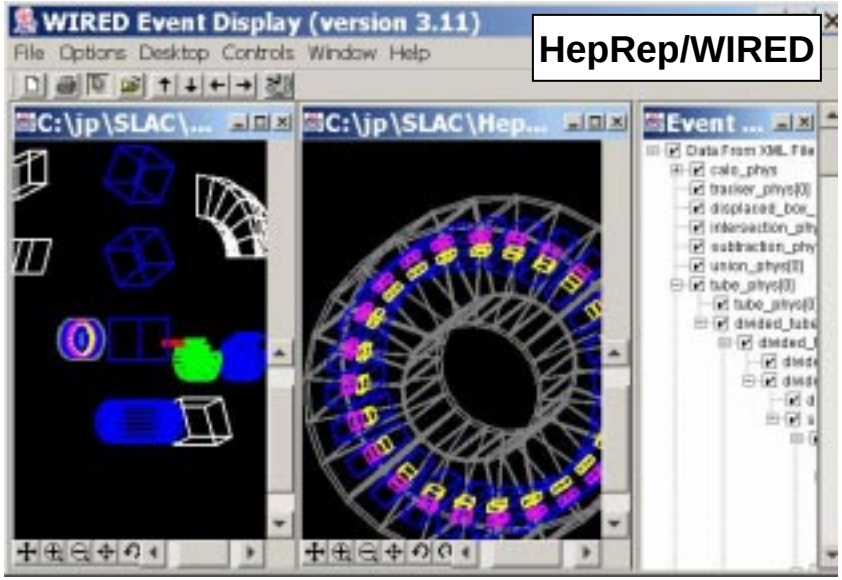
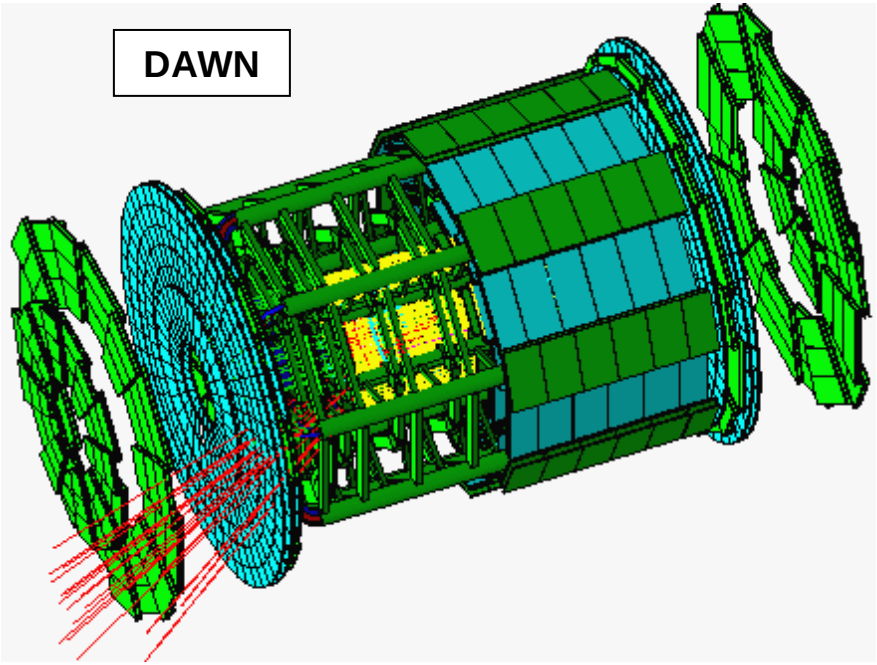
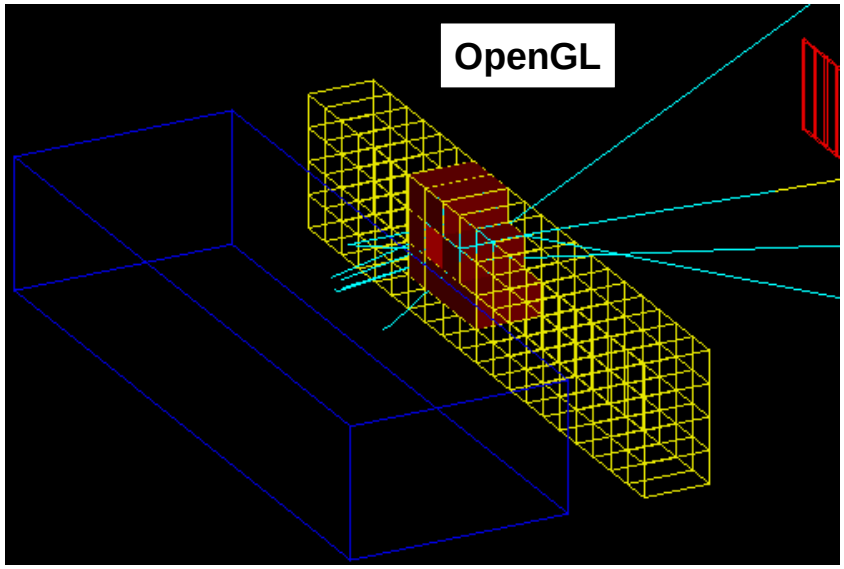
Introduction to Geant4 Visualization

Joseph Perl
SLAC Computing Services







Contents

- **The General Concepts behind Geant4 Visualization**
  - **What you can visualize**
  - **The various visualization drivers**
  - **Visualization attributes**
- **Introduction to the Visualization Commands**
  - **Environment variables**
  - **Commands**

- **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 WIRED Event Display**
  - **Geant4 Workshop Visualization Tutorial using the DAWN Event Display**
  - **Geant4 Workshop Visualization Tutorial using the OpenGL Event Display**

How this Documents Fits in with Other Tutorial Materials

- From here, go on to use three separate hands-on tutorials that give you experience working with three visualization drivers:
  - OpenGL
  - HepRep/WIRED
  - DAWN
- Some other Geant4 visualization drivers are not discussed here simply because the present author is not experienced with them (such as RayTracer, VRML and ASCIITree).

Part 1: The General Concepts behind Geant4 Visualization

- What you can visualize
- The various visualization drivers
- Visualization attributes

Geant4 Visualization serves a Variety of Functions

- Quick response to survey successive events
- Impressive special effects for demonstration
- High-quality output to prepare journal papers
- Flexible camera control for debugging geometry
- Highlighting overlapping of physical volumes
- Interactive picking of visualised objects

What You Can Visualize

- Simulation data can be visualised such as:
  - Detector components
  - A hierarchical structure of physical volumes
  - A piece of physical volume, logical volume, and solid
  - Particle trajectories and tracking steps
  - Hits of particles in detector components

- You can also visualiseother user defined objects such as:
  - A polyline, that is, a set of successive line segments (example: coordinate axes)
  - A marker which marks an arbitrary 3D position (example: eye guides)
  - Texts
    - •character strings for description
    - •comments or titles …

The Various Visualization Drivers

- OpenGL
  - View directly from Geant4
  - Rendered, photorealistic image with some interactive features
    - zoom, rotate, translate
  - Limited printing ability (pixel graphics, not vector graphics)
- HepRepFile
  - View in the WIRED Event Display
  - 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
  - Export to many vector graphic formats (PostScript, PDF, etc.)
- DAWNFILE
  - View in the DAWN Renderer
  - Rendered, photorealistic image
  - Runs on Linux/Unix and Windows
  - No interactive features
  - Highest quality technical rendering - output to vector PostScript

Choose the Driver that Meets your Current Needs

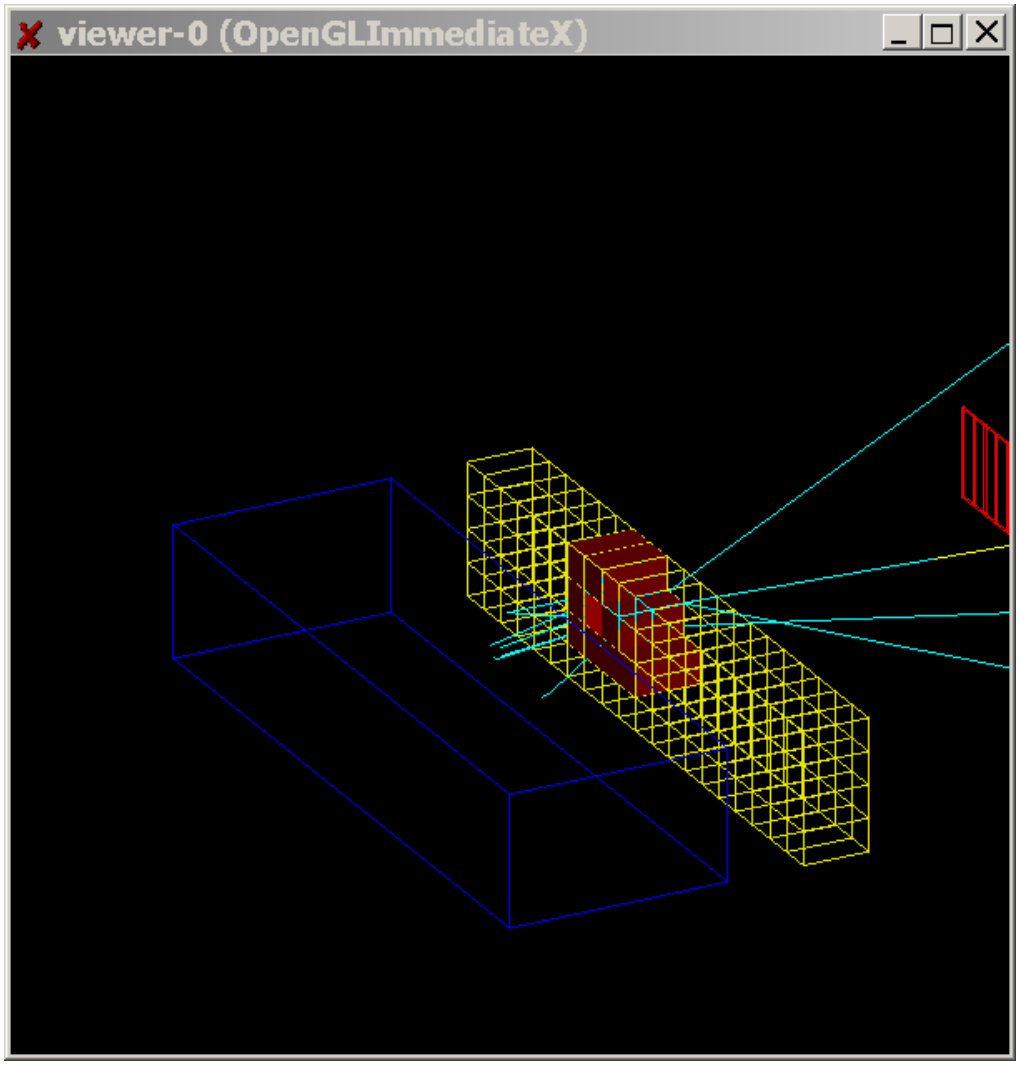
- If you want quick photorealistic graphics with GUI control (and have the necessary libraries installed)
  - OpenGL is a good solution
- If a wireframe look will do, but you still want GUI control and want to be able to pick on items to inquire about them (identity, momentum, etc.)
  - HepRep/WIRED 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

Controlling Visualization

- Your Geant4 code stays basically the same no matter which driver you want to use
  - You can even run all three drivers at the same time.
- Visualisation is performed either with commands or by writing C++ source codes of user-action
  - For the present tutorial, we confine ourselves to command-driven visualization.
- For some visualization drivers all commands go from Geant4
  - OpenGL
- For other visualization drivers, some work is in Geant4, a file is produced, and that file is then rendered by another application (which may have GUI control)
  - HepRep
  - DAWN

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
/vis/viewer/zoom 2
/vis/viewer/reset
/vis/viewer/set/viewpointThetaPhi 40
/vis/viewer/panTo -5 -1
/vis/viewer/zoom 4.
/vis/scene/add/trajectories
/vis/scene/add/hits
/tracking/storeTrajectory 1
/run/beamOn 1
```

HepRep and DAWN work through Files

- **With HepRep and DAWN, Geant4 creates a file:**

Example .heprep File

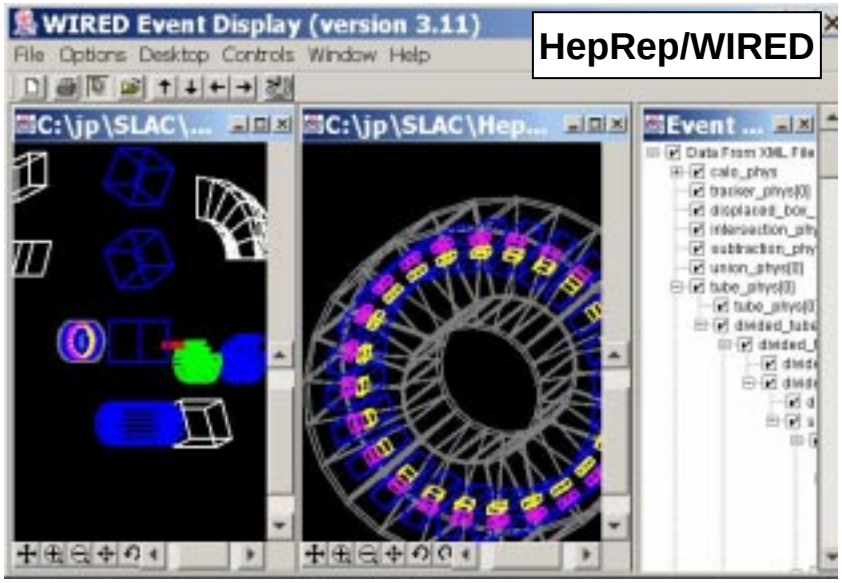
```
<heprep xmlns="http://www.freehep.org/HepRep"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="HepRep.xsd">
 <layerorder="Detector, Event, CalHit, Trajectory, TrajectoryPoint, Hit"/>
 <typetree name="G4GeometryTypes" version="1.0">
  <type name="Detector">
   <attvalue 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">
```

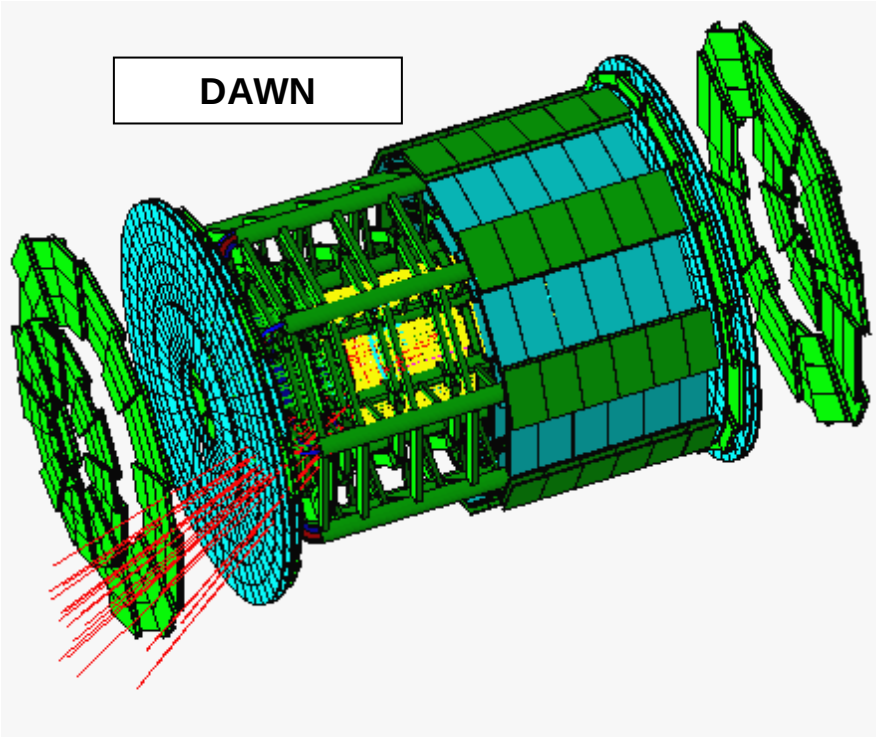
Example .prim File

```
##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:





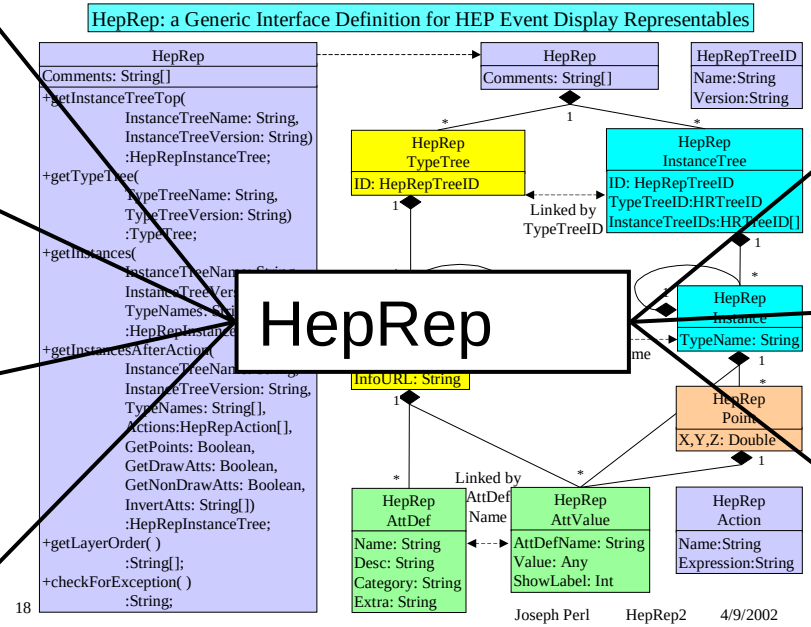
HepRep is Not Just for Geant4 and Not Just for WIRED

BaBar Server

GLAST Gaudi Service

LCD Interface

Geant4 Server



HepRep

WIRED Client (Java)

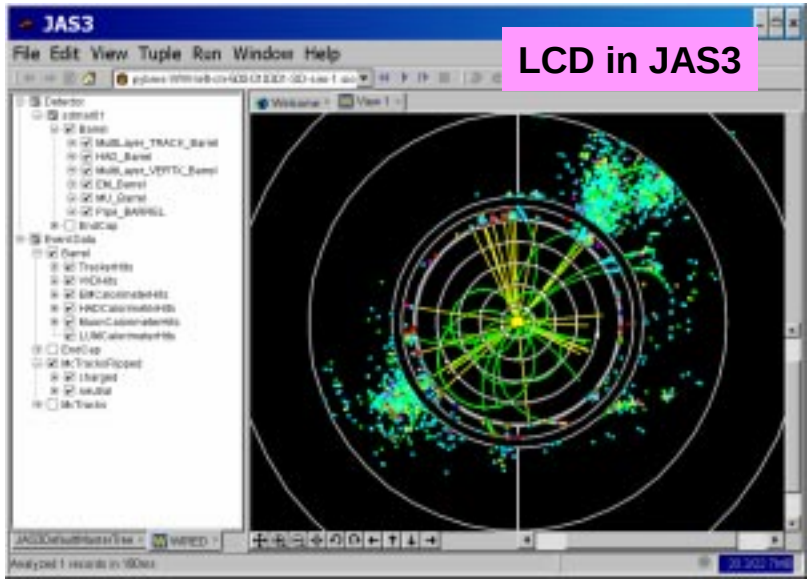
FRED Client (C++/Ruby)

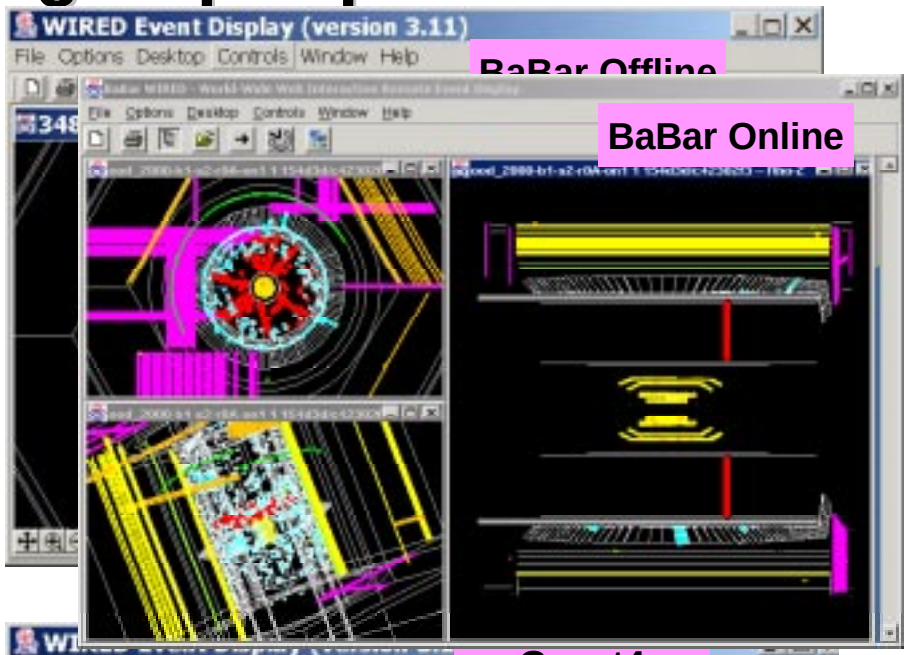
Other HepRep Clients

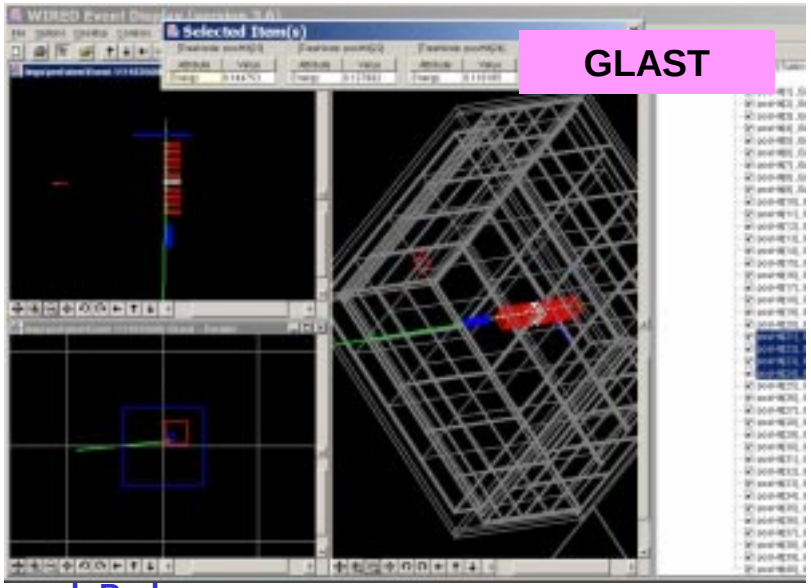
The HepRep interface breaks the dependency between any particular experiment's event display server and any particular event display client.

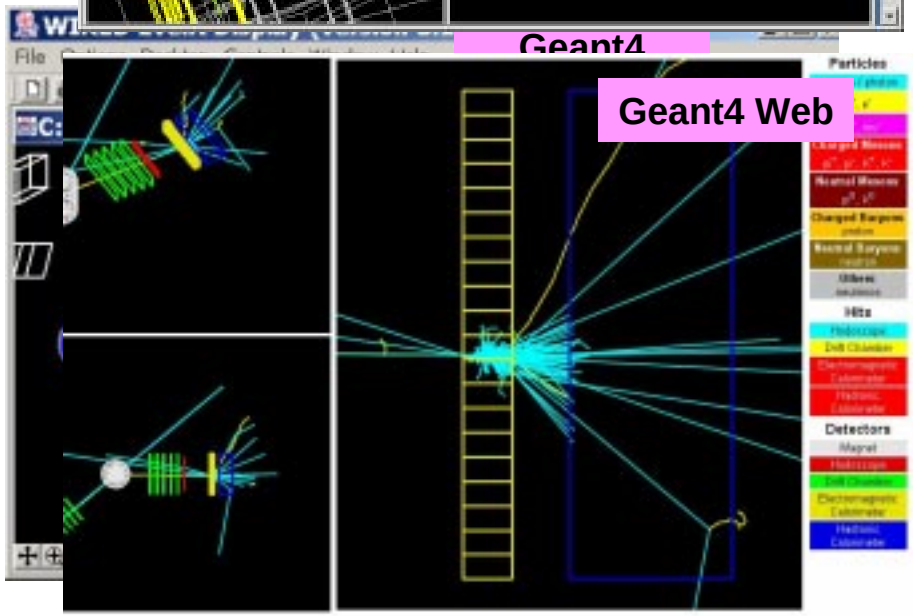
The HepRep format is independent of any one particular language or protocol. It can be used from C++ or Java and can be shipped as Corba, RMI, XML, C++, Java or JNI for consumption by WIRED, FRED or any other HepRep-enabled event display client.

Who's Using HepRep



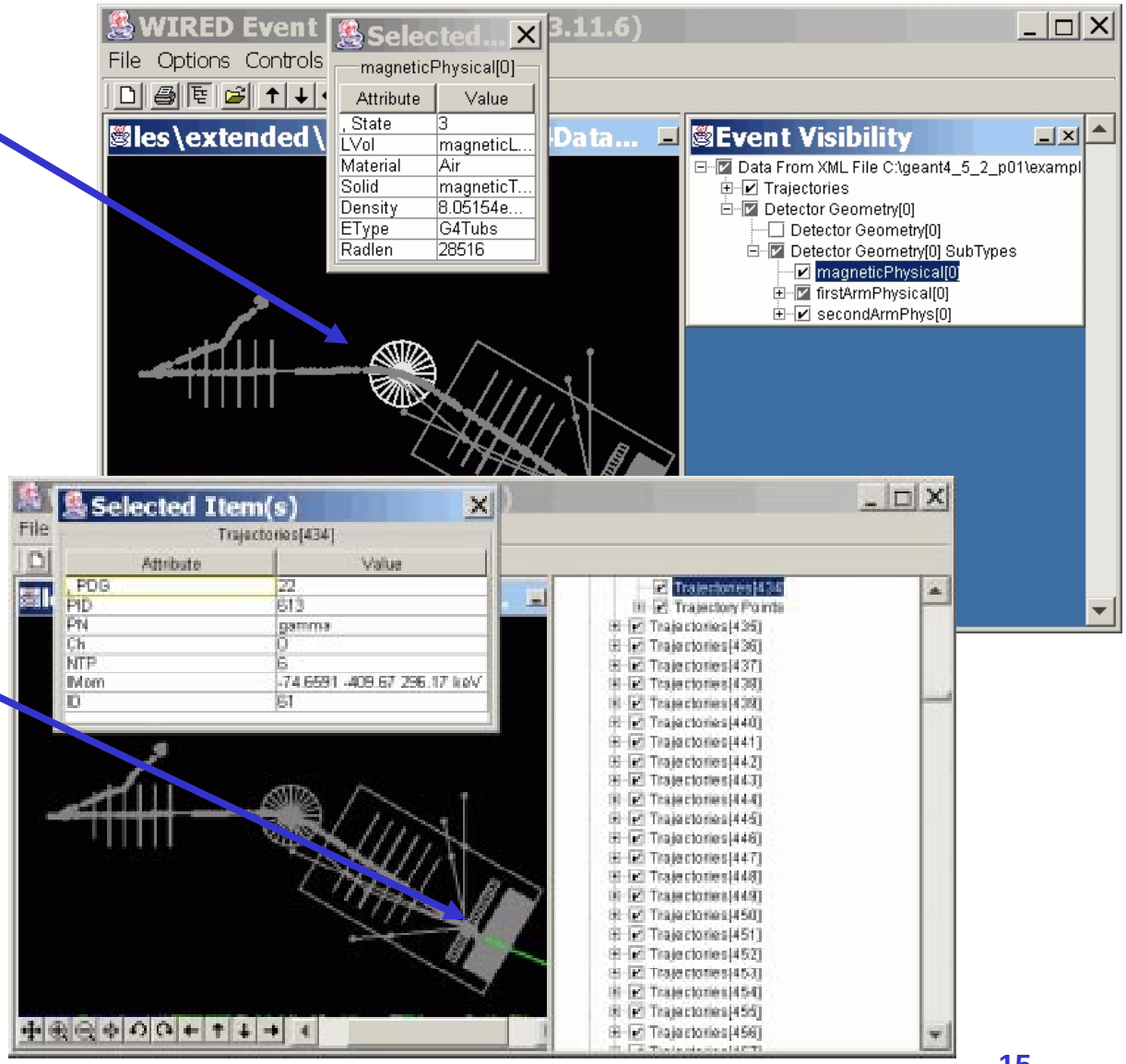






WIRED Let's You Pick to Show Physics Attributes

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



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

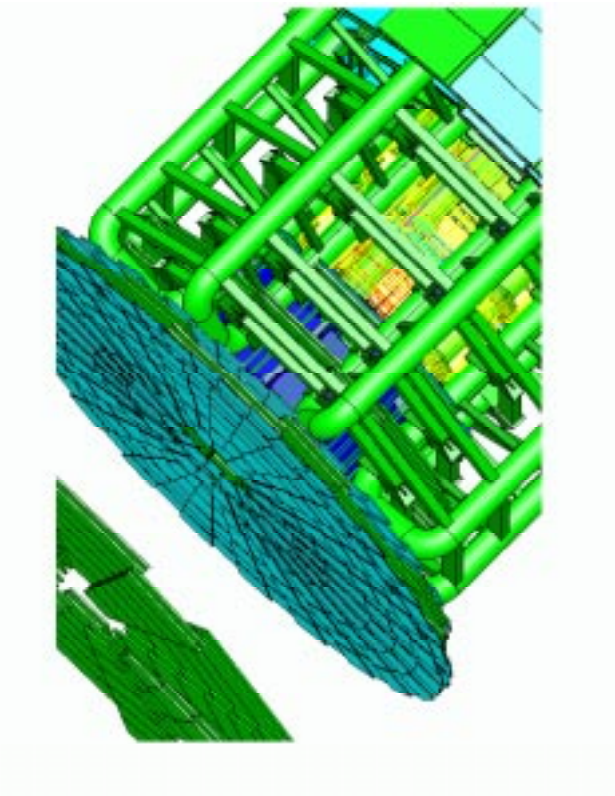
Origins of DAWN

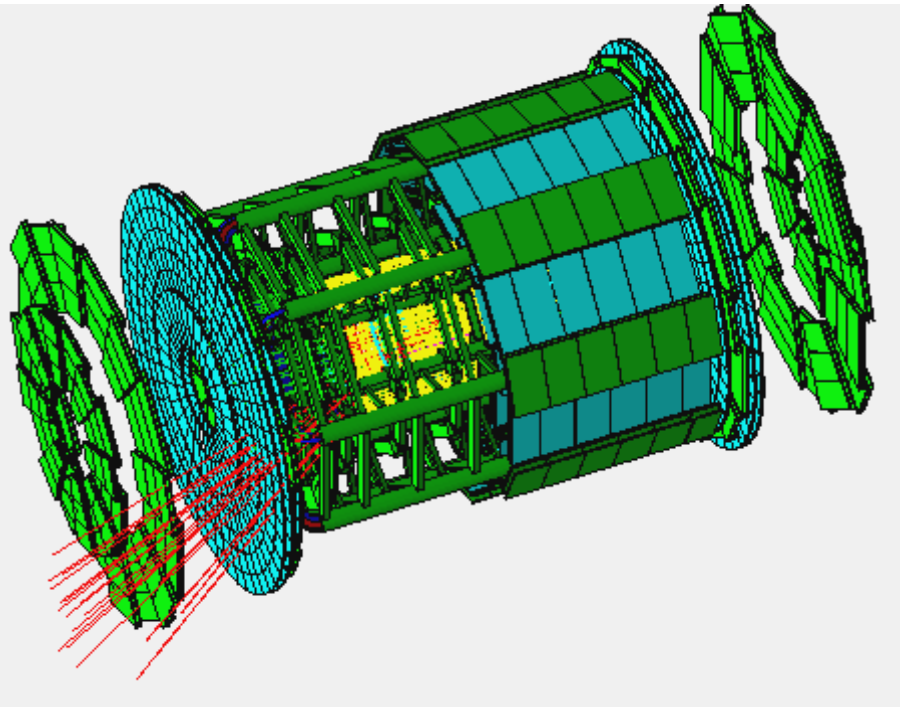
Fukui Renderer DAWN (Drawer for Academic WritiNgs).

- **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.**
- **Calculates all visible parts of the 3D data before drawing.**
- **Produces device-independent vectorized graphics for high quality technical applications.**

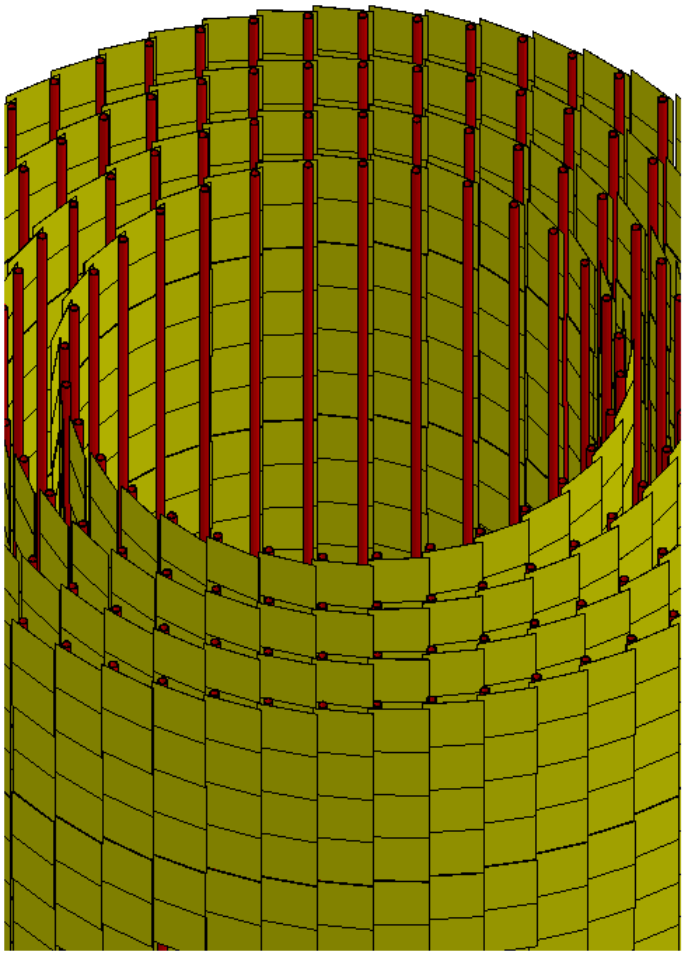
DAWN Examples

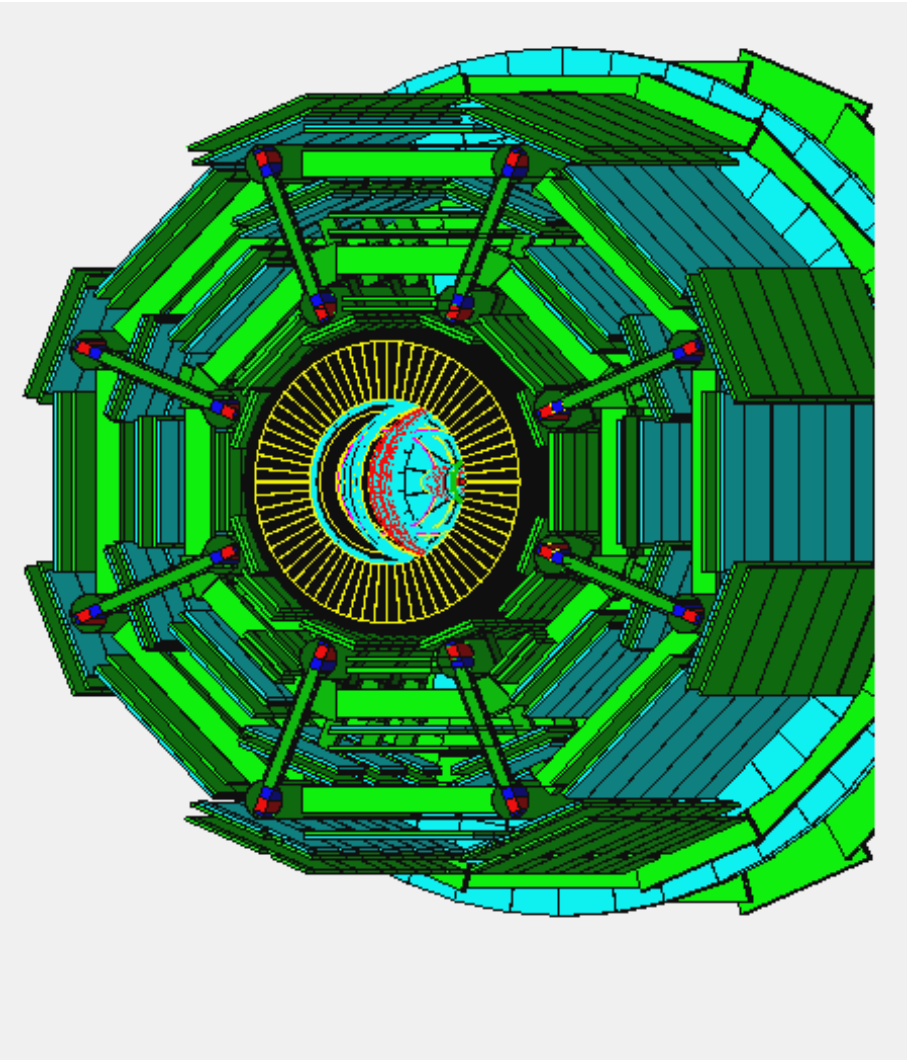
- From a repository of beautiful images at
  - http://geant4.kek.jp/~tanaka/GEANT4/ATLAS_G4_GIFFIG/





DAWN Examples





HepRep and DAWN are 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

Visualization Attributes

- Necessary for visualisation, but not included in geometrical information
  - Colour, visibility, forced-wireframe style, etc
  - A set of visualisation attributes is held by the class G4VisAttributes
- A G4VisAttributes object is assigned to a visualisable object with its method SetVisAttributes() :
  - experimentalHall_logical -> SetVisAttributes (G4VisAttributes::Invisible)
- Study G4 examples or references at end of this presentation to learn more about G4VisAttributes.

Generic 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 event display (and later HepRep clients will be able to turn on and off visibility of objects by cutting 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: Introduction to the Visualization Commands

- Environment variables
- Commands

Environment Variables

- Two of the visualization drivers discussed in this tutorial are always included by default in Geant4 (snce they require no external libraries):
  - HepRepFile
  - DAWNFILE
- Other visualization drivers may require setting environment variables:
  - OpenGL

- To include OpenGL, before you build Geant4, set the appropriate “build” variable to 1 (to cause the necessary code to be linked into your executable), such as:
  - setenv G4VIS_BUILD_OPENGLX_DRIVER 1
- Then, 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, however, 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 - expect a fix when JDK 1.5 comes out.

Command Terminology

- Geant4 visualization may be easier to understand if you know Geant4's concepts of "scene" and "viewer"
  - Scene
    - A set of visualizable 3D data (detectors, events, markers)
  - Viewer
    - Image generator
  - Scene Handler
    - Does the modeling from the scene into the viewer

- You will also sometimes see the term "Visualisation Driver", which is essentially a viewer plus an associated scene handler.

Steps of Visualization

- Open a driver (instantiates scene handler and viewer), 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/reset
  - /vis/viewer/set/viewpointThetaPhi 70 20
  - /vis/viewer/set/style wireframe
- Create an empty scene:
  - /vis/scene/create
- Declare what 3D data should be added to the created scene:
  - /vis/scene/add/volume
  - /vis/scene/add/trajectories
  - /vis/scene/add/hits
- Attach scene to sceneHandler:
  - /vis/sceneHandler/attach
- Run simulation with appropriate options to store trajectory information
  - /tracking/storeTrajectory 1
  - /run/beamOn 1
- Execute the visualization
  - /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

Example 1: Visualizing a Detector

- To visualize a detector, the commands can be as follows:

```
# Create a scene handler and a viewer for OpenGL
    /vis/open OGLIX
# Set camera and drawing style
    /vis/viewer/reset
    /vis/viewer/set/viewpointThetaPhi 70 20
    /vis/viewer/set/style wireframe
# Create a new empty scene
    /vis/scene/create
# Declare that the world volume should be added to the scene
    /vis/scene/add/volume
# Attach scene to scenehandler
    /vis/sceneHandler/attach
# Execute the visualisation
  /vis/viewer/flush
```

Example 2: Visualizing Events

- To visualize events, the commands can be as follows:

```
# Create a scene handler and viewer for HepRep
   /vis/open HepRepFile
# Set camera and drawing style - not needed here since we make
# those adjustments separately in the WIRED or DAWN applications
# Create a new empty scene
    /vis/scene/create
# Declare that the world volume, trajectories and hits should be
# added to the scene
    /vis/scene/add/volume
    /vis/scene/add/trajectories
  /vis/scene/add/hits
# Attach scene to scenehandler
    /vis/sceneHandler/attach
# Store particle trajactories for visualisation
    /tracking/storeTrajectory 1
# Execute the visualization via a /vis/viewer/flush contained within
# the macro “eventAction.mac” that is called at the end of each event
    /run/beamOn 10 eventAction.mac
```

/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 there are some subtleties then about multiple scenes and sceneHandlers.

/vis/viewer/… Commands

To Set Camera Parameters and Drawing Style.
Only needed if using an immediate viewer, such as OpenGL.

- **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 force by "setForceWireframe" or "setForceSolid" commands in the c++ code**

- **Zoom**
  - **/vis/viewer/zoom <scale factor>**
- **for example**
  - **/vis/viewer/zoom 2.**

Commands to Visualize Events

- To tell tracking to make trajectories available for drawing
  - /tracking/storeTrajectory 1

- To add trajectories or hits to the scene
  - /vis/scene/add/trajectories
  - /vis/scene/add/hits

- To accumulate more than one event per drawing
  - /vis/scene/endOfEventAction accumulate
- Otherwise, to have just one event per drawing, take the default
  - /vis/scene/endOfEventAction refresh

- To draw at the end of each event, put the following line into a macro (such as eventAction.mac)
  - /vis/viewer/flush
- Then run with that macro
  - /run/beamOn <number_of_events> <macro_name>
- for example
  - /run/beamOn 10 eventAction.mac

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

Complete List of Commands

- This presentation has shown only a very small subset of the full Geant4 command set. Even for those commands shown, only a few of the options have been presented.

- To see the complete set of commands, look in the Geant4 source code for the file:
  - geant4/source/visualization/README.built_in_commands

- 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 necessary libraries installed), OpenGL 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.
  - Other options, not discussed in this talk simple because the present author is not experienced with them (RayTracer, VRML and ASCIITree).
- Geant4's visualization framework allows one to run multiple visualization drivers side by side
- See the accompanying hands-on tutorials to try three visualization drivers:
  - G4WIREDTutorial.html
  - G4DAWNTutorial.html
  - G4OpenGLTurorial.html

Further Resources

Geant4 Visualization README file:

- geant4/source/visualisation/README

On-line Documentation on Geant4 Visualisation:

- 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

- HepRep: a generic interface definition for HEP event display representables
  http://heprep.freehep.org
- HepRep Complete Presentation (most complete description of HepRep)
  http://heprep.freehep.org/heprep2.Complete.ppt
  http://heprep.freehep.org/heprep2.Complete.pdf
- Fred: oh no, another event display (a HepRep client)
  http://www.fisica.uniud.it/~riccardo/research/fred
- WIRED: world wide web interactive remote event display (a HepRep Client)
  http://www.slac.stanford.edu/BFROOT/www/Computing/Graphics/Wired
- About DAWN
  http://geant4.kek.jp/~tanaka/DAWN/About_DAWN.html
- Satoshi Tanaka's GEANT4 Ritsumeikan University Group Home Page (more information on DAWN, sample PRIM files, images, etc.)
  http://geant4.kek.jp/~tanaka/