Visualization Toolkit: Faster, Better, Open Scientific Rendering and Compute

GTC, San Jose, CA
March, 2015

Marcus D. Hanwell
Robert Maynard
Accelerating Visualization with Partnerships

• NVIDIA and Kitware collaborate to bring advances in scientific visualization

• Collaboration focuses
  – In-site visualization
  – Advanced rendering

• Improved use of NVIDIA GPUs
Kitware, Inc.

- Founded in 1998 by five former GE Research employees
- 98 current employees; 34 with PhDs
- Privately held, profitable from creation, no debt
- Offices
  - Clifton Park, NY
  - Carrboro, NC
  - Santa Fe, NM
  - Lyon, France

- 2011 Small Business Administration’s Tibbetts Award
- HPCWire Readers and Editor’s Choice
- Inc’s 5000 List since 2008
Kitware’s customers & collaborators

Over 75 **academic** institutions including…
- Harvard
- Massachusetts Institute of Technology
- University of California, Berkeley
- Stanford University
- California Institute of Technology
- Imperial College London
- Johns Hopkins University
- Cornell University
- Columbia University
- Robarts Research Institute
- University of Pennsylvania
- Rensselaer Polytechnic Institute
- University of Utah
- University of North Carolina

Over 50 **government** agencies and labs including…
- National Institutes of Health (NIH)
- National Science Foundation (NSF)
- National Library of Medicine (NLM)
- Department of Defense (DOD)
- Department of Energy (DOE)
- Defense Advanced Research Projects Agency (DARPA)
- Army Research Lab (ARL)
- Air Force Research Lab (AFRL)
- Sandia (SNL)
- Los Alamos National Labs (LANL)
- Argonne (ANL)
- Oak Ridge (ORNL)
- Lawrence Livermore (LLNL)

Over 100 **commercial** companies in fields including…
- Automotive
- Aircraft
- Defense
- Energy technology
- Environmental sciences
- Finance
- Industrial inspection
- Oil & gas
- Pharmaceuticals
- Publishing
- 3D Mapping
- Medical devices
- Security
- Simulation
Kitware: Core Technologies
Business Model: Open Source

• Open-source Software
  – Normally BSD-licensed
  – Collaboration platforms
• Collaborative Research and Development
• Technology Integration
• Services, support, and consulting
• Training and webinars
Overview of Software Process

• Openly developed, reusable frameworks
  – Open-source frameworks
  – Developed openly
  – Cross-platform compatibility
  – Tested and verified
  – Contribution model
  – Supported by Kitware experts

• Liberally-licensed to facilitate research
The Visualization Toolkit

• Founded in 1993 as example code for “The Visualization Textbook”.

• Used in many projects developed all over the world:
  – ParaView, VisIt
  – Osirix, 3D Slicer
  – Mayavi, MOOSE
Going From Data to Visualization
VTK Visualizations

HPC Visualization

Large Displays and Virtual Reality

Interactive Medical Application and Visualization
VTK Architecture

• Hybrid approach
  – Compiled C++ core (faster algorithms)
  – Interpreted applications (rapid development)
  – Interpreted layer generated automatically
The Visualization Pipeline

• A sequence of algorithms that operate on data objects to generate geometry
VTK Organization

• Libraries with public APIs
• Cross-platform, open-source, for reuse
• Implementation modules use factories
  – Rendering API uses OpenGL backend
  – Core rendering does not link to/use OpenGL
Basic Library Hierarchy

vtkCommonCore

vtkRenderingCore  vtkFreeType

OpenGL  OpenGL2  OpenGL  OpenGL2
Legacy Rendering

• Based on OpenGL 1.1 APIs
  – Optionally uses some extensions
• Heavy use of display lists for interaction
• A “Painter” API to enable custom rendering
  – Virtual functions, switches, …
    • In tight loops for all vertices, normals, colors, etc
Polygonal Rendering Rewrite

- New minimum OpenGL version
  - OpenGL 2.1, OpenGL ES 2.0
- Rewrite to use minimal common subset
- Major overhaul of the rendering code
  - Use VBOs, VAOs, shaders, “new” OpenGL
- Retain same high level API
Volume Rendering Rewrite

• Improve portability of GPU code
  – Works well on Linux, Mac, and Windows
  – Uses less extensions, more core GL 2.1+

• Refactored to compute more in shaders

• Replicates important features

• Easier to develop new techniques
Removing Old Calls

• Not using matrix stacks
• GLSL, using modern approaches
• Optional extensions detected at runtime
• Not a single glVertex call, highly batched
• Some data structures need further work
  – vtkPolyData needs packed triangles
Performance Improvements

- In many cases now GPU bound
  - Previously large systems CPU bound
- Large polygonal models >100x faster!
- Much more portable depth peeling
- Reduced memory footprint significantly
- Initial render times reduced
Performance: Old vs New

• Looking at static scenes
  – Time to first render
  – Average time of rotated subsequent renders

• Legacy rendering hits maximum size
  – Memory errors/limits
  – Only possible to compare smaller geometries
Benchmarking Tools (Polygonal)

• Added some new benchmarking tools
• Aim to provide systematic comparison
Time For First Frame (K6000)

<table>
<thead>
<tr>
<th>Triangles</th>
<th>Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 million</td>
<td>0</td>
</tr>
<tr>
<td>5 million</td>
<td>2</td>
</tr>
<tr>
<td>20 million</td>
<td>14</td>
</tr>
<tr>
<td>30 million</td>
<td>15</td>
</tr>
</tbody>
</table>

- Legacy
- Rewrite
Time for Subsequent Frames (K6000)

<table>
<thead>
<tr>
<th>Triangles</th>
<th>Legacy</th>
<th>Rewrite</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 million</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 million</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>20 million</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>30 million</td>
<td>2.5</td>
<td></td>
</tr>
</tbody>
</table>
Rendering Speeds

• Two orders of magnitude faster!
• Legacy rendering maxes out at 30 million
  – Not possible to compare above this
• Measured on a modern Linux system
  – Same on Windows, and Mac
• Memory footprint about half for triangles
Comparison of Cards (Rewrite)

![Graph showing the comparison of cards with different numbers of triangles per second and the number of triangles.]

- **K2200**
- **K5200**
- **K6000**
Benchmarking Tools (Volume)

• Uses same framework as polygonal
• Volumes of increasing size
Time For First Frame (K40c)

- **Time (s)**
- **Voxels**

### Graph Details:
- **X-axis**: Voxels (10 million, 50 million, 100 million, 500 million, 1000 million)
- **Y-axis**: Time (s) (0 to 25)
- **Bars**:
  - Blue: Legacy
  - Red: Rewrite

#### Key Points:
- The time for the first frame increases significantly with the number of voxels.
- The Legacy version consistently takes more time compared to the Rewrite version for all voxel counts shown.
- At 100 million voxels, both versions show similar time.
- The 1000 million voxel count is notably higher for both versions, indicating a more computationally intensive task.
Mobile/Embedded

- New rendering can target ES 2.0+
- Some testing on Android and iOS
- Largely shared code with desktop code
- Simple multitouch interaction support
Custom Rendering

- Shaders can be overridden in mappers
- VBOs/IBOs created by reusable helpers
- Override the vtkMapper class
- Several examples of different rendering
  - Glyphing, impostors, composite data
  - Offer a reasonable starting point
Porting/Using New Rendering

• Many applications just change backend
  – VTK_RENDERING_BACKEND=OpenGL2
  – Compile time option, with possible link change
  – vtkRenderingOpenGL ->
    vtkRendering${VTK_RENDERING_BACKEND}

• Custom OpenGL will need to be ported
VTK-m Project Goals

• A single place for the visualization community to collaborate, contribute, and leverage massively threaded algorithms.

• Reduce the challenges of writing highly concurrent algorithms by using data parallel algorithms.
VTK-m Project Goals

• Make it easier for simulation codes to take advantage of these parallel visualization and analysis tasks on a wide range of current and next-generation hardware.
VTK-m Architecture

• Combines strengths of multiple projects:
  – EAVL, Oak Ridge National Laboratory
  – DAX, Sandia National Laboratory
  – PISTON, Los Alamos National Laboratory
VTK-m Arbitrary Composition

- VTK-m allows clients to access different memory layouts through the Array Handle and Dynamic Array Handle.
  - Allows for efficient in-situ integration
  - Allows for reduced data transfer
VTK-m Arbitrary Composition

- VTK-m allows clients to construct data sets from cell and point arrangements that exactly match their original data
  - In effect, this allows for hybrid and novel mesh types

<table>
<thead>
<tr>
<th>Cells</th>
<th>Coordinates</th>
<th>Point Arrangement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structured</td>
<td>Strided</td>
<td>Explicit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Logical</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Implicit</td>
</tr>
<tr>
<td>Separated</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Unstructured</td>
<td>Strided</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Separated</td>
<td></td>
</tr>
</tbody>
</table>
Functor Mapping
Applied to Topologies

[Baker, et al. 2010]
Functor Mapping
Applied to Topologies

[Baker, et al. 2010]
What We Have So Far

• Features
  – Core Types
  – Statically and Dynamically Typed Arrays
  – Device Interface (Serial, Cuda, TBB under development)
  – Basic Worklet and Dispatcher
What We Have So Far

• Compiles with
  – gcc (4.8+), clang, msvc (2010+), icc, and pgi

• User Guide
• Ready for larger collaboration
2 x Intel Xeon CPU E5-2620 v3 @ 2.40GHz + NVIDIA Tesla K40c
Data: $1024^3$ (floats)

Marching Cubes

- VTK-m Cuda [No Transfer]
- VTK-m Cuda
- VTK-m Serial
- VTK Serial

0.524
1.514
17.28
30.2

Marching Cubes performance comparison.
2 x Intel Xeon CPU E5-2620 v3 @ 2.40GHz + NVIDIA Tesla K40c
Data: $1024^3$ (floats)

![Graph showing time in seconds for different triangle counts with and without transfer.]
Future Directions

• Make custom rendering easier
• Improved support for mobile
• Improved support for multitouch
• Extend approaches to the web
• Optionally use new features (OpenGL 4.4)
Coprocessing/In-situ

• Use of VTK and VTK-m
  – Process data in place using VTK-m
  – Visualize and analyze using VTK

• Bringing highly parallelized visualization and analytics in science to all

• Create bridges between VTK and VTK-m
Thank You!

Marcus D. Hanwell
• mhanwell@kitware.com
• @mhanwell
• +MarcusHanwell

Robert Maynard
• robert.maynard@kitware.com
• @robertjmaynard

Checkout out Kitware @ www.kitware.com and VTK @ www.vtk.org

Please complete the Presenter Evaluation sent to you by email or through the GTC Mobile App. Your feedback is important!