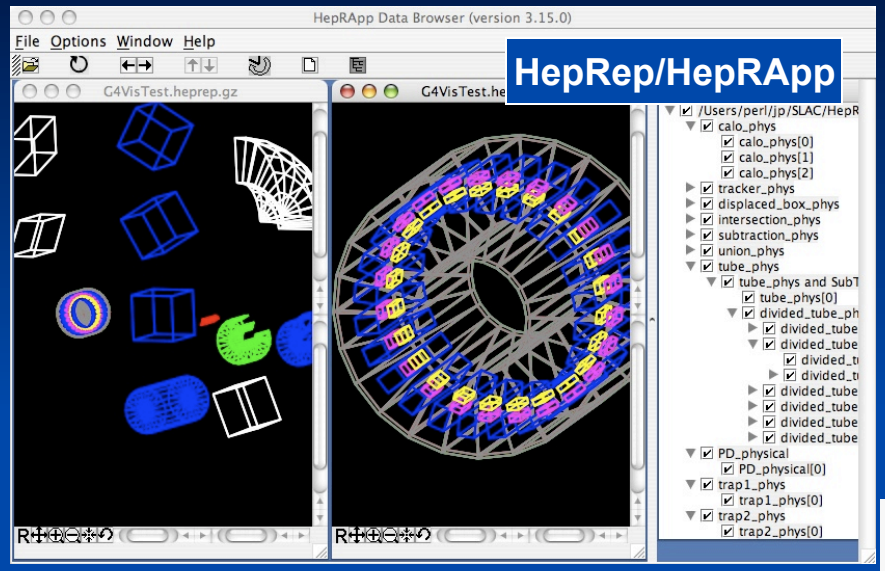
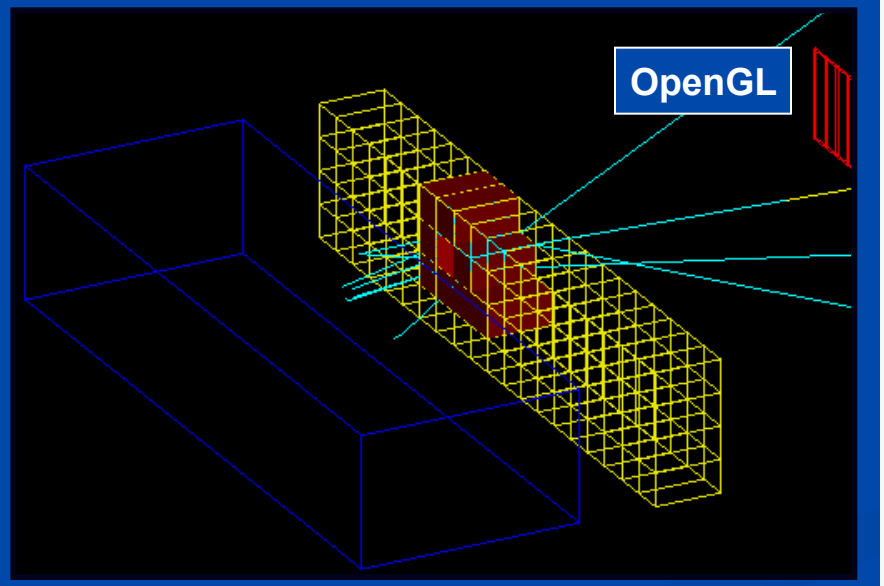
Geant4 Advanced Visualization

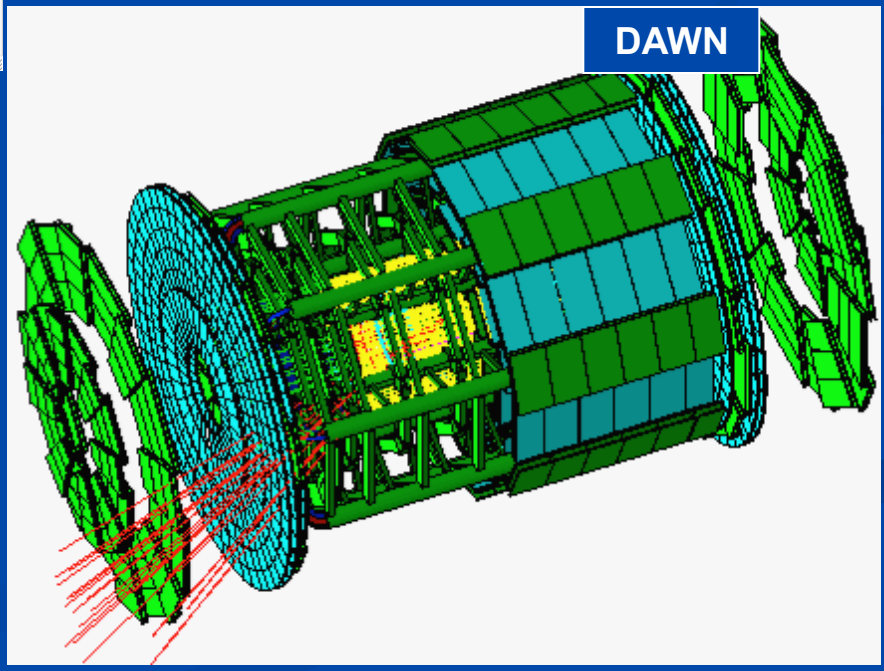
Geant4 v10.0.p1

Joseph Perl, SLAC

How to Control the Drawing Style and Filtering of Detector Geometry and Trajectories, plus a few other topics







Contents

- Visualization Attributes
  - to control color, line style, etc.
  - to represent particle type, charge, etc.
- Trajectory Modeling
- Advanced Trajectory Modeling
- Trajectory and Hit Filtering
- Additional Topics
  - Controlling level of detail in geometry
  - Section planes
  - Reviewing kept events
  - Standalone visualization

Visualization Attributes

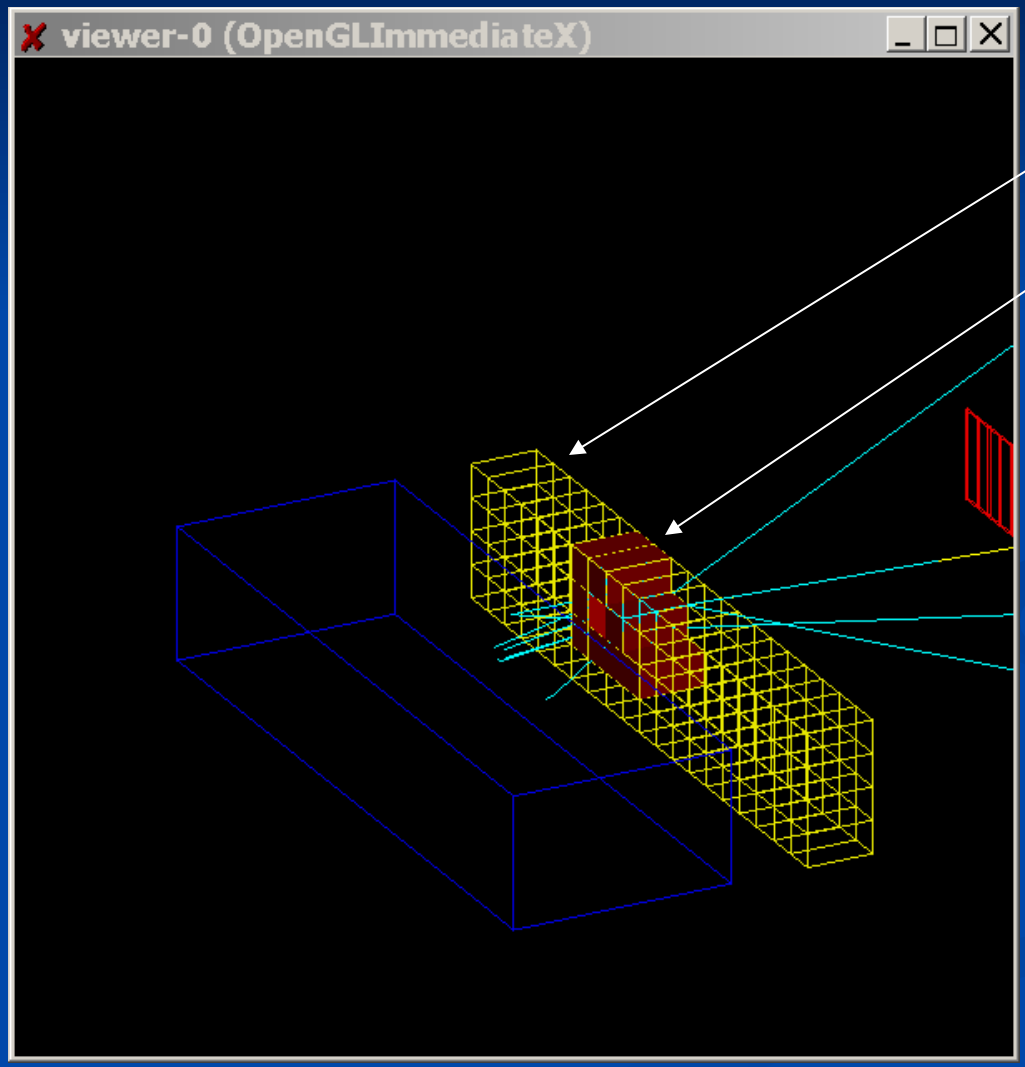
Two Kinds of Visualization Attributes

- G4VisAttributes - carry standard drawing information:
  - Color
  - Visible/Invisible
  - Wireframe/Solid, etc.
- HepRep-Style Attributes - carry arbitrary, user-defined information:
  - for trajectories:
    - momentum
    - particle type, etc.
  - for hits:
    - energy
    - electronics channel number, etc.
  - for geometry volumes:
    - logical volume name
    - material, etc.

G4VisAttributes

- G4VisAttributes - carry standard drawing information:
  - Color
  - Visible/Invisible
  - Wireframe/Solid
  - etc.
- For the first half of Geant4's life,
  these were the only kind of Visualization Attributes

LineStyle: WireFrame or Solid



Setting Geometry Vis Attributes from C++

- Create G4VisAttributes object and assign it to a volume:
  - experimentalHall_logical -> SetVisAttributes (G4VisAttributes::Invisible)

Setting Geometry Vis Attributes from Commands

- You can control the color, linewidth and other attributes of detailed geometry drawing using /vis/geometry commands such as:
  - /vis/geometry/set/colour <myvolume> blue
  - For the full set of options, see the built-in command guidance.
- A few more examples:
  - Change the line style of a particular volume to dashed
    - /vis/geometry/set/lineStyle <myvolume> dashed
  - Change the line width of all volumes to 3
    - /vis/geometry/set/lineWidth all 3
  - Change the number of line segments used to approximate a circle for all volumes to 100
    - /vis/geometry/set/forceLineSegmentsPerCircle all 0 100

Setting Overall Vis Attributes from Commands

- While the previously shown /vis/geometry commands allow the most detailed control over geometry drawing,
  the /vis/viewer commands allow you to control a few overall settings
  - /vis/viewer/set/style wireframe
  - /vis/viewer/set/globalLineWidthScale
  - etc.
  - see online command guidance for details

- Watch out for fact that interactive commands do not override C++ or /vis/geometry commands that have the “force” prefix, such as:
  - experimentalHallVisAtt->SetForceWireframe(true)
  - or
  - /vis/geometry/set/forceSolid experimentalHall

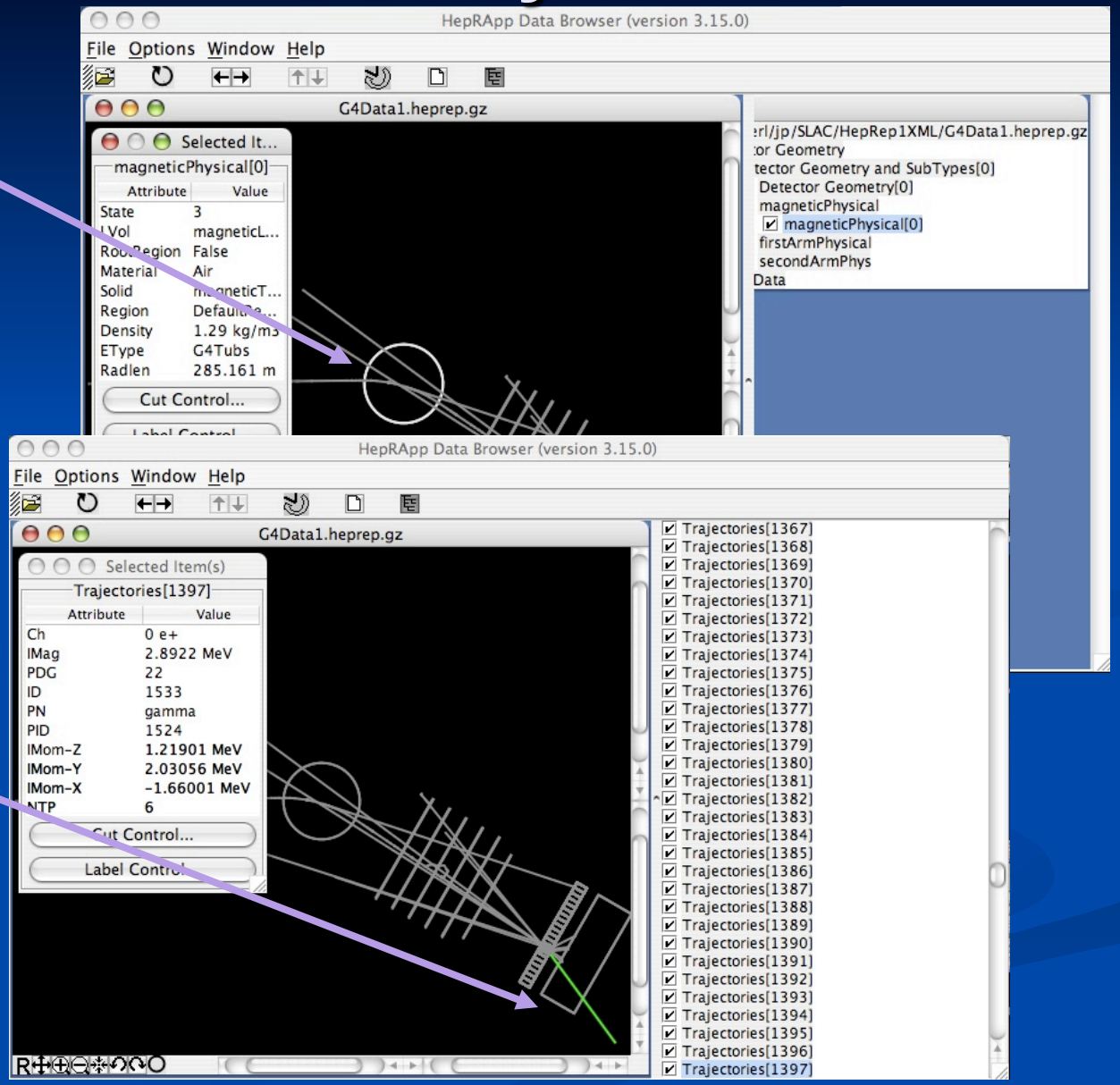
HepRep-Style Attributes

- HepRep-Style Attributes - carry arbitrary, user-defined information:
  - for trajectories:
    - momentum
    - particle type, etc.
  - for hits:
    - energy
    - electronics channel number, etc.
  - for geometry volumes:
    - logical volume name
    - material, etc.
- These attributes were added to Geant4 over the last few years to support advanced features in the HepRep browsers (HepRApp, Wired4 and FRED).
  - display the attributes when you click on the graphics object
  - perform cuts on these values
  - label objects by these values

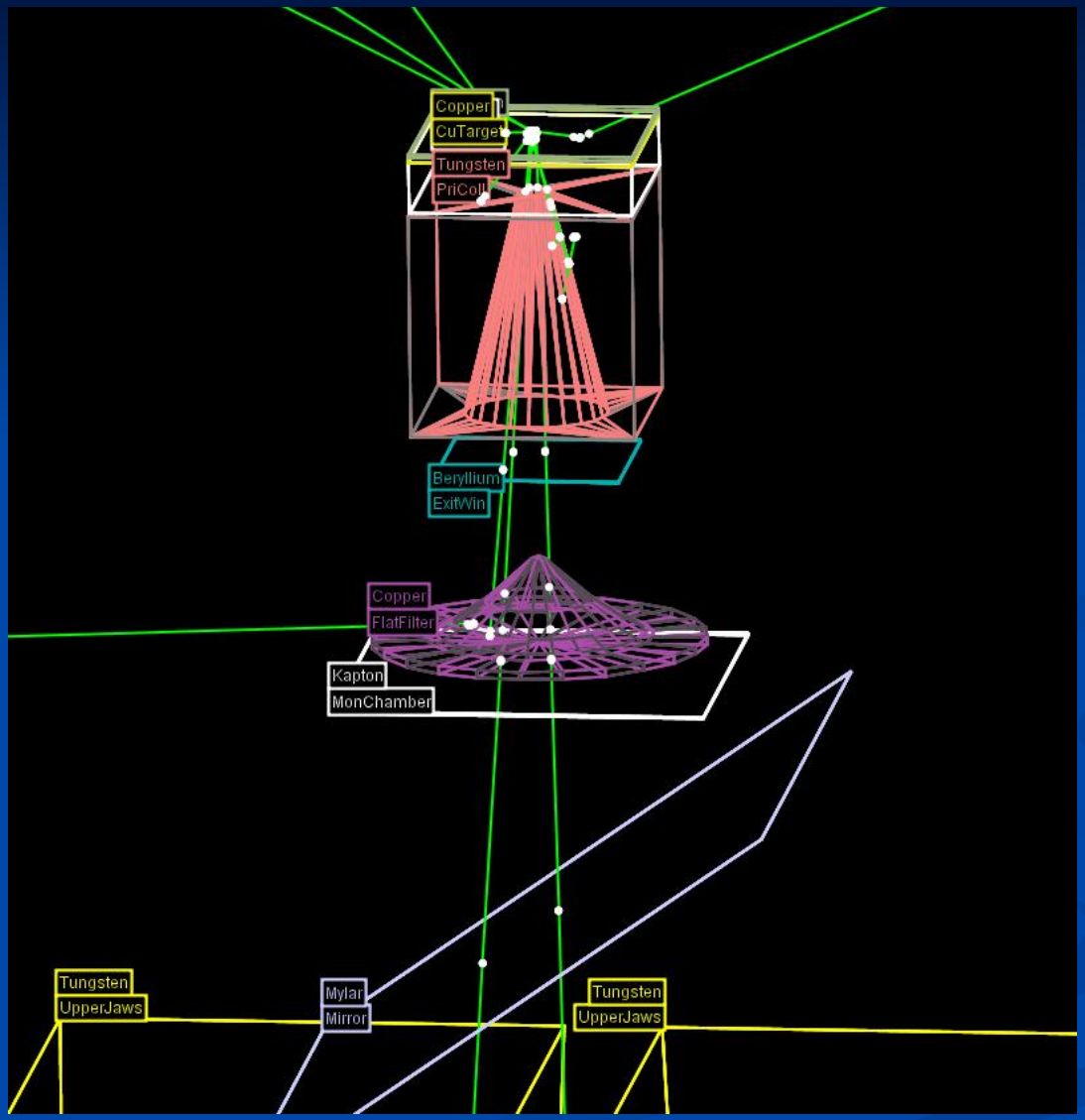
HepRApp: Pick to Show Physics Attributes

Picked on this volume to show
- **Material**
- **Density**
- **Radlen**
- **etc**

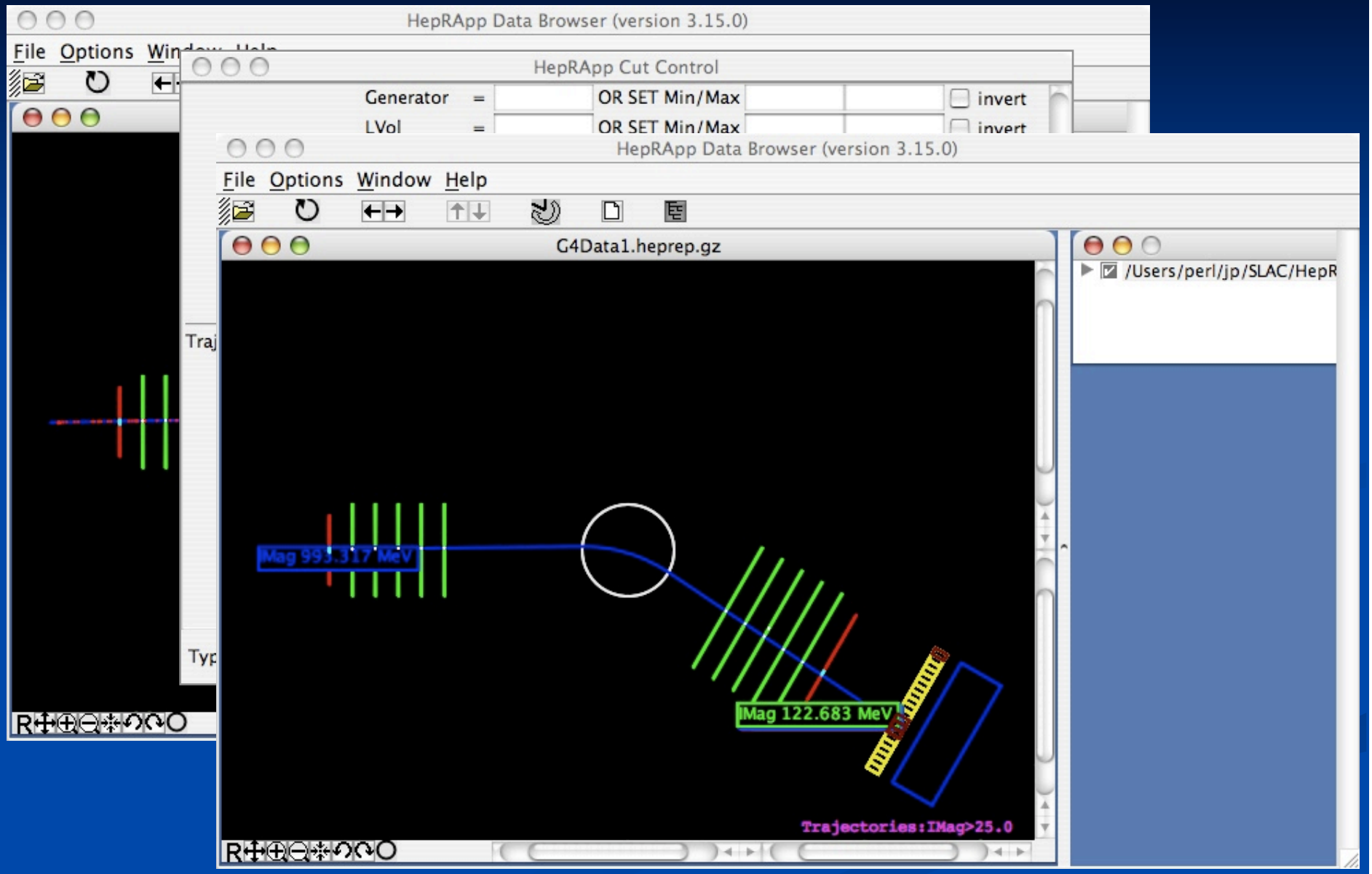
Picked on this trajectory to show
- **Particle ID**
- **Charge**
- **Momentum**
- **etc.**



HepRApp: Labeling by Any Attribute



HepRApp: Cut by Any Attribute



HepRep Attributes beyond the HepRep Browsers

- Other visualization drivers can also use the HepRep-style attributes:
  - Qt and Open Inventor will show the attributes when you click on the relevant object
  - OpenGL also does this, with attributes shown in standard output window

- Advanced visualization features described later in this presentation also use HepRep attributes and work for all Geant4 visualization drivers:
  - Trajectory Modeling
  - Trajectory and Hit Filtering

Defining Your Own HepRep Attributes

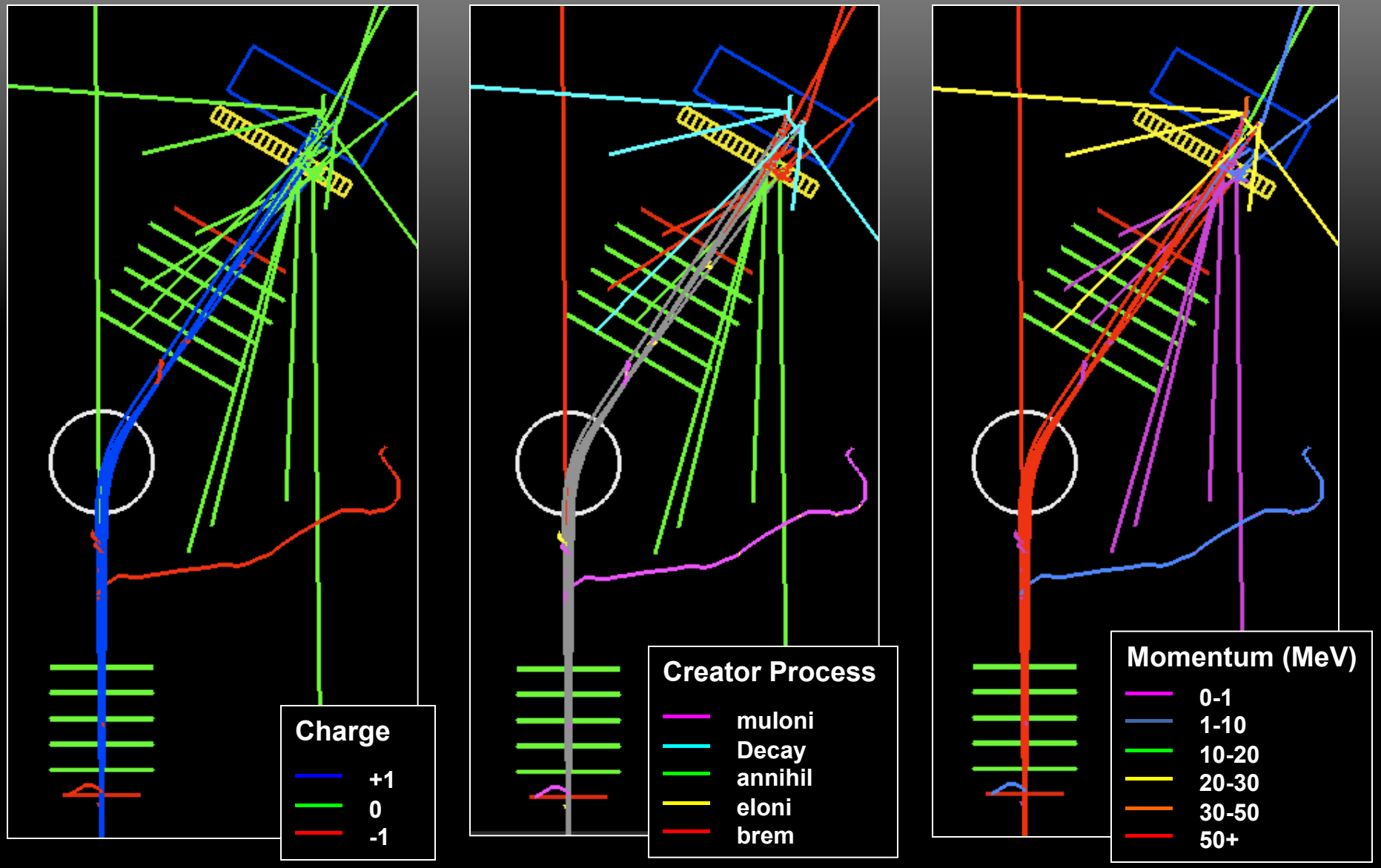
- While a useful set of attributes is already defined by default in the standard Geant4 trajectory, you are also free to define your own attributes:
  - std::map<G4String,G4AttDef>* store = G4AttDefStore::GetInstance("G4Trajectory",isNew);
  - G4String PN("PN");
  - (*store)[PN] = G4AttDef(PN,"Particle Name","Physics","","G4String");
  - G4String IMom("IMom");
  - (*store)[IMom] = G4AttDef(IMom, "Momentum of track at start of trajectory", "Physics","","G4ThreeVector");
- Then fill the attributes with lines such as:
  - std::vector<G4AttValue>* values = new std::vector<G4AttValue>;
  - values->push_back(G4AttValue("PN",ParticleName,""));
  - s.seekp(std::ios::beg);
  - s << G4BestUnit(initialMomentum,"Energy") << std::ends;
  - values->push_back(G4AttValue("IMom",c,""));
- See geant4/source/tracking/src/G4Trajectory.cc for a good example.

Trajectory Modeling

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
- Most examples include a vis.mac that demonstrates some of these features

Example A01, five events, drawBy various models



Standard Models Supplied with Geant4

- drawByCharge
  - Default model
  - Colors trajectories according to charge

+1	Blue
-1	Red
0	Green

- drawByParticleID
  - Colors trajectories according to particle type
  - Configure to highlight chosen particle types with chosen colors
  - No limit on the number of particle types that can be highlighted
- drawByOriginVolume
  - Colors trajectories according to volume in which they originated
  - for example, to highlight all particles that backscattered from a particular collimator
- generic
  - Draws all trajectories the same
- control over more than just color, including:
  - Whether to show trajectory as line, step points or both
  - width of trajectory lines, type of marker to use for points, point size, etc.

Model Configuration

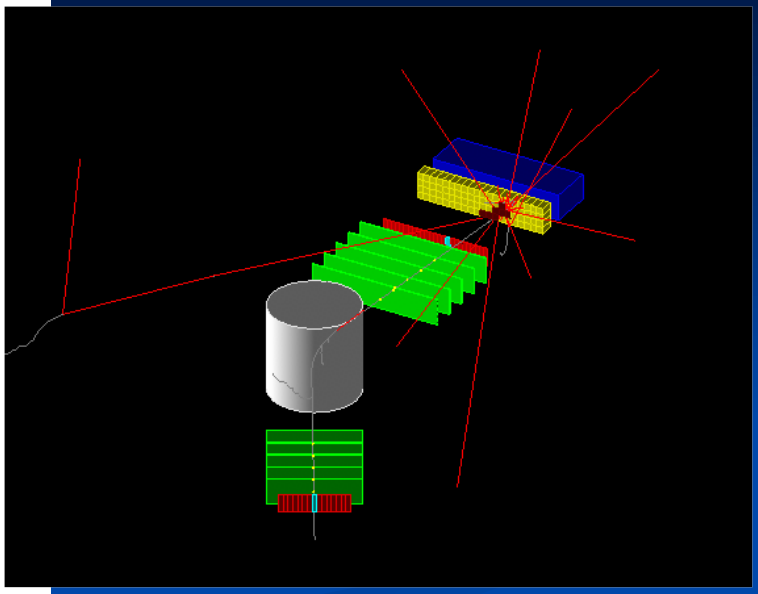
- You can create and configure multiple models through either
  - ➔ Interactive commands
  - ➔ Compiled code
- Interactive commands
  - Located in /vis/modeling/trajectories directory
  - Possible to have multiple instances of given model type
    - e.g., toggle between two different ways of coloring by charge
  - List and select instantiated models with commands:
    - ➔ /vis/modeling/trajectories/list
    - ➔ /vis/modeling/trajectories/select <model-instance-name>
- Note that detailed help for a given model is not shown in the help system until you have created an instance of that model
  - e.g., only after you do your first:
    - /vis/modeling/trajectories/create/drawByCharge
  - will the help system will include details on drawByCharge

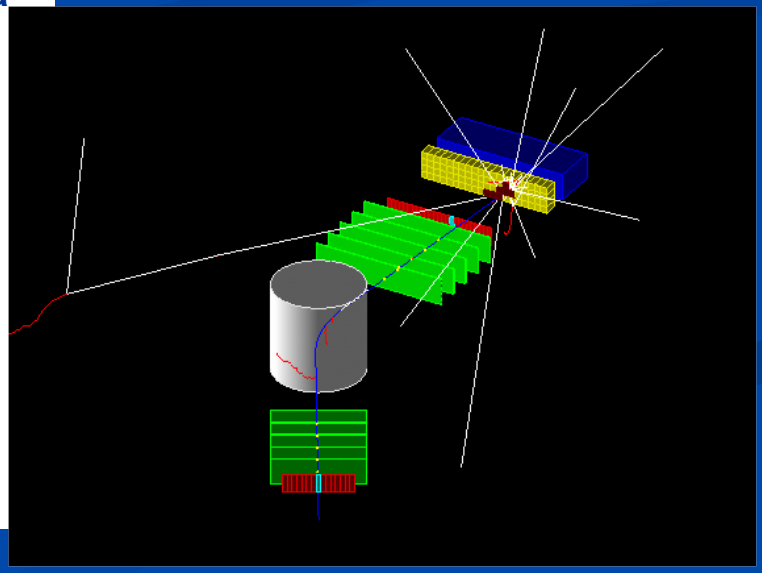
Trajectory Modeling Examples

Example macro
#Standard setup
/vis/scene/create
/vis/open OGL
/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, coloring neutrals 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





More 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

Default colors correspond to:
/vis/modeling/trajectories/drawByParticleID-0/set e- red
/vis/modeling/trajectories/drawByParticleID-0/set e+ blue
/vis/modeling/trajectories/drawByParticleID-0/set proton cyan
/vis/modeling/trajectories/drawByParticleID-0/set gamma green
/vis/modeling/trajectories/drawByParticleID-0/set neutron yellow
/vis/modeling/trajectories/drawByParticleID-0/set pi+ magenta
/vis/modeling/trajectories/drawByParticleID-0/set pi- magenta
/vis/modeling/trajectories/drawByParticleID-0/set pi0 magenta
```

More Sample Commands: drawByCharge Model and Toggling Between Two Models

```
# 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

# Configure these two drawByCharge models and visualize using either one of them

# 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 (showing different way of setting colors)
/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
```

Trajectory Modeling by HepRep-Style Attributes

- For even greater flexibility, you can model the trajectory based on any of the HepRep-style attributes, whether it is one of the default ones or an attribute that you have defined for yourself.
  - E.g., set color based the value of the attribute CPN (for "creator process name")

```
/vis/modeling/trajectories/create/drawByAttribute
/vis/modeling/trajectories/drawByCharge-0/verbose true
/vis/modeling/trajectories/drawByCharge-0/setAttribute CPN
```

Draw by the attribute called CPN

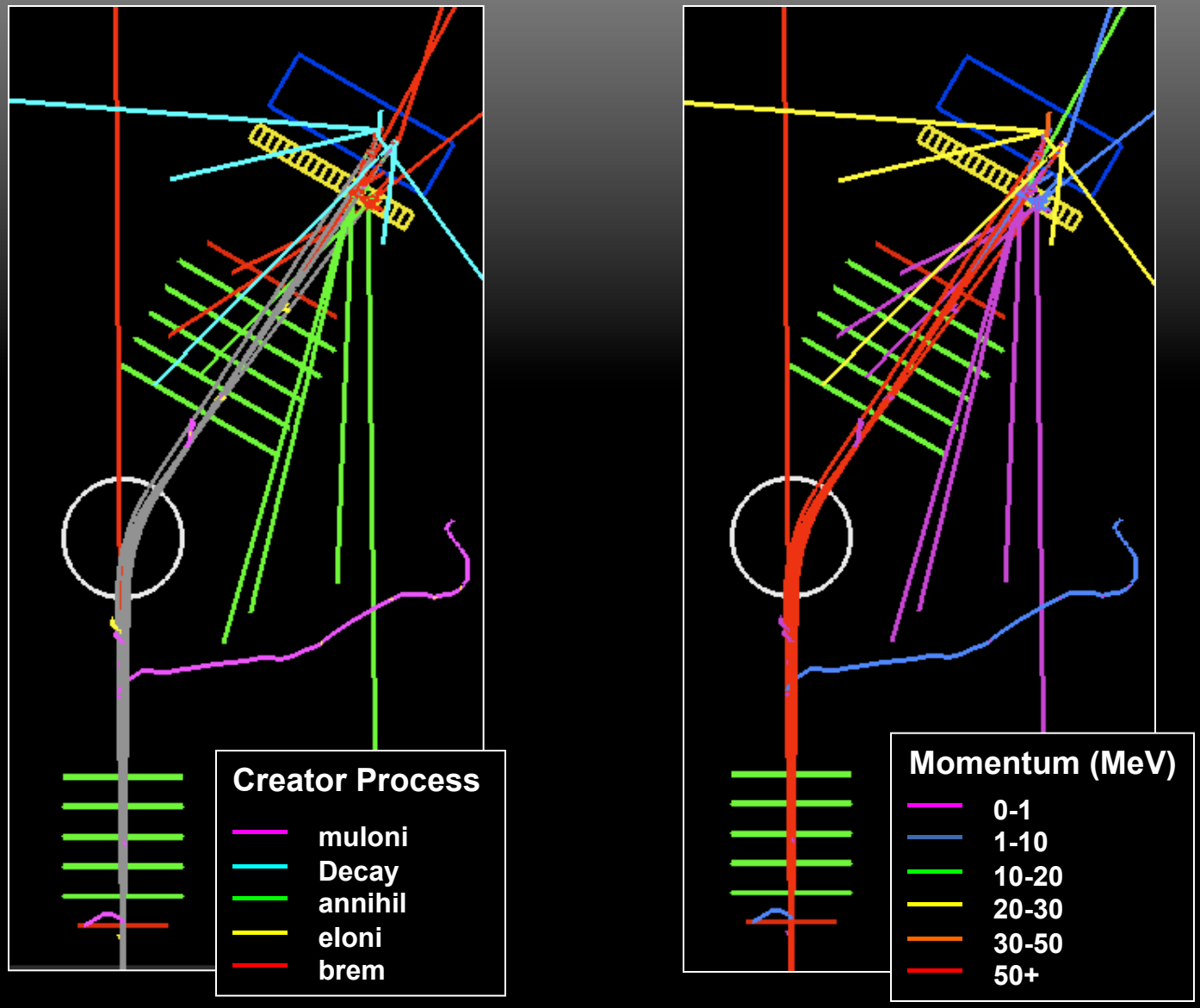
```
/vis/modeling/trajectories/drawByAttribute-0/addValue brem_key  eBrem
/vis/modeling/trajectories/drawByAttribute-0/addValue annihil_key annihil
/vis/modeling/trajectories/drawByAttribute-0/addValue decay_key Decay
/vis/modeling/trajectories/drawByAttribute-0/addValue muIon_key muIoni
/vis/modeling/trajectories/drawByAttribute-0/addValue eIon_key  eIoni
```

Select relevant attribute values

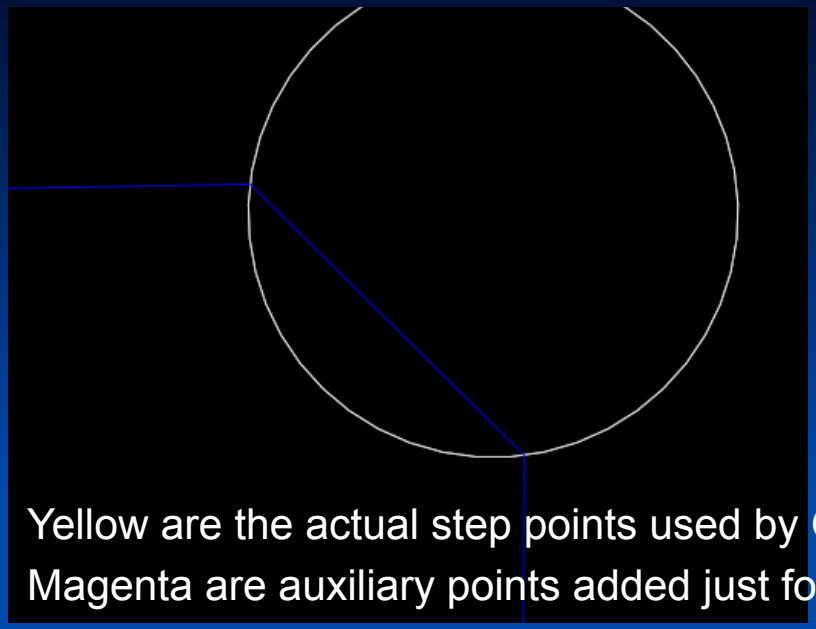
```
/vis/modeling/trajectories/drawByAttribute-0/brem_key/setLineColour     red
/vis/modeling/trajectories/drawByAttribute-0/annihil_key/setLineColour  green
/vis/modeling/trajectories/drawByAttribute-0/decay_key/setLineColour    cyan
/vis/modeling/trajectories/drawByAttribute-0/eIon_key/setLineColour     yellow
/vis/modeling/trajectories/drawByAttribute-0/muIon_key/setLineColour magenta
```

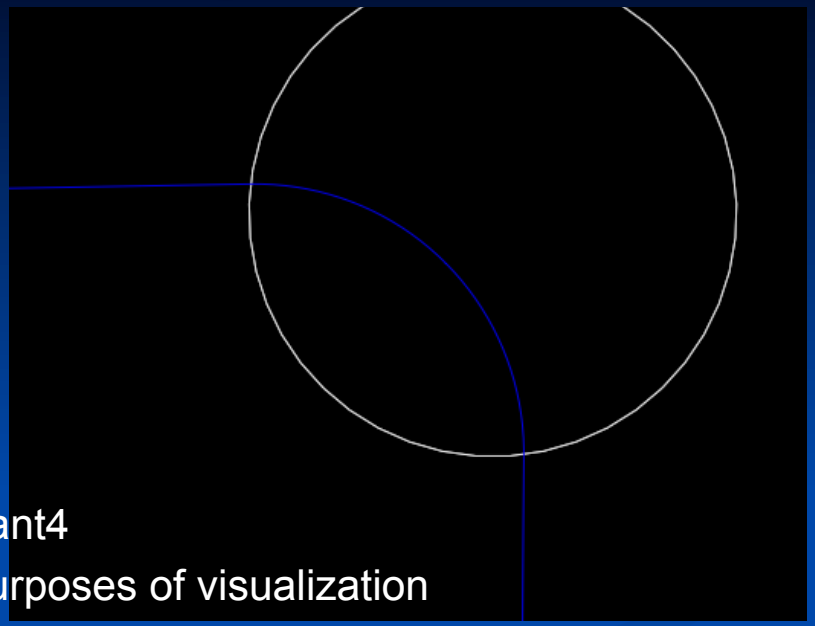
Configure visualisation parameters

Example A01, five events, drawByAttribute models



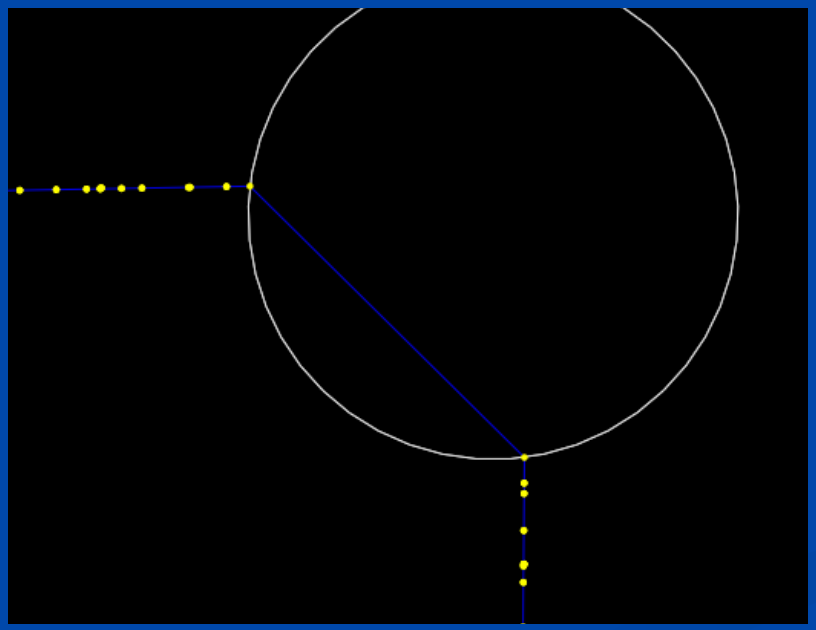
Trajectory Lines, Step Points, Regular and Smooth

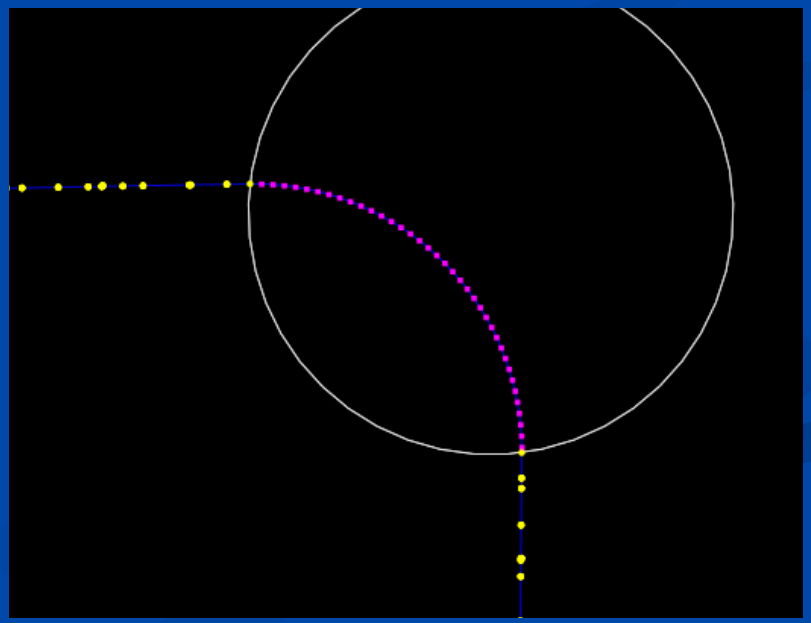




Yellow are the actual step points used by Geant4

Magenta are auxiliary points added just for purposes of visualization





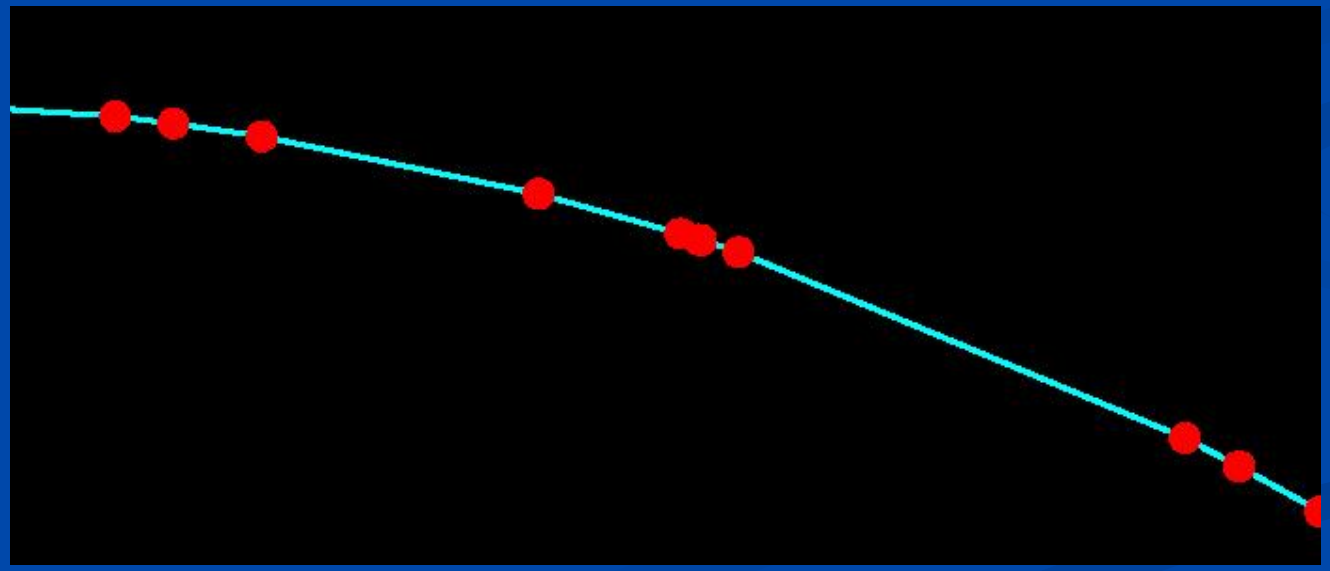
Modeling Trajectory as Line, Step Points or Both

- In the discussion of models up to now, we've only shown you how to set the color, but a model can include many more options including whether to show the trajectory as a line, a set of step points or both:
  - setDrawLine * Set draw line command
  - setLineVisible * Set line visibility command
  - setLineColour * Set colour through a string
  - setLineColourRGBA * Set colour through red, green, blue and alpha components
  - setDrawStepPts * Set draw step points command
  - setStepPtsVisible * Set step points colour command
  - setStepPtsColour * Set colour through a string
  - setStepPtsColourRGBA * Set colour through red, green, blue and alpha components
  - setStepPtsSize * Set step points colour command
  - setStepPtsType * Set step points type.
  - setStepPtsFillStyle * Set step fill style type.
  - (the following items relate to special kind of points discussed later under “smooth trajectory”)
  - setDrawAuxPts * Set draw auxiliary points command
  - setAuxPtsVisible * Set auxiliary points visibility command
  - setAuxPtsColour * Set colour through a string
  - setAuxPtsColourRGBA * Set colour through red, green, blue and alpha components
  - setAuxPtsSize * Set auxiliary points size command
  - setAuxPtsType * Set auxiliary points type.
  - setAuxPtsFillStyle * Set auxiliary fill style.

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
```



Advanced Trajectory Modeling

Controlling Model from Compiled Code

- Instantiate model
- Configure model
- Register with visualization manager

main.cc

```
// Create and initialise visualization manager
G4VisManager* visManager = new G4VisExecutive;
visManager->Initialize();

// Create new drawByParticleID model
G4TrajectoryDrawByParticleID* model =
   new G4TrajectoryDrawByParticleID;

// Configure model
model->SetDefault("cyan");
model->Set("gamma", "green");
model->Set("e+", "magenta");
model->Set("e-", G4Color(0.3, 0.3, 0.3));

//Register model with visualization manager
visManager->RegisterModel(model);
```

Defining Your Own Model

- New trajectory models must inherit from G4VTrajectoryModel and implement these pure virtual methods:

```
virtual void Draw(const G4VTrajectory&, G4int i_mode = 0) const = 0;
virtual void Print(std::ostream& ostr) const = 0;
```

- New models can be used directly in compiled code
  - ➔ Need to be registered with visualization manager

main.cc

```
// Create custom model
MyCustomTrajectoryModel* myModel = new MyCustomTrajectoryModel("custom");

// Configure it if necessary and then register with G4VisManager
...
visManager->RegisterModel(myModel)
```

To Make User Defined Model Available from Interactive Commands

- You will need to write Messenger classes
  - Messengers to configure the model should inherit from G4VModelCommand. The concrete trajectory model type should be used for the template parameter

G4ModelCommandDrawByParticleIDSet.cc

```
class G4ModelCommandDrawByParticleIDSet : public
                : G4VModelCommand<G4TrajectoryDrawByParticleID> {
 ...
 };
```

- and a Factory class
  - A factory class responsible for the model and associated messenger creation must also be written. The factory should inherit from G4VModelFactory. The abstract model type should be used for the template parameter, e.g.:

G4TrajectoryDrawByChargeFactory.cc

```
class G4TrajectoryDrawByChargeFactory
  : public G4VModelFactory<G4VTrajectoryModel> {
 ...
};
```

Construct the Model and Associated Messengers

G4TrajectoryDrawByParticleIDFactory.cc

```
ModelAndMessengers
G4TrajectoryDrawByParticleIDFactory::Create(const G4String& placement, const G4String& name) {
 // Create model with given name
 G4TrajectoryDrawByParticleID* model = new G4TrajectoryDrawByParticleID(name);

  // Create associated messengers with commands in ``placement'' command directory.
  Messengers messengers;
  messengers.push_back(new G4ModelCommandDrawByParticleIDSet(model, placement));
  ...
  return ModelAndMessengers(model, messengers);
}
```

G4VisExecutive.cc

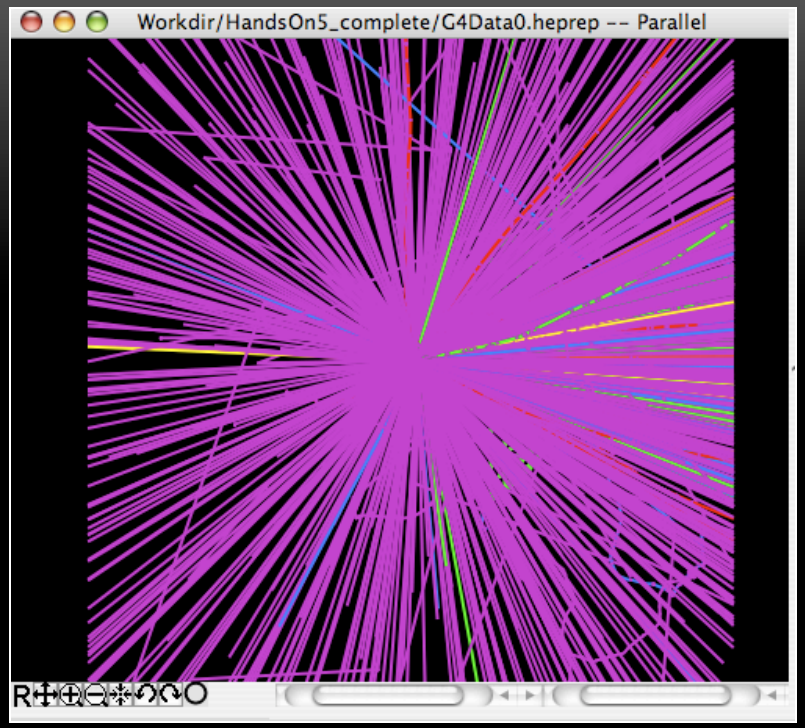
```
G4VisExecutive::RegisterModelFactories() {...
   RegisterModelFactory(new G4TrajectoryDrawByParticleIDFactory());
}
```

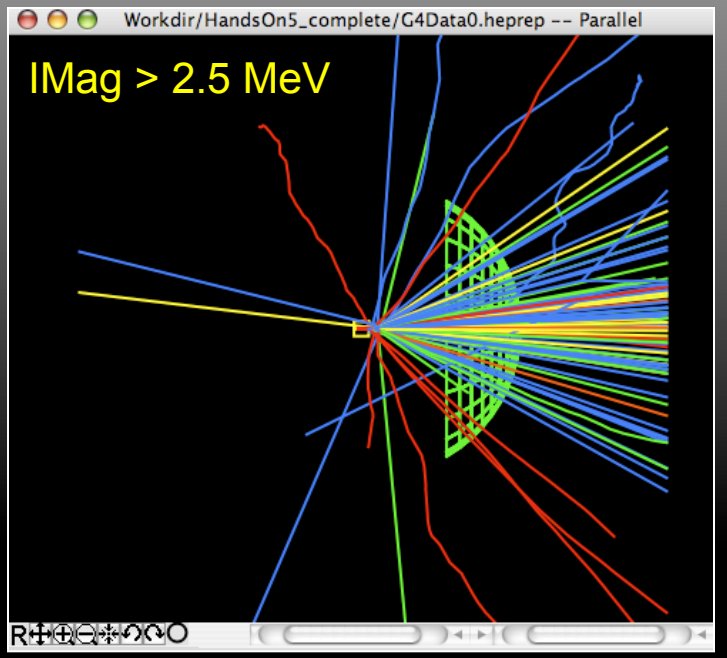
Trajectory and Hit Filtering

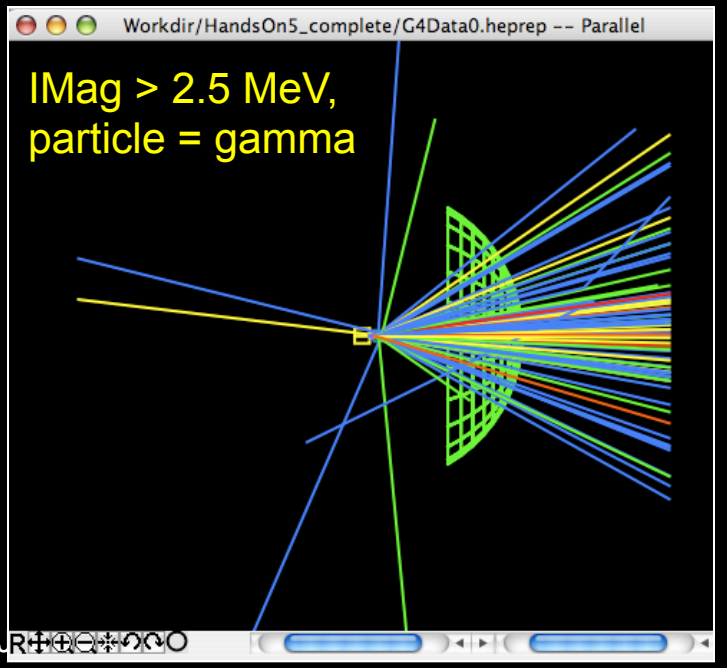
Trajectory and Hit Filtering

- Display user-defined subset of trajectories
  - Solves problems with overly busy graphics or excessively large graphics files
- Similar structure to enhanced trajectory drawing
  - Set of simple filter models
  - Similar Interactive creation/configuration structure
- chargeFilter
  - ➔ Filters trajectories according to charge
- particleFilter
  - ➔ Filters trajectories according to particle type
- originVolumeFilter
  - ➔ Filters trajectories according to volume in which they originated
- Project Lead: Jane Tinslay

HandsOn5, McGill tutorial, 1000 events, Attribute Filter







Momentum (MeV)

- 0-2.5
- 2.5-5
- 5-7.5
- 7.5-10
- 10-10.25
- 12.5+

Trajectory Filtering

- Simplest example:
  - /vis/filtering/trajectories/create/particleFilter
  - /vis/filtering/trajectories/particleFilter-0/add e-
  - will cause everything except electrons to be filtered out

- You can chain multiple filters
  - e.g., filter out gammas
  - and filter out particles with momentum less than 100 MeV

- Two modes of filtering:
  - Important issue when working with those visualization drivers that allow you to turn visibility on and off from the vis application (HepRApp or OpenInventor)
  - One mode has rejected trajectories marked invisible but still sent to vis app
    - user can toggle them back to visible from within the vis app
    - but if there is a very large number of these invisible trajectories, application may be slowed down and files may be very large
  - Other mode has rejected trajectories entirely omitted
    - user cannot toggle them back to visible from within the vis app
    - but application stays fast and files stay small

Filtering Example

```
/vis/modeling/trajectories/drawByAttribute-0/setAttribute IMag
/vis/modeling/trajectories/drawByAttribute-0/addInterval interval1 0.0 keV 2.5MeV
/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:

```
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

```
void MyHit::Draw() {
  …
  if (! pVVisManager->FilterHit(*this)) return;
  …
```

Additional Topics

Controlling Detail Level of Detector Geometry

- By default, /vis/drawVolume will draw the entire detector geometry. This is equivalent to the commands:
  - /vis/scene/create
  - /vis/scene/add/volume world
- You can specify additional arguments to limit the amount of geometry detail shown:
  - /vis/scene/add/volume [<physical-volume-name>] [<copy-no>] [<depth-of-descending>]
    - 1st parameter: volume name (default "world").
    - 2nd parameter: copy number (default -1 meaning first occurrence of physical-volume-name is selected.
    - 3rd parameter: depth of descending geometry hierarchy (default G4Scene::UNLIMITED (-1)).
- Still more arguments can be given to specify a clipping volume.
  - vis/scene/add/volume world -1 -1 box km 0 1 0 1 0 1 will draw the world with the positive octant cut away.

Even more Control over Level of Detail in Detector Geometry

- Additional commands allow finer control including whether or not to draw Boolean components, voxels and readout geometries:
  - /vis/specify <logical-volume-name> [depth-of-descent] [<booleans-flag>] [<voxels-flag>] [<readout-flag>]
  - /vis/scene/add/logicalVolume <logical-volume-name> [<depth-of-descending>] [<voxels-flag>] [<readout-flag>]
- Culling allows you to specify that covered daughters or low density volumes are omitted:
  - /vis/viewer/set/culling global|coveredDaughters|invisible|density [true|false] [density] [unit]
  - HepRepFile will still include these culled objects, but just make them initially invisible.
    - Idea is that you might later decide you want to see these.
    - To really omit them from the HepRepFile, as you may wish to do to make the file smaller:
      - /vis/heprep/setCullInvisibles true

Section Planes / Cutaways

- Some drivers allow you to section the view, that is, cut it away along a specified plane (but this generally works only for simple geometries)
  - /vis/viewer/set/sectionPlane [on|off]
    <3 vector of point> [unit of point] <3 vector of plane normal>
  - e.g., for a y-z plane at x = 1 cm:
    /vis/viewer/set/sectionPlane on 1 0 0 cm 1 0 0

Reviewing Kept Events

- If you have accumulated several events in your visualization, you can still go back afterwards and view the events individually. For each event, you can execute various vis commands to rotate, zoom, output to a different vis driver, etc.
  - /vis/reviewKeptEvents
  - Each time you type “continue”, you will get to the next kept event.
- To quit reviewing events:
  - /vis/abortReviewKeptEvents
  - and then again type “continue”
- You can also use a command or c++ calls to force keeping of specific events regardless of how visualization is accumulating them.
  - e.g., keep events based on a particular hit or trigger pattern
- From the command line:
  - /event/keepCurrentEvent
- From C++
  - G4EventManager->KeepTheCurrentEvent()
- This feature makes it easy to do a large run and then recall for visualization only those events that are of interest

Reviewing Kept Events

- Event Keeping is turned on by default when you have Vis and Trajectories
- By default it will keep 100 events.
  - WARNING: G4VisManager::EndOfEvent: Automatic event keeping suspended.
  - The number of events exceeds the maximum, 100, that can be kept by the vis manager.
  - And we generate the following message at the end of the run:
  - WARNING: G4VisManager::EndOfRun: Automatic event keeping has been suspended.
  - The number of events in the run exceeded the maximum, 100, that can be kept by the vis manager.
  - The number of events kept by the vis manager can be changed with
  - "/vis/scene/endOfEventAction accumulate <N>", where N is the
  - maximum number you wish to allow. N < 0 means "unlimited".
  - 100 events have been kept for refreshing and/or reviewing.
- This is just an information message. It is not a problem
  - However it's been confusing to enough users that we will try to find a better way to express this in future releases

Reviewing Kept Events flagged from C++

- If you're using C++ to flag which events to keep, so that you can do a large run but only visualize a few special events, mix event keeping with the vis disable/enable commands, as follows:
  - /vis/open HepRepFile
  - /vis/drawVolume
  - /vis/scene/add/trajectories
  - /vis/disable
  - /run/beamOn 1000
  - /vis/enable
  - /vis/reviewKeptEvents
- In this way, the visualization is disabled when you first run through the 1000 events, and is enabled only when you are reviewing the small number of special events that you flagged from your C++ call:
  - G4EventManager->KeepTheCurrentEvent()

Geant4 Visualization in Standalone Mode

- The Geant4 Visualization system can be used on its own - without the rest of Geant4.
- Build something “by hand” from the Geant4 geometry primitives and placement apparatus, but without any of the main parts of Geant4 such as detector construction, run manager or physics list.
- Still preserves all of the interactive apparatus of the visualization system.
- See the example: /examples/extended/visualization/standalone

```
// Simple box...
    pVisManager->Draw(      G4Box("box",2*m,2*m,2*m),
                                    G4VisAttributes(G4Colour(1,1,0)));

// Boolean solid...
    G4Box boxA("boxA",3*m,3*m,3*m);
    G4Box boxB("boxB",1*m,1*m,1*m);
    G4SubtractionSolid subtracted(      "subtracted_boxes",&boxA,&boxB,
                                    G4Translate3D(3*m,3*m,3*m));
    pVisManager->Draw(      subtracted,
                            G4VisAttributes(G4Colour(0,1,1)),
                                    G4Translate3D(-6*m,-6*m,-6*m));
```

Geant4 Visualization Resources

Hands on Qt Tutorial

- http://geant4.in2p3.fr/spip.php?article60

Hands on HepRApp Tutorial

- http://geant4.slac.stanford.edu/Presentations/vis/G4HepRAppTutorial/G4HepRAppTutorial.html

Hands on DAWN Tutorial

- http://geant4.slac.stanford.edu/Presentations/vis/G4DAWNTutorial/G4DAWNTutorial.html

Hands on OpenGL Tutorial

- http://geant4.slac.stanford.edu/Presentations/vis/G4OpenGLTutorial/G4OpenGLTutorial.html

How to Make a Movie

- http://geant4.slac.stanford.edu/Presentations/vis/HowToMakeAMovie.ppt (and .pdf)

Visualization Chapter of the Geant4 User's Guide for Application Developers

- http://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/ForApplicationDeveloper/html/

List of Visualization Commands:

- http://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/ForApplicationDeveloper/html/AllResources/Control/UIcommands/_vis_.html

For Questions or Comments: Geant4 Visualization Online Forum:

- http://geant4-hn.slac.stanford.edu:5090/HyperNews/public/get/visualization.html