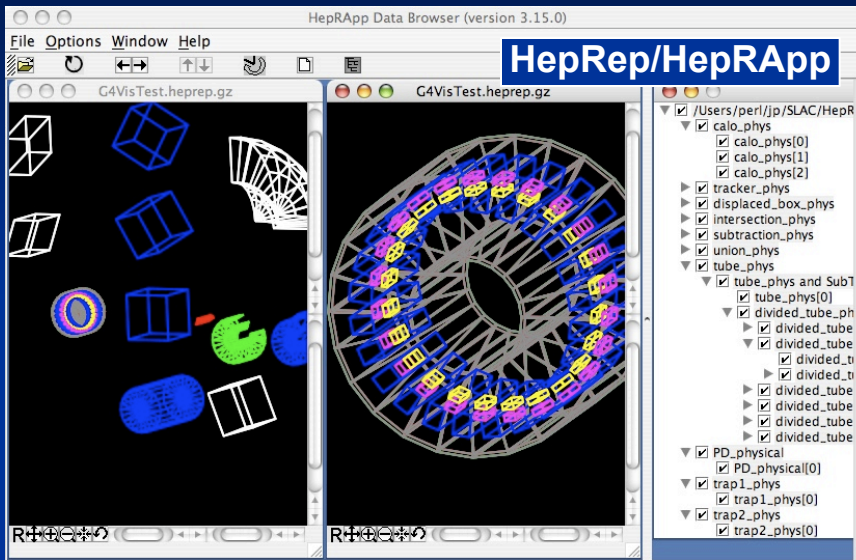


# Geant4 Visualization Drivers

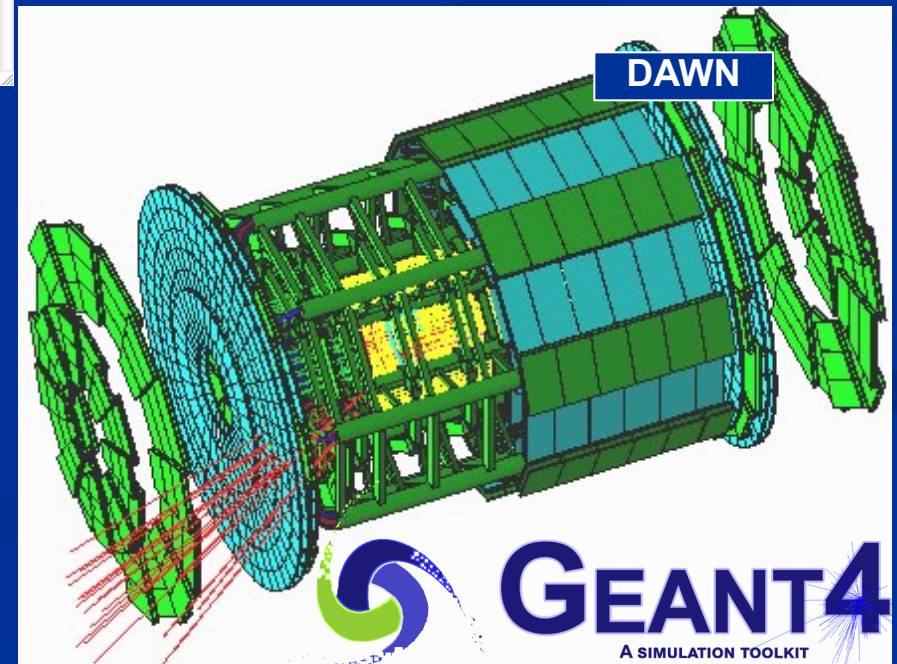
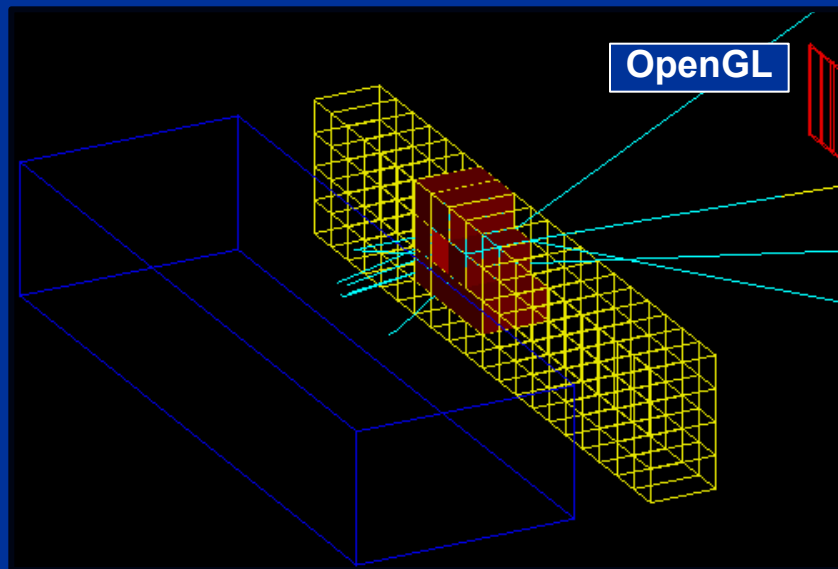


Igor Semeniouk

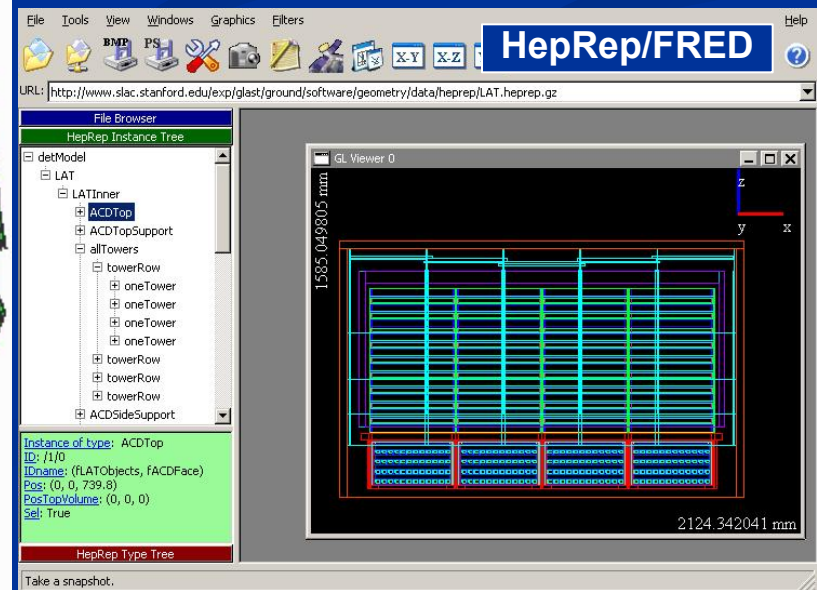
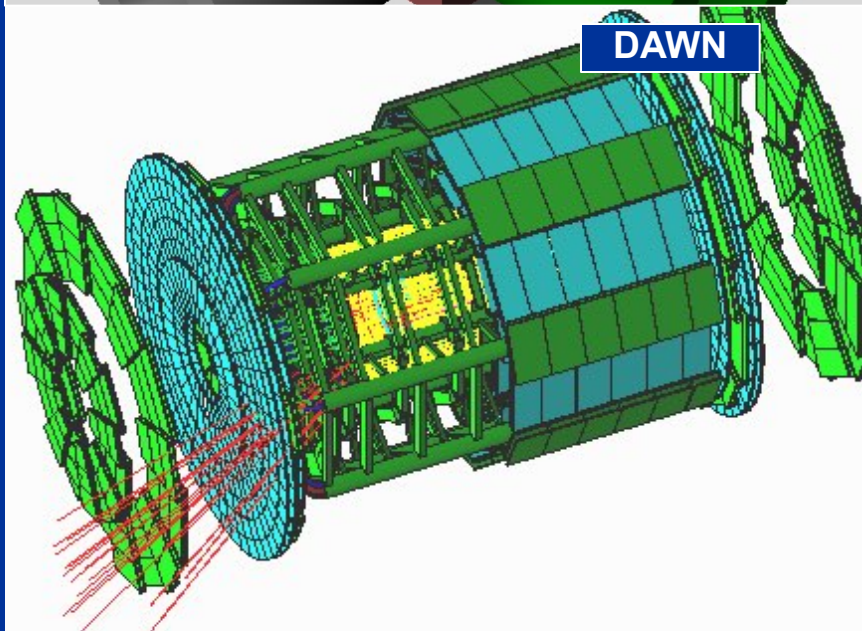
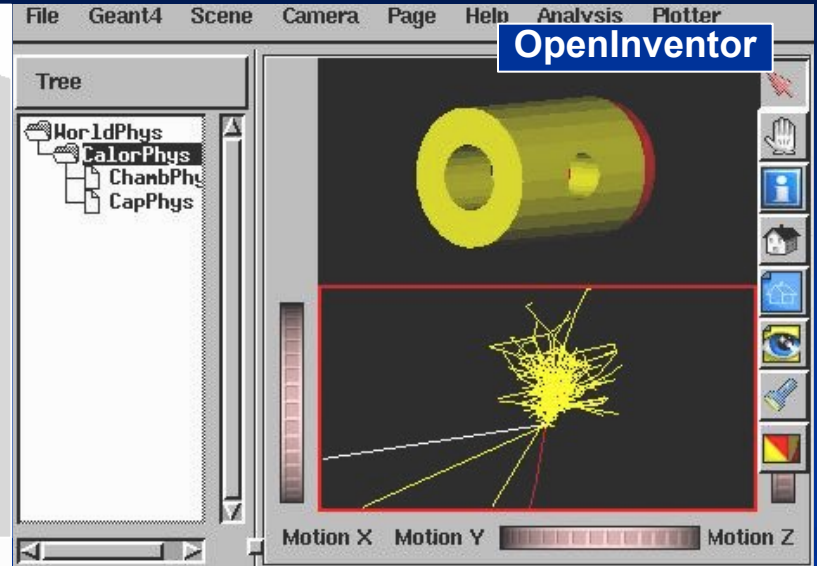
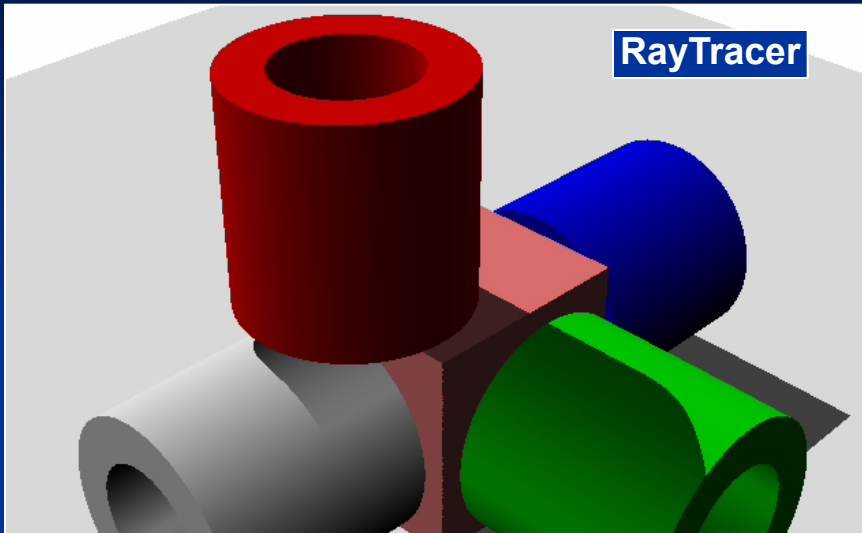
LLR, CNRS - Ecole Polytechnique

Slides from Laurent GARNIER ( IRISA )

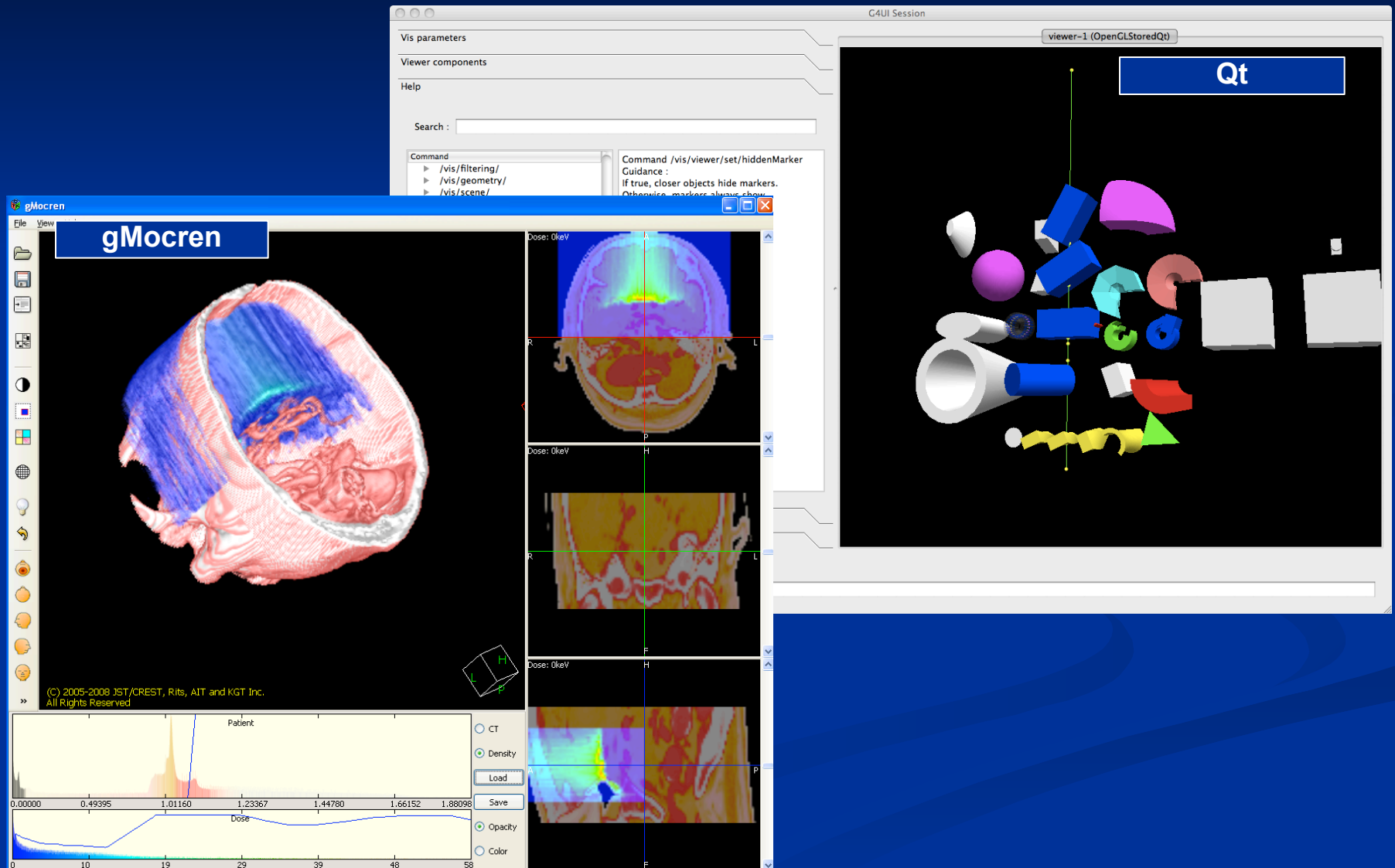
Based on Joseph Perl (SLAC) slides



# Geant4 Visualization Drivers



# Geant4 Visualization Drivers



# Geant4 Visualization Drivers

The image displays two windows from the Geant4 visualization interface. The top window, titled 'viewer-0 (Qt3D)', shows a 3D wireframe cube with a central cluster of green lines representing particle tracks. The bottom window, titled 'viewer-0 (OpenGLStoredQt)', shows the same 3D visualization but with a more detailed scene tree on the left. The scene tree includes a search bar and a list of commands such as 'control', 'units', 'profiler', 'gui', 'tracking', 'geometry', 'particle', 'event', 'cuts', 'run', 'random', 'process', 'material', 'testem', 'analysis', 'vis', 'globalField', and 'gun'. Below the 3D view in the bottom window is an 'Output' panel displaying simulation parameters and results.

**Qt3D**

**OGL**

```
Threads: All
```

```
Energy thresholds : gamma 101.843 keV e-  
1.36749 MeV e+ 1.27862 MeV proton 100 keV  
Region(s) which use this couple :  
DefaultRegionForTheWorld
```

```
Index : 2 used in the geometry : Yes  
Material : G4_lAr  
Range cuts : gamma 1 mm e- 1 mm e+ 1 mm proton 1 mm  
Energy thresholds : gamma 6.19986 keV e-  
349.521 keV e+ 337.972 keV proton 100 keV  
Region(s) which use this couple :  
DefaultRegionForTheWorld
```

```
=====  
Session :
```

```
used in the geometry : Yes  
p  
 : gamma 1 mm e- 1 mm e+ 1 mm proton 1 mm  
cuts : gamma 101.843 keV e-  
27862 MeV proton 100 keV  
use this couple :  
onForTheWorld  
used in the geometry : Yes  
LAr  
 : gamma 1 mm e- 1 mm e+ 1 mm proton 1 mm  
cuts : gamma 6.19986 keV e-  
7.972 keV proton 100 keV  
use this couple :
```

**Qt3D since 10.7.x  
Need Qt > 5.15**



# Geant4 Visualization Drivers

## ToolsSG Driver

The screenshot displays the ToolsSG driver interface. On the left is a command list with categories like control, units, profiler, gui, tracking, geometry, process, particle, event, cuts, run, random, B5, material, physics\_lists, analysis, vis, hits, physics\_engine, and gun. The main window shows a 3D visualization of a detector with a central source and various components. Below the visualization is an output window with the following text:

```
Threads: All
G4WT0 > ... write file : B5_t0.root - done
G4WT1 > ... delete empty file : B5_t1.root - done
G4WT5 > ... merge all H2 - done
G4WT5 > ... merge all H3 - done
G4WT5 > ... merge all P1 - done
G4WT5 > ... merge all P2 - done
G4WT0 > ... close file : B5_t0.root - done
G4WT5 > ... merge slave ntuples - done
G4WT0 > ... delete empty file : B5_t0.root - done
G4WT5 > ... write file : B5_t5.root - done
G4WT5 > ... close file : B5_t5.root - done
G4WT5 > ... delete empty file : B5_t5.root - done
```

On the right side, there are three histograms:

- Drift Chamber 1 Hits**: A histogram showing hits in the drift chamber 1. Statistics: Entries: 5, Mean: 6.2, RMS: 193907.
- Drift Chamber 2 Hits**: A histogram showing hits in the drift chamber 2. Statistics: Entries: 5, Mean: 4.2, RMS: 2.13542.
- Drift Chamber 2 X vs Y**: A scatter plot showing the relationship between X and Y coordinates for hits in the drift chamber 2. Statistics: Entries: 21, MeanX: -242.273, MeanY: -0.467175, RMS X: 85.0265, RMS Y: 219306.

Below the histograms is an output window with the following text:

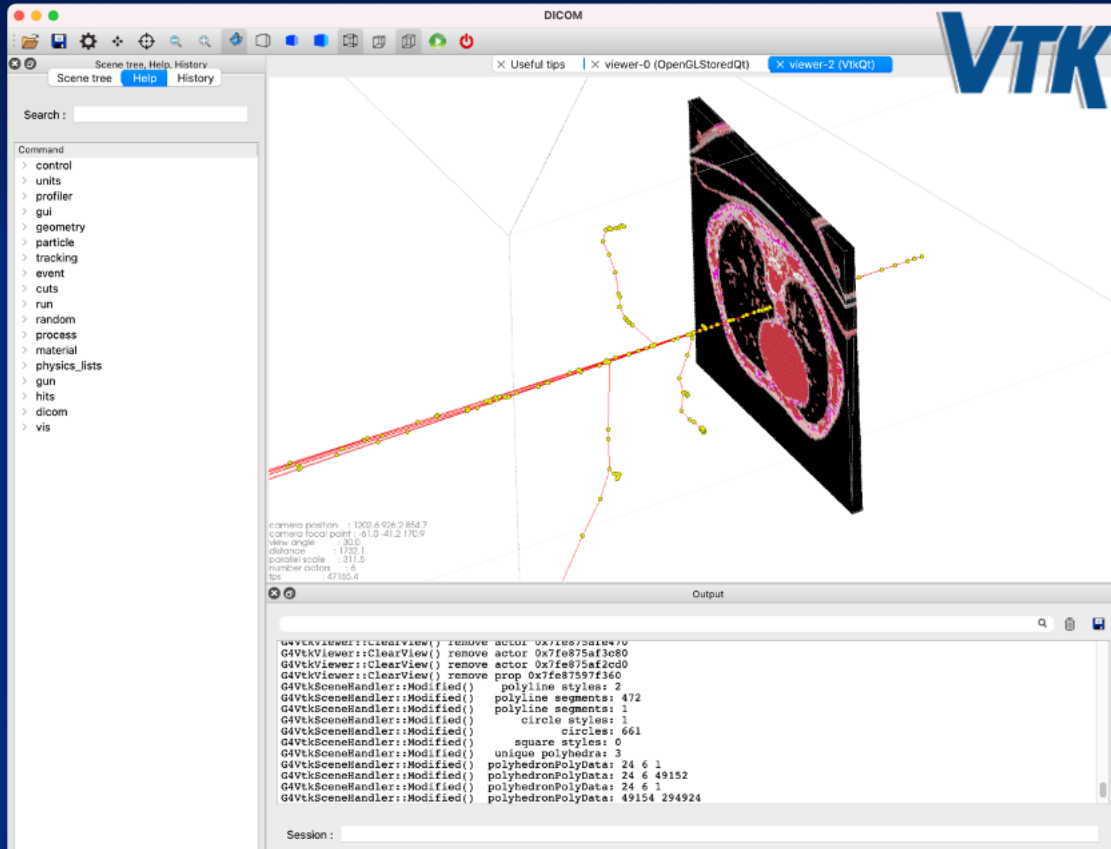
```
if on a Mac.
with ImageMagick on a Linux.
... be long if the scene as a lot of primitives):

if on a Mac.
with ImageMagick on a Linux.
... to produce an image and put it in a PostScript file:

if on a Mac.
with ImageMagick on a Linux.
```

TSG since 10.7.x  
Qt, X11, Xm, WIN32  
Interactive Histos

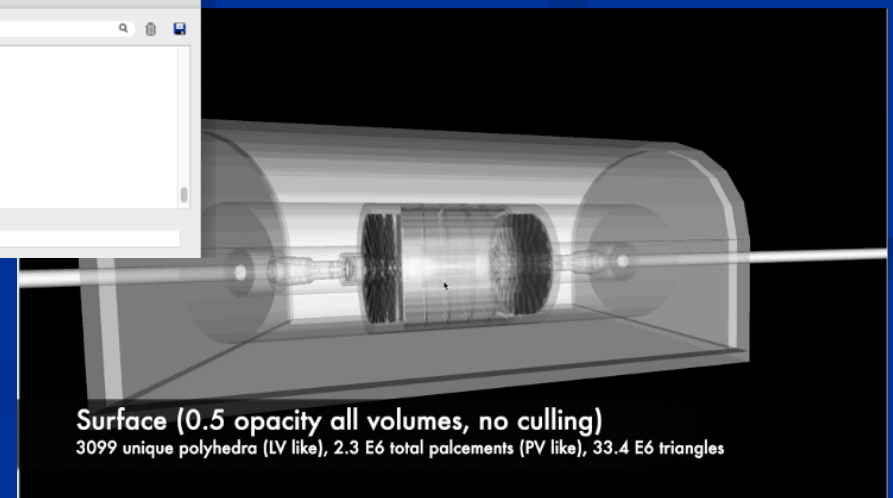
# Geant4 Visualization Drivers



## VTK visualisation Driver

- Medical
- Large GDML Files
- OBJ/GLTF output

Since 10.7



**Surface (0.5 opacity all volumes, no culling)**  
3099 unique polyhedra (LV like), 2.3 E6 total placements (PV like), 33.4 E6 triangles

# What Can be Visualized

- Simulation data can be visualized:
  - Geometrical components
  - Particle trajectories and tracking steps
  - Hits of particles in the geometry
  - Scored energy, dose, etc.
  - Histograms Plotting
- 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

# Visualization Driver Choices 1

- First I'll explain why there are so many visualization driver choices
- If you want more details about each visualization driver, see “references” at the end of this presentation

Driver	Variant	Hight quality print	Interactive	browse geometry hierarchies	Direct access to G4 kernel	Make movies	Web/3D
OpenGL	X	Green	Green	Red	Green	Green	Red
	Xm	Green	Green	Red	Green	Green	Red
	Qt	Green	Green	Green	Green	Green	Red
OpenInventor	Win32	Green	Green	Red	Green	Green	Red
	Xt	Green	Green	Red	Green	Red	Red
	Win32	Green	Green	Red	Green	Red	Red
Qt3D	Qt	Green	Green	Red	Green	Red	Red
	Qt	Green	Green	Green	Green	Green	Red
ToolsSG	X	Green	Green	Red	Green	Green	Red
	Xm	Green	Green	Red	Green	Green	Red
	Qt	Green	Green	Red	Green	Green	Red
	Win32	Green	Green	Red	Green	Green	Red

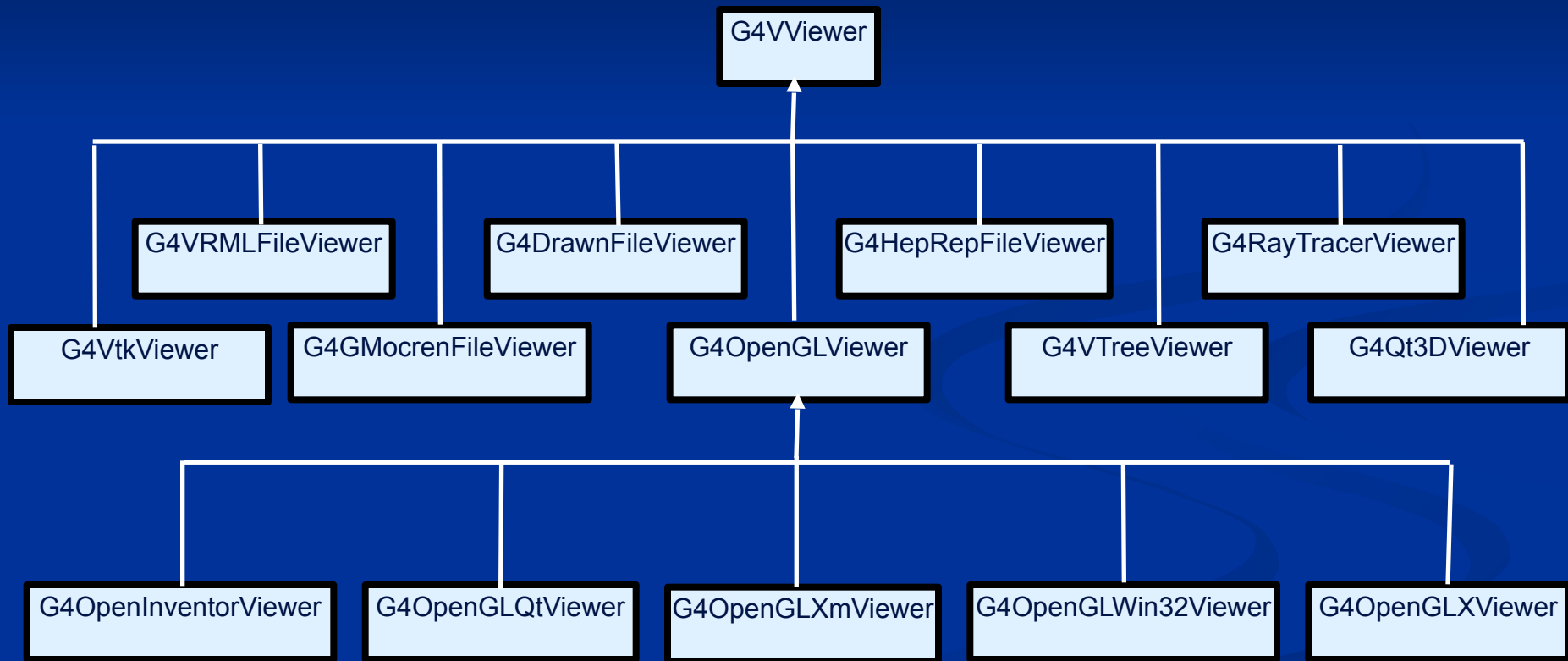


# Visualization Driver Choices 2

- First I'll explain why there are so many visualization driver choices
- If you want more details about each visualization driver, see “references” at the end of this presentation

Driver	Variant	Hight quality print	Interactive	browse geometry hierarchies	Direct access to G4 kernel	Make movies	Web/3D
VTK	Qt	Green	Green	Red	Green	Green	Green
	Native	Green	Green	Red	Green	Green	Green
VRML		Red	Red	Red	Red	Red	Green
DAWN		Green	Red	Red	Red	Red	Red
gMocren		Red	Green	Red	Red	Red	Red
RayTracer		Green	Red	Red	Red	Red	Red
ACSII File		Red	Red	Green	Green	Red	Red

# Class hierarchy diagram



# Many Visualization Drivers

- No Single Visualization Solution Can Meet all of Our Demands
  - Quick response with flexible camera control
  - High-quality Output for Publications
  - Interactive Picking to Get More Information
  - Complex Boolean Solids and Transparent or Reflective Surfaces
  - 3D Format Suitable for Web Distribution
  - Visualize Volume Data
  - Understand Geometry Hierarchies
- 
- By exploiting the same interface design that we need anyway to support visualization systems of existing frameworks
  - We are able to take advantage of the best features of several different visualization drivers
  - With a common set of user commands
  - And minimal maintenance for many of the drivers
- 
- We take advantage of the best features of many pre-existing visualization systems without having to reinvent those systems.

# 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
  - Qt3D
  - VTK
  - ToolSG
  - 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)
  - HepRepFile
  - DAWNFILE
  - VRML2FILE
  - gMocrenFile
- You can open more than one driver at a time.
  - For example, do a quick check in OpenGL, then save the same event for a beautiful DAWN plot

# Controlling Which Drivers are Available

- Six of the visualization drivers are always included by default (since they require no external libraries):
  - HepRepFile
  - DAWNFILE
  - VRMLFILE
  - RayTracer
  - gMocrenFile
  - ASCIITree
- Other visualization drivers are included only if appropriate flags are set in CMake
- 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.



# Simplest command Example

- Visualize your geometry in OpenGL:
  - `/vis/open OGL`
  - `/vis/drawVolume`
  
- Most examples come with a visualization macro more complete (including our exercise), which will be explained in more details later

# To Open Visualization

- To Open a Driver
  - `/vis/open <driver name>`
- for example
  - `/vis/open OGL`
  - `/vis/open DAWNFILE`
  - `/vis/open HepRepFile`
  - `/vis/open VRML2FILE`
- 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`

# OpenGL Additional Modes

- For all OpenGL drivers, 2 modes available :
  - Immediate mode
    - draws only to screen, no “memory”; detector can be redrawn after view changes but event data is lost.  
=> Slow if you want to rotate/move the scene
  - Stored mode
    - Stored mode: creates graphical database (display lists); nothing is lost on simple operations like change of viewing angle  
=> Slower at first draw, but faster after if you want to rotate/move the scene

exampleB2b run	Immediate mode	Stored mode
command	/vis/open OGLI	/vis/open OGLS
First draw time (sec)	1	15
Frame per sec	1	12

# References

## [Users Guide For Application Developer - Visualization](#)

[http://geant4-userdoc.web.cern.ch/geant4-userdoc/  
UsersGuides/ForApplicationDeveloper/html/Visualization/  
visdrivers.html](http://geant4-userdoc.web.cern.ch/geant4-userdoc/UsersGuides/ForApplicationDeveloper/html/Visualization/visdrivers.html)

## [DAWN download](#)

<https://geant4.kek.jp/~tanaka/>