3D Visualization with ParaView

Adam Lyon, SCD/Muon g-2
Computing Techniques Seminar November 3, 2015
Why we’re here

- Why we’re (g-2, JimK & I) are looking at ParaView?
- What makes ParaView unique?
- The niche it may fill
- Questions you should ask yourself before considering
- Just DEMO it already (OK - will go through the above quickly)!

Why do I have anything to say on this?
I’m a Scientist II on Muon g-2 & Head of SCD SSA Quadrant;
In 1998 I arrived at Cornell as a CLEO/U of Rochester Postdoc to find everyone only used black and white monitors (ick!). Wrote a python based event display in order to get a color one (wouldn’t you like to see that track in purple?)
The problems

Muon g-2 was facing several problems where visualization could be very helpful

- **Validation of our Geant geometry**
  We had hints of incorrect positions in the geometry

- **Debugging of Magnetic fields in Geant**
  We have some complicated and time-varying fields (kicker magnets) — needed verifying

- **Debugging Tracking (the usual stuff)**
  Comparison of reconstructed hits & tracks to truth hits & trajectories — needs to run post-grant
No easy solutions

- Validation of our Geant geometry
  We seemed to reach the performance limit of Geant 3D viewers
  Geant/OpenGL hard to control, HepRAp & JAS3/Wired4 painful,
  Didn’t try Geant/Qt since wouldn’t help with track debugging

- Debugging of magnetic fields in Geant
  Not clear how to visualize the fields on top of geometry with
  Geant/Root tools

- Debugging Tracking
  Need to superimpose reconstruction & Geant views
  for comparison (view Geant info without running Geant)
  Root/Eve could not faithfully display our Geant-generated GDML
  Not many other options here

Bigger Problem: The SCD no longer has expertise in visualization. All experiments writing their own viz apps - no one solution
Simple requirements

This visualization niche: Aid for validation, debugging, development

Requirements (or maybe expectations, perhaps realized after the fact):

Fast image manipulation (rotate, pan, zoom)

Ability to hide objects (e.g. certain detector components)

Ability to ingest data from simple text or CSV files (for debugging) [e.g. how do you debug stuck Geant when you can’t get the regular output?]

Ability to superimpose data from different sources [e.g. see 3D Model from CAD on top of Geant geometry]

I didn’t want to have to write a lot of code, nor generate code that would be hard to maintain (e.g. I don’t want to change Geant)

Ability to make pretty pictures/movies doesn’t hurt (but not a specific requirement)
ParaView

Jim Kowalkowski had mentioned and briefly played with ParaView – thinking about it to fill visualization needs of art users

So I started looking at it and liked what I saw

Written with 3D performance as top priority

Based on very mature VTK toolkit

Note the logos ————>

Kitware
Sandia National Laboratories
Los Alamos National Laboratory
ASC
ARL
Features of ParaView

Applies principles from the visualization & supercomputing communities

Extremely responsive scene manipulation (rotate, pan, zoom)
  Image quality is degraded (Level of Detail) during manipulation for speed. Configurable

Many tools for visualization of scalar and vector fields
  Arrows, streamlines, heat maps, volume rendering (opacity)

Easy slicing, cutting, thresholds, “warping”

Can make 2D plots of aspects of the 3D data

Animation

Pipeline metaphor – sources, filters, and sinks

Very easy to overlay data using multiple sources (easy transformation if necessary - cm to mm)

Can write your own sources and filters in Python with built in numpy & matplotlib

Deep automation with Python

Less domain specific (certainly not tuned to our domain)
Who uses ParaView?

Mesh based simulations, especially computational fluid dynamics (CFD) @ Supercomputer Centers

See [http://www.paraview.org/gallery/](http://www.paraview.org/gallery/)

Chen, et. al., [http://dx.doi.org/10.1090/noti1236](http://dx.doi.org/10.1090/noti1236)
ParaView Support

HEP simulations are not usually mesh-based:
- We’re pushing on a tool that’s not our domain
- But this tool seems to be flexible enough to work in many cases
- There’s a lot of interest in adding HEP users

I’ve found the support from Kitware to be excellent and they are interested in adding HEP science to their portfolio – Enabling Science is important.

Every Supercomputing center has a visualization group, and they do ParaView (+ other applications [VisIt])

The Argonne ALCF Visualization group (Joe Insley) is interested – and they have a big visualization system.
How do we get our data into ParaView?

ParaView speaks VTK!

Many readers for file formats we don’t use

BUT, it’s not hard to make VTK objects (in C++ & Python) and integrate directly in ParaView

– It’s a pain to translate (but not hard after learning)

+ Make better use of ParaView/VTK’s features when you speak its language (my opinion)
Readers

- ParaView Data (.pvd)
- VTK (.vt, .vtu, .vti, .vts, .vtr)
- VTK Legacy (.vtk)
- VTK Multi Block (.vtm, .vtmb, .vtmg, .vthd, .vthb)
- Partitioned VTK (.pvtu, .pvti, .pvtm, .pvtb)
- ADAPT (.nc, .cdf, .elev, .ncd)
- ANALYZE (.img, .hdr)
- ANSYS (.inp)
- AVS UCD (.inp)
- BOV (.bov)
- BYU (.g)
- CCSM MTSD (.nc, .cdf, .elev, .ncd)
- CCSM STSD (.nc, .cdf, .elev, .ncd)
- CEaudc (.ucd, .inp)
- CMAT (.cmt)
- CTRL (.ctr)
- Chombo (.hdf5, .h5)
- Claw (.claw)
- Comma Separated Values (.csv)
- Cosmology Files (.cosmo, .gadget2)
- Curve2D (.curve, .ultra, .ult, .u)
- DDCMD (.ddcmd)
- Digital Elevation Map (.dem)
- Dyna3D (.dyn)
- EnSight (.case, .sos)
- Enzo boundary and hierarchy
- ExodusII (.g, .e, .exe, .ex2, .ex2v, .etc)
- ExtrudedVol (.exvol)
- FVCOM (MTMD, MTSD, Particle, STSD)
- Facet Polygonal Data
- Flash multiblock files
- Fluent Case Files (.cas)
- GGCM (.3df, .mer)
- GTC (.h5)
- GULP (.trg)
- Gadget (.gadget)
- Gaussian Cube Files (.cube)
- JPEG image (.jpg, .jpeg)
- LAMMPS Dump (.dump)
- LAMMPS Structure Files
- LODI (.nc, .cdf, .elev, .ncd)
- LODI Particle (.nc, .cdf, .elev, .ncd)
- LS-DYNA (.k, .lsdyna, .d3plot, .d3plot)
- M3DCI (.h5)
- MFIX Unstructured Grid (.RES)
- MM5 (.mm5)
- MPAS NetCDF (.nc)
- Meta Image (.mhd, .mha)
- Miranda (.mir, .raw)
- Multilevel 3D Plasma (.m3d, .h5)
- NASTRAN (.nas, .f06)
- NeK5000 Files
- Nrrd Raw Image (.nrrd, .nhdr)
- OpenFOAM Files (.foam)
- PATRAN (.neu)
- PFLOTTRAN (.h5)
- PLOT2D (.p2d)
- PLOT3D (.xyz, .q, .x, .vp3d)
- PLY Polygonal File Format
- PNG Image Files
- POP Ocean Files
- ParaDIS Files
- Phasta Files (.pht)
- Pixie Files (.h5)
- ProSTAR (.cel, .vrt)
- Protein Data Bank (.pdb, .ent, .pdb)
- Raw Image Files
- Raw NRRD image files (.nrrd)
- SAMRAI (.samrai)
- SAR (.SAR, .sar)
- SAS (.sasgeom, .sas, .sasdata)
- SESAME Tables
- SLAC netCDF mesh and mode data
- SLAC netCDF particle data
- Silo (.silo, .pdb)
- Spherical (.spherical, .sv)
- SpyPlot CTH
- Spy Plot (.case)
- Stereo Lithography (.stl)
- TFT Files
- TIFF Image Files
- TSurf Files
- Tecplot ASCII (.tec, .tp)
- Tecplot Binary (.pit)
- Tetrad (.hdf5, .h5)
- UNIC (.h5)
- VASP CHGCA (.CHG)
- VASP OUT (.OUT)
- VASP POSTCAR (.POS)
- VPIC (.vpc)
- VRML (.wrl)
- Velodyne (.vl, .rst)
- VizSchema (.h5, .vsh5)
- Wavefront Polygonal Data (.obj)
- WindBlade (.wind)
- XDMF and hdf5 (.xmf, .xdmf)
- XMol Molecule

From Bill Sherman, IU, SC14 Workshop
Ingesting Data

Need to convert your data to VTK objects
So far I’ve done
  - Geant’s HepRepXML format via my GeantToVTK Python ParaView Plugin
  - Text [csv] files via external python scripts (pvpython) - I have a generic script
  - Direct translation of art objects within art via C++/Catalyst [A real event display]
  - Specialty “Python Programmable source” within ParaView (e.g. Read numpy file)

Translation from Root files would probably not be hard (if you build ParaView and Root with the same Python, ParaView could do “import ROOT”)

Dumb text files can be crucial (e.g. stuck Geant use case)

  - Can connect ParaView to a running process supplying VTK objects (in-situ processing/Catalyst) - but communication back to process is very limited

  - ParaView can run in parallel mode, using multiple CPUs and GPUs on a remote cluster displaying locally (e.g. our Lattice QCD GPU cluster). I’ve tried 4 CPU’s/GPU’s and had mixed results (probably don’t have a big enough problem yet) – don’t underestimate the power of the GPU in your laptop.
Questions for you

What are your specific visualization needs?
Where does viz fit in to your work?
  o Perhaps many places
  o g-2’s specific use cases have been a good guide

What niche(s) are you trying to fill?
  o The “one event display to rule them all” remains elusive
  o While ParaView is very useful & flexible, I don’t think it’ll fit every use case
  o E.g. Probably not good for highly interactive tracking debugging with real-time algorithm communication
  o E.g. Probably not good for display kiosks operated by public

What’s a demo that would help you decide?
[Requirements? Expectations?]
Next steps

On g-2 we’ve been pleasantly surprised/amazed by ParaView. Extremely useful for debugging Geant

There’s a lot more to visualization than we know

Careful - it’s fun (and a bit addictive, you’ll want to use it everywhere)

SCD Visualization Strategy may involve ParaView
[attractive: Mature system with broad outside support, aligns us with HPC/ASCR community]

Documenting things so far
http://lyon-fnal.github.io/paraviewForHep/

Just DEMO it already…
Demo 1 - General ParaView

- Download it yourself from www.paraview.org
- Note pipeline
- Make Wavelet (see help); Apply Button
- Info - Scalar field
- Coloring; Surface; Volume (slow) - change opacity
- Camera movement (Camera Undo); Note LOD
- Filters - slice (one, multiple) - delete
- Clip (eyeballs) - delete
- Contour - then slice
Demo 2 - g-2 Ring

- show file, unzip -l
- Load g-2 ring from HepRepXML file; note options
- -Y view, change colors
- Multi-block inspector
- Front face culling
- Right click on inflector green circle thing
- Color by material
- Change center of rotation
- Wireframe/surface
- save state; restart; reload
- Load event (single); -SUPERIMPOSING OF VIZs- Rename sources
- multi block inspector on events (note e is very short)
- Event particle table
- Color by PDG
- VCR Controls
- On event (time) 4, select block, See in field data spreadsheet
- Select muons only from Event source
- Look at event 23
- Remember - these are reading in from a file
- Remove event; Load overlay events 0:5
The problem

Calorimeter Placement in gm2ringsim (Jarek / GM2-doc-3032) - using Inventor and ParaView

kalorimeter placement
kaspar@uw.edu, 8/11/15
The problem
The problem
Demo 3 - Live art

- Start ParaView with CatalystLiveButtons
- Note that anything to do with time (animation) will crash ParaView
- Flip through some events
- Save an event for later
- Make a normalized time $\frac{\text{globalTime}}{\max(\text{globalTime})}$
- Try event 8
- Overlay ring
- Color by pdg
- Make line width 4
- Threshold by global time
- Can really study the events
- Show making a movie
- Show movies

- Go change FCL to stop on calorimeter hits
- Show making spheres out of hits
Demo 4 - Kicker field

- Look at files in demo3/kickerdata
- Load geometry (choosing \([03]\), fix colors, cull front face
- Load field scan (note multi-timesteps), show info (29M points)
- Try surface
- Try volume - VOLUME only works on scalar field, use calculator, By is what matters
- Try slices - y, camera normal
- Try stream tracer on point -1358, 0, 6851, 10000 points, radius 1000
- Glyphs - arrows - scale by vector, factor 120, 100000 points
- Maybe try a contour

Show animation

Try front animation
- Load geometry (wireframe), fields
- Set 2D
- Slice in along camera
- Set colors to 1.4, 1.5 (hit upper gear for annotation),
- Need two calculators - one for By and another for y
- Threshold
- plot over line in front of calorimeter (use 2d, just show slice), left axis custom range
- Add time annotation
- Do animation - 10 frame/s, 1 per frame
Demo 5 - LArTPC fun!

- Load wv3d_clean.pvsm
- Show histograms, warps (noisy first)

- Load wv3d_add.pvsm
- Try volume rendering

- Load wv3d_match.pvsm
- Histogram
- Threshold at 50 - all scalars off!

- Maybe try others