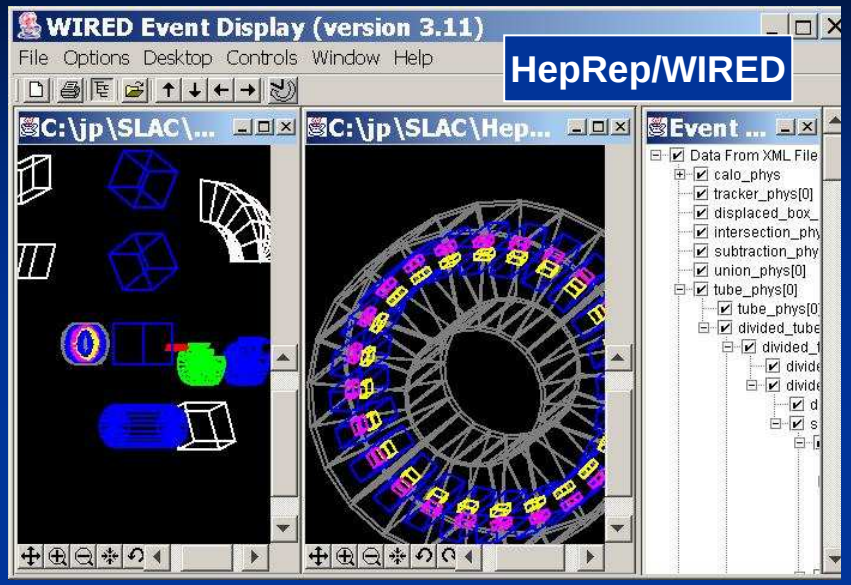
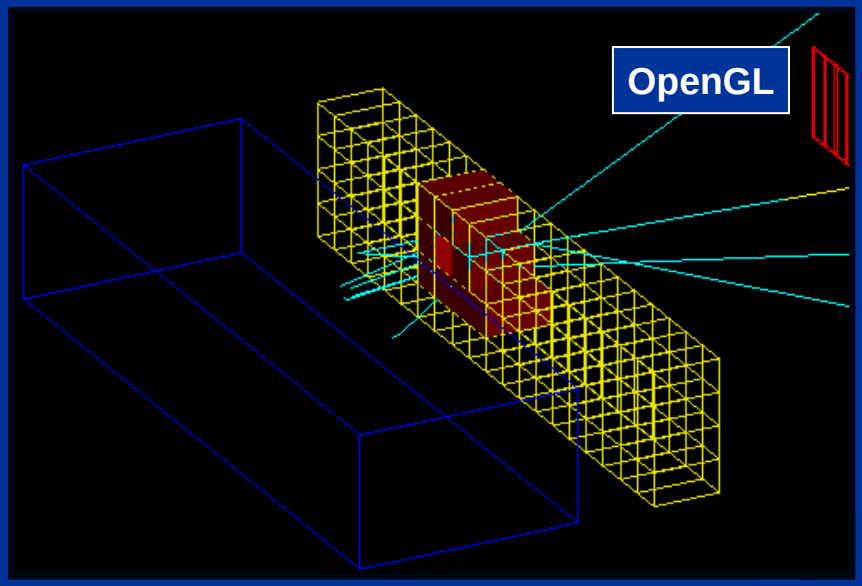
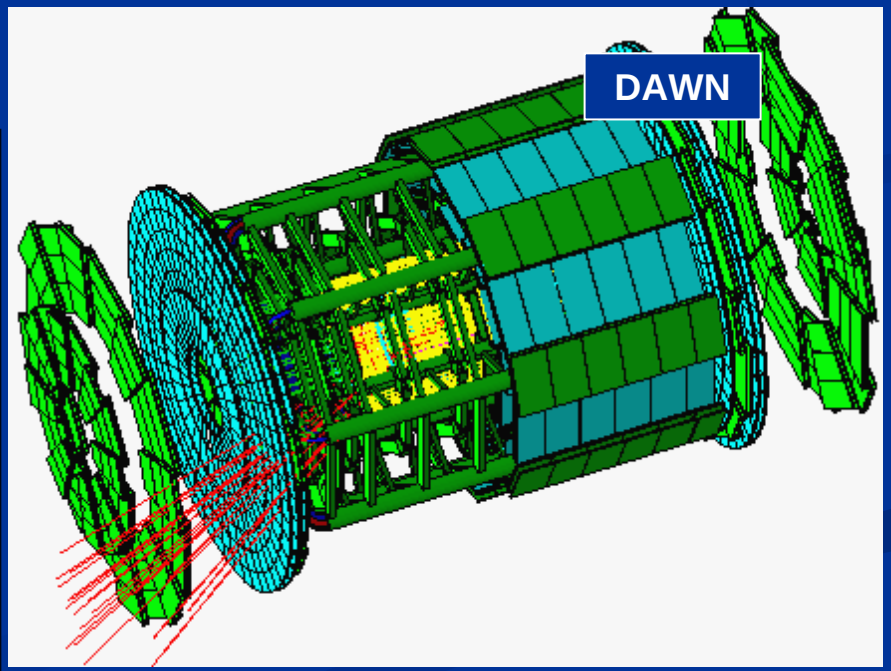
Introduction to Geant4 Visualization

Joseph Perl, SLAC

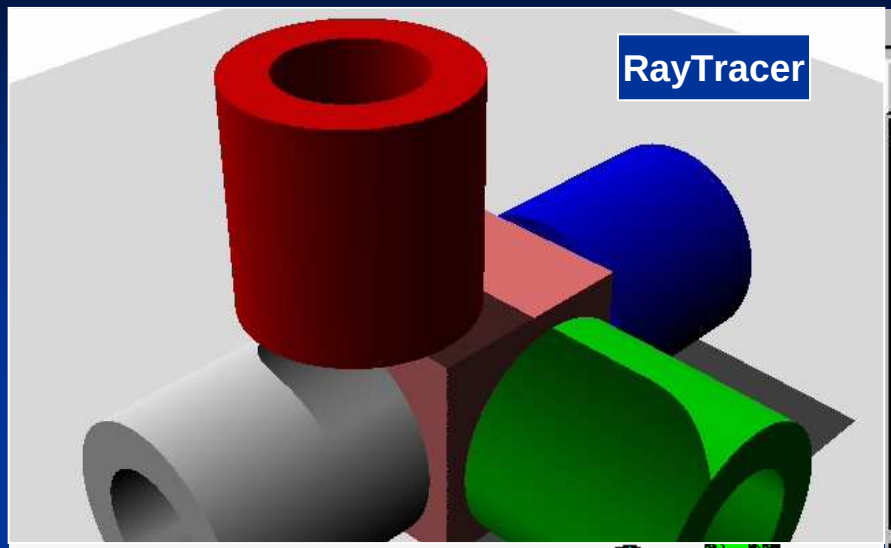


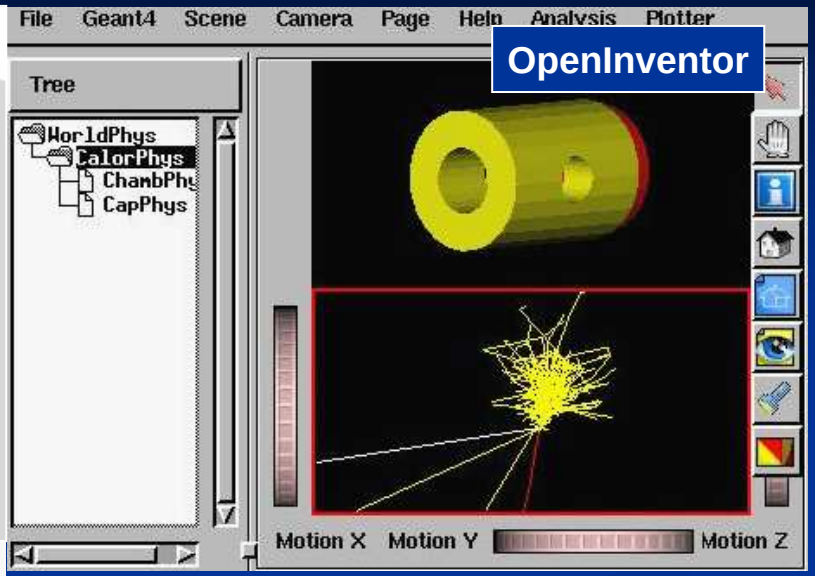
So many options, it needs two title pages

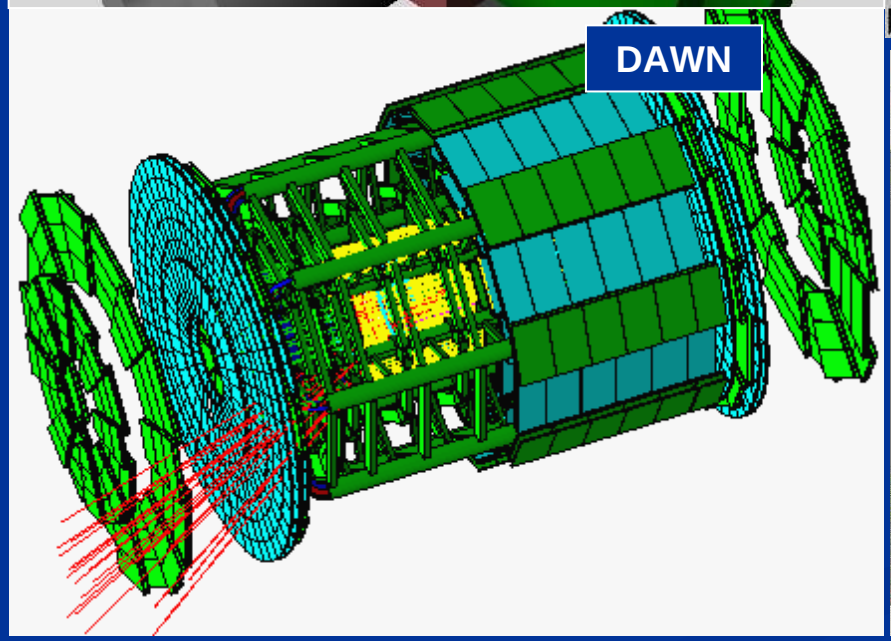


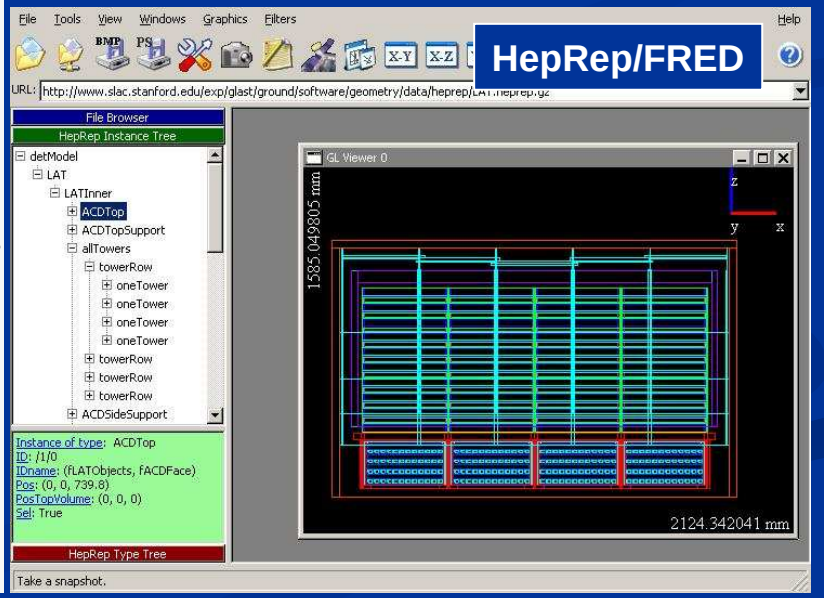


Introduction to Geant4 Visualization









Contents

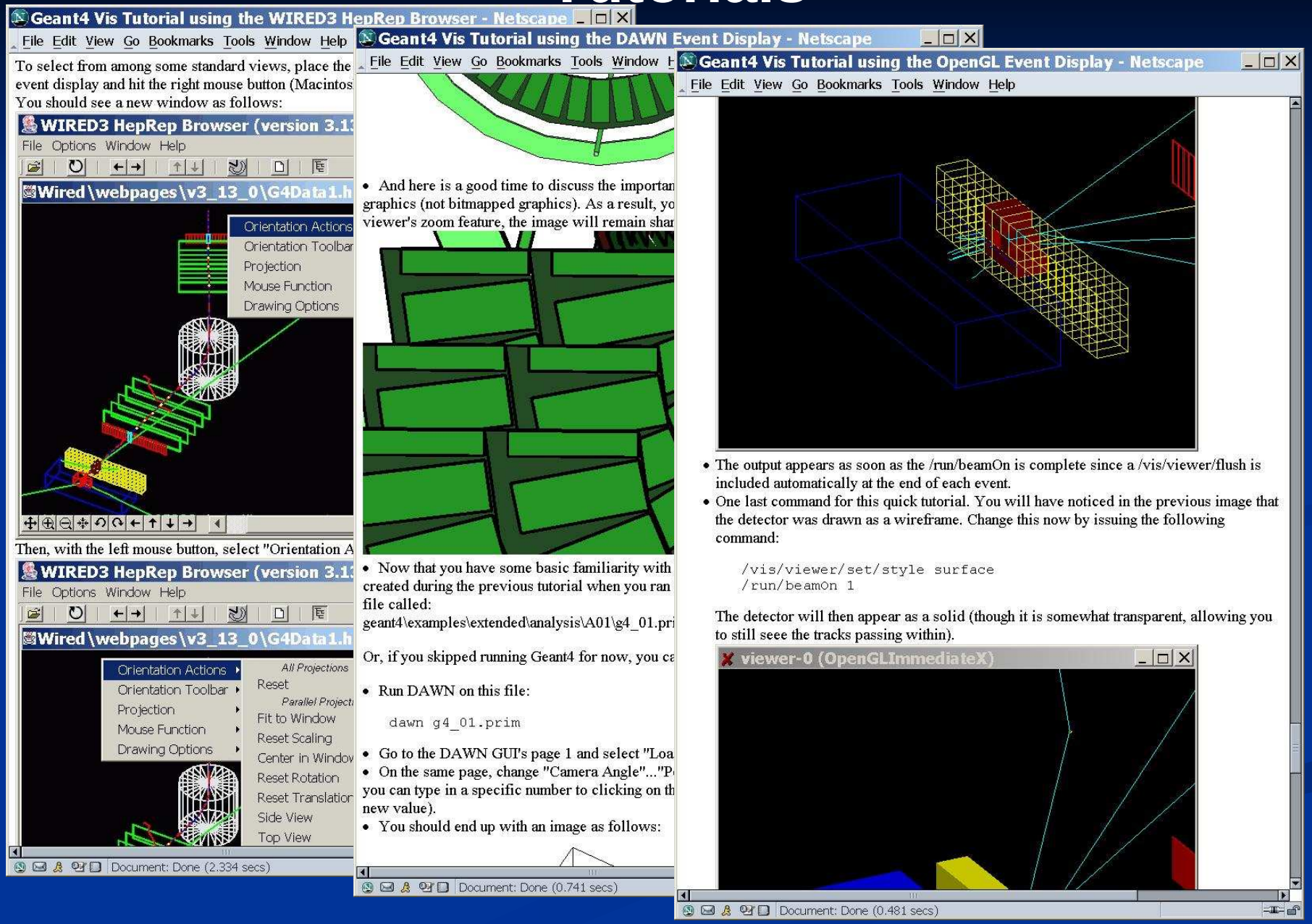
- The General Concepts behind Geant4 Visualization
  - Purpose of Geant4 Visualization
  - What can be Visualized
  - You have a Choice of Visualization Drivers
  - Visualization Attributes
- Seven Visualization Drivers
  - OpenGL
  - OpenInventor
  - HepRep/WIRED (and FRED)
  - DAWN
  - VRML
  - RayTracer
  - ASCIITree
- How to Run Geant4 Visualization
  - Environment Variables
  - Commands

How this Document Fits with Other Tutorial Materials

- This presentation can be used on its own, but gives the most comprehensive introduction to Geant4 visualization when used as part of the following full set of documents:
  - Geant 4 Tutorial
  - Introduction to Geant4 Visualization
  - Geant4 Workshop Visualization Tutorial using the WIRED3 Event Display
  - Geant4 Workshop Visualization Tutorial using the DAWN Event Display
  - Geant4 Workshop Visualization Tutorial using the OpenGL Event Display
  - See the URLS at the end of this presentation

- This presentation discusses seven visualization drivers:
  - OpenGL
  - OpenInventor
  - HepRep/WIRED
  - DAWN
  - VRML
  - RayTracer
  - ASCIITree

Tutorials


Geant4 Vis Tutorial using the WIRED3 HepRep Browser - Netscape
File Edit View Go Bookmarks Tools Window Help
To select from among some standard views, place the
event display and hit the right mouse button (Macintos
You should see a new window as follows:
WIRED3 HepRep Browser (version 3.1
File Options Window Help
Wired\webpages\v3_13_0\G4Data1.h
Orientation Actions
Orientation Toolbar
Projection
Mouse Function
Drawing Options
Then, with the left mouse button, select "Orientation A
WIRED3 HepRep Browser (version 3.1
File Options Window Help
Wired\webpages\v3_13_0\G4Data1.h
Orientation Actions
Orientation Toolbar
Projection
Mouse Function
Drawing Options
All Projections
Reset
Parallel Projecti
Fit to Window
Reset Scaling
Center in Windov
Reset Rotation
Reset Translatior
Side View
Top View
Document: Done (2.334 secs)
Geant4 Vis Tutorial using the DAWN Event Display - Netscape
File Edit View Go Bookmarks Tools Window H
• And here is a good time to discuss the importan
graphics (not bitmapped graphics). As a result, yo
viewer's zoom feature, the image will remain shar
• Now that you have some basic familiarity with
created during the previous tutorial when you ran
file called:
geant4\examples\extended\analysis\A01\g4_01.pri
Or, if you skipped running Geant4 for now, you ca
• Run DAWN on this file:
dawn g4_01.prim
• Go to the DAWN GUI's page 1 and select "Loa
• On the same page, change "Camera Angle"..."P
you can type in a specific number to clicking on th
new value).
• You should end up with an image as follows:
Document: Done (0.741 secs)
Geant4 Vis Tutorial using the OpenGL Event Display - Netscape
File Edit View Go Bookmarks Tools Window Help
• The output appears as soon as the /run/beamOn is complete since a /vis/viewer/flush is
included automatically at the end of each event.
• One last command for this quick tutorial. You will have noticed in the previous image that
the detector was drawn as a wireframe. Change this now by issuing the following
command:
/vis/viewer/set/style surface
/run/beamOn 1
The detector will then appear as a solid (though it is somewhat transparent, allowing you
to still seee the tracks passing within).
viewer-0 (OpenGLImmediateX)
Document: Done (0.481 secs)

Part 1: The General Concepts behind Geant4 Visualization

- Purpose of Geant4 Visualization
- What can be Visualized
- You have a Choice of Visualization Drivers
- Visualization Attributes

Purpose of Geant4 Visualization

- Quick response to study geometries, trajectories and hits
- High-quality output for publications
- Flexible camera control to debug complex geometries
- Tools to show volume overlap errors in detector geometries
- Interactive picking to get more information on visualized objects

What Can be Visualized

- Simulation data can be visualized:
  - Detector components
  - Particle trajectories and tracking steps
  - Hits of particles in detector components
- Other user defined objects can be visualized:
  - Polylines
    - such as coordinate axes
  - 3D Markers
    - such as eye guides
  - Text
    - descriptive character strings
    - comments or titles …

You have a Choice of Visualization Drivers

- OpenGL
  - View directly from Geant4
  - Uses GL libraries that are already included on most Linux systems (plus some Windows availability)
  - Rendered, photorealistic image with some interactive features
    - zoom, rotate, translate
  - Fast response (can usually exploit full potential of graphics hardware)
  - Limited printing ability (pixel graphics, not vector graphics)
- OpenInventor
  - View directly from Geant4
  - Requires addition of OpenInventor libraries (freely available for most Linux systems).
  - Rendered, photorealistic image
  - Many interactive features
    - zoom, rotate, translate
    - click to “see inside” opaque volumes
  - Fast response (can usually exploit full potential of graphics hardware)
  - Expanded printing ability (vector and pixel graphics)
- HepRep/WIRED
  - Create a file to view in the WIRED3 HepRep Browser
  - Requires WIRED browser (a Java application easily to install on all operating systems)
  - Wireframe or simple area fills (not photorealistic)
  - Many interactive features
    - zoom, rotate, translate
    - click to show attributes (momentum, etc.)
    - special projections (FishEye, etc.)
    - control visibility from hierarchical (tree) view of data
  - Hierarchical view of the geometry
  - Export to many vector graphic formats (PostScript, PDF, etc.)

More Choices of Visualization Drivers

- DAWN
  - Create a file to view in the DAWN Renderer
  - Requires DAWN, available for all Linux and Windows systems.
  - Rendered, photorealistic image
  - No interactive features
  - Highest quality technical rendering - output to vector PostScript
- VRML
  - Create a file to view in any VRML browser (some as web browser plug-ins).
  - Requires VRML browser (many different choices for different operating systems).
  - Rendered, photorealistic image with some interactive features
    - zoom, rotate, translate
  - Limited printing ability (pixel graphics, not vector graphics)
- RayTracer
  - Create a jpeg file
  - Forms image by using Geant4's own tracking to follow photons through the detector
  - Can show geometry but not trajectories
  - Can render any geometry that Geant4 can handle (such as Boolean solids)
  - Supports shadows, transparency and mirrored surfaces
- ASCIITree
  - Text dump of the geometry hierarchy
  - Not graphical
  - Control over level of detail to be dumped
  - Can calculate mass and volume of any hierarchy of volumes

Choose the Driver that Meets Your Needs

- If you want very responsive photorealistic graphics (and have the OpenGL libraries installed)
  - OpenGL is a good solution

  (if you have the Motif extensions, this also gives GUI control)
- If you want very responsive photorealistic graphics plus more interactivity (and have the OpenInventor libraries installed)
  - OpenInventor is a good solution
- If you want GUI control, want to be able to pick on items to inquire about them (identity, momentum, etc.), perhaps want to render to vector formats, and a wireframe look will do
  - HepRep/WIRED will meet your needs
- If you want to render highest quality photorealistic images for use in a poster or a technical design report, and you can live without quick rotate and zoom
  - DAWN is the way to go
- If you want to render to a 3D format that others can view in a variety of commodity browsers (including some web browser plug-ins)
  - VRML is the way to go
- (more options on next page)

Choose the Driver (continued)

- (continued from previous slide)
- If you want to visualize a geometry that the other visualization drivers can't handle, or you need transparency or mirrors, and you don't need to visualize trajectories
  - RayTracer will do it
- If you just want to quickly check the geometry hierarchy, or if you want to calculate the volume or mass of any geometry hierarchy
  - ASCIITree will meet your needs
- You can also add your own visualization driver.
  - Geant4's visualization system is modular. By creating just three new classes, you can direct Geant4 information to your own visualization system.

Controlling Visualization

- Your Geant4 code stays basically the same no matter which driver you use
- Visualization is performed either with commands or from C++ code
  - For the present tutorial, we confine ourselves to command-driven visualization.
- Some visualization drivers work directly from Geant4
  - OpenGL
  - OpenInventor
  - RayTracer
  - ASCIITree
- For other visualization drivers, you first have Geant4 produce a file, and then you have that file rendered by another application (which may have GUI control)
  - HepRep/WIRED
  - DAWN
  - VRML

Basic Visualization Attributes

- Color, Visible/Invisible, Wireframe/Solid, etc.
- Set from C++ by creating a G4VisAttributes object and assigning it to a volume:
  - experimentalHall_logical -> SetVisAttributes (G4VisAttributes::Invisible)
- Can also be set interactively from the command prompt.
- Study G4 examples or references at end of this presentation to learn more about G4VisAttributes.

Additional User-Defined Attributes

- Geant4 Trajectories and Hits can be assigned additional arbitrary attributes that will be displayed when you click on the relevant object in the WIRED or FRED HepRep browsers.
- WIRED then lets you label objects by any of these attributes or cut visibility based on these attributes.
- Define the attributes with lines such as:
  - std::map<G4String,G4AttDef>* store = G4AttDefStore::GetInstance("G4Trajectory",isNew);
  - G4String PN("PN");
  - (*store)[PN] = G4AttDef(PN,"Particle Name","Physics","","G4String");
  - G4String IMom("IMom");
  - (*store)[IMom] = G4AttDef(IMom, "Momentum of track at start of trajectory", "Physics","","G4ThreeVector");
- Then fill the attributes with lines such as:
  - std::vector<G4AttValue>* values = new std::vector<G4AttValue>;
  - values->push_back(G4AttValue("PN",ParticleName,""));
  - s.seekp(std::ios::beg);
  - s << G4BestUnit(initialMomentum,"Energy") << std::ends;
  - values->push_back(G4AttValue("IMom",c,""));
- See geant4/source/tracking/src/G4Trajectory.cc for a good example.

Part 2: Seven Visualization Drivers

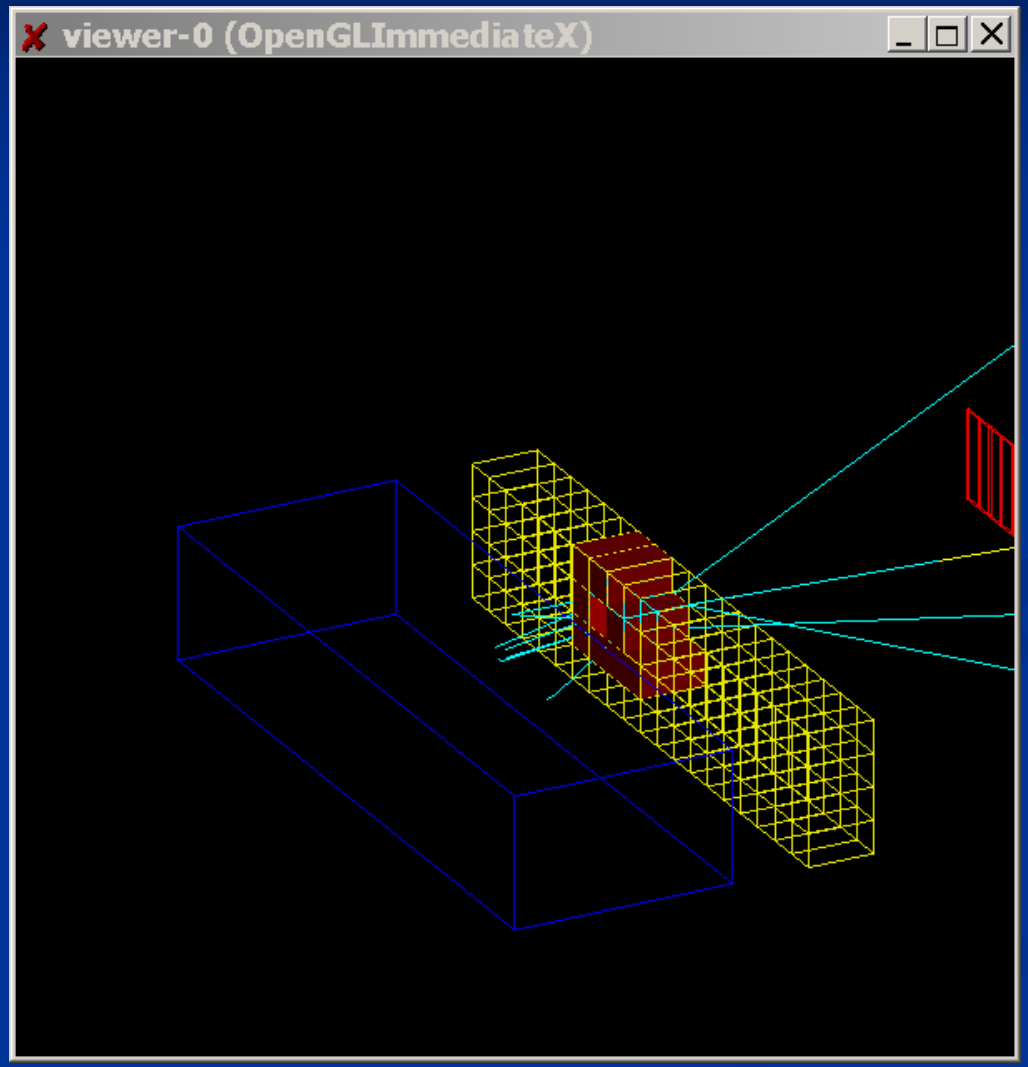
- OpenGL
- OpenInventor
- HepRep/WIRED (and FRED)
- DAWN
- VRML
- RayTracer
- ASCIITree

OpenGL

- Run directly from Geant4
  - /vis/open OGLIX

OpenGL: Runs Directly from Geant4

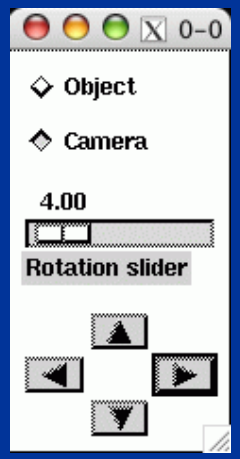
- With OpenGL, all commands go through Geant4:

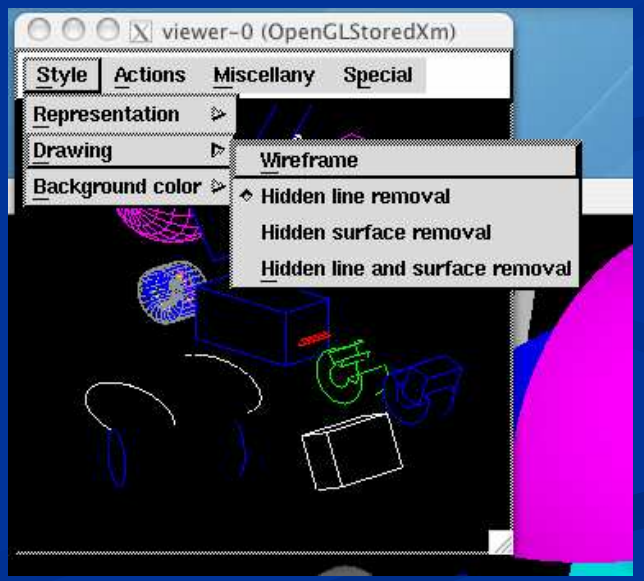



```
vis/open OGLIX
/vis/scene/create
/vis/scene/add/volume
/vis/sceneHandler/attach
/vis/viewer/flush
/vis/viewer/set/viewpointThetaPhi 70
20
/vis/viewer/zoom 2
/vis/viewer/reset
/vis/viewer/set/viewpointThetaPhi 40
40
/vis/viewer/panTo -5 -1
/vis/viewer/zoom 4.
/vis/scene/add/trajectories
/vis/scene/add/hits
/tracking/storeTrajectory 1
/run/beamOn 1
```

OpenGL with Motif Control

- If you don’t have Motif, all control is done from Geant4 commands:
  - /vis/open OGLIX
  - /vis/viewer/set/viewpointThetaPhi 70 20
  - /vis/viewer/zoom 2
  - etc.
- But if you have Motif libraries, you can control Geant4 from Motif widgets:
  - /vis/open OGLIXm





OpenGL Additional Modes

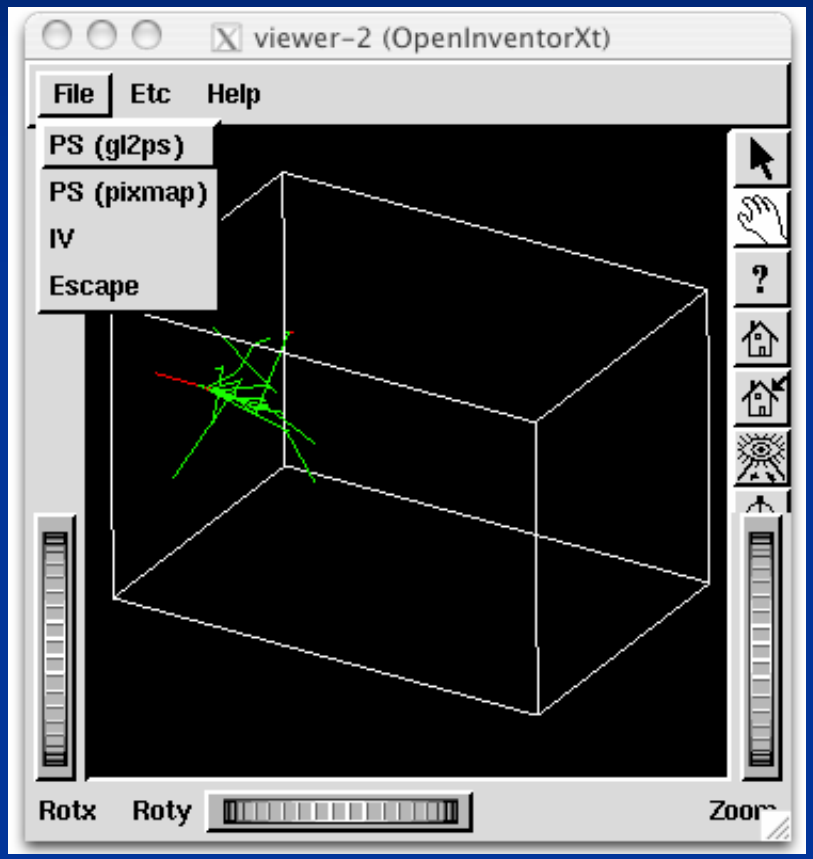
- There are actually 6 OpenGL drivers - OGLxy:
  - x = I (Immediate) or S (Stored)
  - y = X, Xm (Motif) or Win32
- Immediate mode: draws only to screen, no “memory”; detector can be redrawn after view changes but event data is lost.
- Stored mode: creates graphical database (display lists); nothing is lost on simple operations like change of viewing angle
- Also note: Smooth shading and Transparency are coming in the next Geant4 release (Geant4.8.0)

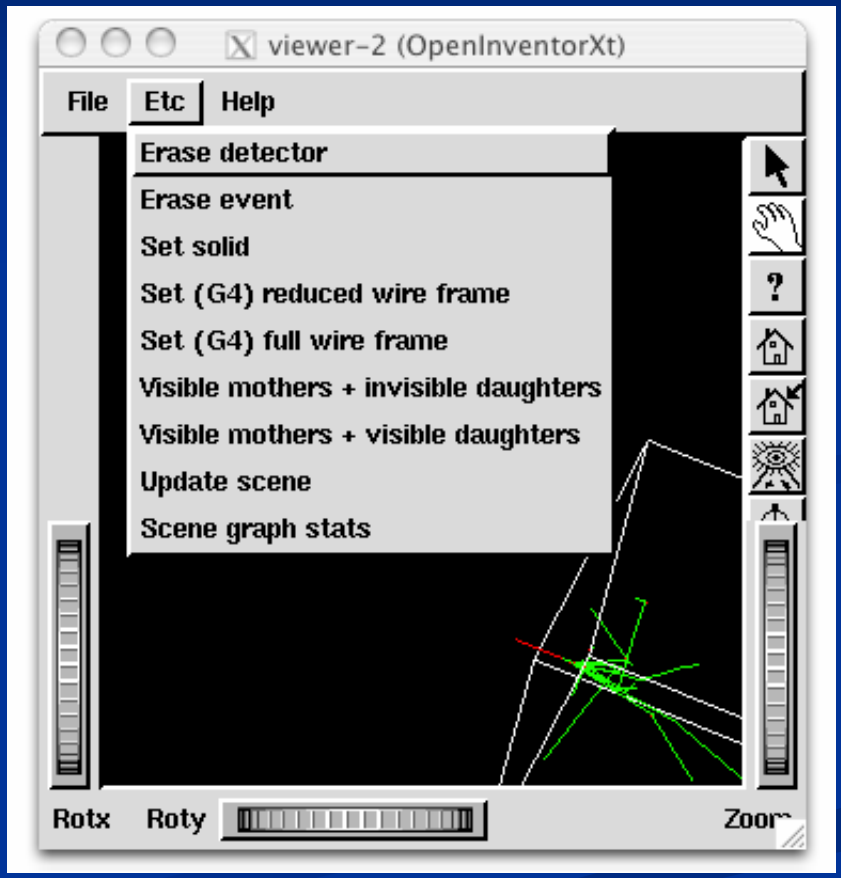
OpenInventor

- Start from Geant4
  - /vis/open OpenInventor
- But then control from the OpenInventor GUI

OpenInventor: Start from Geant4

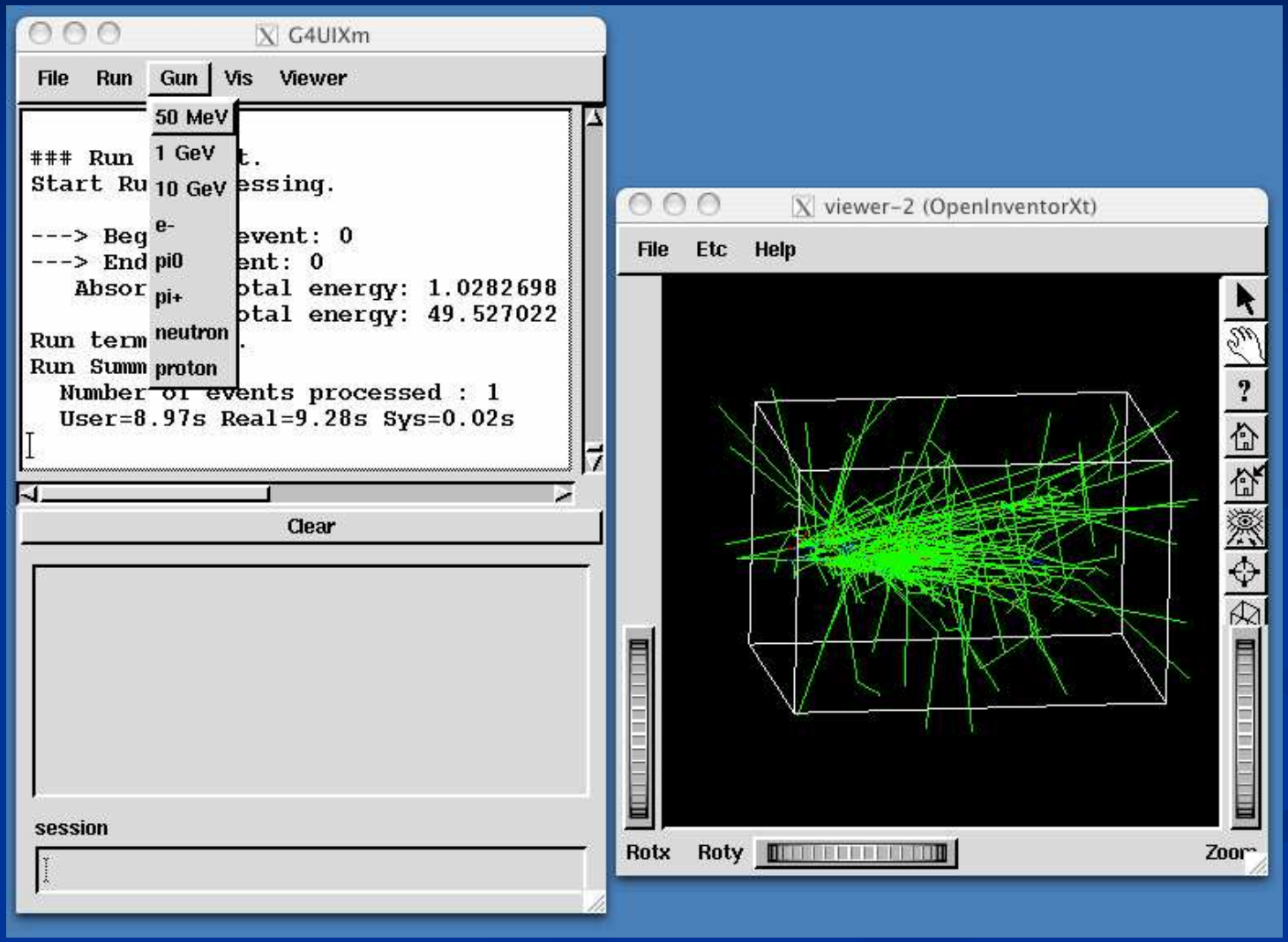
- With OpenInventor, start from Geant4, but then some control from OpenInventor GUI





OpenInventor: More GUI Control

- You can also choose to control the Geant4 run from OpenInventor.



OpenInventor Details

- Connected directly to the Geant4 kernel, using same language as that kernel (C++):
  - Can have direct access to Geant4 data (geometry, trajectories, etc.).
- Use of OpenGL for rendering:
  - Supports lighting and transparency
- Thumbwheel control to rotate an zoom
- Picking to ask about data
- “Control Clicking” on a volume turns on rendering of that volume’s daughters. “Shift Clicking” a daughter turns that rendering off:
  - If modeling opaque solid, effect is like opening a box to look inside
- OpenInventor is part of OpenScientist:
  - Can do analysis as well as visualization
- OpenScientist Home Page http://openscientist.lal.in2p3.fr
  - Follow the “Geant4 and Inventor” link at the left hand side of that page for details.

HepRep

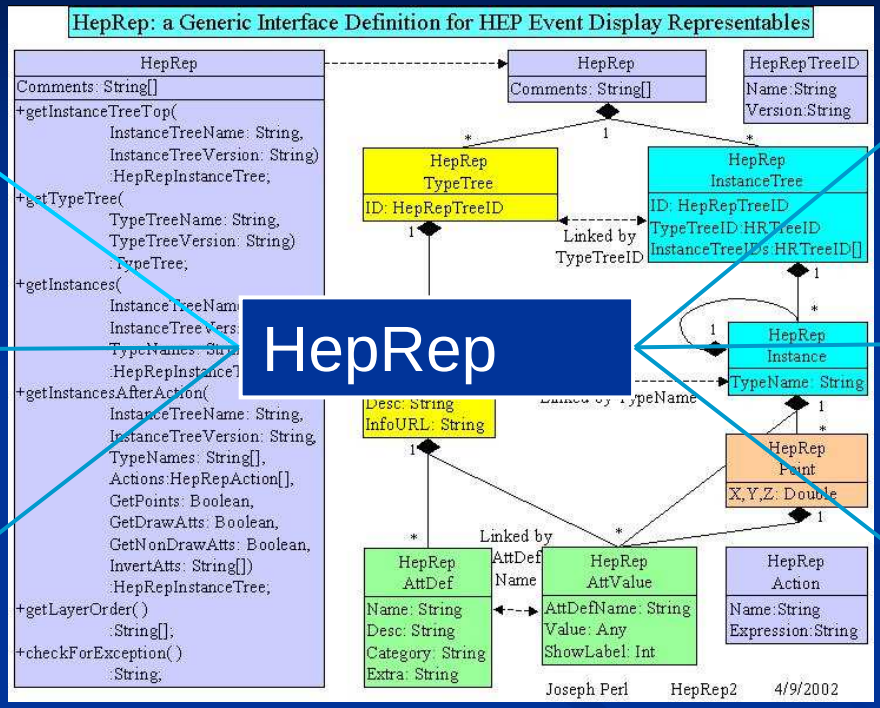
- Geant4 creates HepRepFile
  - /vis/open HepRepFile
- View file in WIRED3 or FRED HepRep Browsers
  - WIRED3 can export to various graphics formats

HepRep is Not Just for Geant4 and Not Just for WIRED

BaBar Server

GLAST Gaudi Service

Geant4 Server



HepRep

WIRED3 Client (Java)

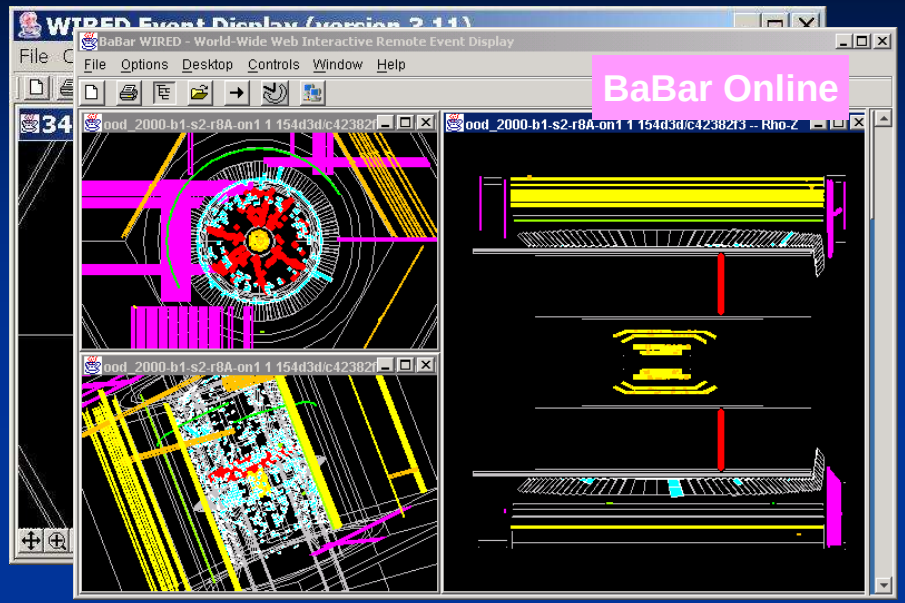
FRED Client (C++/Ruby)

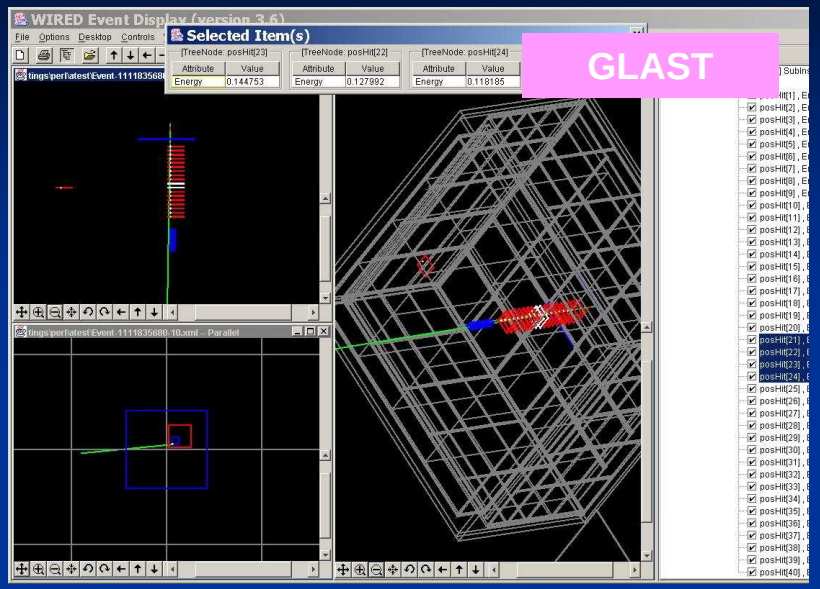
Other HepRep Clients

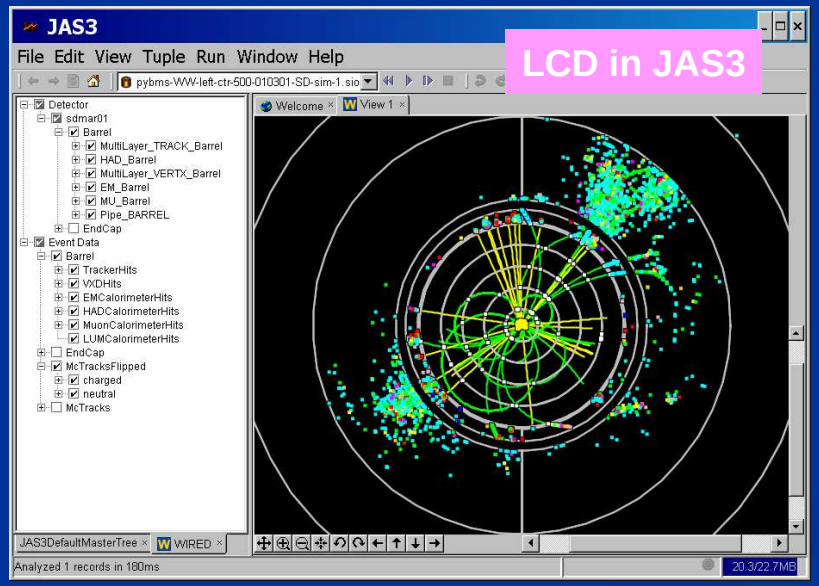
The HepRep interface breaks the dependency between any particular experiment's event display server and any particular event display client.

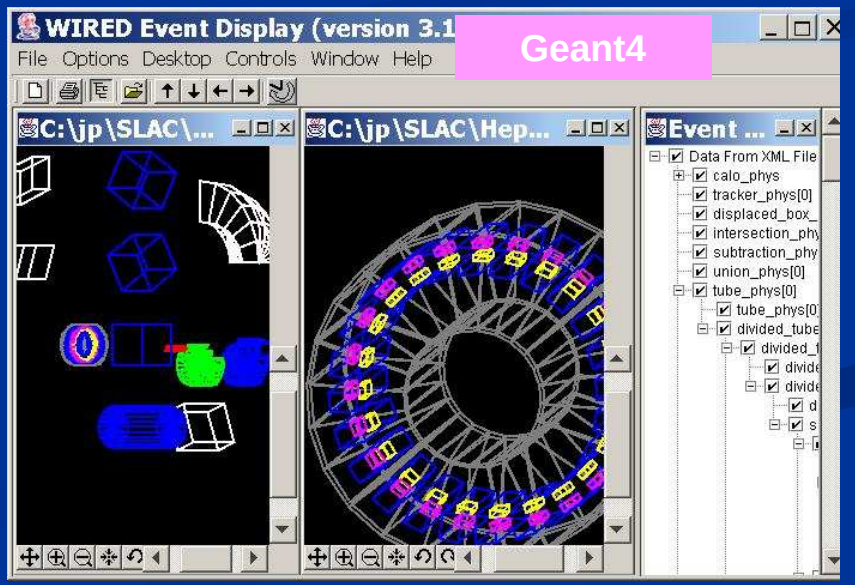
The HepRep format is independent of any one particular language or protocol. It can be used from C++ or Java and can be shipped as Corba, RMI, XML, C++, Java or JNI for consumption by WIRED, FRED or any other HepRep-enabled event display client.

Who's Using HepRep



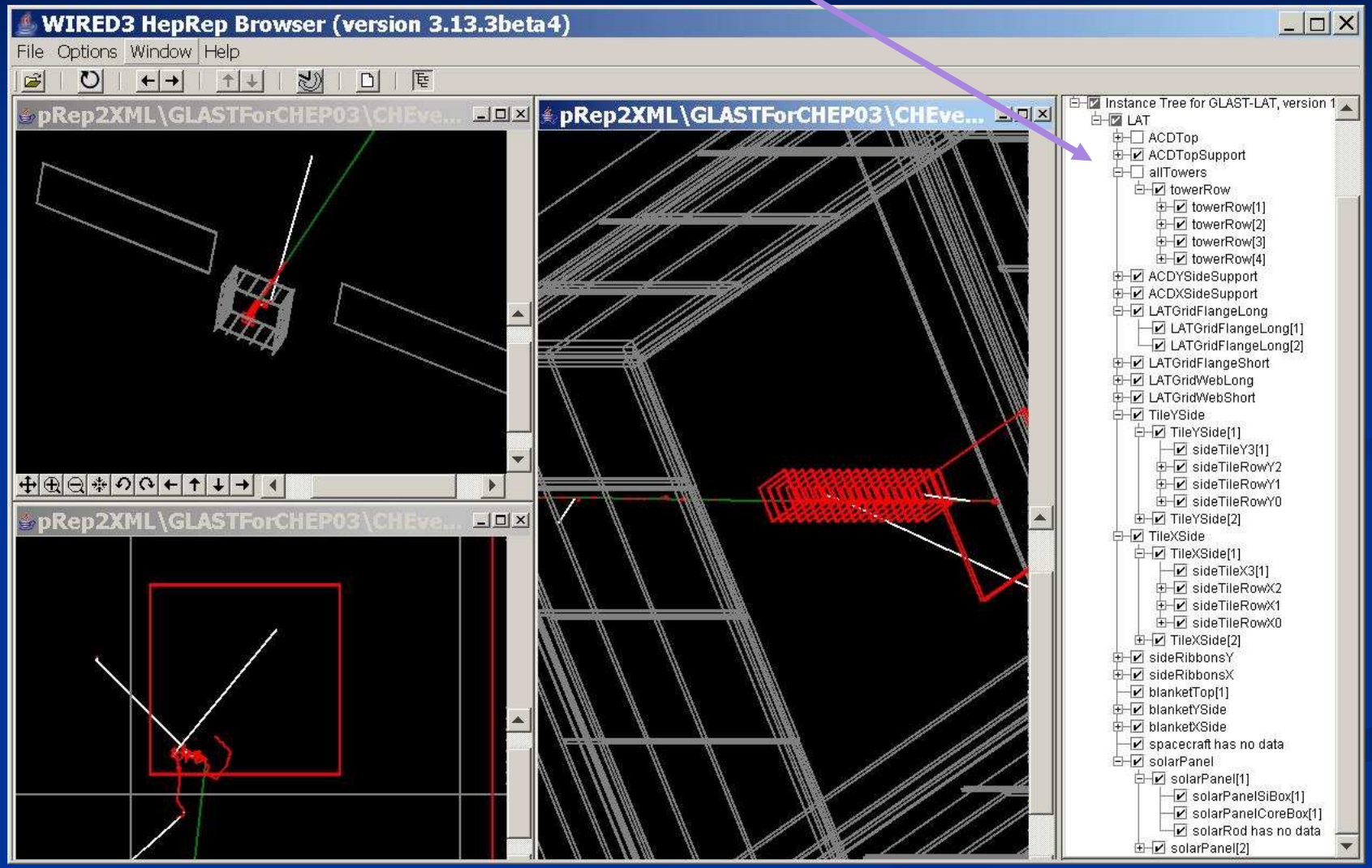






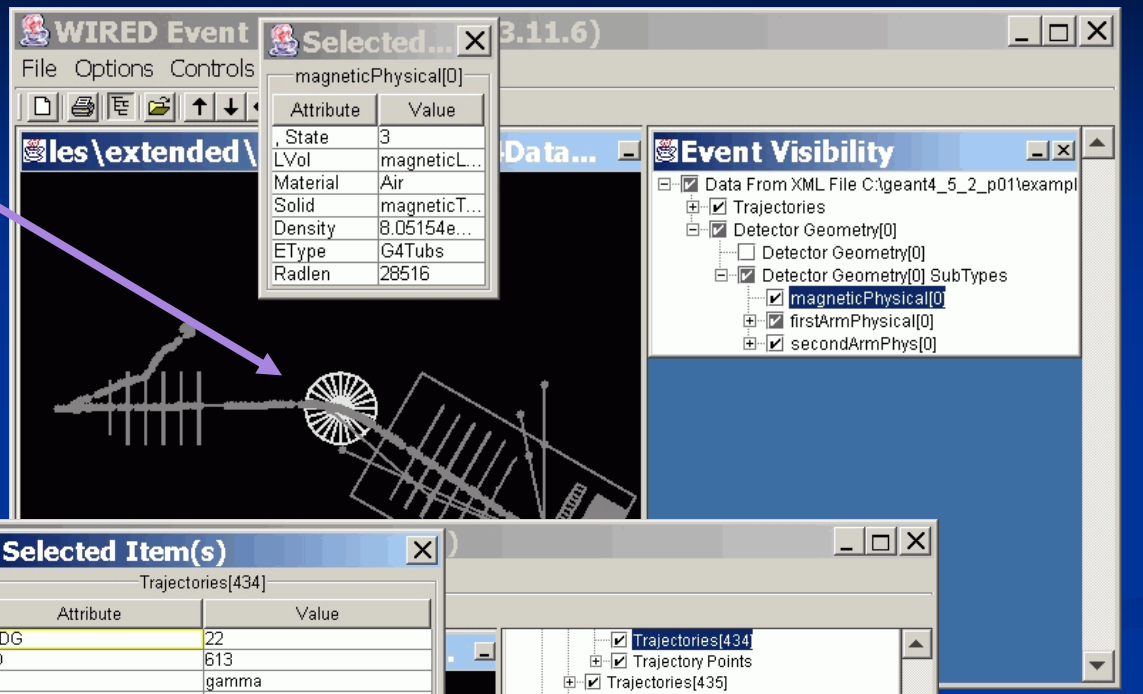
WIRED3: Shows Geometry Hierarchy

Turn visibility on and off from hierarchical control

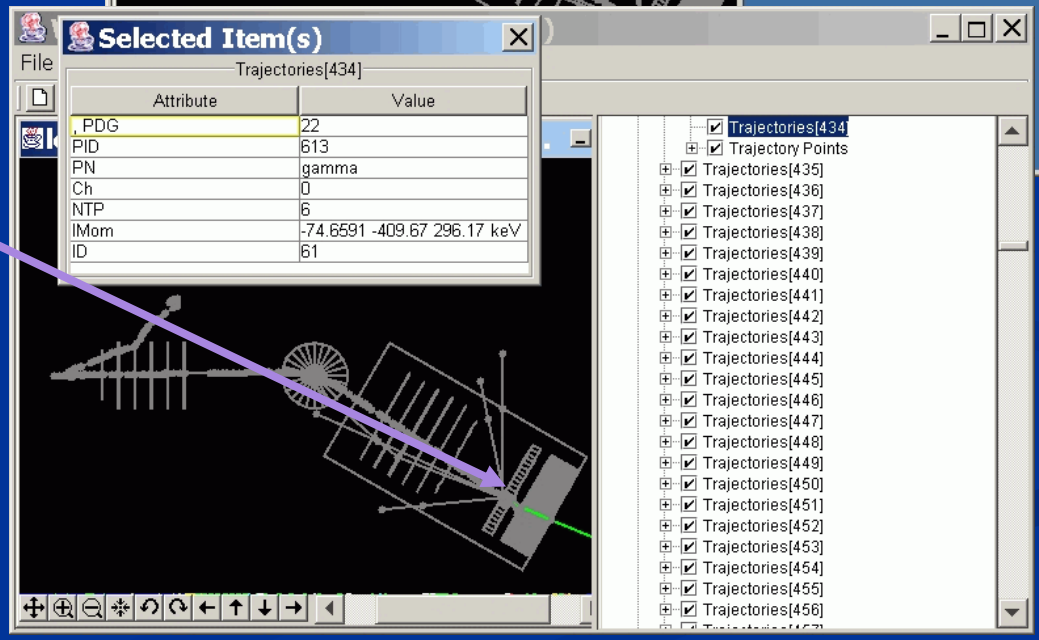


WIRED3: Pick to Show Physics Attributes

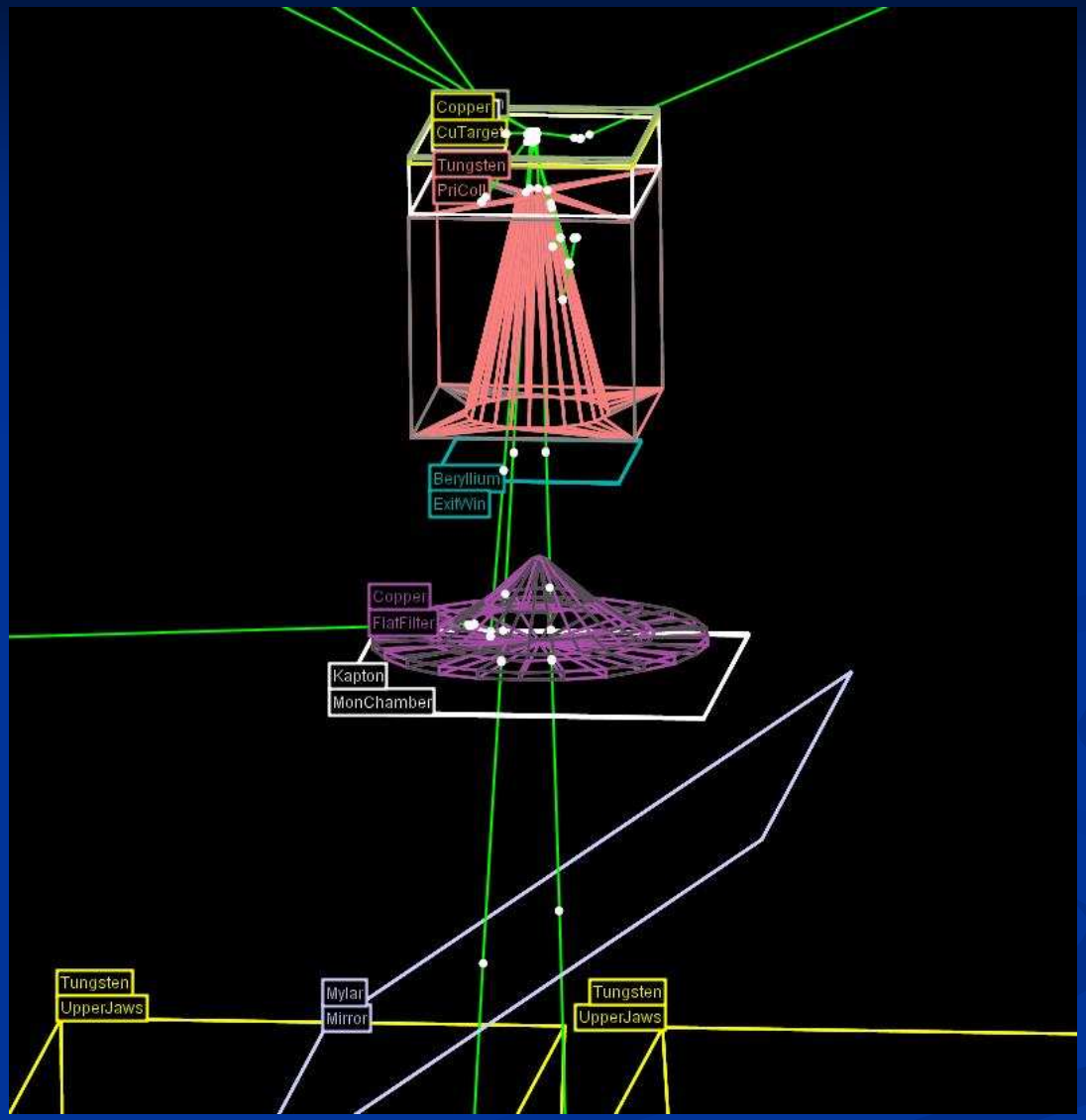
Picked on this volume to show
- **Material**
- **Density**
- **Radlen**
- **etc**



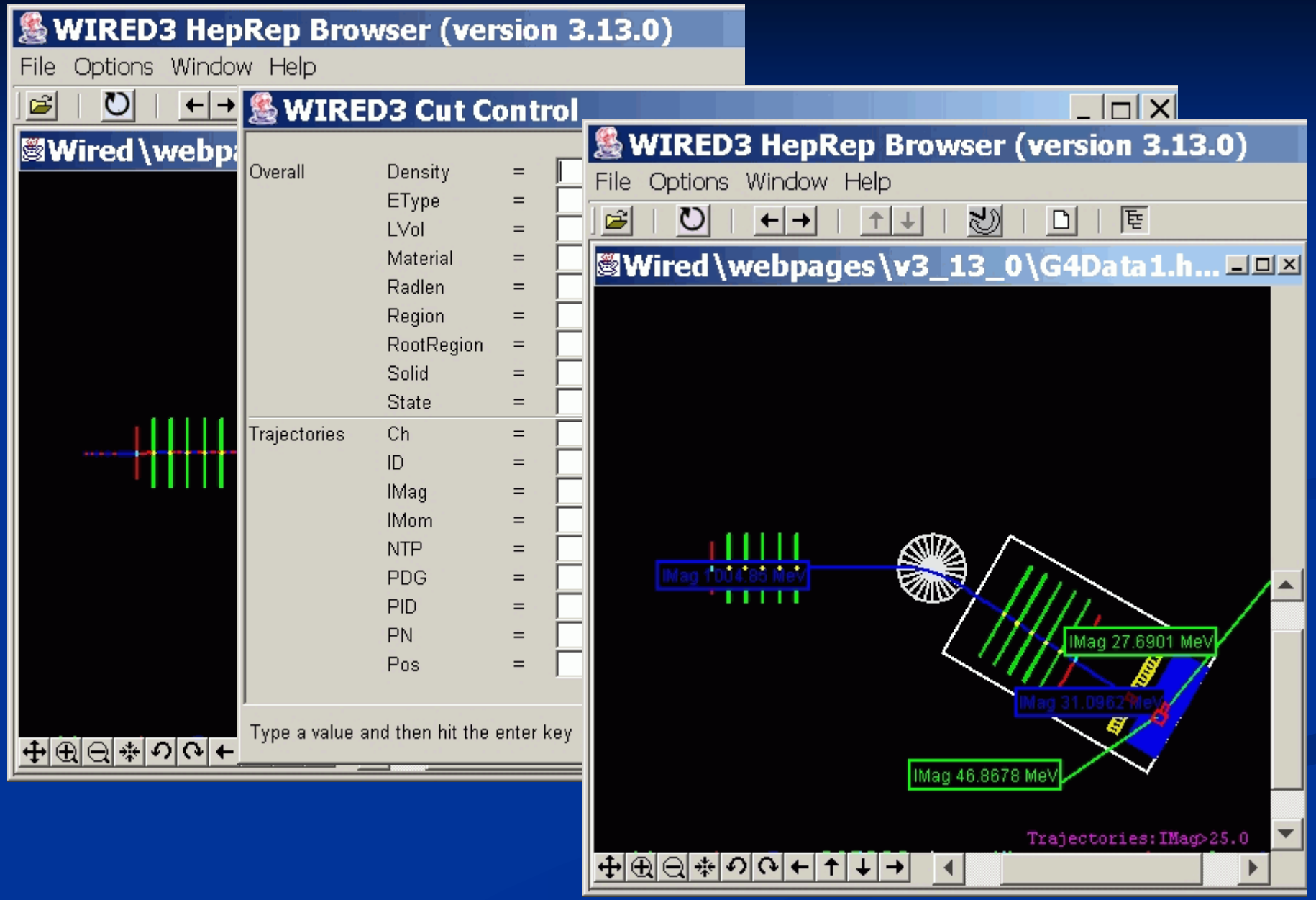
Picked on this trajectory to show
- **Particle ID**
- **Charge**
- **Momentum**
- **etc.**



WIRED3: Labeling by Any Attribute

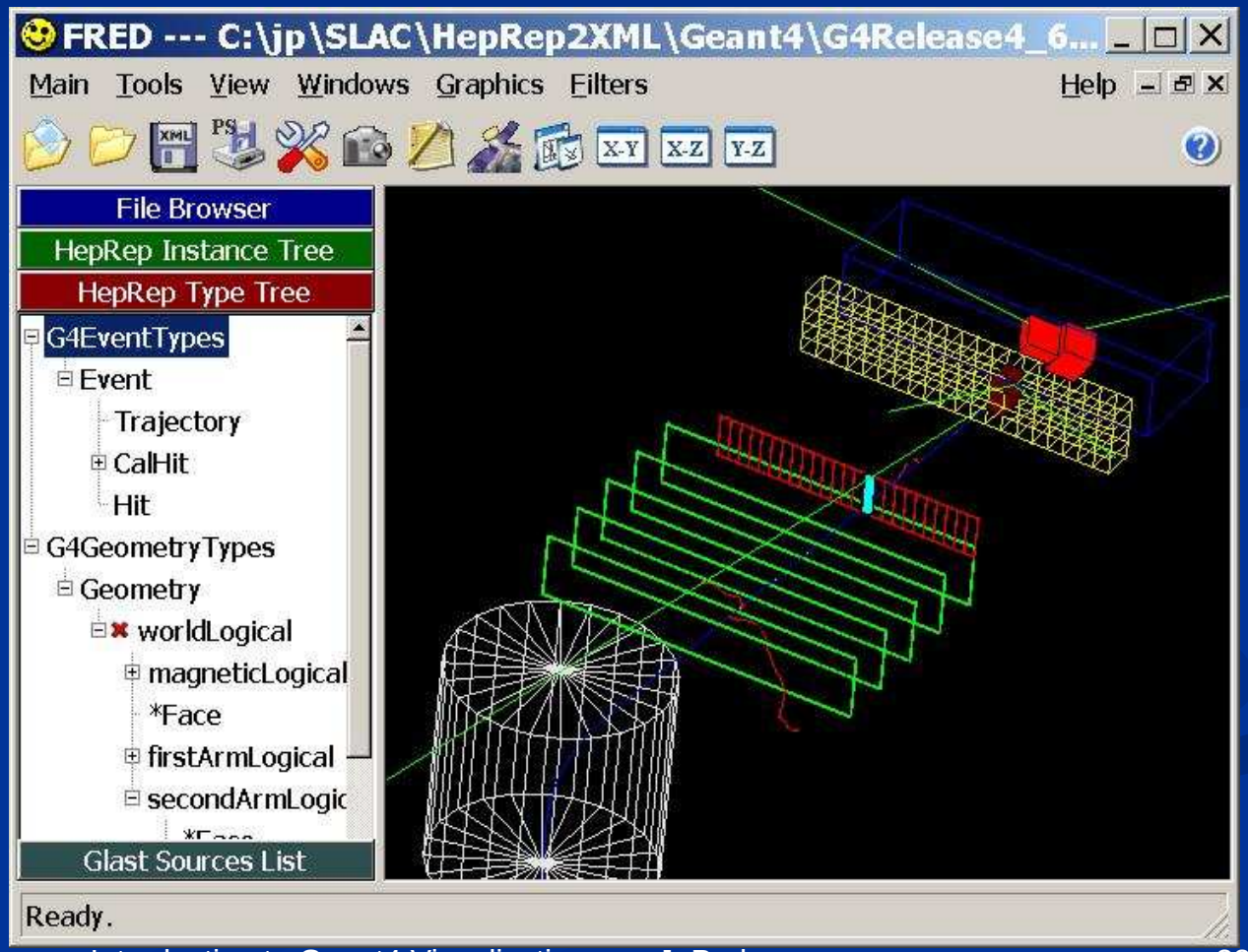


WIRED3: Cut by Any Attribute



FRED: Fox Ruby Event Display

- An additional HepRep-compatible browser developed by members of the GLAST space telescope collaboration.
- Includes the fast rotations and beautiful rendering of GL plus HepRep interactivity
- Allows scripting to change any attribute based on logic involving other attributes, hence things like "color by momentum" are scriptable.



DAWN

- Geant4 creates .prim file
  - /vis/open DAWNFILE
- DAWN renders .prim file into PostScript
- View or print from your favorite PostScript application

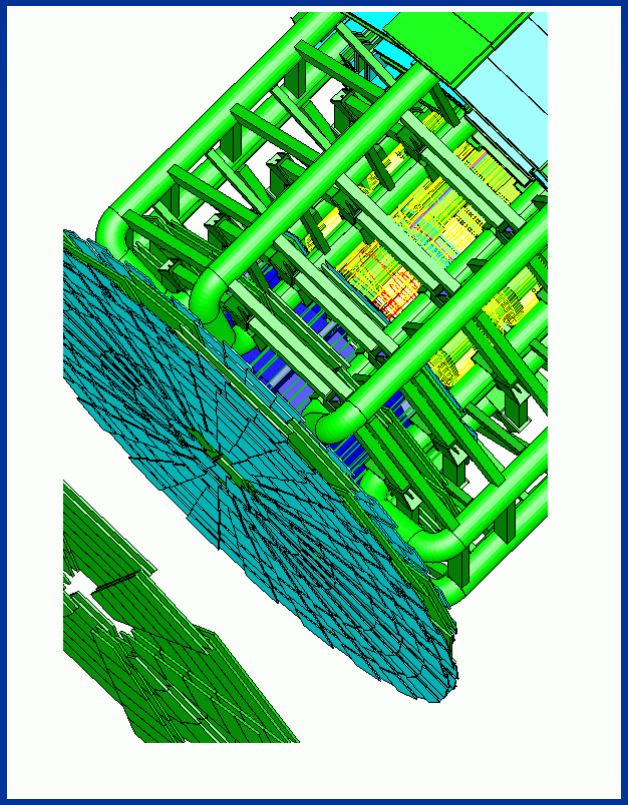
Origins of DAWN

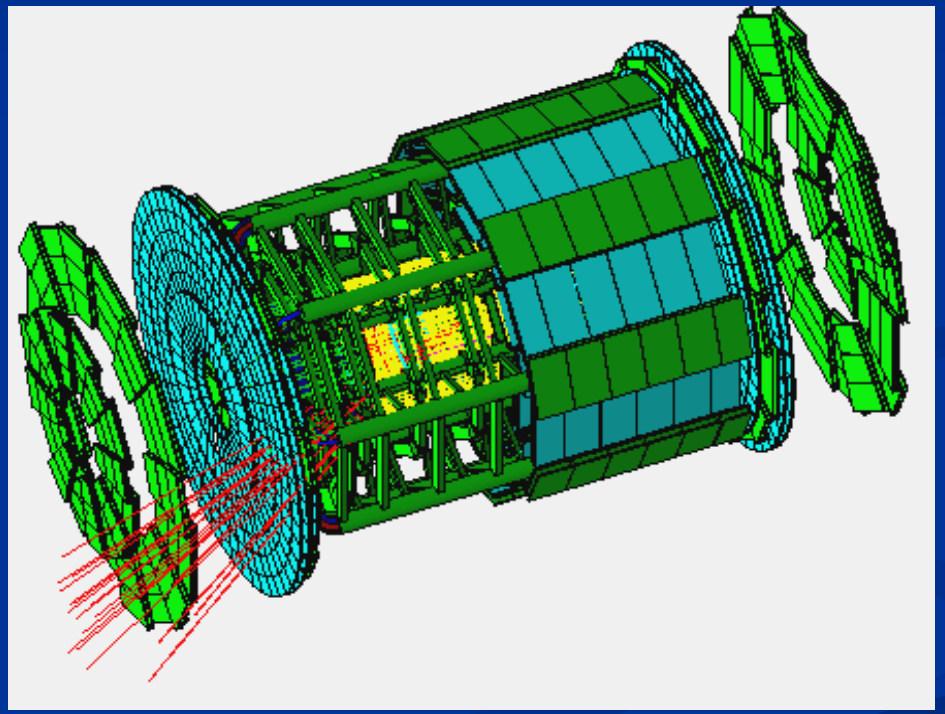
Fukui Renderer DAWN (Drawer for Academic WritiNgs).

- A vectorized 3D PostScript processor with analytical hidden line/surface removal intended for precise technical drawing of complicated objects.
- Specifically designed for Geant4.
- Primitives set is same as Geant4 primitives set.
- Produces device-independent vectorized graphics for high quality technical applications.

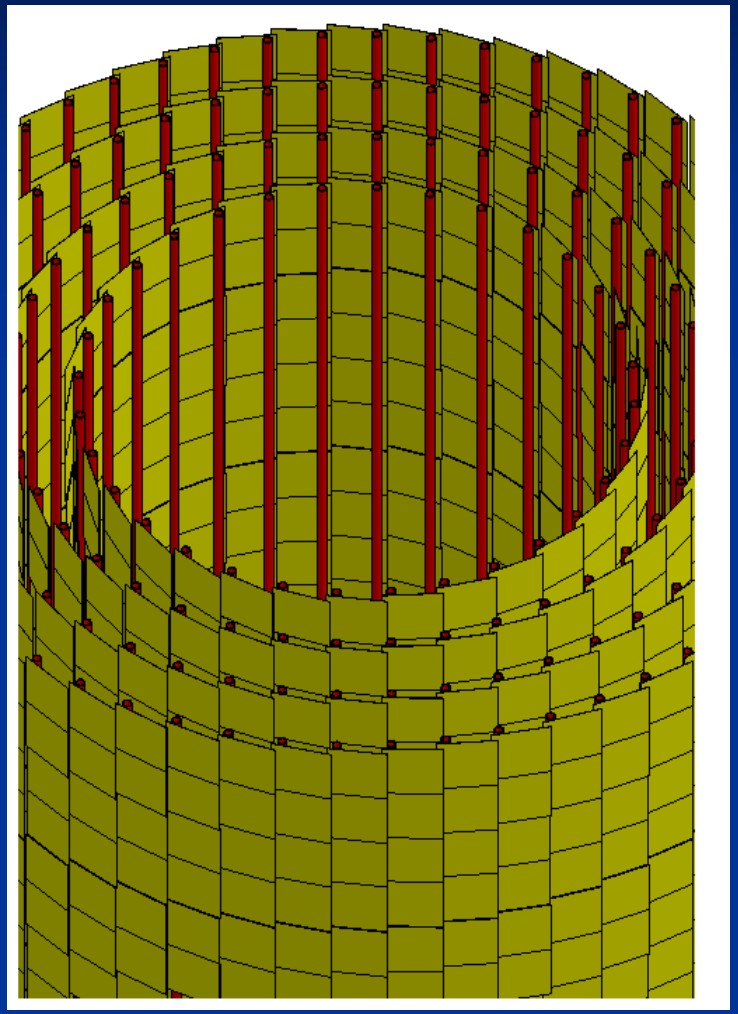
DAWN Examples

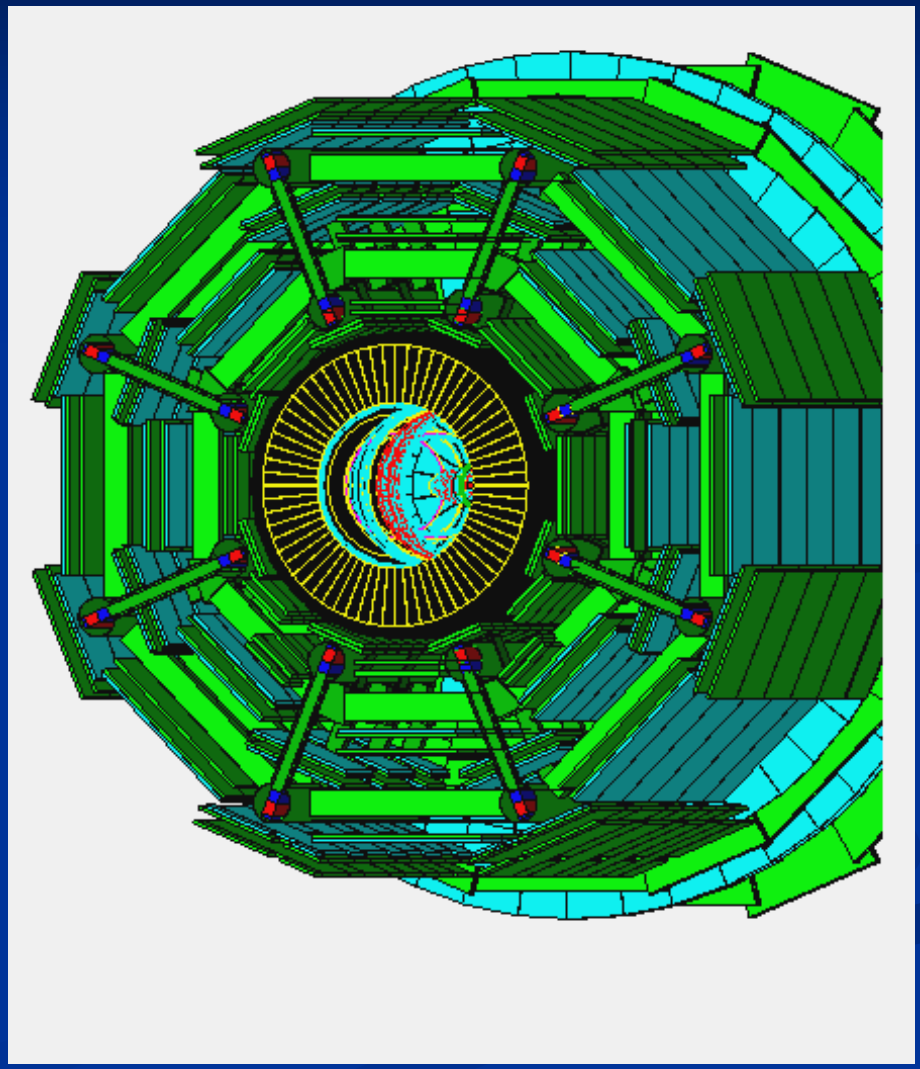
- From a repository of beautiful images at
  - http://geant4.kek.jp/~tanaka/GEANT4/ATLAS_G4_GIFFIG/





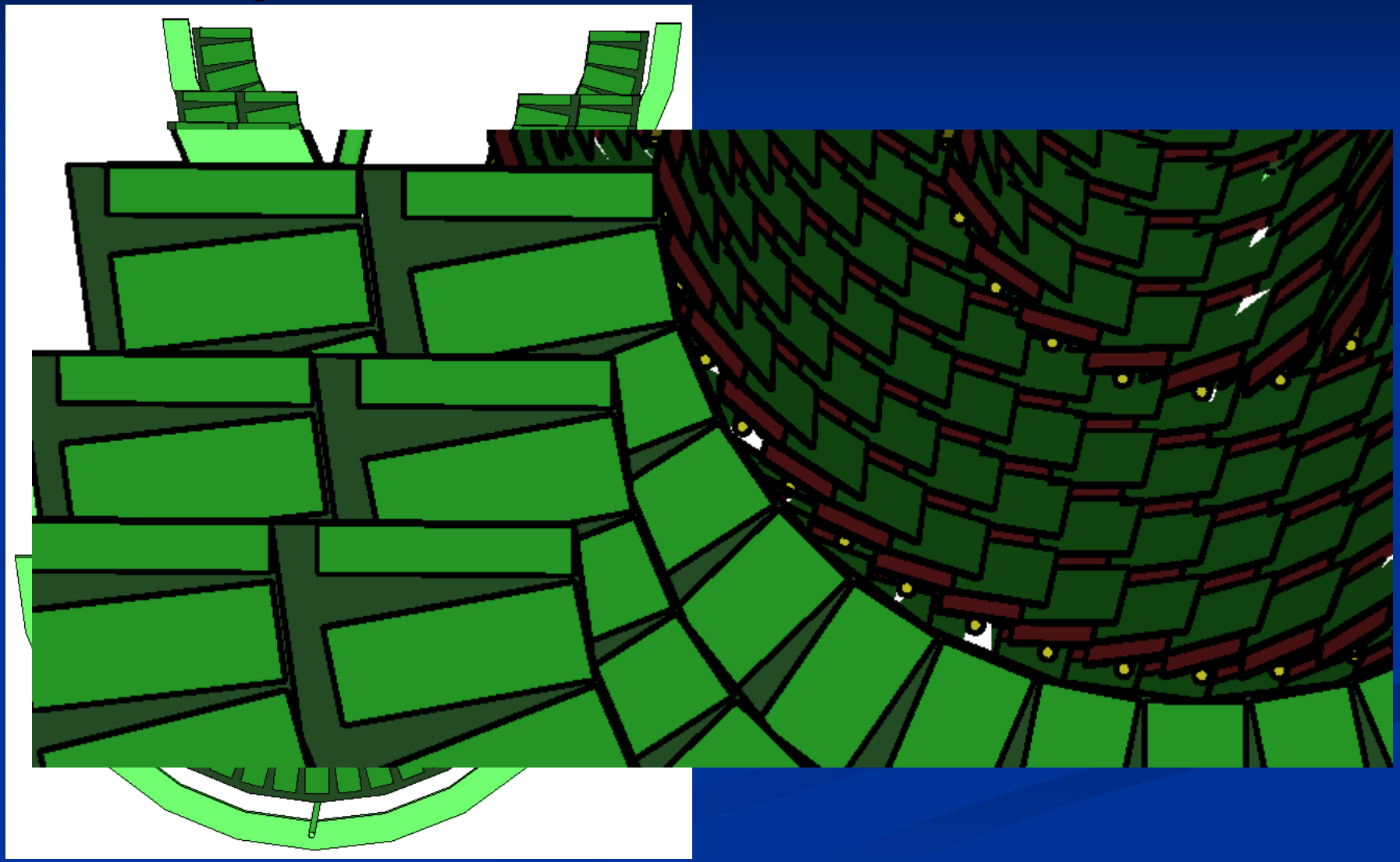
DAWN Examples





DAWN makes True Vector PostScript

- So when you zoom in with your PostScript browser, the images retain high resolution



DAWNCUT and DAVID

- A standalone program, DAWNCUT, can perform a planar cut on a DAWN image.
  - DAWNCUT takes as input a .prim file and some cut parameters. Its output is a new .prim file to which the cut has been applied.
- Another standalone program, DAVID, can show you any volume overlap errors in your geometry.
  - DAVID takes as input a .prim file and outputs a new .prim file in which overlapping volumes have been highlighted.
- Details at http://geant4.kek.jp/~tanaka/

HepRep and DAWN work through Files

- With HepRep and DAWN, Geant4 creates a file:

Example .heprep File

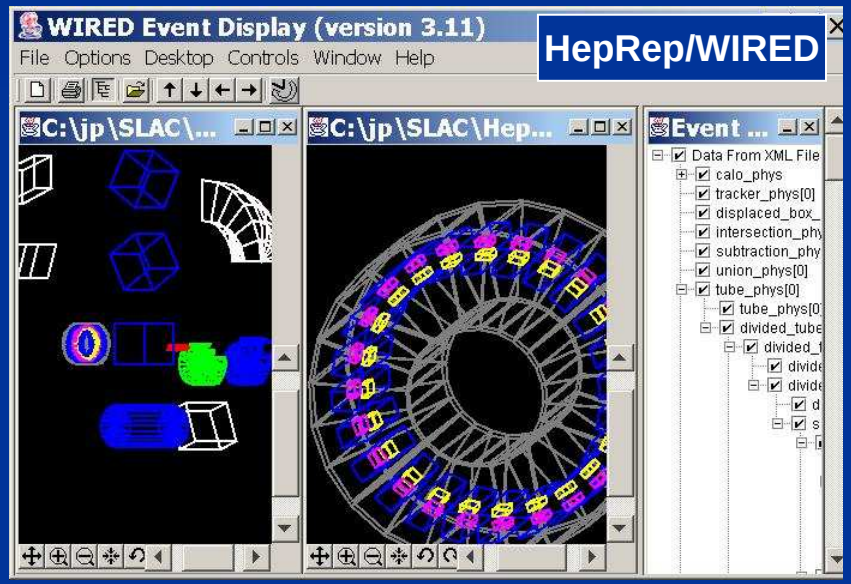
```
<heprep xmlns="http://www.freehep.org/HepRep"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="HepRep.xsd">
 <layerorder="Detector, Event, CalHit, Trajectory, TrajectoryPoint, Hit"/>
 <typetree name="G4GeometryTypes" version="1.0">
  <type name="Detector">
   <attvalue name="Layer" showLabel="NONE" type="String" value="Detector"/>
   <attdef category="Physics" desc="Logical Volume" extra="" name="LVol"/>
   <attdef category="Physics" desc="Material Name" extra="" name="Material"/>
   <type name="Detector/World">
    <type name="Detector/World/Calorimeter">
     <type name="Detector/World/Calorimeter/Layer">
      <type name="Detector/World/Calorimeter/Layer/Lead">
      </type>
     </type>
    </type>
   </type>
  </type>
 </typetree>
 <typetree name="G4EventTypes" version="1.0">
  <type name="Event">
   <attvalue name="Layer" showLabel="NONE" type="String" value="Event"/>
   <type name="Event/Trajectory">
```

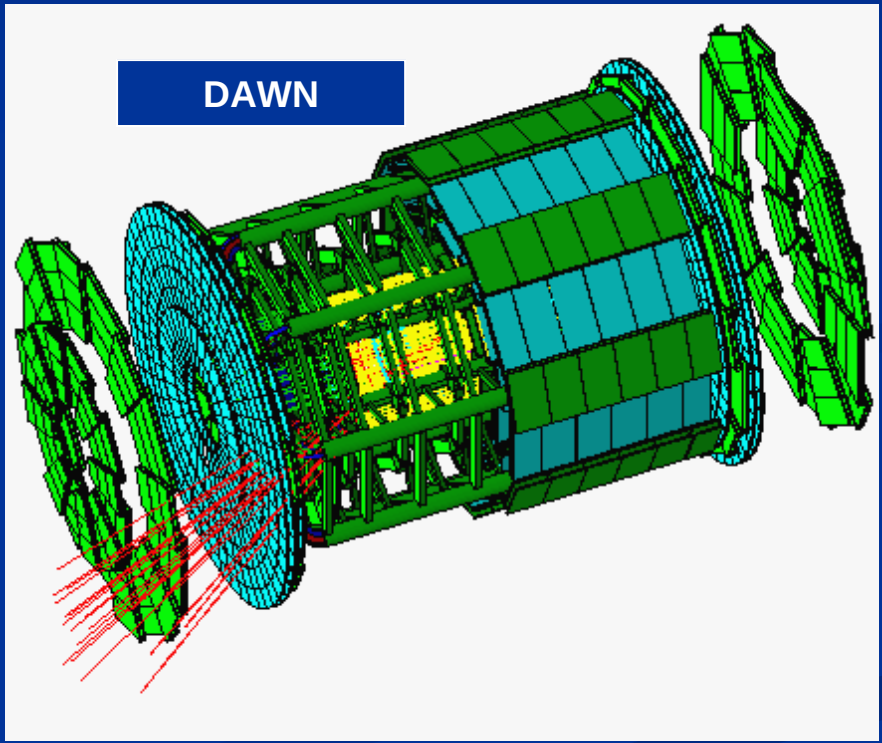
Example .prim File

```
##G4.PRIM-FORMAT-2.4

###### List of primitives 1 ######
/BoundingBox -1.0  -1.0 -5.0   8.0  4.0  6.0
!SetCamera
!OpenDevice
!BeginModeling

# Box
/Origin   0.0  0.0  0.0
/ColorRGB 1.0  0.0  0.0
/Box 0.5  2.0  4.5

# Column
/Origin   4.0  0.0  0.0
/ColorRGB 0.0  1.0  0.0
/Ndiv 50
/Column 1.5  2.0

# Trd
/Origin   0.0  0.0  0.0
/ColorRGB 0.0  1.0  1.0
/Origin   7.0  0.0  0.0
/Trd      1 0.5 1 0.5 4

# Cone segment
/Origin   1.0  5.0  0.0
/ColorRGB 0.0  1.0  1.0
```

HepRep and DAWN work through Files

- And you then run an application to visualize that file:





HepRep and DAWN: complimentary file formats, each with its own strengths

HepRep	DAWN
Hierarchical	Flat
Simple Primitives	All Geant4 Primitives
General Purpose	Just for Geant4
Representables have Attributes	No Attributes
No Camera or Lighting Information	Camera and Lighting Information

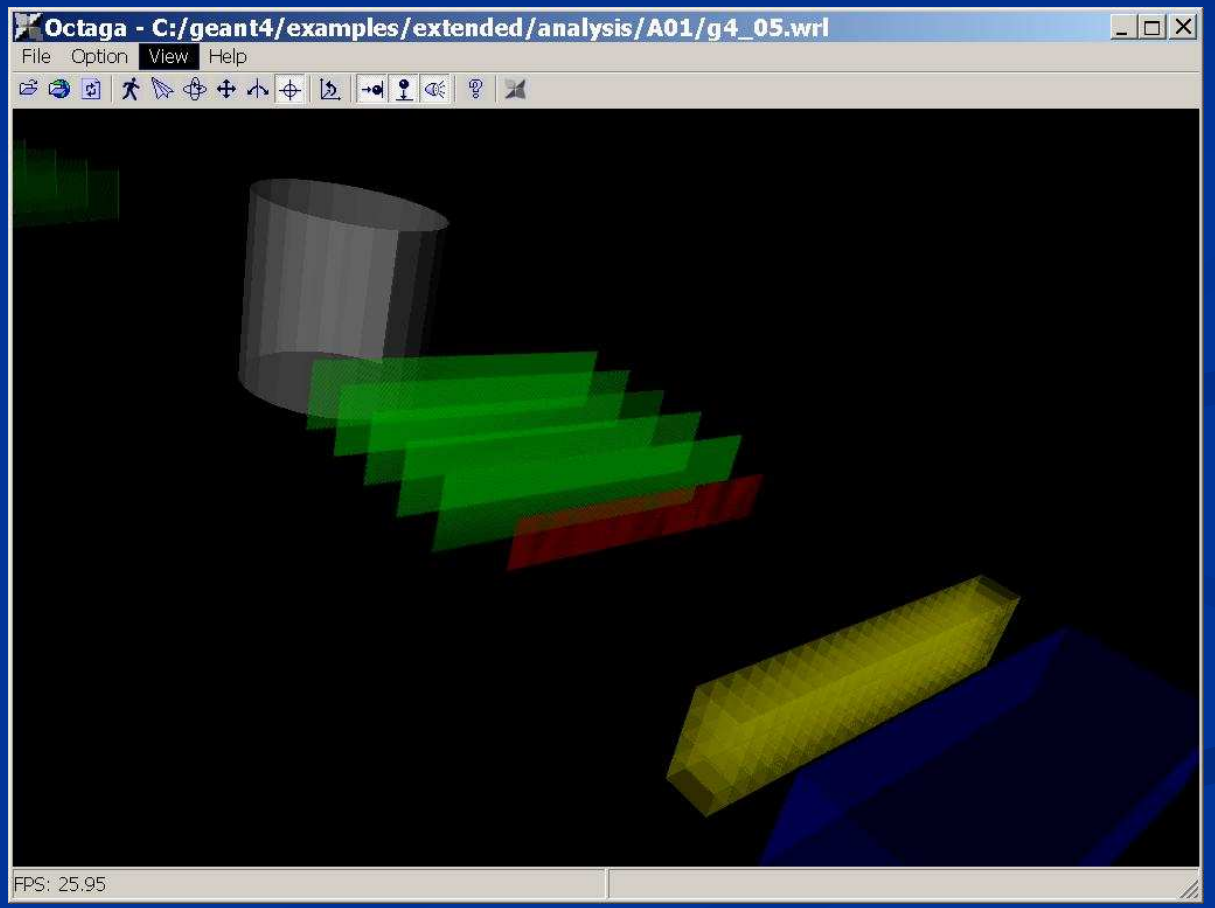
- Plans to eventually use HepRep/WIRED and DAWN together
  - use WIRED to select view (rotate, translate, zoom, pick to understand data), then when view selected, have WIRED call DAWN to render to photorealistic vector postscript

VRML

- Geant4 creates a .vrml file (vrml version 1 or 2)
  - /vis/open VRMLFILE1 or /vis/open VRMLFILE2
- View file in a VRML Browser

VRML

- Geant4 creates VRML File
  - /vis/open VRML1FILE or /vis/open VRML2FILE
- View file in a VRML Browser
  - Many free options, for example, here is one from octaga.com



RayTracer

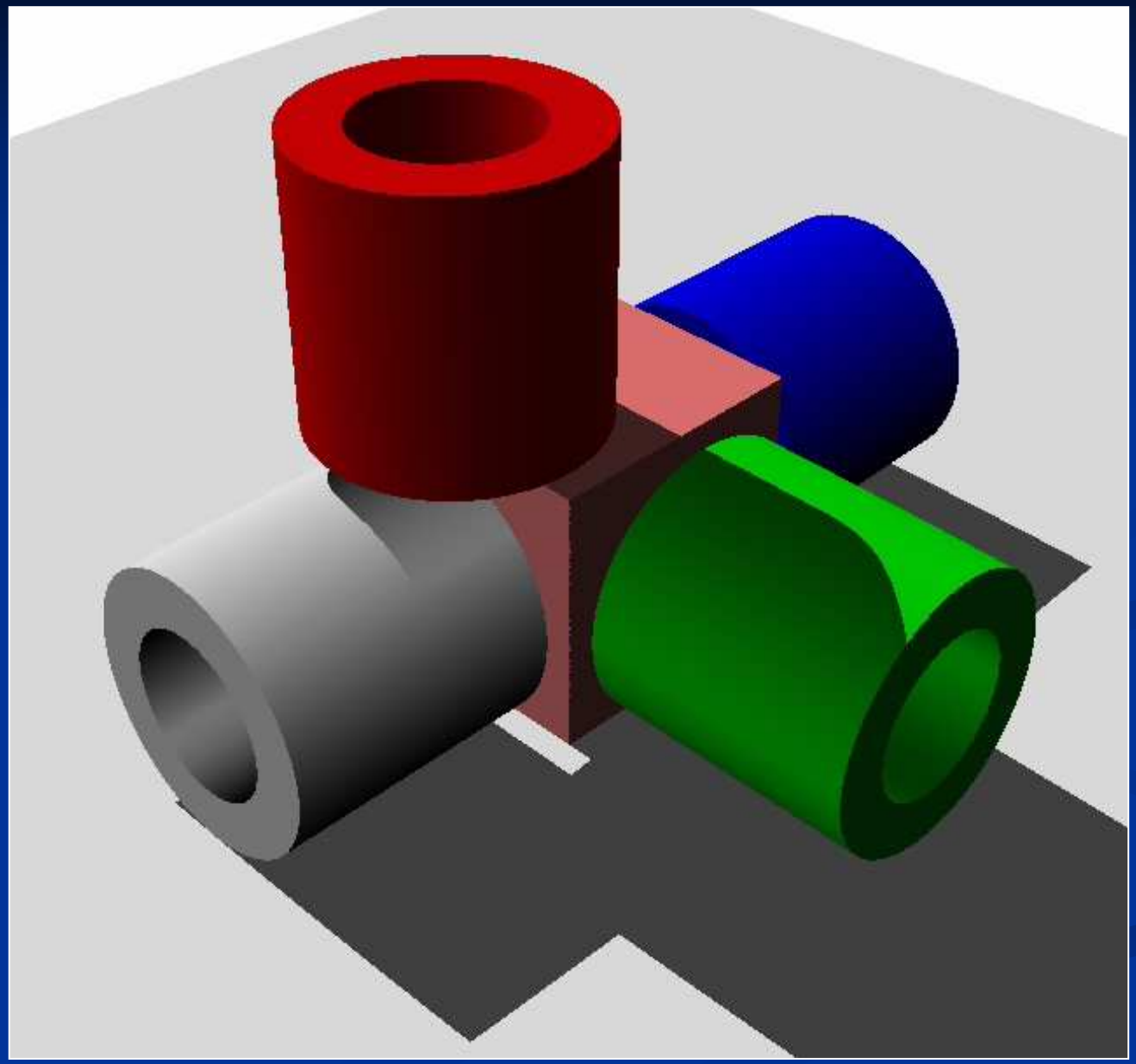
- Run directly from Geant4
  - /vis/open RayTracer
- Creates a jpeg file

- The next Geant4 release (Geant4.8.0) will add RayTracerX
  - /vis/open RayTracerX
  - Simultaneously renders to screen and to jpeg file, so that you can watch as the rendering grows progressively smoother

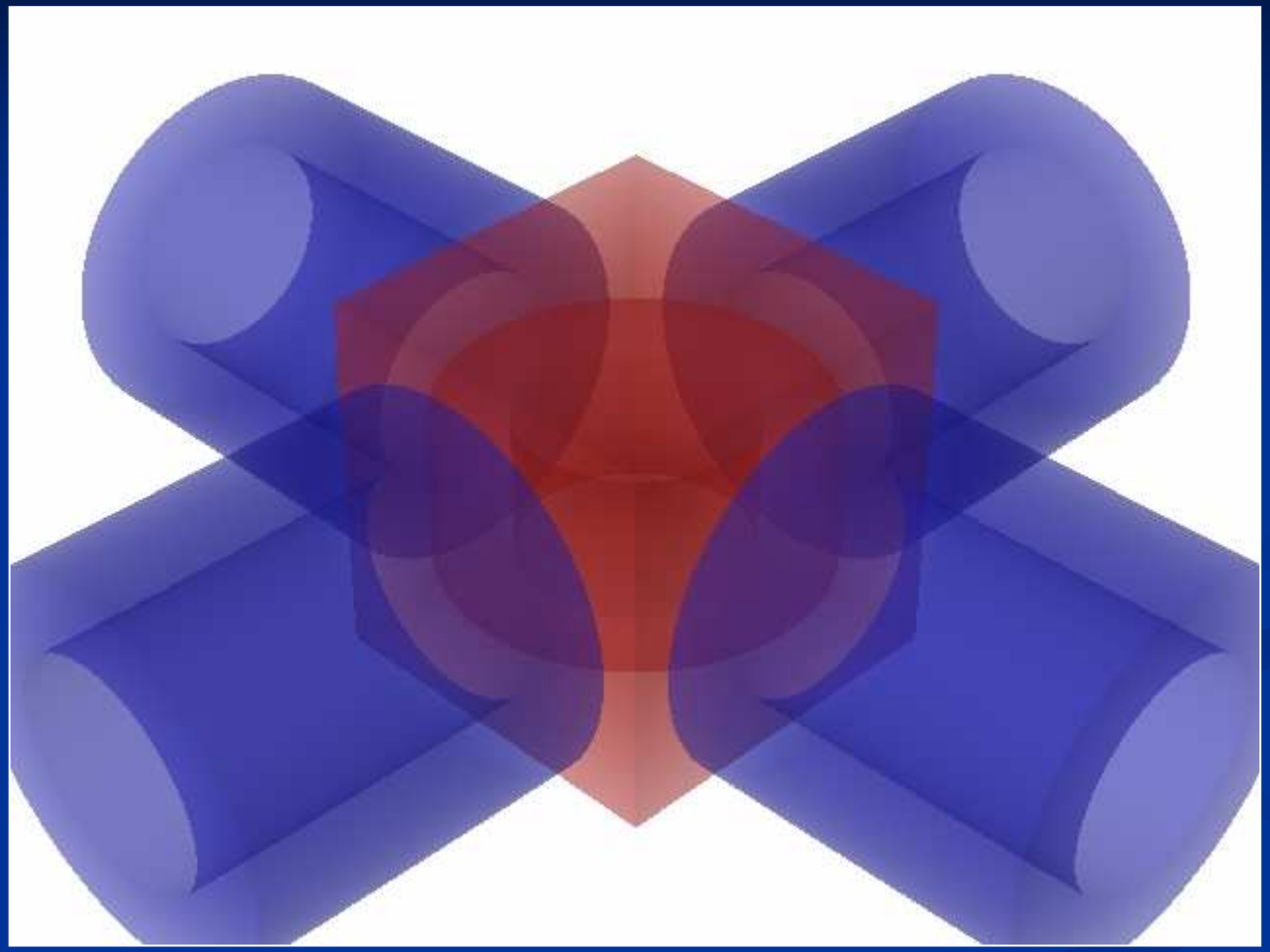
RayTracer

- RayTracer works by using Geant4's own tracking to shoot photons through the detector onto a sensitive plane. The resulting image is presented as a jpeg file.
  - /vis/open RayTracer
- Some pieces of geometries may fail to show up in other visualization drivers (due to algorithms those drivers use to compute visualizable shapes and polygons), but RayTracer can handle any geometry that the Geant4 navigator can handle.
- RayTracer can not be used to visualize Trajectories.
- Commands:
  - 1) trace * Start the ray tracing.
  - 2) column * Define the number of horizontal pixels.
  - 3) row * Define the number of vertical pixels.
  - 4) target * Define the center position of the target.
  - 5) eyePosition * Define the eye position.
  - 6) lightDirection * Define the direction of illumination light.
  - 7) span * Define the angle per 100 pixels.
  - 8) headAngle * Define the head direction.
  - 9) attenuation * Define the attenuation length for transparent material.
  - 10) distortion * Distortion effect of the fish eye lens.
  - 11) ignoreTransparency * Ignore transparency even if the alpha of G4Colour < 1
  - 12) backgroundColour * Set background colour: red green blue: range 0.->1.

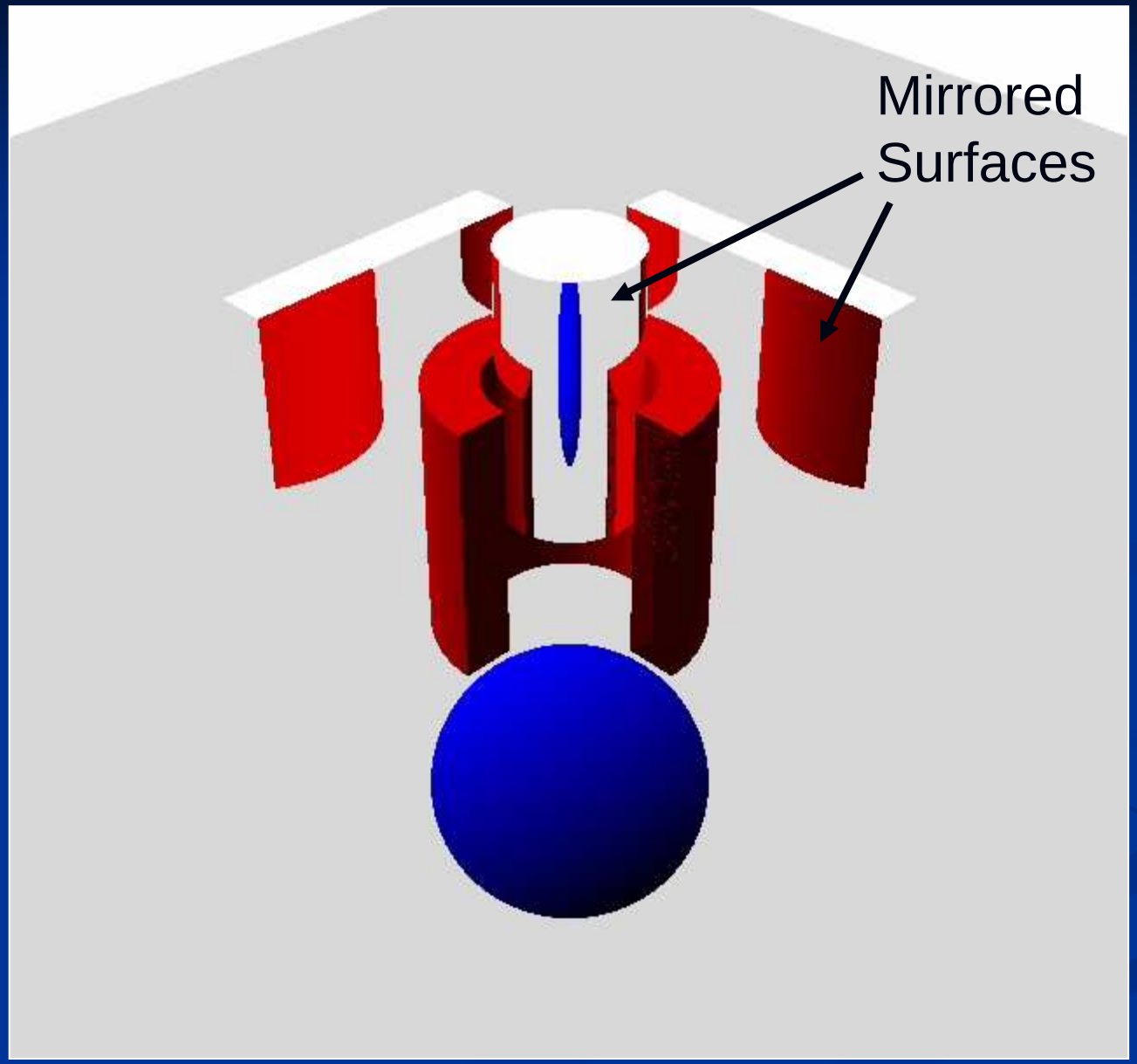
RayTracer Shows Shadows



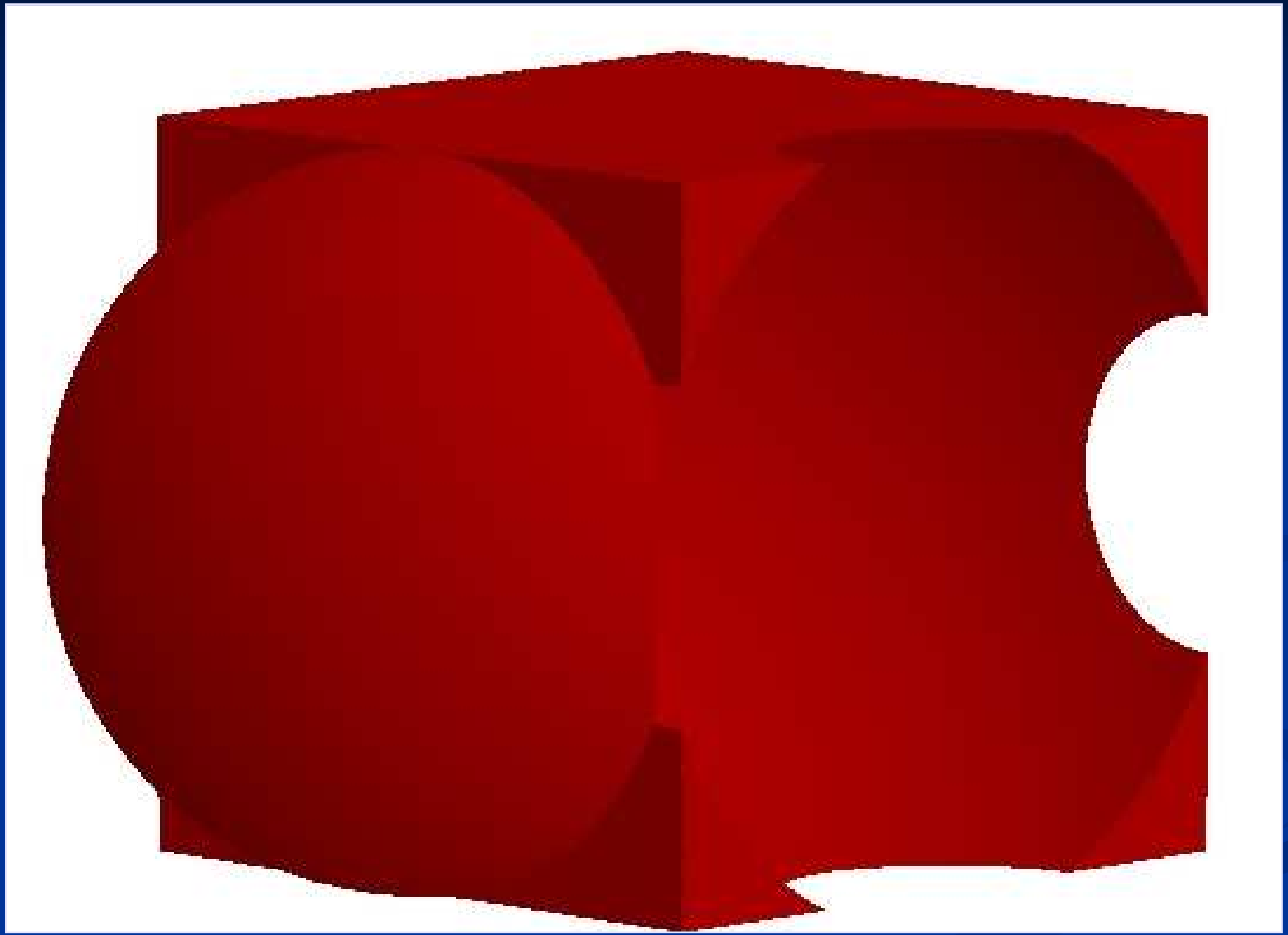
RayTracer Supports Transparency



RayTracer Handles Mirrored Surfaces

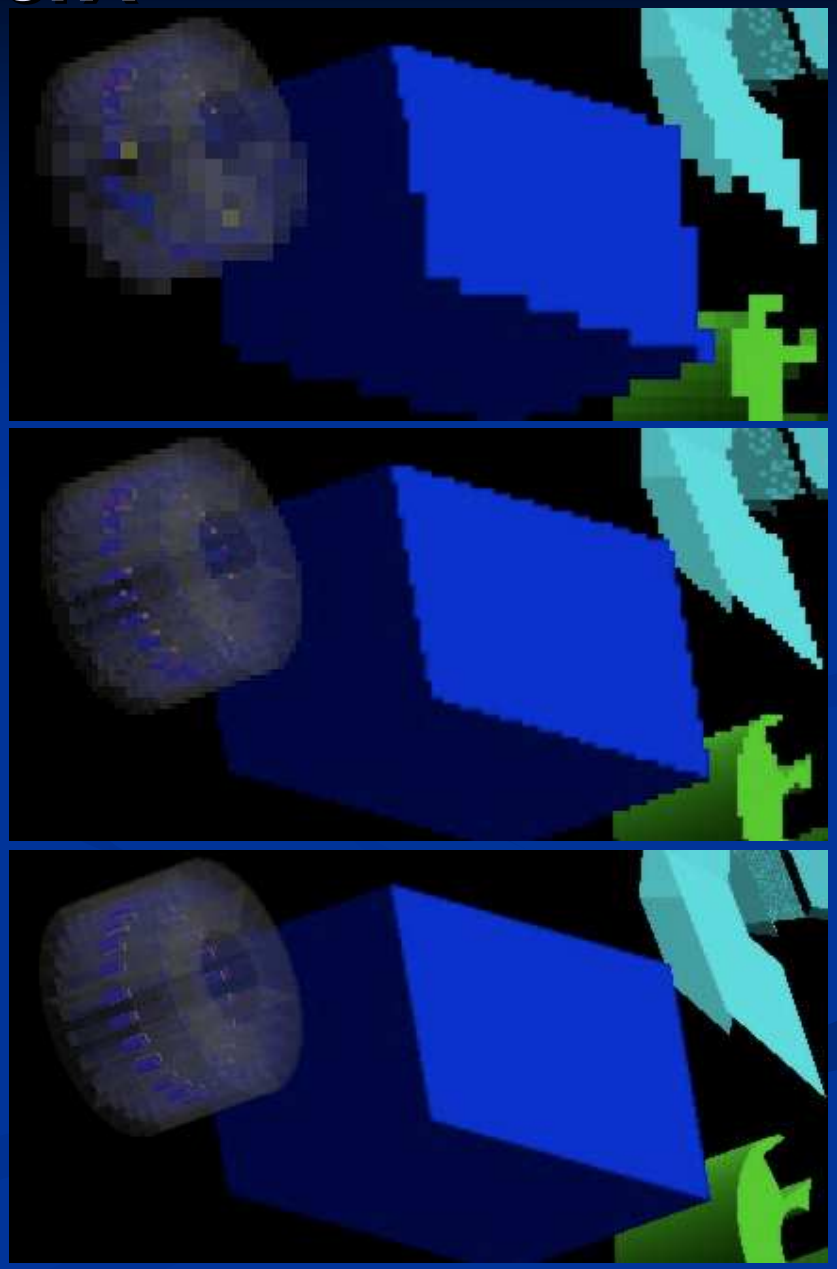


RayTracer Handles Boolean Solids



RayTracerX

- Coming in next release (Geant4.8.0)
- In addition to
  - /vis/open RayTracer
- You will have the option of
  - /vis/open RayTracerX



- Builds same jpeg file as RayTracer, but simultaneously renders to screen so you can watch as rendering grows progressively smoother.
- Means you can abort and retry the rendering with different view parameters without having to wait for the complete refinement of the image.

ASCIITree

- Run directly from Geant4
  - /vis/open ATree

ASCIITree

- ASCIITREE is a visualization driver that is not actually graphical, but that dumps the hierarchy as a simple text tree.
  - /vis/open ATree
- /vis/viewer/flush
  - "worldPhysical":0
  - "magneticPhysical":0
  - "firstArmPhysical":0
  - "hodoscope1Physical":0
  - "hodoscope1Physical":1 (repeated placement)
  - "hodoscope1Physical":2 (repeated placement)
  - "hodoscope1Physical":3 (repeated placement)
  - "hodoscope1Physical":4 (repeated placement)
- Can be set to various levels of detail
  - /vis/ASCIITree/verbose <verbosity>
  - 0: prints physical volume name.
  - 1: prints logical volume name.
  - 2: prints solid name and type.
  - 3: prints volume and density of solid.
  - 4: calculates and prints mass(es) of volume(s) in scene.
  - By default, shows only daughters of first placement and not repeat replicas.
  - Add 10 to the above to also show repeated placements and replicas.

ASCIITree: Calculate Volume and Mass

- At verbosity level 4, ASCIITree calculates the mass of the complete geometry tree taking into account daughters up to the depth specified for each physical volume.
- The calculation involves subtracting the mass of that part of the mother that is occupied by each daughter and then adding the mass of the daughter, and so on down the hierarchy.
- /vis/ASCIITree/Verbose 4
- /vis/viewer/flush
- "HadCalorimeterPhysical":0 / "HadCalorimeterLogical" / "HadCalorimeterBox"(G4Box), 1.8 m3 , 11.35 g/cm3
  - "HadCalColumnPhysical":-1 (10 replicas) / "HadCalColumnLogical" / "HadCalColumnBox"(G4Box), 180000 cm3, 11.35 g/cm3
    - "HadCalCellPhysical":-1 (2 replicas) / "HadCalCellLogical" / "HadCalCellBox"(G4Box), 90000 cm3, 11.35 g/cm3
      - "HadCalLayerPhysical":-1 (20 replicas) / "HadCalLayerLogical" / "HadCalLayerBox"(G4Box), 4500 cm3, 11.35 g/cm3
        - "HadCalScintiPhysical":0 / "HadCalScintiLogical" / "HadCalScintiBox"(G4Box), 900 cm3, 1.032 g/cm3
- Calculating mass(es)...
  - Overall volume of "worldPhysical":0, is 2400 m3
  - Mass of tree to unlimited depth is 22260.5 kg

Part 3: How to Run Geant4 Visualization

- Environment Variables
- Commands

Environment Variables

- Five of the visualization drivers discussed here are always included by default in Geant4 (since they require no external libraries):
  - HepRepFile
  - DAWNFILE
  - VRMLFILE
  - RayTracer
  - ASCIITree
- Other visualization drivers may require setting environment variables:
  - OpenGL
    - Before you build Geant4, set the appropriate “build” variable to 1 (causes the necessary code to be linked into your executable):
      - setenv G4VIS_BUILD_OPENGLX_DRIVER 1
    - Before you run Geant4, set the corresponding “use” variable to 1. (Geant4 separates the BUILD and USE variables so that you can BUILD in drivers that you might not necessarily want to USE during some executions):
      - setenv G4VIS_USE_OPENGLX 1
    - Note that you cannot run JAIDA/JAS if OpenGL is in your build.
      - the OpenGL libraries pre-load the library libXt.so which makes the Java virtual machine crash when it tries to open its first Window.
  - Similarly for OpenInventor or the upcoming RayTracerX

Visualization Commands

- Create an empty scene:
  - /vis/scene/create
- Open a visualization driver, such as:
  - /vis/open HepRepFile
- If using an immediate viewer, such as OpenGL, set camera parameters and drawing style (wireframe/surface), such as:
  - /vis/viewer/set/style wireframe
  - /vis/viewer/set/viewpointThetaPhi 70 20
- Declare what data should be added to the scene (default is to just add full set of detector volumes)
  - /vis/scene/add/trajectories
  - /vis/scene/add/hits
- Run simulation with appropriate options to store trajectory information:
  - /tracking/storeTrajectory 1
  - /run/beamOn 1
- Execute the visualization (done automatically with each /run/beamOn, but needed if you want to output geometry without running an event):
  - /vis/viewer/flush
- If using an external viewer, such as for HepRepFile or DAWNFILE:
  - import the .heprep or .prim file into WIRED or DAWN, set camera parameters, drawing style, etc., view the visualization

Examples Visualization Command Sequences

- Visualize a detector in OpenGL:
  - /vis/scene/create
  - /vis/open OGLIX
  - /vis/viewer/flush
- Visualize trajectories and hits for 10 events using HepRep/WIRED
  - /vis/scene/create
  - /vis/open HepRepFile
  - /vis/scene/add/trajectories
  - /vis/scene/add/hits
  - /tracking/storeTrajectory 1
  - /run/beamOn 10

Command Guidance

- Complete guidance on all commands is available from the command line:
  - Idle> help
  - Command directory path : /
  - Sub-directories :
  - 1) /control/ UI control commands.
  - 2) /units/ Available units.
  - 3) /geometry/ Geometry control commands.
  - 4) /tracking/ TrackingManager and SteppingManager control commands.
  - 5) /event/ EventManager control commands.
  - 6) /run/ Run control commands.
  - 7) /random/ Random number status control commands.
  - 8) /particle/ Particle control commands.
  - 9) /process/ Process Table control commands.
  - 10) /vis/ Visualization commands.
  - 11) /mydet/ A01 detector setup control commands.
  - 12) /hits/ Sensitive detectors and Hits
  - 13) /gun/ Particle Gun control commands.
  - Commands :
  - Type the number (0:end, -n:n level back) :

Guidance Detail

- Guidance is hierarchical, providing full detail on all commands.
  - Sub-directories :
  - 1) /vis/ASCIITree/ Commands for ASCIITree control.
  - 2) /vis/GAGTree/ Commands for GAGTree control.
  - 3) /vis/heprep/ HepRep commands.
  - 4) /vis/rayTracer/ RayTracer commands.
  - 5) /vis/scene/ Operations on Geant4 scenes.
  - 6) /vis/sceneHandler/ Operations on Geant4 scene handlers.
  - 7) /vis/viewer/ Operations on Geant4 viewers.
  - Commands :
  - 8) enable * Enables/disables visualization system.
  - 9) disable * Disables visualization system.
  - 10) verbose * Simple graded message scheme - digit or string (1st character defines):
  - 11) drawTree * (DTREE) Creates a scene consisting of this physical volume and produces a representation of the geometry hierarchy.
  - 12) drawView * Draw view from this angle, etc.
  - 13) drawVolume * Creates a scene consisting of this physical volume and asks the current viewer to draw it.
  - 14) open * Creates a scene handler ready for drawing.
  - 15) specify * Draws logical volume with Boolean components, voxels and readout geometry.

- Command /vis/open
- Creates a scene handler ready for drawing.
- The scene handler becomes current (the name is auto-generated).
- Parameter : graphics-system-name
  - Parameter type : s
  - Omittable : False
  - Candidates : ATree DAWNFILE GAGTree HepRepXML HepRepFile RayTracer VRML1FILE VRML2FILE OGLIX OGLSX
- Parameter : window-size-hint
- pixels
  - Parameter type : i
  - Omittable : True
  - Default value : 600

Details of the /vis/open Command

- To Open a Driver
  - /vis/open <driver name>
- for example
  - /vis/open OGLIX
  - /vis/open HepRepFile
  - /vis/open DAWNFILE
- The set of available drivers is listed when you first start Geant4, but you can also get this list with the command:
  - help /vis/open
- You can even open more than one driver at a time, but this requires storing and replaying the random seed for the given event.

Details of the /vis/viewer/… Commands

- To Set Camera Parameters and Drawing Style.
  - Only needed if using an immediate viewer, such as OpenGL
  - For HepRepFile or DAWNFILE, these sorts of adjustments are made later, in the WIRED or DAWN viewer programs
- Reset viewpoint
  - /vis/viewer/reset
- Set view angles
  - /vis/viewer/set/viewpointThetaPhi <theta_angle> <phi_angle>
  - for example
    - /vis/viewer/set/viewpointThetaPhi 70 20
- Set drawing style
  - /vis/viewer/set/style <style>
  - for example
    - /vis/viewer/set/style wireframe
    - /vis/veiwer/set/style surface
  - but note that this will not affect volumes that have style explicitly forced by “setForceWireframe” or “setForceSolid” commands in the c++ code
- Zoom
  - /vis/viewer/zoom <scale factor>
  - for example
    - /vis/viewer/zoom 2.
- Coming in next Geant4 release (Geant4.8.0), some drivers will support different zoom along different axes (e.g., zoom more in X and Y but not in Z). New command will be
  - /vis/viewer/scale <3 vector of scale factors>

Controlling Detail Level of Detector Geometry

- By default, Geant4 will draw the entire detector geometry.
  This is equivalent to the command:
  - /vis/scene/add/volume world
- You can specify additional arguments to limit the amount of geometry detail shown:
  - /vis/scene/add/volume [<physical-volume-name>] [<copy-no>] [<depth-of-descending>]
    - 1st parameter: volume name
      (default "world").
    - 2nd parameter: copy number
      (default -1 meaning first occurrence of physical-volume-name is selected.
    - 3rd parameter: depth of descending geometry hierarchy
      (default G4Scene::UNLIMITED (-1)).
- Still more arguments can be given to specify a clipping volume.
  - vis/scene/add/volume world -1 -1 box km 0 1 0 1 0 1
    will draw the world with the positive octant cut away.
- Additional commands allow finer control including whether or not to draw Boolean components, voxels and readout geometries:
  - /vis/specify <logical-volume-name> [depth-of-descent] [<booleans-flag>] [<voxels-flag>] [<readout-flag>]
  - /vis/scene/add/logicalVolume <logical-volume-name> [<depth-of-descending>] [<voxels-flag>] [<readout-flag>]

Details of Visualizing Trajectories and Hits

- To add trajectories or hits to the scene
  - /vis/scene/add/trajectories
  - /vis/scene/add/hits
- But also, you must tell tracking to make trajectories available for drawing
  - /tracking/storeTrajectory 1
  - Otherwise, in the interests of saving memory, trajectory information is deleted before the visualization system gets a chance to handle it.
  - From next Geant4 release (Geant4.8.0), this command will be done for you automatically any time you do /vis/scene/add/trajectories.
- Run using the command
  - /run/beamOn
- If you place a number after beamOn, the run will go for that many events
  - /run/beamOn 10

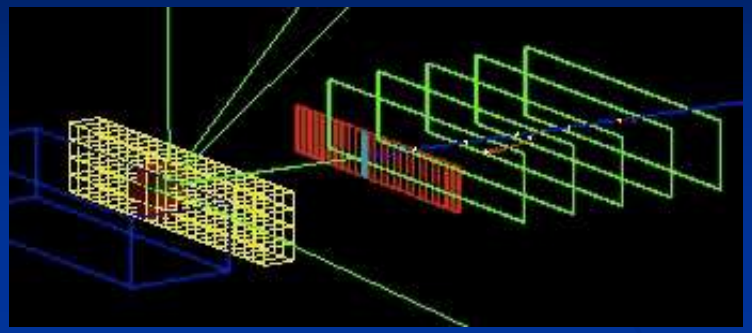
Accumulating Trajectories and Hits

- By default, you will get a drawing after each event. To instead get just one drawing with all of the accumulated events from that run
  - /vis/scene/endOfEventAction accumulate
- This overrides the default
  - /vis/scene/endOfEventAction refresh

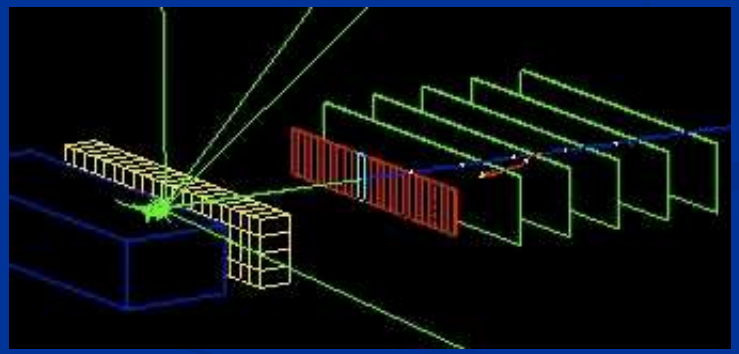
- To even suppress that one drawing from the end of the /run/beamOn, use
  - /vis/scene/endOfRunAction accumulate
- This overrides the default
  - /vis/scene/endOfRunAction refresh
- When you actually want to draw, you then have to explicitly issue the command
  - /vis/viewer/flush

Hidden Line Removal

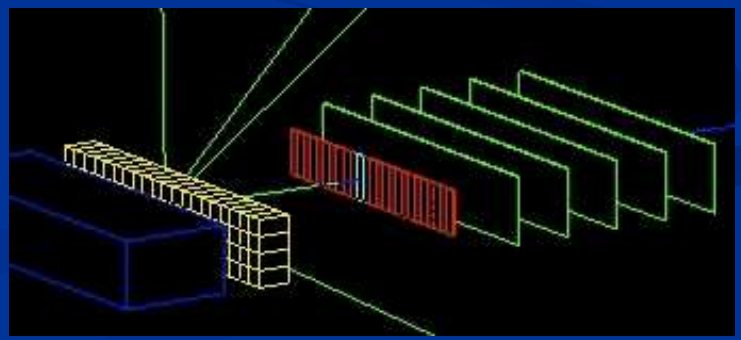
- Some viewers such as OpenGL support hidden line removal.
- You can control whether this removal is done and whether trajectories and hits are affected by this feature.
- By default, hidden line removal is disabled



- To turn on hidden line removal
  - /vis/viewer/set/hiddenEdge 1
- This hides edges of geometry, but lets trajectories through.



- To hide trajectories and hits as well
  - /vis/viewer/set/hiddenMarker 1



Compound Commands

- To allow you to work quickly, Geant4 visualization lets you issue the equivalent of several common commands at one time by using a “compound command”.
- Some of the commands you have already seen in this presentation are actually compound commands:
  - /vis/open
    - /vis/sceneHandler/create
    - /vis/viewer/create
  - /vis/viewer/flush
    - /vis/veiwer/refresh
    - /vis/viewer/update
- Another commonly used compound commands is:
  - /vis/drawVolume
    - /vis/scene/create
    - /vis/scene/add/volume

Enhanced Trajectory Drawing

- Coming in the next Geant4 release (Geant4.8.0), you will be able to interactively change how trajectories are modeled.
- Initial options will let you select to either color trajectories by charge, or by particle ID.
  - /vis/model/trajectories/drawByCharge
  - /vis/model/trajectories/drawByParticleID
  - And then additional commands to set specific colors
- Then, either in release Geant4.8.0 or soon thereafter, there will be many more options for trajectory modeling, such as:
  - drawByMomentum
    - where color tells you a particle's momentum
  - drawByOriginLogicalVolume
    - where color tells you where a particle originated
  - drawByOriginInteraction
    - where color tells you what interaction created the particle
  - cutByMomentum
    - where you can assign a momentum cut on what trajectories should be shown
  - and so forth
- Already in Geant4.8.0 will be a mechanism to let you define your own "drawBy…" class and then invoke it from the command line.

Geant4 Visualization in Standalone Mode

- The Geant4 Visualization system can be used on its own - without the rest of Geant4.
- Build something “by hand” from the Geant4 geometry primitives and placement apparatus, but without any of the main parts of Geant4 such as detector construction, run manager or physics list.
- Still preserves all of the interactive apparatus of the visualization system.
- From next Geant4 release (Geant4.8.0) there will be an example: /examples/extended/visualization/standalone

```
// Simple box...
    pVisManager->Draw(      G4Box("box",2*m,2*m,2*m),
                                    G4VisAttributes(G4Colour(1,1,0)));

// Boolean solid...
    G4Box boxA("boxA",3*m,3*m,3*m);
    G4Box boxB("boxB",1*m,1*m,1*m);
    G4SubtractionSolid subtracted(      "subtracted_boxes",&boxA,&boxB,
                                        G4Translate3D(3*m,3*m,3*m));
    pVisManager->Draw(      subtracted,
                            G4VisAttributes(G4Colour(0,1,1)),
                                    G4Translate3D(-6*m,-6*m,-6*m));
```

Complete List of Commands

- This presentation has shown only a very small subset of the full Geant4 visualization command set. Even for those commands shown, only a few of the options have been presented.
- Each visualization driver may have its own set of additional commands.
- To see the complete set of commands, use the interactive command guidance (i.e., type help and then type the appropriate number for “vis”).
- Or see the extensive Geant4 documentation on the web:
  - http://cern.ch/geant4/G4UsersDocuments/UsersGuides/ForApplicationDeveloper/html/Visualization/UIcommands/vis.txt

Summary

- Choose a driver based on your current needs:
  - If you want quick photorealistic graphics with GUI control (and have the OpenGL installed), OpenGL is a good solution.
  - If you want photorealistic graphics plus more interactivity (and have the OpenInventor libraries installed), OpenInventor is a good solution.
  - If a wireframe look will do, but you still want GUI control and want to be able to pick on items to inquire about them (identity, momentum, etc.), HepRep/WIRED will meet your needs.
  - If you want to render highest quality photorealistic images for use in a poster or a technical design report, and you can live without quick rotate and zoom, DAWN is the way to go.
  - If you want to render to a 3D format that others can view in a variety of commodity browsers (including some web browsers), VRML is the way to go.
  - If you want to visualize a geometry that the other visualization drivers can't handle, or you need transparency or mirrors, and you don't need to visualize trajectories, RayTracer will do it.
  - If you just want to quickly check the geometry hierarchy, or if you want to calculate the volume or mass of any geometry hierarchy, ASCIITree will meet your needs.
  - Advanced developers can even create their own visualization drivers.
- Some drivers are always present, others require setting of environment variables (since they require external libraries).
- Explore the online command guidance (help) to learn the extensive set of visualization commands.

Further Resources

Geant4 Tutorial CD:

- http://geant4.slac.stanford.edu/g4cd/March2004

  In particular:
  Hands on WIRED Tutorial
  - http://geant4.slac.stanford.edu/g4cd/March2004/Documentation/Visualization/G4WIREDTutorial/G4WIREDTutorial.html

  Hands on DAWN Tutorial
  - http://geant4.slac.stanford.edu/g4cd/March2004/Documentation/Visualization/G4DAWNTutorial/G4DAWNTutorial.html

  Hands on OpenGL Tutorial
  - http://geant4.slac.stanford.edu/g4cd/March2004/Documentation/Visualization/G4OpenGLTutorial/G4OpenGLTutorial.html

On-line Documentation on Geant4 Visualization:

- http://cern.ch/geant4/G4UsersDocuments/UsersGuides/ForApplicationDeveloper/html/Visualization

List of Visualization Commands:

- http://cern.ch/geant4/G4UsersDocuments/UsersGuides/ForApplicationDeveloper/html/Visualization/UIcommands/vis.txt

Another Presentation that Introduces Visualization,
with More Focus on Controlling Visualization from C++:

- http://www.ge.infn.it/geant4/training/portland/visualisation.pdf

For Questions or Comments: Geant4 Visualization Online Forum:

- http://geant4-hn.slac.stanford.edu:5090/HyperNews/public/get/visualization.html

References

- OpenScientist Home Page
  http://openscientist.lal.in2p3.fr
- HepRep: a generic interface definition for HEP event display representables
  http://www.slac.stanford.edu/~perl/heprep
- Fred: oh no, another event display (a HepRep client)
  http://www.fisica.uniud.it/~glast/FRED
- WIRED3 HepRep Browser
  http://www.slac.stanford.edu/BFROOT/www/Computing/Graphics/Wired
- DAWN Hot Information
  http://geant4.kek.jp/Geant4/vis
- DAWN Home Page
  http://geant4.kek.jp/~tanaka/DAWN/About_DAWN.html
- DAWNCUT Home Page
  http://geant4.kek.jp/~tanaka/DAWN/About_DAWNCUT.html
- DAVID Home Page
  http://geant4.kek.jp/~tanaka/DAWN/About_DAVID.html
- Satoshi Tanaka's GEANT4 Ritsumeikan University Group Home Page (more information on DAWN, sample PRIM files, images, etc.)
  http://geant4.kek.jp/~tanaka/