IEEE Nuclear Science Symposium and Medical Imaging Conference

Short Course

Simulation Techniques Using Geant4

Maria Grazia Pia (*INFN Genova, Italy)*
MariaGrazia.Pia@ge.infn.it

Dresden, 18 October 2008

http://www.ge.infn.it/geant4/events/nss2008/geant4course.html

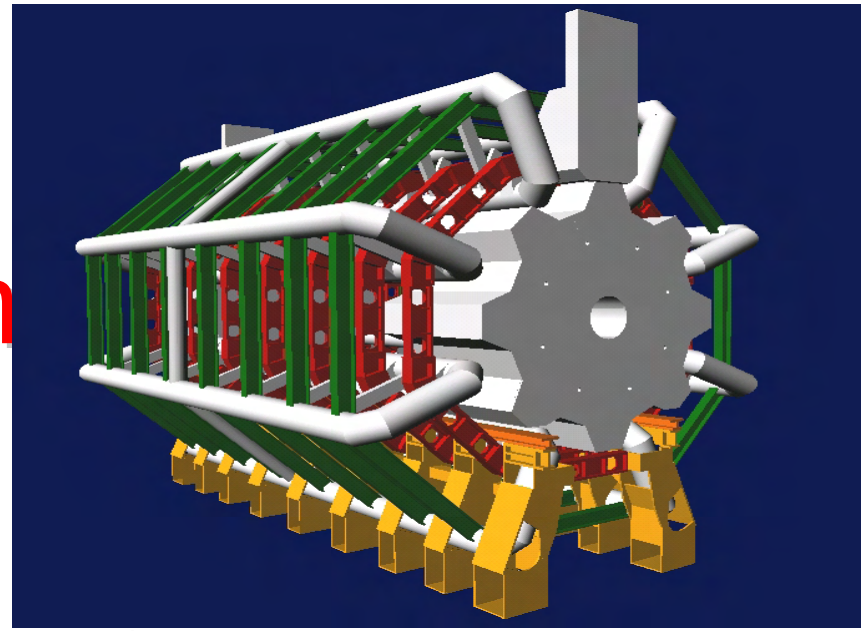
This course exploits training material developed by several Geant4 Collaboration members: thanks to all of them!

Visualisation

Basic functionality

PART I

Geant4 visualisation



. Introduction

- Geant4 Visualisation must respond to varieties of **user requirements**
  - 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

Visualisable Objects

- Simulation data you would like to see:
  - 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
- Visualisation is performed either with commands (macro or interactive) or by writing C++ source codes of user-action classes

Visualisable Objects

- You can also visualize other 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)
  - Text
    - character strings for description
    - comments or titles ...

Visualization Attributes

- Necessary for visualization, 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 (e.g. a logical volume) with its method `SetVisAttributes()` :

```
myVolumeLogical
   ->SetVisAttributes (G4VisAttributes::Invisible)
```

Assigning **G4VisAttributes** to a logical volume

- Class **G4LogicalVolume** holds a pointer of **G4VisAttributes**
- Access functions of **G4LogicalVolume**
  - **SetVisAttributes** (const G4VisAttributes* pva)
- For instance:

```
G4Colour brown(0.7, 0.4, 0.1);
G4VisAttributes* copperVisAttributes = new
  G4VisAttributes(brown);
copper_liquid_log
->SetVisAttributes(ccpperVisAttributes);
```

logical volume

Visualisation Drivers

- **Visualization drivers** are interfaces of Geant4 to 3D graphics software
- You can select your favorite one(s) depending on your purposes such as
  - Demo
  - Preparing precise figures for journal papers
  - Publication of results on Web
  - Debugging geometry
  - etc.

Available Graphics Software

Geant4 provides several visualization drivers tailored to different purposes:

- **DAWN** : Technical High-quality PostScript output
- **OPACS**: Interactivity, unified GUI
- **OpenGL**: Quick and flexible visualisation
- **OpenInventor**: Interactivity, virtual reality, etc
- **RayTracer** : Photo-realistic rendering
- **VRML**: Interactivity, 3D graphics on Web
- ...

Available Visualisation Drivers

- DAWN → Fukui Renderer DAWN
- OPENGLX → OpenGL with Xlib
- HepRep → HepRep graphics
- OIX → OpenInventor with Xlib
- RayTracer → JPEG files
- VRML → VRML 1.0/2.0
- Etc.

How to Use Visualization Drivers

- Visualization should be switched on using the variable `G4VIS_USE`
- You can select/use visualisation driver(s) by setting environmental variables before compilation, according to what is installed on your computer:
  - `setenv G4VIS_USE_DRIVERNAME 1`
- Example (DAWN, OpenGLXlib, and VRML drivers):
  - `setenv G4VIS_USE_DAWN 1`
  - `setenv G4VIS_USE_OPENGLX 1`
  - `setenv G4VIS_USE_VRML 1`

`main()` Function

To have a Geant4 executable able to **handle visualization**, you have two choices:

- Instantiate and initialize your own Visualization Manager in the `main()`
  - It must inherit from `G4VisManager` and implement the `void RegisterGraphicSystem()` method
- (Easier) Use the **`G4VisExecutive`** class available in Geant4.
  - It must be instantiated and initialized in the **`main()`** program

main() Function

```
//----- C++ source codes: Instantiation and
   initialization of G4VisManager in main()

#include "G4VisExecutive.hh“

// Instantiation and initialization of the Visualization
   Manager
#ifdef G4VIS_USE
G4VisManager* visManager =
   new G4VisExecutive;
visManager -> initialize();
#endif

#ifdef G4VIS_USE
delete visManager;
#endif
```

Includes the G4VisExecutive class

Instantiate and initialize the Visualization Manager if G4VIS_USE is “true”

Don’t forget to **delete** the pointer to G4VisExecutive at the end of main()

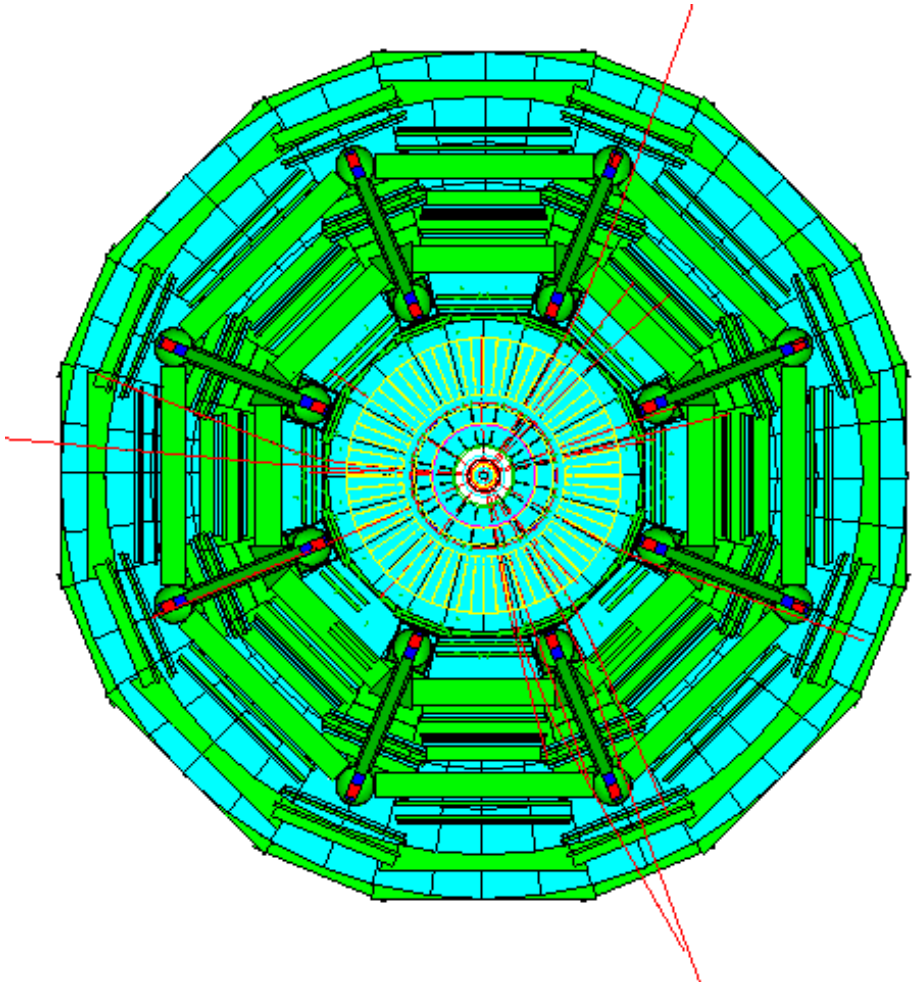
Visualisation commands

- There are some frequently-used built-in visualization commands in Geant4, that you may like to try

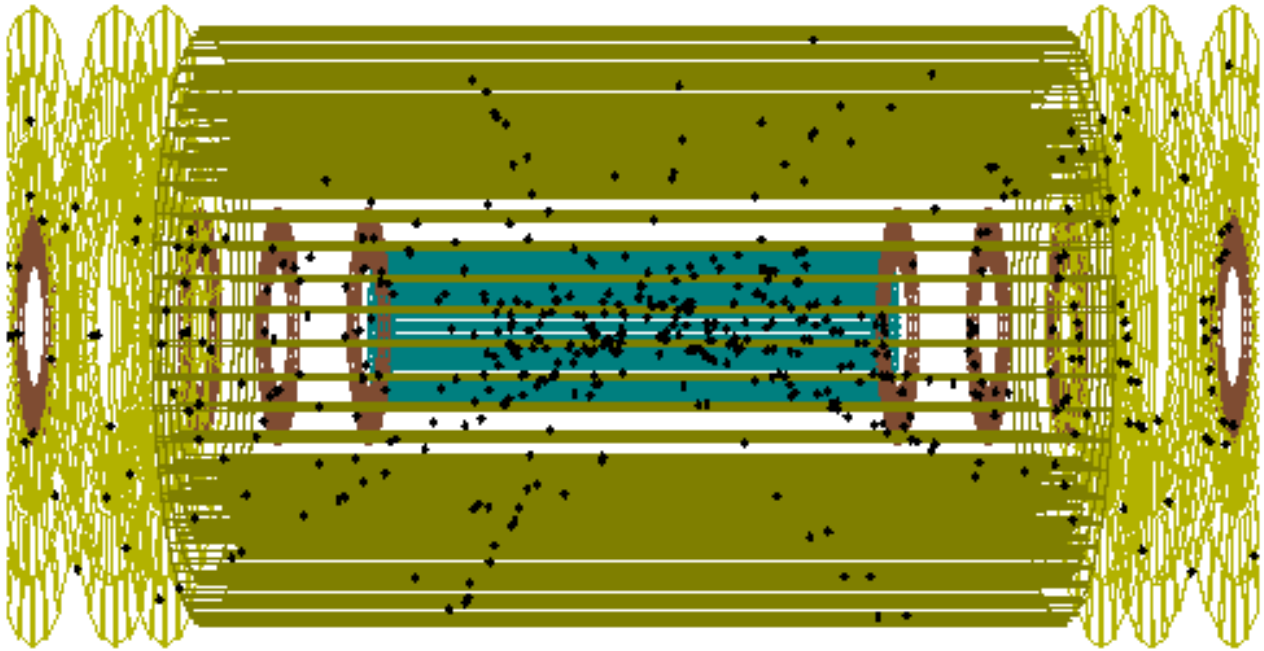
- Setting environment variables
  - `setenv G4VIS_USE_DAWN    1`
  - `setenv G4VIS_USE_OPENGLX 1`
  - `setenv G4VIS_USE_VRML`
  - `  1`
- Examples

```
/vis/drawVolume
/vis/viewer/flush
/vis/viewer/viewpointThetaPhi 70 20
/vis/viewer/set/style wireframe
```

Sample Visualization



Sample Visualization



Sample Visualization