  – 410: due Tues Nov 17th
• Class canceled Mon Nov 16th
• JB^2?
Project #7A

• You write VTK program.
  – I expect that your answers will be partially derived from previous VTK programs
  – Python OK

• 4 renderers in 1 window
CIS 410/510: Project #7A
Due November 19th, 2015
(which means submitted by 6am on November 20th, 2015)
Worth 11% of your grade

Assignment:
1) Download file proj7.vtk.
2) You will write a VTK program. The program should have one render window & 4 renderers, in a 2x2 layout.
   a. Renderer #1 should cover viewpoint X:0->0.5, Y: 0->0.5. It should contain an isosurface of the variable hardyglobal with isovalue 2.5 and 5.0. You can color this however you like.
   b. Renderer #2 should cover viewport X:0->0.5, Y:0.5->1.0. It should contain two slices of the variable hardyglobal. It should use the rainbow colormap, which is the default colormap.
   c. Renderer #3 should cover viewport X:0.5->1.0, Y:0->0.5. It should contain hedgehog glyphs of the variable grad. You can choose the density and colors.
   d. Renderer #4 should cover viewport X:0.5->1.0, Y:0.5->1.0. It should contain streamlines of the variable grad. Use RK4 for integration. The seed locations should be in a line from (-9, 0, 0) to (9, 0, 0). There should be 19 total seeds, meaning they should cover each integer.
3) Upload your source code and a screenshot of it working to Canvas.
Project 7B (510 only)

CIS 410/510: Project #7B
Due November 19th, 2015
(which means submitted by 6am on November 20th, 2015)
Worth 4 points toward your grade

Assignment:
1) Download file proj7B.ascii.
   a. Proj7B.ascii contain a scalar field for a data set that is 50x50x50.
   b. There is one entry per line, for a total of 125,000 lines
   c. We have been dealing with data where X varies most quickly, and Z
      varies most slowly. For this file, Z varies most quickly, and X varies
      most slowly.
2) You will write a VTK program. Where 7A was about using VTK filters, this
   project is about manipulating VTK data structures.
   a. Load data from file proj7B.ascii into a VTK data set.
   b. Compare the vtkDataArray (from 2a) of scalar field in proj7B.ascii to
      that the scalar field “hardyglobal” from proj7.vtk. They should have 3
      spots where they differ. Tell me what the three indices are.
   c. In project 7A, you made hedgehogs and thinned them. I want you to
      modify your thinning procedure for this project. Write a function that
      takes a vtkDataSet corresponding to the data in proj7.vtk and outputs
      a vtkPolyData. The vtkPolyData should contain only the vectors that
      have magnitude greater than or equal to 1.
3) Upload your source code and a screenshot of it working to Canvas.

Note: solutions are expected to be within the context of the VTK library. If you were
to solve 2B by writing code that does not involve VTK, then you won’t get credit.
(This is designed to teach you VTK!)
Final Project

• Two general flavors:
  – Here is some data I find interesting and I want to visualize
    • Data source:
      – Find yourself
      – I have some
    • How to visualize:
      – Use VTK
      – Use VisIt
      – Other
  – Here is a visualization algorithm I want to implement
Aneurysm data set

Outline

- **Description of the data**
- **Static Analysis**
  - Pressure display
  - Velocity glyphs
  - Picking
  - Streamlines
  - Flux thru a surface
- **Transient Analysis**
  - Multi-view of Transient data
  - Pressure Curve vs. Time
  - Average over one heart beat
  - Pathlines
  - Expressions
  - Python code
  - Making it pretty
Water Dam data set
Explore Data

- **Aneurysm**

- **Water flow**
Volume Rendering Primer

• Goal: get you enough information to make a decision whether to do self-defined project or pre-defined project

• Volume rendering project:
  – implement
  – make some pictures
  – do a performance study
X-rays
Volume rendering

• Important visualization technique for 3D data
• Use combination of color and transparency to see entire 3D data set at one time.
Volume rendering overview

Camera

Pixels on the screen

3D data
Volume rendering overview

Ray casting game plan:
For every pixel on the screen,
  Find ray for that pixel
  Intersect volume with ray
  Calculate color from intersection
  Assign color to pixel
VisIt Overview

• PPT material
• Live demo
Tutorial materials

VisIt is an open source, turnkey application for data analysis and visualization of mesh-based data.

- Production end-user tool supporting scientific and engineering applications.
- Provides an infrastructure for parallel post-processing that scales from desktops to massive HPC clusters.
- Source released under a BSD style license.
VisIt supports a wide range of use cases.

- **Data Exploration**
- **Comparative Analysis**
- **Quantitative Analysis**
- **Visual Debugging**
- **Presentation Graphics**

Project Introduction
VisIt is a vibrant project with many participants.

- The VisIt project started in 2000 to support LLNL’s large scale ASC physics codes.
- The project grew beyond LLNL and ASC with research and development from DOE SciDAC and other efforts.
- VisIt is now supported by multiple organizations:
  - LLNL, LBNL, ORNL, UC Davis, Univ of Utah, Intelligent Light, …
- Over 75 person years of effort, 1.5+ million lines of code.
### Machine Specifications

<table>
<thead>
<tr>
<th>Machine</th>
<th>Architecture</th>
<th>Problem Size</th>
<th># of Cores</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph</td>
<td>X86_64</td>
<td>$20,001^3$ (8 T cells)</td>
<td>12K</td>
</tr>
<tr>
<td>Dawn</td>
<td>BG/P</td>
<td>$15,871^3$ (4 T cells)</td>
<td>64K</td>
</tr>
<tr>
<td>Franklin</td>
<td>Cray XT4</td>
<td>$12,596^3$ (2 T cells)</td>
<td>32K</td>
</tr>
<tr>
<td>JaguarPF</td>
<td>Cray XT5</td>
<td>$12,596^3$ (2 T cells)</td>
<td>32K</td>
</tr>
<tr>
<td>Juno</td>
<td>X86_64</td>
<td>$10,000^3$ (1 T cells)</td>
<td>16K</td>
</tr>
<tr>
<td>Franklin</td>
<td>Cray XT4</td>
<td>$10,000^3$ (1 T cells)</td>
<td>16K</td>
</tr>
<tr>
<td>Ranger</td>
<td>Sun</td>
<td>$10,000^3$ (1 T cells)</td>
<td>16K</td>
</tr>
<tr>
<td>Purple</td>
<td>IBM P5</td>
<td>$8,000^3$ (0.5 T cells)</td>
<td>8K</td>
</tr>
</tbody>
</table>

**Scaling Studies of Isosurface Extraction and Volume Rendering (2009)**

VisIt is also used daily by domain scientists.
The VisIt team focuses on making a robust, usable product for end users.

- **Regular releases (~ 6 / year)**
  - Executables for all major platforms
  - End-to-end build process script `build_visit`

- **User Support and Training**
  - visitusers.org, wiki for users and developers
  - Email lists: visit-users, visit-developers
  - Beginner and advanced tutorials
  - VisIt class with detailed exercises

- **Documentation**
  - “Getting data into VisIt” manual
  - Python interface manual
  - Users reference manual

*Slides from the VisIt class*
VisIt’s core abstractions

- **Databases**: How datasets are read
- **Plots**: How you render data
- **Operators**: How you manipulate data
- **Expressions**: Mechanism for generating derived quantities
- **Queries**: How to access quantitative information
Examples of VisIt Pipelines

- Databases: how you read data
- Plots: how you render data
- Operators: how you transform/manipulate data
- Expressions: how you create new fields
- Queries: how you pull out quantitative information

Open a database, which reads from a file (example: open file1.hdf5)

Make a plot of a variable in the database (example: Volume plot)
Examples of VisIt Pipelines

- **Databases**: how you read data
- **Plots**: how you render data
- **Operators**: how you transform/manipulate data
- **Expressions**: how you create new fields
- **Queries**: how you pull out quantitative information

**Diagram**

- **Database**: Open a database, which reads from a file (example: open file1.hdf5)
- **Operator**: Apply an operator to transform the data (example: Slice operator)
- **Plot**: Plot a variable in the database (example: Pseudocolor plot)
Examples of VisIt Pipelines

- Databases: how you read data
  
- Plots: how you render data

- Operators: how you transform/manipulate data

- Expressions: how you create new fields

- Queries: how you pull out quantitative information

- Database

  Open a database, which reads from a file (example: open file1.hdf5)

- Operator 1

  Apply an operator to transform the data (example: Slice operator)

- Operator 2

  Apply a second operator to transform the data (example: Elevate operator)

- Plot

  Plot a variable in the database (example: Pseudocolor plot)
Examples of VisIt Pipelines

- Databases: how you read data
- Plots: how you render data
- Operators: how you transform/manipulate data
- Expressions: how you create new fields
- Queries: how you pull out quantitative information

Database: Open a database, which reads from a file (example: open file1.hdf5)

Expression: Create derived quantities from fields in the file (example: magnitude(velocity))

Plot: Plot the expression variable (example: Pseudocolor plot)
Examples of VisIt Pipelines

- Databases: how you read data
- Plots: how you render data
- Operators: how you transform/manipulate data
- Expressions: how you create new fields
- Queries: how you pull out quantitative information

- **Database**: Open a database, which reads from a file (example: open file1.hdf5)
- **Plot**: Plot a field from the file (example: density + Pseudocolor plot)
- **Query**: Extract quantitative information (example: integrate density to find mass)
Examples of VisIt Pipelines

- **Databases**: how you read data
  - Open a database, which reads from a file (example: open file1.hdf5)

- **Plots**: how you render data
  - Create derived quantities from fields in the file (example: magnitude(velocity))

- **Operators**: how you transform/manipulate data
  - Apply an operator to transform the data (example: Slice operator)

- **Expressions**: how you create new fields
  - Apply a second operator to transform the data (example: Elevate operator)

- **Queries**: how you pull out quantitative information
  - Plot a field (example: speed + Pseudocolor plot)
  - Extract quantitative information (example: maximum speed over cross-section)