Recent Developments in Geant4 Visualization

Geant4 Lecture Course
Tsukuba, Japan
Joseph Perl, SLAC
Sorry, Vis Talks always need two pages of Cover Graphics
Contents

- Status: Seven Visualization Drivers
- New Features: Enhanced Trajectory Drawing
- New Features: Trajectory and Hit Filtering
- New Features: Smooth and Rich Trajectories
- New Features: Movies
- Coming Perhaps by June 2007: Visualization of Fields
Status: Seven Visualization Drivers
OpenGL

- /vis/open OGLIX

- Features
  - Control directly from Geant4
  - Uses GL libraries that are already included on most Linux and Windows systems
  - 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)

- Supported by John Allison with Guy Barrand
OpenGL with Motif Control

- If you don’t have Motif, all control is done from Geant4 commands:
  - /vis/open OGLIX or OGLSWin32
  - /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
Hidden Line Removal

- OpenGL supports 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`
OpenInventor

- `/vis/open OIX` or `/vis/open IOWin32`

- **Features**
  - Control from the OpenInventor GUI
  - Requires addition of OpenInventor libraries (freely available for most Linux systems and Windows).
  - 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)

- Supported by Guy Barrand
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.
/vis/open HepRepFile

Features
- Create a file to view in the
  - WIRED3 HepRep Browser
  - WIRED4 JAS Plugin
  - or FRED 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
- Hierarchical view of the geometry
- WIRED3 and WIRED4 can export to many vector graphic formats (PostScript, PDF, etc.)

Supported by Joseph Perl and Mark Donszelmann
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
DAWN

- /vis/open DAWNFILE

- Features
  - Create a .prim file
  - Requires DAWN, available for all Linux and Windows systems
  - DAWN creates a rendered, photorealistic PostScript image
  - No interactive features once at PostScript stage
  - Highest quality technical rendering - vector PostScript
  - View or print from your favorite PostScript application

- Supported by Satoshi Tanaka
  - Recently ported DAWN, DAWNCUT and DAVID to FedoraCore 4.0 (g++ 4.0)
DAWN Examples

- From a repository of beautiful images at
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/
VRML

- `/vis/open VRML1FILE` or `/vis/open VRML2FILE`

**Features**
- 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)

**Supported by Satoshi Tanaka**
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

/vis/open RayTracer

Features
- 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) - no other Vis driver can handle every case
- Supports shadows, transparency and mirrored surfaces

As of release Geant4.8.0, also now RayTracerX
- /vis/open RayTracerX
- Simultaneously renders to screen and to jpeg file, so that you can watch as the rendering grows progressively smoother

Supported by Makoto Asai and John Allison
RayTracer Shows Shadows
RayTracer Supports Transparency
RayTracer Handles Mirrored Surfaces

Mirrored Surfaces
RayTracerX

- New since Geant4.8.0
- In addition to
  - /vis/open RayTracer
- You 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

- `/vis/open ATree`

**Features**
- 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

- Supported by John Allison
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.
At verbosity level 4, ASCIIITree 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/ASCIIITree/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
Summary of Visualization Drivers

We have Seven Visualization Drivers with complimentary strengths. All well supported.

- OpenGL
- OpenInventor
- HepRep/WIRED (and FRED)
- DAWN
- VRML
- RayTracer
- ASCIITree
New Features:
Enhanced Trajectory Drawing
Enhanced Trajectory Drawing

- Ability to change trajectory drawing model through interactive commands

- Lets you, for example,
  - declare that trajectories should be color-coded by charge,
  - then change to have them color-coded by particle type

- Eliminates the most common reason users had to code their own trajectory classes

- Project Lead: Jane Tinslay
Example A01, five events, draw by various models
Sample Commands: generic trajectory model

# Create a generic model (will get default name of generic-0)
# From here we can set overall defaults for things like line color,
# whether to show step points or just the trajectory line, etc.
/vis/modeling/trajectories/create/generic

# Configure the generic model to colour all trajectories cyan and to show step points
/vis/modeling/trajectories/generic-0/default/setDrawStepPts true
/vis/modeling/trajectories/generic-0/default/setStepPtsSize 16
/vis/modeling/trajectories/generic-0/default/setLineColour cyan
/vis/modeling/trajectories/generic-0/default/setStepPtsColour red
Sample Commands: drawByCharge model

# Create a drawByCharge model (will get default name of drawCharge-0)
/vis/modeling/trajectories/create/drawByCharge

# Create another drawByCharge model with an explicit name of testChargeModel
/vis/modeling/trajectories/create/drawByCharge testChargeModel

# We can now go on to configure these two different drawByCharge models and then
# can choose to visualize using either one of them at any time

# Configure drawByCharge-0 model
/vis/modeling/trajectories/drawByCharge-0/set 1 red
/vis/modeling/trajectories/drawByCharge-0/set -1 red
/vis/modeling/trajectories/drawByCharge-0/set 0 white

# Configure testCharge model through G4Colour components
/vis/modeling/trajectories/testChargeModel/setRGBA 1 0 1 1 1
/vis/modeling/trajectories/testChargeModel/setRGBA -1 0.5 0.5 0.5 1
/vis/modeling/trajectories/testChargeModel/setRGBA 0 1 1 0 1

# List available models
/vis/modeling/trajectories/list

# select drawByCharge-0 to be current
/vis/modeling/trajectories/select drawByCharge-0
Sample Commands: drawByParticleID model

# Create a drawByParticleID model named drawByParticleID-0
/vis/modeling/trajectories/create/drawByParticleID

# Configure drawByParticleID-0 model
/vis/modeling/trajectories/drawByParticleID-0/set gamma red
/vis/modeling/trajectories/drawByParticleID-0/set proton yellow
/vis/modeling/trajectories/drawByParticleID-0/setRGBA e+ 1 0 1 1
Example A01, five events, drawByAttribute models
Sample Commands: drawByAttribute

/vis/modeling/trajectories/create/drawByAttribute
/vis/modeling/trajectories/drawByCharge-0/verbose true
/vis/modeling/trajectories/drawByCharge-0/setAttribute CPN
/vis/modeling/trajectories/drawByAttribute-0/addValue brem_key eBrem
/vis/modeling/trajectories/drawByAttribute-0/addValue annihil_key annihil
/vis/modeling/trajectories/drawByAttribute-0/addValue decay_key Decay
/vis/modeling/trajectories/drawByAttribute-0/addValue muIon_key muIoni
/vis/modeling/trajectories/drawByAttribute-0/addValue eIon_key eIoni
/vis/modeling/trajectories/drawByAttribute-0/brem_key/setLineColour red
/vis/modeling/trajectories/drawByAttribute-0/annihil_key/setLineColour green
/vis/modeling/trajectories/drawByAttribute-0/decay_key/setLineColour cyan
/vis/modeling/trajectories/drawByAttribute-0/eIon_key/setLineColour yellow
/vis/modeling/trajectories/drawByAttribute-0/muIon_key/setLineColour magenta
Enhanced Trajectory Drawing: Releases

- **Since Release 8.0:**
  - Color by Charge
  - Color by Particle Type

- **Since Release 8.1**
  - Color by Origin Volume
  - Control over more than just color, including:
    - whether to show just trajectory line, or just trajectory points, or both
    - width of trajectory lines, type of marker to use for points, point size, etc.

- **Coming In Release 8.2**
  - Color by Any of the HepRep-Style Attributes, such as:
    - Initial volume name
    - Magnitude of momentum
    - Number of trajectory points
    - User defined attributes
    - Creator process name & type (if using Rich Trajectory)
    - Next volume name (if using Rich Trajectory)
New Features:
Trajectory and Hit Filtering
Trajectory and Hit Filtering

- Display user-defined subset of trajectories
  - Solves problems with large graphics files, busy events

- Two modes of operation
  - Rejected trajectories not drawn at all
  - Rejected trajectories drawn but marked invisible
    - Useful in interactive graphics browsers such as Wired or OpenInventor, where you have the option to toggle visibility later from the browser.
    - Drawback is that the files remain large, since all of the data is still in the file.

- Similar structure to enhanced trajectory drawing
  - Set of simple filter models
  - Similar Interactive creation/configuration structure

- You can chain multiple filters

- Project Lead: Jane Tinslay
HandsOn5, McGill tutorial, 1000 events, Attribute Filter

\[ \text{IMag} > 2.5 \text{ MeV} \]

particle = gamma
Sample Macro Commands

/vis/modeling/trajectories/drawByAttribute-0/setAttribute IMag
/vis/modeling/trajectories/drawByAttribute-0/addInterval interval1 0.0 keV 2.5 MeV
/vis/modeling/trajectories/drawByAttribute-0/addInterval interval2 2.5 MeV 5 MeV
/vis/modeling/trajectories/drawByAttribute-0/addInterval interval3 5 MeV 7.5 MeV
/vis/modeling/trajectories/drawByAttribute-0/addInterval interval4 7.5 MeV 10 MeV
/vis/modeling/trajectories/drawByAttribute-0/addInterval interval5 10 MeV 12.5 MeV
/vis/modeling/trajectories/drawByAttribute-0/addInterval interval6 12.5 MeV 10000 MeV
/vis/modeling/trajectories/drawByAttribute-0/interval1/setLineColourRGBA 0.8 0 0.8 1
/vis/modeling/trajectories/drawByAttribute-0/interval2/setLineColourRGBA 0.23 0.41 1 1
/vis/modeling/trajectories/drawByAttribute-0/interval3/setLineColourRGBA 0 1 0 1
/vis/modeling/trajectories/drawByAttribute-0/interval4/setLineColourRGBA 1 1 0 1
/vis/modeling/trajectories/drawByAttribute-0/interval5/setLineColourRGBA 1 0.3 0 1
/vis/modeling/trajectories/drawByAttribute-0/interval6/setLineColourRGBA 1 0 0 1
/vis/filtering/trajectories/create/attributeFilter
/vis/filtering/trajectories/attributeFilter-0/setAttribute IMag
/vis/filtering/trajectories/attributeFilter-0/addInterval 2.5 MeV 1000 MeV
/vis/filtering/trajectories/create/particleFilter
/vis/filtering/trajectories/particleFilter-0/add gamma

Momentum
filter
Momentum
interval based
colour scale
Configure
visualisation
properties
Momentum
filter
Gamma filter
Hit Filtering

- The Attribute-Based filtering discussed above for Trajectories was implemented generically so that it can apply to any class which implements the Geant4 generic attributes method:

```cpp
const std::map<G4String, G4AttDef> * GetAttDefs() const;
std::vector<G4AttValue> * CreateAttValues() const;
```

- Whatever your Hit class, you can basically get interactive hit filtering for free

- To activate, add a filter call to G4VVisManager in Draw method of hit class

```cpp
void MyHit::Draw() {
  ...
  if (! pVVisManager->FilterHit(*this)) return;
  ...
```
Trajectory and Hit Filtering: Releases

- **Since Release 8.1**
  - Filter by Charge
  - Filter by Particle Type
  - Filter by Origin Volume

- **Coming In Release 8.2**
  - Filter by Any of the HepRep-Style Attributes, such as:
    - Initial volume name
    - Magnitude of momentum
    - Number of trajectory points
    - User defined attributes
    - Creator process name & type (if using Rich Trajectory)
    - Next volume name (if using Rich Trajectory)
New Features:
Smooth and Rich Trajectories
G4SmoothTrajectory and G4RichTrajectory

- Most users use the standard G4Trajectory. But two other trajectory classes have been available in Geant4 tracking for some time.
  - Smooth Trajectory adds auxiliary points to allow smoother line in visualization (not Geant4 Steps, no physics at auxiliary points)
  - Rich Trajectory encodes additional information at every step point

- Project Lead: John Allison, Joseph Perl
Regular versus Smooth Trajectory

- Regular Trajectory is a polyline made up of only the actual steps used by Geant4.

- Smooth Trajectory includes additional points to make the polyline appear smoother.

- These additional points are not Geant4 steps. They are only used by visualization.
Rich Trajectory
Smooth and Rich Trajectories Now Easier to Use

- Up to now, users who wanted to use these trajectories have had to write their own tracking action and explicitly instantiate one of these special kinds of trajectory.

- December release will make this possible from an interactive command:
  - /vis/scene/add/trajectories
  - /vis/scene/add/trajectories smooth
  - /vis/scene/add/trajectories rich
  - /vis/scene/add/trajectories smooth rich

- Remember that if you actually want to display the trajectory points, you need to issue the appropriate trajectory modeling commands, such as:
  - /vis/modeling/trajectories/create/generic
  - /vis/modeling/trajectories/generic-0/default/setDrawStepPts true
  - /vis/modeling/trajectories/generic-0/default/setStepPtsSize 16
  - /vis/modeling/trajectories/generic-0/default/setStepPtsColour red
New Features: Movies
Movies that Just Involve Changing Camera Position

- Making movies with just changes to camera position has been possible for years using macros.
- Conversion to mpeg somewhat complicated
  - will be documented in December release
  - or see Friday’s Visualization Lecture
Movies: Stitched Together from Multiple Stills

QuickTime and a YUV420 codec decompressor are needed to see this picture.
New Technique: Time Development of the Event

- New features in the December release will allow you to do a new kind of Geant4 movie. You will be able to make movies that show Time Development of an event:
  - I.e., a shower in slow motion

- Based on technique of “time-slicing”, breaking trajectories into individual slices, each with a time attribute:
  - Requires newer visualization features, rich trajectory and some extensions to the OpenGL driver
  - You can run these animations directly from Geant4, does NOT involve stitching together a movie by hand

- A collection of example movies has been prepared by John Allison:

- How-To Presentation:
  - [http://geant4.slac.stanford.edu/Presentations/vis/HowToMakeAMovie.ppt](http://geant4.slac.stanford.edu/Presentations/vis/HowToMakeAMovie.ppt)
  - [http://geant4.slac.stanford.edu/Presentations/vis/HowToMakeAMovie.pdf](http://geant4.slac.stanford.edu/Presentations/vis/HowToMakeAMovie.pdf)

- Project Lead: John Allison
Movies: Live from Geant4 OpenGL

QuickTime and a decompressor are needed to see this picture.
Coming Perhaps by June 2007: Visualization of Fields
Summary

- From the very beginning, Geant4 visualization has been very flexible. The experienced programmer could exploit the full flexibility of C++ and a well-designed series of abstract interfaces to code any desired visualizations.

- The work of the Geant4 Visualization group over the last few years has been to identify those features that users want most, and make them easily achievable from simple interactive commands.

- So, in the last year or so we have added:
  - Trajectory Modeling commands
  - Trajectory and Hit Filtering commands
  - Smooth and Rich Trajectory commands
  - Time-Development Animation, etc.

- More great new features will be coming, such as:
  - Visualization of Fields
References and Further Resources
Where to Find Exact Command Syntax

- For the exact syntax of any of the commands shown in this presentation, use the help system from within Geant4.

- Note that many of the command details are only loaded into the help system once you start using the given command.
  - e.g., when you first look at the help for /vis/modeling, you will see only
    - /vis/modeling/trajectories/create
    - /vis/modeling/trajectories/list
  - But once you have done your first
    - /vis/modeling/trajectories/create/drawByParticleID
  - you will see many subcommands such as
    - /vis/modeling/trajectories/drawByParticleID-0/set
    - /vis/modeling/trajectories/drawByParticleID-0/setRGBA
    - etc.
Detailed List of What Was Recently Done and What is Still To Do

For the complete list of what has been done since release 8.0 and what is to do, see:

http://geant4.slac.stanford.edu/Presentations/vis/workplans/VisToDoList
and .pdf

There are some new features there that I haven’t even had time to tell you about today, such as the vis/geometry commands to interactively control color, etc. of geometry volumes

Feedback welcome
Geant4 Visualization Resources

Geant4 Installation Guides
➢ http://geant4.slac.stanford.edu/installation

Hands on WIRED3 Tutorial

Hands on DAWN Tutorial

Hands on OpenGL Tutorial
➢ http://geant4.slac.stanford.edu/Presentations/vis/G4OpenGLTutorial/G4OpenGLTutorial.html

Geant4 Visualization Commands
➢ http://geant4.slac.stanford.edu/Presentations/vis/G4VisCommands.ppt (and .pdf)

Geant4 Advanced Visualization
➢ http://geant4.slac.stanford.edu/Presentations/vis/G4VisAdvanced.ppt (and .pdf)

On-line Documentation on Geant4 Visualization:

List of Visualization Commands:

For Questions or Comments: Geant4 Visualization Online Forum:
References

- OpenScientist Home Page

- 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

- 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/