

# GEANT4 Visualization and Analysis Tools

Short Tutorial

Mark Dönszelmann  
Stanford Linear Accelerator Center (SLAC)

Space Users' Forum  
20-22 January 2003

ESTEC, Noordwijk, The Netherlands

# Content

- **Context**
- **Geant4 Visualization**
  - Data, Objects and Attributes
  - Steps in Visualization
  - Different Drivers
  - HepRep and WIRED
- **Geant4 Analysis**
  - AIDA – Abstract Interfaces for Data Analysis
  - JAS – Java Analysis Studio
- **Demonstration**

# Context

- Only recently joined the Geant4 collaboration
- Not an expert of Geant4, neither a User
- Developer of the WIRED Event Display
- Implemented a Geant4 graphics driver for WIRED
- Work on the JAS Analysis Studio team

# Geant4 Visualization

- Geant4 Visualisation must respond to varieties of user requirements:
  - Scanning a set of events
  - High-quality output for publication
  - Special effects for demonstrations
  - Flexible control for debugging geometry
  - Highlighting overlapping of physical volumes
  - Interactive picking of visualized objects
  - ...

# Visualizable Data

- You can visualize simulation data such as:
  - Detector components
  - A hierarchical structure of physical volumes
  - A piece of a physical volume, logical volume or solid
  - Particle trajectories and tracking steps
  - Hits of particles in detector components
- Visualization is performed either with commands or by writing C++ user code

# Visualizable Objects

- **Polyline**
  - A set of successive line segments
  - examples: trajectories, coordinate axes
- **PolyHedron**
  - A set of polygons / faces
  - examples: faces of geometry, calorimeter hits
- **Marker**
  - Marks an arbitrary 3D position with a symbol (square) that always faces the eye
  - example: hits
- **Text**
  - Special marker type which uses a character string as marker
  - examples: description, comments or titles

# Attributes of the Visualizable Objects

- In addition to the geometrical information one can set Attributes on Visualizable Objects
  - Color, Visibility, Style, etc
  - A set of visualization attributes is held by class *G4VisAttributes*
- *G4VisAttributes* is set on a logical volume in the C++ user code.

# Scenes, Driver, Handlers and Viewers

- **Scene**

- a set of visualizable objects, such as detector components, hits, trajectories, axes, etc

- **Visualization Driver**

- **Scene Handler**

- a graphics-data modeler, which processes raw data in a scene for later visualization

- **Viewer**

- generates image/output based on data processed by a scene handler

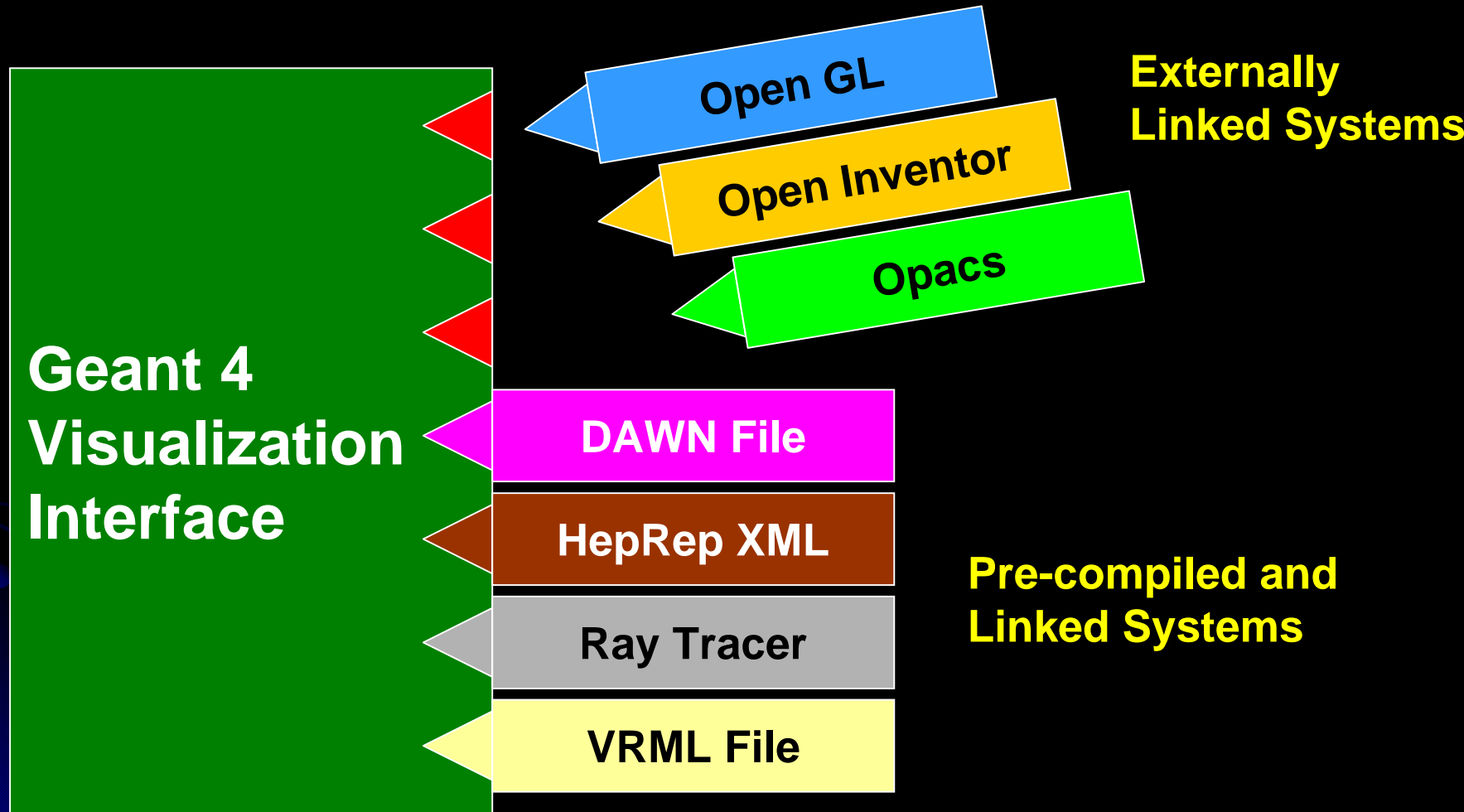


# Visualization Steps and Commands

1. Create a scene handler and a viewer
2. Create an empty scene
3. Add 3D data to the created scene
4. Attach the current scene handler to the current scene
5. Set camera parameters, drawing style (wireframe/surface), etc
6. Make the viewer execute visualization
7. Declare the end of visualization

- 1: /vis/open OGLIX
- 2,3,4: /vis/drawVolume
- 5: /vis/viewer/reset
- 5: /vis/viewer/viewpointThetaPhi 70 20
- 5: /vis/viewer/set/style wireframe
- 7: /vis/viewer/update
  
- /tracking/storeTrajectory
- 1: /vis/open HepRepXML
- 2: /vis/scene/create
- 3: /vis/scene/add/volume
- 3: /vis/scene/add/trajectories
- 4: /vis/sceneHandler/attach
- 6: /run/beamOn 10
- 7: /vis/viewer/update

# Visualization Abstract Interface



# Visualization Drivers

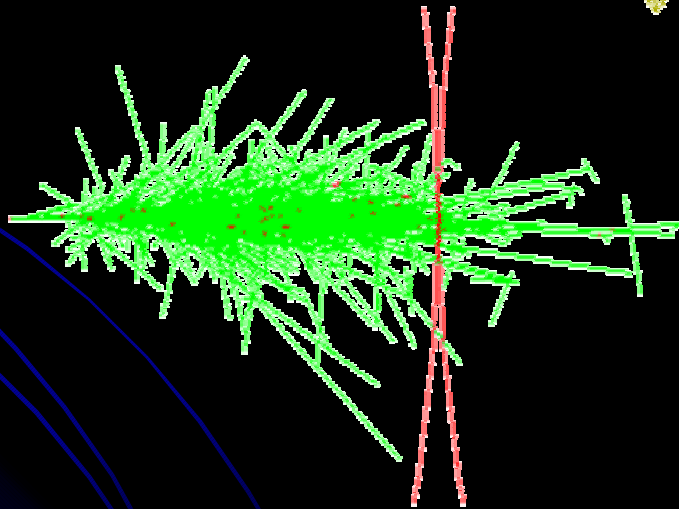
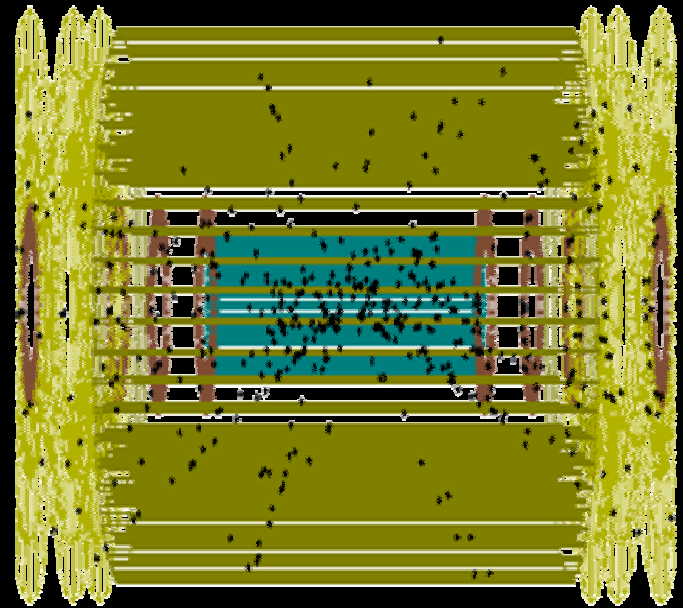
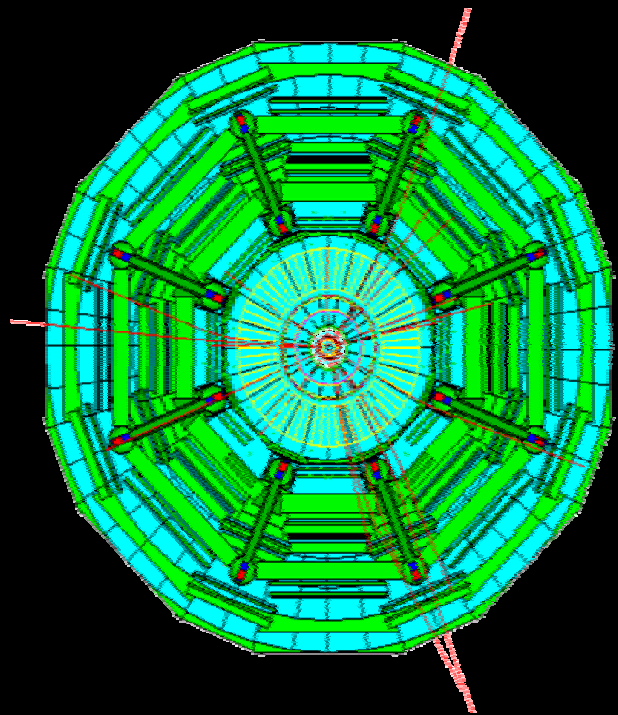
- **DAWN-File**
  - Technical High-quality PostScript output
- **HepRepFile and HepRepXML**
  - Model of Representables for FRED or WIRED
- **RayTracer**
  - Photo-realistic rendering in JPEG format
- **VRML-File (version 1 and 2)**
  - Interactivity, 3D graphics on Web

# External Visualization Drivers

- **OPACS**
  - Interactivity, unified GUI
- **OpenGL (on X, on X<sub>m</sub> and WIN32)**
  - Quick and flexible visualization
- **OpenInventor (on X and WIN32)**
  - Interactivity, virtual reality, etc
- **DAWN-Network-FukuiRenderer**
- **VRML-Network**

# Compiling and Linking in External Visualization Drivers

- In the code add a Visualization Manager (example MyVisManager)
  - This registers the available graphics systems
- **Compilation of G4 Code**
  - Set one or more of the preprocessor variables G4VIS\_BUILD\_\*\_DRIVER
  - Example: G4VIS\_BUILD\_OPENGLX\_DRIVER
- **Compilation and Linking of User Code**
  - Set one or more of the preprocessor variables G4VIS\_USE\_\*
  - Example: G4VIS\_USE\_OPENGLX
- **The configure script**
  - `.../geant4/Configure --install`
- asks which drivers should be supported, and sets the variables accordingly.
- To disable visualization completely set the environment variable **G4VIS\_NONE**



10 February 2003

Mark Donszelmann, Stanford Linear Accelerator Center

14

# The HEPREP Driver

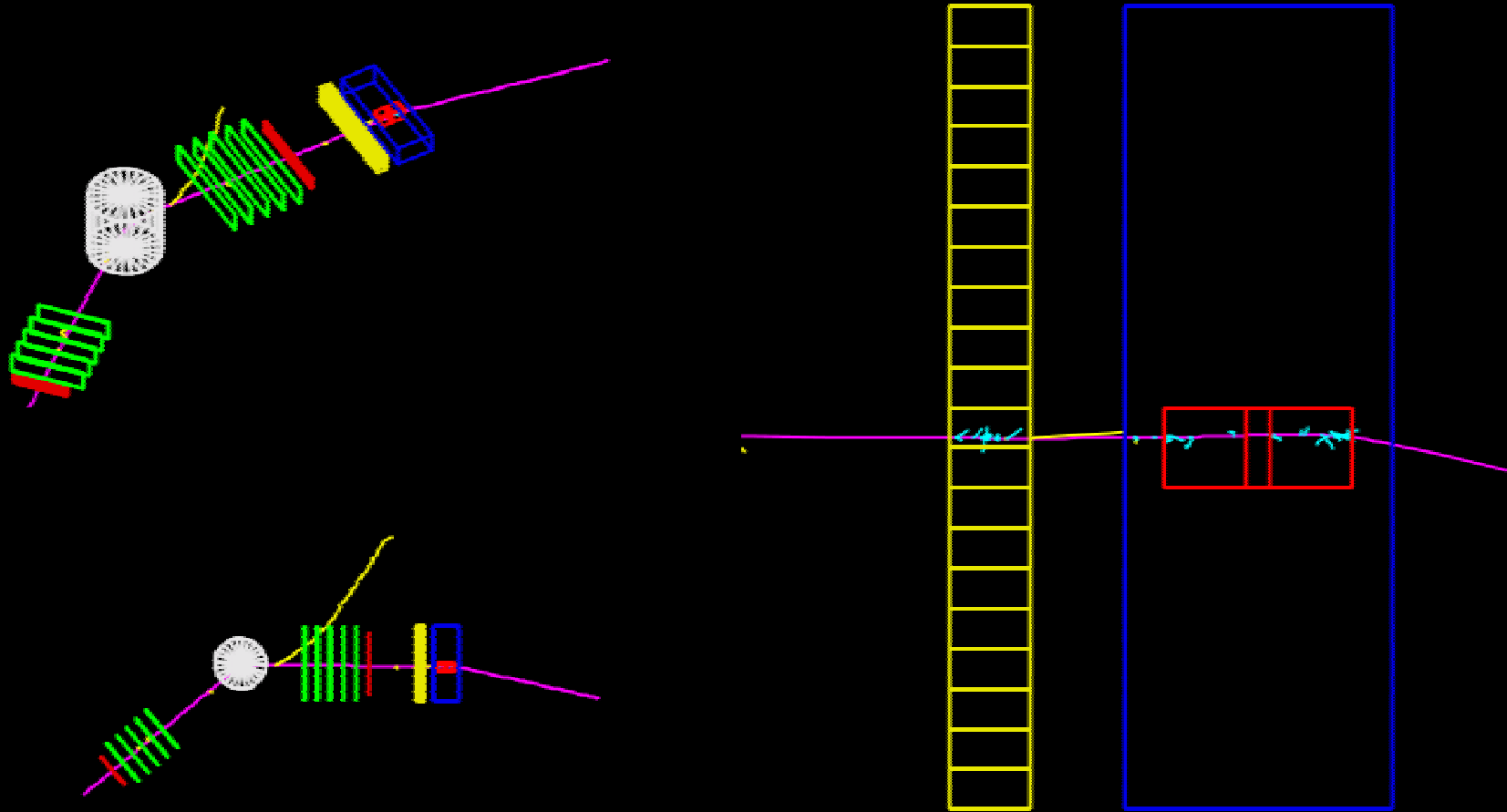
- Pure C++
- Writes out the graphics model, rather than the viewpoint. All viewer commands are ignored.
- Streams Events and Geometry as HepRep XML Trees
- HepRep compatible viewers:
  - FRED (OpenGL)
  - WIRED (wire-frames only)

# The WIRED Viewer I

- Pure Java viewer with its own graphics engine
- Reads HepRep
- Projections
  - Linear 3D projections
  - Non-Linear, Fish-Eye – Blow Up projections
- Layering Model
  - Allows hits to be on top of tracks
- Outputs to many vector graphics formats
  - PDF, PS, EMF, SVG

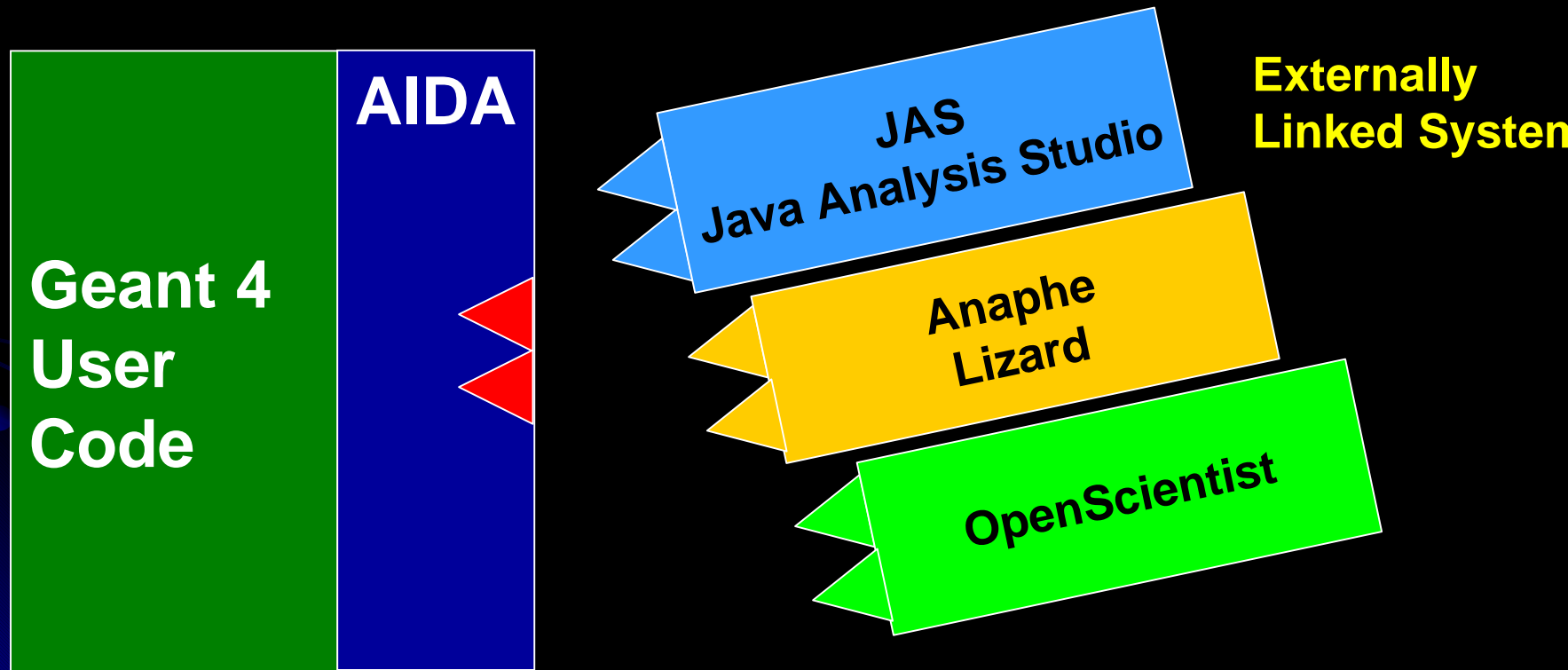


# The WIRED Viewer II



# Geant4 Analysis Design

- **AIDA: Abstract Interfaces for Data Analysis**



# AIDA Design

- **Goal**

- Provide the user with a powerful set of interfaces independent of the analysis tool

- **Advantages**

- User has to learn system only once
- Easier development, developers of different analysis systems can help each other
- Interoperability (common data format, for now in XML)

# AIDA Interfaces

- **Histograms**
  - Binned 1-,2-,3- dimensional histograms
  - Unbinned 1-,2-,3- dimensional histograms (Clouds)
  - 1-,2- dimensional profile histograms
- **Tuples**
- **DataPointSet (Vector of Points)**
- **Functions**
- **Fitting interfaces**
- **Plotting interfaces**
- **Management of analysis objects**
  - Tree
  - Factories

# AIDA Implementations

- **C++**

- Anaphne/LHC++/Lizard – AIDA 3.0
- Open Scientist – AIDA 2.2

- **Java**

- JAS – Java Analysis Studio – AIDA 3.0

# Usage of AIDA

- Example to create and fill a histogram

- Once

- `IAnalysisFactory* af = AIDA_createAnalysisFactory();`
- `ITreeFactory* tf = af->createTreeFactory();`
- `IHistogramFactory* hf = af->createHistogramFactory(tf);`
- `Histogram1D* h1 = hf->createHistogram1D(  
"Drift Chamber 1 # Hits", 10, 0, 10);`

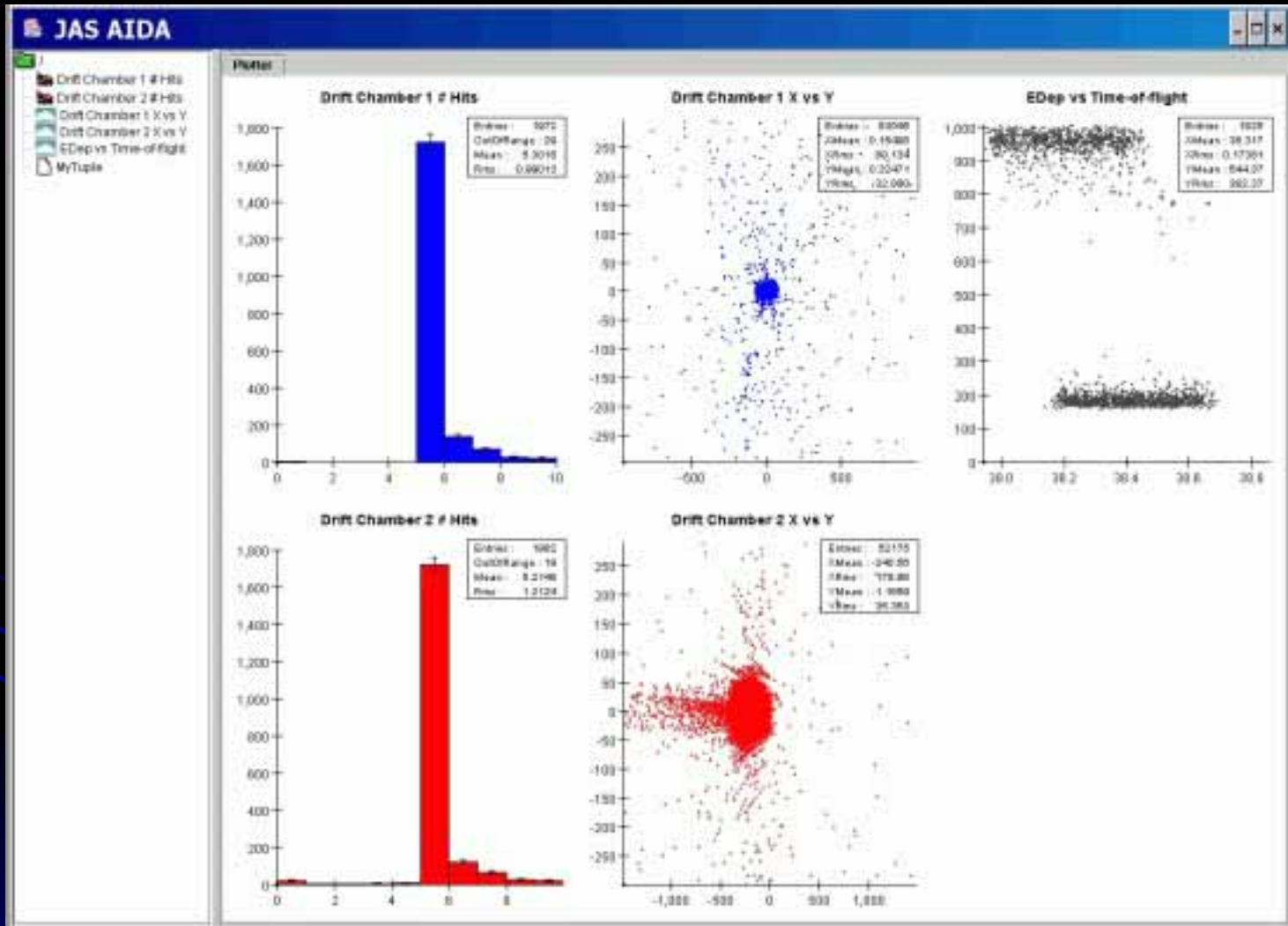
- In a loop

- `h1->fill(data);`

# Linking in an AIDA Implementation

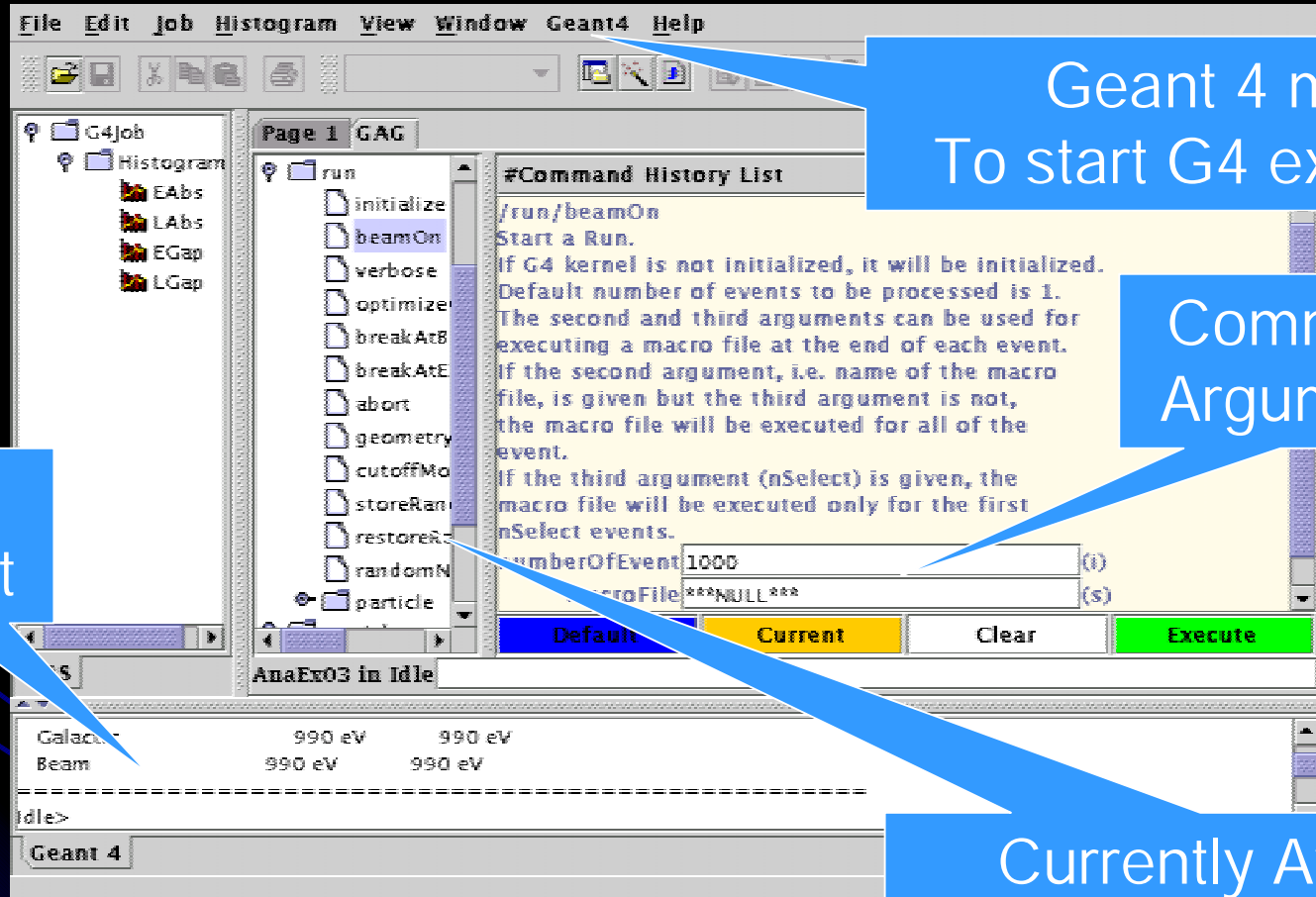
- **Add a `AnalysisManager` to the user code**
  - This instantiates the analysis system
- **One AIDA implementation can be linked into Geant4**
  - Set `G4ANALYSIS_AIDA_CONFIG_LIBS` to point to one or more libraries
- **Relink**

# G4 and JAS – Java Analysis Studio





# G4 and JAS – Future Interface



Geant 4 menu.  
To start G4 executable

Command Arguments

G4  
Output

Currently Available  
Commands

# Demonstration

- Web Server running Geant4 showing A01App extended analysis example
- *Geant4 writing an HepRep XML file*
- WIRED showing this file
- Geant4 linked with JAS
- JAS exploring the NTuple

# URLs

- **AIDA: Abstract Interfaces for Data Analysis**
  - <http://aida.freehep.org>
- **JAS: Java Analysis Studio**
  - <http://jas.freehep.org>
- **HepRep: a generic interface definition for HEP event display representables**
  - <http://heprep.freehep.org>
- **WIRED: WWW interactive remote event display (a HepRep Client)**
  - <http://www.slac.stanford.edu/BFROOT/www/Computing/Graphics/Wired>
- **Fred: oh no, another event display (a HepRep client)**
  - <http://www.fisica.uniud.it/~riccardo/research/fred>
- **Geant4 Users Workshop at SLAC Feb 2002, Tutorials**
  - <http://geant4.slac.stanford.edu/UsersWorkshop/tutorial.html>