

Geant 4

March 2006, Geant4 v8.0p01

Visualisation UI Commands

Jane Tinslay

SLAC

Stanford
Linear
Accelerator
Center

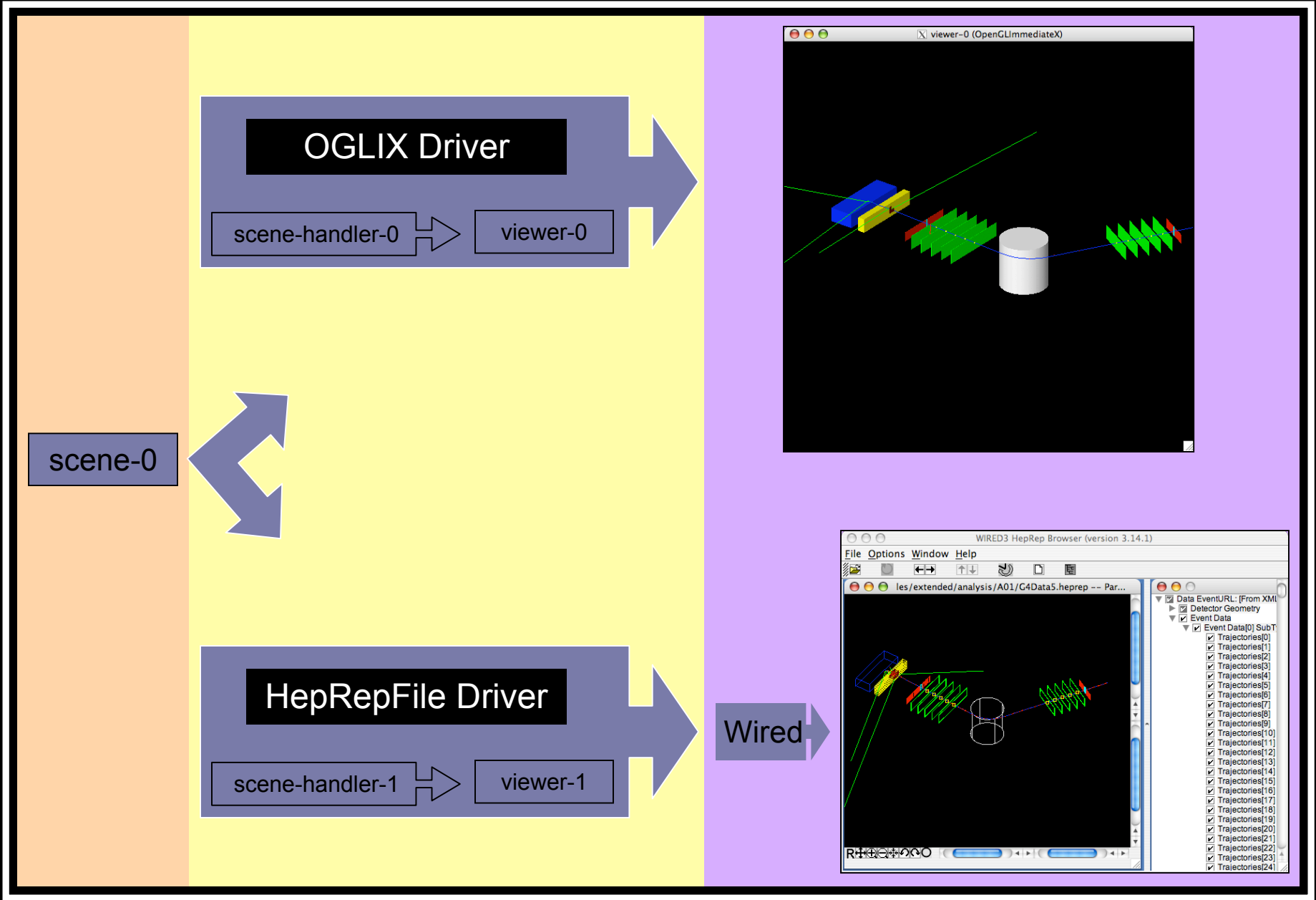
Outline

- Key Concepts
- Command structure and guidance
- Quick start guide
- Principle scene and driver commands
- Trajectory model commands
- Geometrical level of detail
- Accumulating trajectories and hits
- Information resources

Key Concepts

- **Scene**
 - Collection of objects which can be seen with a viewer
 - For example, text, axes, volumes, trajectories, hits, etc
- **Scene handler**
 - Processes raw scene data and produces output viewable by a specific graphics system
- **Viewer**
 - Generates an image using data produced from scene handler
 - May open a window and write to the screen, or dump data into files
- **“Current” object**
 - Possible to construct multiple scenes, viewers etc
 - Most recently created or selected is “current”

**Visualisation
driver**



Command Structure and Guidance

- Extensive set of visualisation commands
- **Low level commands**
 - Manipulate multiple scenes, scene handlers and viewers
 - Set camera parameters, drawing style etc
- **Compound commands**
 - Effectively wrap two or more other commands
 - Allows user to work more efficiently
 - Hides scene/scene handler/viewer interaction
 - Don't need detailed understanding of low level commands to do visualisation
- Typical macro will use mixture of compound and low level commands

- Command guidance is available from the interactive 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) /BeamTest/ ...Title not available...
11) /vis/ Visualization commands.
12) /gun/    Particle Gun control commands.
13) /hits/   Sensitive detectors and Hits
Commands :

Type the number ( 0:end, -n:n level back ) :
```

■ Detailed help is available for each visualisation command

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/modeling/ Modeling commands.
- 6) /vis/scene/ Operations on Geant4 scenes.
- 7) /vis/sceneHandler/ Operations on Geant4 scene handlers.
- 8) /vis/viewer/ Operations on Geant4 viewers.

Commands :

- 9) enable * Enables/disables visualization system.
- 10) disable * Disables visualization system.
- 11) verbose * Simple graded message scheme - digit or string (1st character defines):
- 12) drawTree * (DTREE) Creates a scene consisting of this physical volume and
produces a representation of the geometry hierachy.
- 13) drawView * Draw view from this angle, etc.
- 14) drawVolume * Creates a scene consisting of this physical volume and asks the current viewer to draw it.
- 15) open * Creates a scene handler ready for drawing.**
- 16) specify * Draws logical volume with Boolean components, voxels and readout geometry.

Command /vis/open

Guidance :

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

...

Quick Start Guide

- Create an empty scene
- Create a visualisation driver
- Add data to the scene
- Configure drawing style
- Execute visualisation

Example macro

```
# Create a new empty scene
/vis/scene/create

# Create an OpenGL driver (ie, a
# scene handler and viewer)
/vis/open OGLIX

# Add world volume, trajectories and hits
# to the scene
/vis/scene/add/volume
/vis/scene/add/trajectories
/vis/scene/add/hits

# Configuration
/vis/viewer/set/style surface

# Visualisation is executed
# automatically with the beamOn command
/run/beamOn 1
```

Principle Scene and Driver Commands

- See guidance for full set of commands and configuration parameters
 - Scene Commands
 - **/vis/scene/create <name>**
 - Create an empty scene. Uses a default name if not supplied
 - Eg, /vis/scene/create myscene
 - **/vis/scene/add/trajectories <drawing-mode>**
 - Add trajectories to scene
 - **/vis/scene/add/hits**
 - Add hits to scene
 - **/vis/scene/add/volume <physical-volume-name>**
 - Add physical volume to current scene.
 - If no volume is specified, “world” is added.
 - **/vis/scene/endOfEventAction <action>**
 - If action set to accumulate, multiple events drawn in the same scene. If action set to refresh (default), screen is refreshed for each event
- Add data to scene**

■ Driver Commands

- `/vis/sceneHandler/create <graphics-system-name>`
 - ➔ Create a scene handler for a specific system
- `/vis/sceneHandler/attach <scene-name>`
 - ➔ Attach scene to current scene handler
- `/vis/viewer/create <scene-handler>`
 - ➔ Create a viewer for given scene handler
- `/vis/viewer/update <viewer-name>`
 - ➔ Trigger graphical database post-processing
- `/vis/viewer/refresh <viewer-name>`
 - ➔ Refresh specified viewer
- `/vis/viewer/set/style <style>`
 - ➔ Set style of drawing. Can be set to either wireframe or surface
- `/vis/viewer/set/viewpointThetaPhi <theta> <phi>`
 - ➔ Set direction from target to camera
- `/vis/viewer/set/hiddenEdge <hidden-edge>`
 - ➔ Edges become hidden/seen through boolean parameter
- `/vis/viewer/zoom <multiplier>`
 - ➔ Incremental zoom

Creation

Execution

Settings

■ Compound commands

■ `/vis/open <graphics-system-name>`

- `/vis/sceneHandler/create <graphics-system-name>`
- `/vis/viewer/create <graphics-system-name>`
- For example `/vis/Open HepRepFile`

■ `/vis/viewer/flush`

- `/vis/viewer/refresh`
- `/vis/viewer/update`

■ `/vis/drawVolume <physical-volume-name>`

- `/vis/scene/create`
- `/vis/scene/add/volume <physical-volume-name>`
- World volume will be added to scene if none specified

Example macro

```
# Create a new scene
/vis/scene/create
# Create OpenGL driver
/vis/open OGLIX

# Add data to the scene
/vis/scene/add/volume
/vis/scene/add/trajectories
/vis/scene/add/hits

# Configure viewer
/vis/viewer/set/lightsThetaPhi 90. 0.
/vis/viewer/set/viewpointThetaPhi 150. 90.
/vis/viewer/set/style surface
/vis/viewer/set/hiddenEdge true

# Run for 10 events
/run/beamOn 10

# Open HepRepFile driver
/vis/open HepRepFile

#output just detector geometry
/vis/viewer/flush
# Run for one event. Visualisation will
#be executed automatically
/run/beamOn 1
```

**Create scene and
OpenGL driver**

Add data to scene

Configuration

OpenGL execution

**Create HepRepFile
driver**

HepRepFile execution

Model Commands

- Trajectory drawing commands located in `/vis/modeling/trajectories/`
 - This enhanced trajectory drawing functionality was introduced in Geant4 v8.0
 - Introduces the concept of models - ie, draw trajectories according to a particular style
- By default, trajectories are coloured according to charge
- Two models currently available with Geant4 distribution
 - `drawByCharge`
 - Colour trajectories according to charge
 - `drawByParticleID`
 - Colour trajectories according to particle type
- Users able to define and use own models

+1	Blue
-1	Red
0	Green

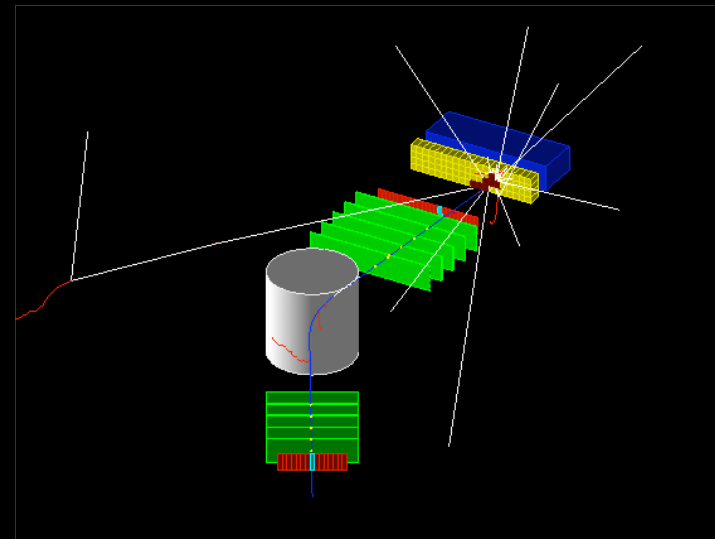
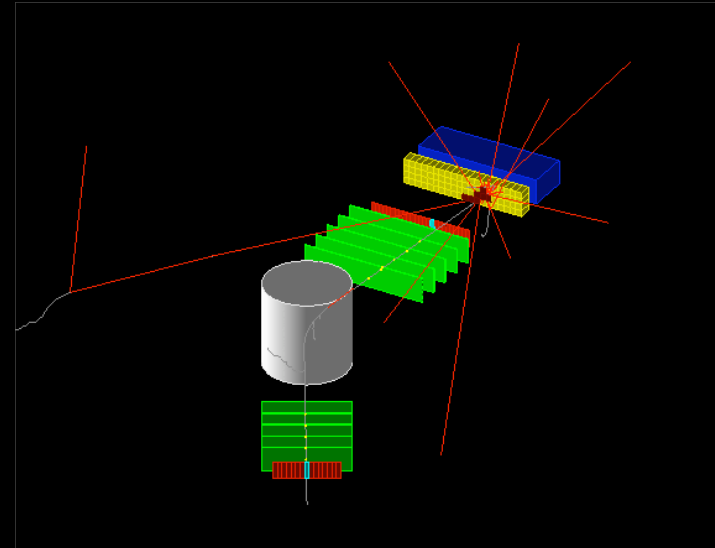
■ examples/extended/analysis/A01

Example macro

```
#Standard setup
/vis/scene/create
/vis/open OGLIX
/vis/scene/add/volume
/vis/scene/add/trajectories
/vis/scene/add/hits
/vis/viewer/set/lightsThetaPhi 90. 0.
/vis/viewer/set/viewpointThetaPhi 150. 90.
/vis/viewer/set/style surface
/vis/viewer/set/hiddenEdge true
#Create drawByParticleID model, highlighting photons
/vis/modeling/trajectories/create/drawByParticleID
/vis/modeling/trajectories/drawByParticleID-0/set gamma red
/run/beamOn 1

...

#Create drawByCharge model, colouring photons white
/vis/modeling/trajectories/create/drawByCharge
/vis/modeling/trajectories/drawByCharge-0/set 1 blue
/vis/modeling/trajectories/drawByCharge-0/set -1 red
/vis/modeling/trajectories/drawByCharge-0/set 0 white
/run/beamOn 1
```



Accumulating Trajectories and Hits

- Can choose to visualise trajectories:
 - At the end of each event (default behaviour)
 - `/vis/scene/endOfEventAction refresh`
 - At the end of run of X events
 - `/vis/scene/endOfEventAction accumulate`
 - Data from X events will be superimposed
 - At the end of Y runs
 - `/vis/scene/endOfRunAction accumulate`
 - Data from Y runs will be superimposed

Geometrical Level of Detail

- Geant4 draws world volume if `/vis/scene/add/volume` command invoked with no arguments
- Can specify parameters to control the geometrical level of detail
- `/vis/scene/add/volume <physical-volume-name> <copy-no> <depth-of-descent>`
 - `physical-volume-name`
 - Volume name. Default = world
 - `copy-no`
 - Copy number. By default, first occurrence of `physical-volume-name` is selected
 - `depth-of-descent`
 - Depth of descent of geometry hierarchy. Default = unlimited
- See command guidance for more details

Information Resources

- Only small sub-set of visualisation commands have been presented
- Full set of commands along with detailed help is available through the interactive guidance
- Example macros in Geant4 distribution
 - E.g., `examples/novice/N03`
- Also see visualisation section in Geant4 documentation:
<http://geant4.web.cern.ch/geant4/G4UsersDocuments/UsersGuides/ForApplicationDeveloper/html/index.html>