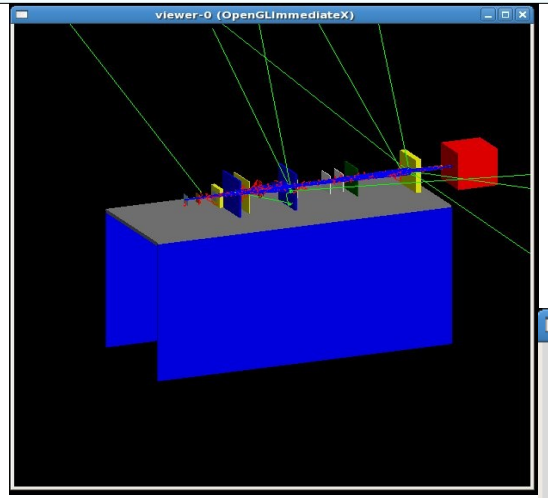
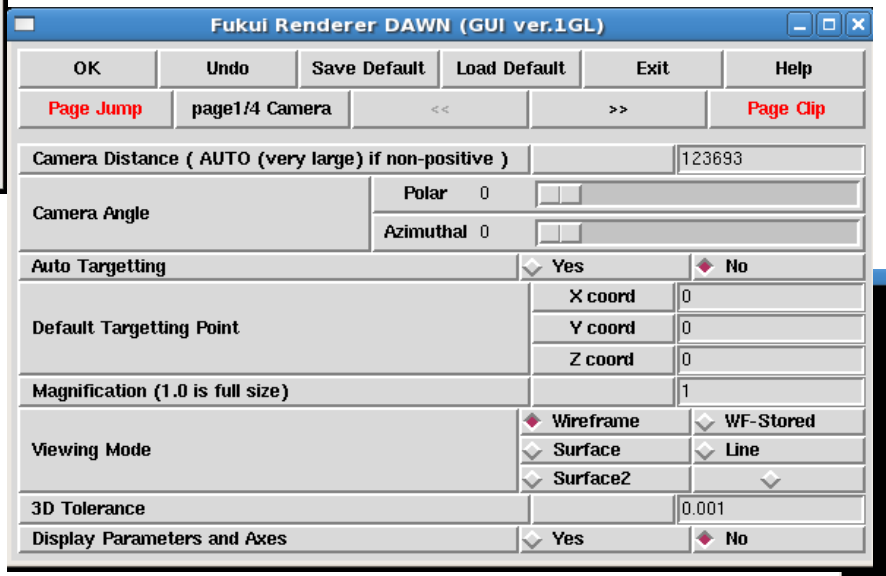
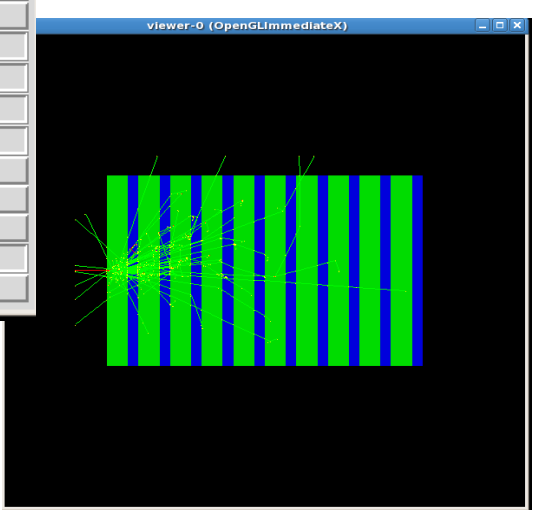
Visualizzazione in Geant4







Overview

Assicuriamoci che nel setup siano settate a 1 le variabili che attivano la compilazione: **export** (**setenv**) **G4VIS_USE_DRIVERNAME**=1
e **G4VIS_BUILD_DRIVERNAME_DRIVER**=1 prima di compilare il kernel G4.

Driver di Visualizzazione

- ***OpenGL[x]:*** Richiede le librerie grafiche OpenGL
- OpenInventor: Idem come sopra ma maggiore interattività
- **VRML*:*** Interattività
- Dawn: Genera files *postscript*
- HepRep (HepRepXml) : Wired per visualizzare
- RayTracer: Immagini simili a foto
- AsciiTree: Utile nel caso di strutture complesse, genera un albero ascii dei volumi

Help me!

I comandi di visualizzazione possono essere impartiti:
Interattivamente → macro files, G4UITerminal (idle command line)
Tramite il codice C++.

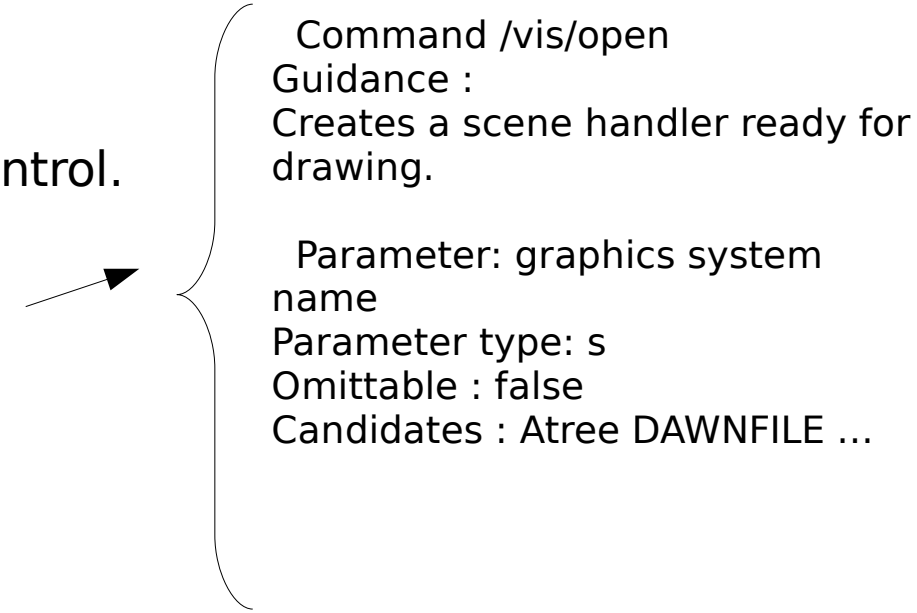
La lista di comandi di visualizzazione è molto estesa, alcuni sono specifici del driver di visualizzazione. E' consigliabile dunque usare la guida interattiva.

Digitando help al terminale **UI** → comandi /vis/:

1) /vis/ASCIITree/ Commands for ASCIITree control.

...

20) open * Creates a scene handler ready for drawing



Scene, scene handler and Viewer

- **Scene**: E' un'insieme di dati (oggetti) visualizzabili
- **Scene handler**: E' un handle associato a scene
- **Viewer**: E' il generatore di immagini finale

In ordine temporale:

- Creazione di uno *scene handler* con annesso *viewer*
- Impostazione dei parametri di scena (zoom, vista, wireframe, etc)
- [beamOn]
- Pulizia finale (nel sorgente)

Comandi interattivi Comuni

/vis -> Contenitore/directory per i comandi di visualizzazione
/vis/scene/create -> scena vuota

/vis/open <VISUALIZATION DRIVER> (è possibile aprire più driver)

Aggiungiamo visibilità per le traiettorie e Hits

/vis/scene/add/trajectories

/vis/scene/add/hits

Distinguiamo le particelle in base alla carica e visualizziamo lo step

/vis/modeling/trajectories/create/drawByCharge

/vis/modeling/trajectories/drawByCharge-0/default/setDrawStepPts true

Accumuliamo le traiettorie in memoria e simuliamo **n** eventi

/vis/scene/endOfEventAction accumulate n

/run/beamOn **n**

Colori & Particelle

E' possibile colorare le tracce in base al parent ID/carica

/gun/particle gamma
/gun/energy 10 MeV
Creiamo prima un filtro basato sul tipo di particella:
a) /vis/modeling/trajectories/create/drawByParticleID
/vis/modeling/trajectories/drawByParticleID-0/set gamma red
/vis/modeling/trajectories/drawByParticleID-0/setRGBA gamma r g b

O sulla carica:
b) /vis/modeling/trajectories/create/drawByCharge

Oppure possiamo visualizzare solo determinate particelle senza ulteriori restrizioni:

/particle/list

Creiamo prima un filtro:
/vis/filtering/trajectories/create/particleFilter

Poi gli diciamo di visualizzare solo fotoni
/vis/filtering/trajectories/create/particleFilter-0/add gamma

Accumulare eventi e Hits

/vis/scene/add/trajectories

/vis/scene/add/hits

[/vis/scene/endOfEventAction refresh]

Permettono di visualizzare traiettorie e hits in tempo reale

Il comando :

/vis/scene/endOfRun(Event)Action accumulate

mostra tutto solo alla fine di un run (singolo evento)

Ovviamente molti eventi sovrapposti non sono molto leggibili!

/vis/reviewKeptEvents

Permette di rivedere un determinato evento immagazzinato in memoria! (interfaccia con l' EventAction class)

Tramite codice C++ (permette di memorizzare solo gli eventi interessanti:

```
G4EventManager -> KeepTheCurrentEvent();
```

Controllare la geometria

Se vogliamo visualizzare l'intera [o parti della] geometria:

/vis/drawVolume [*Volume name taken from /vis/geometry/list*]

Possiamo però specificare **solo** una parte della geometria anche con:

/vis/scene/add/volume *[nome del volume **fisico**!] [numero copia]*

I valori di default sono rispettivamente: world/-1/-1

Lo stato di visibilità dei volumi può essere controllato anche dal codice C++

```
G4VisAttributes* simpleBoxVisAtt = new G4VisAttributes(G4Colour(1.0,1.0,0.0));
simpleBoxVisAtt -> SetForceSolid(true);
simpleBoxVisAtt -> SetVisibility(true);
logicCalor -> SetVisAttributes(simpleBoxVisAtt);// collego gli attributi di visualizzazione
```

Colori e grafica in C++

- Ricordarsi di includere G4VisExecutive.hh nel main
- Istanziare ed inizializzare il visualization manager:

```
#ifdef G4VIS_USE
  G4VisManager* visManager = new G4VisExecutive;
  visManager -> Initialize();
#endif
```

- Cancellando poi tutto all'uscita:

```
#ifdef G4VIS_USE
  delete visManager;
#endif
```

G4VIS_USE è definita automaticamente ad 1 se, in fase di compilazione è stata attivata la visualizzazione

I comandi interattivi possno essere inviati anche tramite UI manager

```
G4UImanager* UI = G4UImanager::GetUIpointer();
UI->ApplyCommand("/vis/...")
```

I colori dei volumi in C++

- G4VisAttributes
  - Colore
  - Visibilità
  - Solido / spigoli (solid/wireframe)

Costruttori:

- G4VisAttributes ();
- G4VisAttributes (G4bool visibility);
- G4VisAttributes (const G4Colour& colour);// rgb colors
- G4VisAttributes (G4bool visibility, const G4Colour& colour);

Metodi associati:

- void SetVisibility (G4bool); // default true
- void SetDaughtersInvisible (G4bool);
- void SetColor (G4double red, G4double green, G4double blue,
- G4double alpha = 1.); //RGB
- void SetLineStyle (LineStyle);
- void SetForceWireframe (G4bool); // this wins over interactive commands
- void SetForceSolid (G4bool);

Infine colleghiamo al volume logico d' interesse:

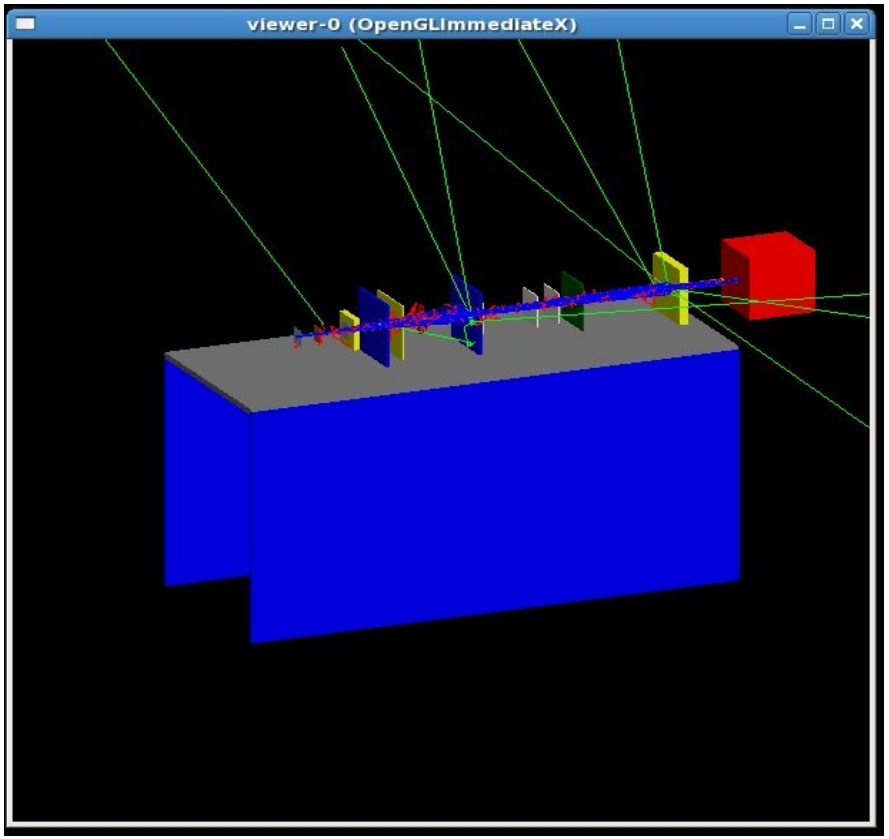
LogicalVolumePtr -> SetVisAttributes(G4VisAttributes* attrib)

LogicalVolumePtr -> SetVisAttributes(G4VisAttributes::Invisible)

OpenGL

Comando di attivazione:

/vis/open **OGLIX**



Caratteristiche

- Controllo diretto da Geant tramite macro
- Utilizza le OpenGL
- Immagini in alta risoluzione
- zoom, rotazioni, traslazioni
- Risposta veloce
- Grafica vettoriale / bitmap

I comandi /vis/viewer (*OpenGL*)

Lista dei visualizzatori correnti

/vis/viewer/list

Ripristinare le condizioni iniziali:
/vis/viewer/reset

Per cambiare il punto di vista:
/vis/viewer/set/viewpointThetaPhi 70 20

Cambia lo zoom rispetto all'ingrandimento standard:
/vis/viewer/zoomTo

Sposta la telecamera a destra e in alto rispetto al punto iniziale di riferimento
/vis/viewer/panTo

Forzare il sistema grafico a ridisegnare tutto
/vis/refresh

VRML

Virtual Reality Modeling Language

/vis/open VRML1FILE (VRML2FILE)

Caratteristiche:

- Disponibili Plugin
- Utilizza le OpenGL
- Immagini in alta risoluziione
- zoom, rotate, translate
- Risposta veloce
- Grafica vettoriale/bitmap

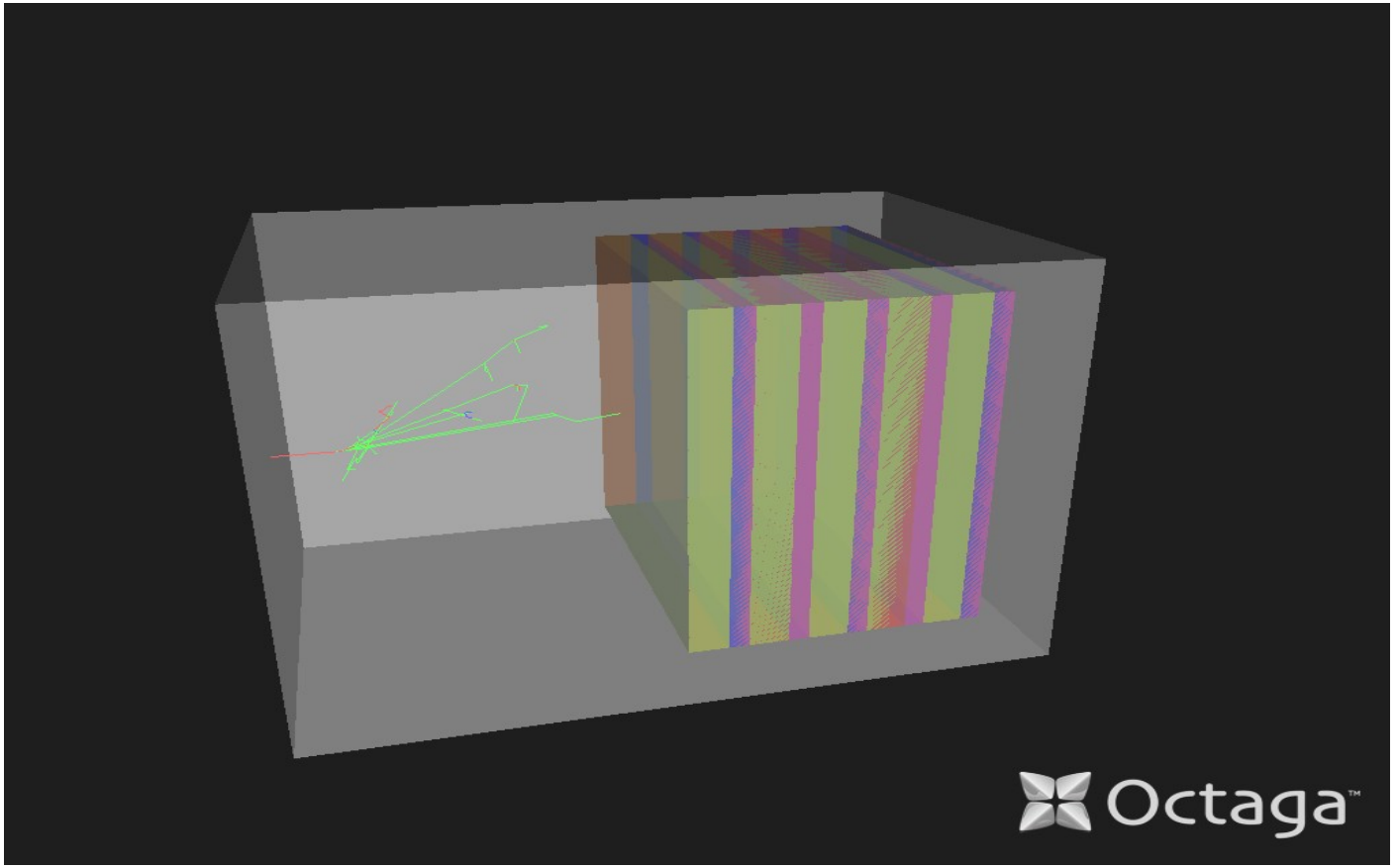
http://cic.nist.gov/vrml/vbdetect.html

Programmi Standalone (Web browser plugin)

- FreeWRL
- Octaga Player
- OpenVRML
- Vrmlview → http://ific.uv.es/~silicio/simulation/athena_installation_vrml.htm

VRML

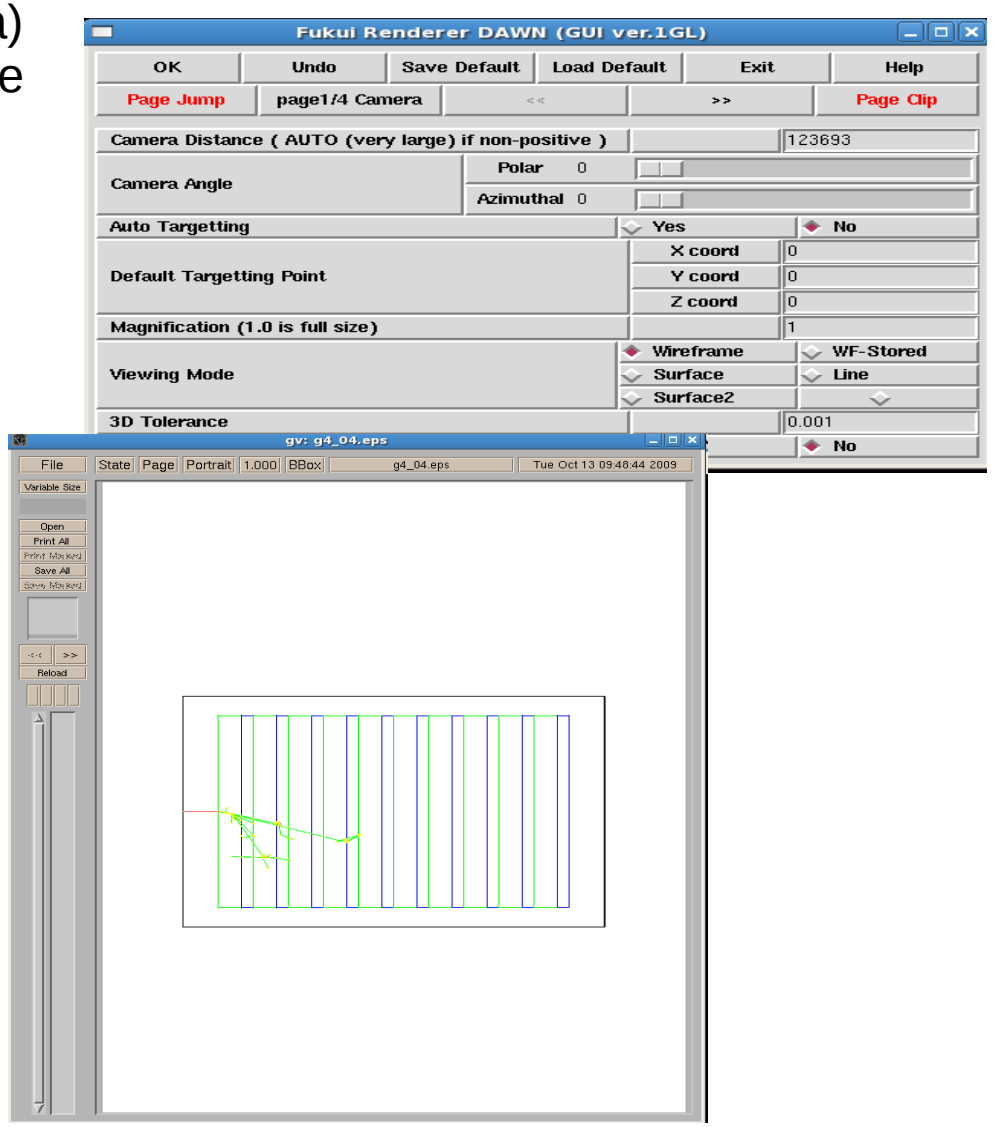
Viene prodotto un file .wrl per singolo evento, fino a G4VRMLFILE_MAX_FILE_NUM



DAWN

Dopo aver installato DAWN (by Satoshi Tanaka) ma non è necessario! DAWN genera comunque files postscript (vettoriali)

/vis/open DAWNFILE
/run/beamOn n



L' Esempio N03

- **Calorimetro** costituito da strati (**repliche**) di un **assorbitore** e un **rivelatore(detector)**.

All'interno della classe DetectorConstruction (file src/ExN03DetectorConstruction.cc) troviamo le istanze necessarie alla definizione dei volumi.

Copiare l'esempio N03:

cp -r /PATH_TO_YOUR_G4_INSTALLATION/examples/novice .

Il Visualization Manager è istanziato nel main() nel modo già visto.

Dopo aver compilato, possiamo mandare in esecuzione:

- Senza alcun parametro (e in tal caso verrà considerato “vis.mac” come parametro),
- Aggiungendo un file di macro sulla riga di comando,
- Eseguendo la macro dal terminale interattivo (any G4UITerminal)

- $G4WORKDIR/bin/Linux-g++/exampleN03 [visTutor/exN03Visxx.mac]
- idle > /control/execute visTutor/exN03VisXX.mac