

# GraXML - Modular Geometric Modeler

Julius Hrivnák  
LAL, Orsay, France

Many entities managed by HEP Software Frameworks represent spatial (3-dimensional) real objects. Effective definition, manipulation and visualization of such objects is an indispensable functionality.

GraXML is a modular Geometric Modeling toolkit capable of processing geometric data of various kinds (detector geometry, event geometry) from different sources and delivering them in ways suitable for further use. Geometric data are first modeled in one of the Generic Models. Those Models are then used to populate powerful Geometric Model based on the Java3D technology. While Java3D has been originally created just to provide visualization of 3D objects, its light weight and high functionality allow an effective reuse as a general geometric component. This is possible also thanks to a large overlap between graphical and general geometric functionality and modular design of Java3D itself. Its graphical functionalities also allow a natural visualization of all manipulated elements.

All these techniques have been developed primarily (or only) for the Java environment. It is, however, possible to interface them transparently to Frameworks built in other languages, like for example C++.

The GraXML toolkit has been tested with data from several sources, as for example ATLAS and ALICE detector description and ATLAS event data. Prototypes for other sources, like Geometry Description Markup Language (GDML) exist too and interface to any other source is easy to add.

## I. HISTORY

GraXML [1],[2],[4] has been originally developed as a simple 3D Detector Display for ATLAS Generic Detector Description (AGDD) [3], i.e. for the detector geometry specified in a generic (application neutral) way using XML. (Since then, many other experiments and Geant4 have followed ATLAS example.)

Later, the support for several other geometry descriptions has been added to GraXML:

- **AliDD** is AGDD extended with geometric elements used in ALICE, but not in ATLAS.
- **AGDD v6** is AGDD extended with mathematical formulas to express complex relations between elements and with an access to a relational database to get initial parameters.
- **AtlasEvent** [6] is a format used to describe ATLAS Event data, like hits and tracks.
- **AtlantisEvent** (prototype) is similar to AtlasEvent, it is used by the Atlantis [19] event display.
- **GDML** [5] (prototype) is an XML format of Geant4 detector description.
- Direct access via API is available from Java directly and from C++ via proxies generated by JACE [11].

Quite soon, it has been realized that requirements on Event and Detector Display functionality often closely correspond to requirements on Geometric Modeler:

- Both have to provide a logical navigation in complex geometric structures.
- Both have to provide a complex geometrical navigation functionality, like:
  - identification of the element occupying a spacial position (“Where am I?”),
  - intersections between elements (picking, collision detection),
  - attaching of materials to elements,
  - displacement and calibration (interactivity).
- Both need highly optimized geometry description in memory, supporting millions of 3D objects.

The only functionality which is special in an Display application is the manipulation of visual object’s properties, like transparency, shading, etc. This functionality has, however, very big demand on system resources (mainly memory).

It has been concluded, that 3D Geometric Engine can be used as Geometric Modeler foundation if its visual overhead can be removed. Java3D, which is used as a Geometric Engine in GraXML, can be used without its visual overhead so it can serve as a general Modeler foundation.

GraXML has been then re-engineered to provide flexible Geometric Modeling Toolkit with optional Display capabilities. Currently, it contains following components:

- **Core** provides the foundation for the toolkit, it uses Java3D library for a description of 3D objects.

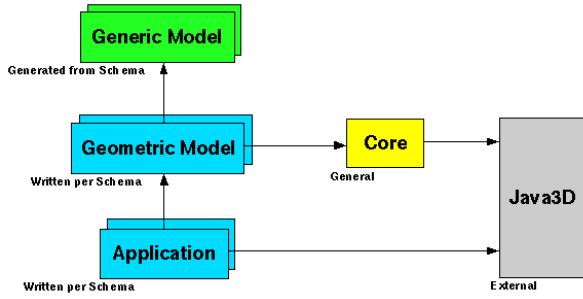


FIG. 1: Overview of the GraXML Toolkit Global Architecture.

- **Generic Modelers** are used to parse XML geometric descriptions and create their memory model.
- **Geometric Modelers** create optimized 3D model of the geometry.
- **Applications** (visual or not) are small programs build on top of the foundation toolkit.
- **Utilities** help to manage and organize data.

## II. ARCHITECTURE

### A. Generic Model

A special Generic Model is created for each source data XML Schema definition. JAXB [15] processor is used to create an object model similar to widely known DOM model, but with several advantages. JAXB model is tuned to the XML Schema which has been used to create it. The main features of the JAXB model are:

- Created classes correspond to available XML elements.
- Their methods (setters/getters) correspond directly to elements attributes.
- Variables have proper types and default values based on a Schema definition and an optional customization file.
- The generation can be customized (via an XML customization file) to capture more complex structures and relations.
- Created classes can be modified by extension or by helper conversion classes.

The generated object model is more natural and faster than traditionally used DOM and SAX-based models. Listing I shows a simple example of a class corresponding to an XML element.

```
< box x = "1.1" y = "2.2" z = "3.3" / >
public class box {
    public double x;
    public double y;
    public double z;
}
```

Listing I: Simplified example of an XML element and corresponding class generated by JAXB processor.

```
< XSQLConfig >
< connectiondefs >
    < connection name = "demo" >
        ...
        < dburl > jdbc:mysql://nova.site.org/NOVA < /dburl >
    < /connection >
< /connectiondefs >
< /XSQLConfig >
...
< var connection = "demo" name = "SCT.length" / >
...
< var name = "SCT.length" value = "123.456" / >
```

Listing II: Definition of the connection to a relational database and the result of the filling operation in AGDD XML v.6.

#### 1. Initial values

A detector description XML file can have its initial values outsourced into a Relational Database. The Nova [7] database is used for AGDD. XSQL [13] Schema is used to define connections between the XML file and the Relational Database. A simple JDBC [14] Connection is then used to fill those values into a parametrised XML file as is demonstrated in the Listing II.

#### 2. Formulas

Symmetries and dependencies between attributes in XML file can be expressed using standard mathematical formulas, like in the Listing III. The XSLT [12] stylesheets with the simple Java evaluator are then used to expand structures with formulas to concrete elements. Formulas together with initial values read in from a relational database help to make the XML files very compact.

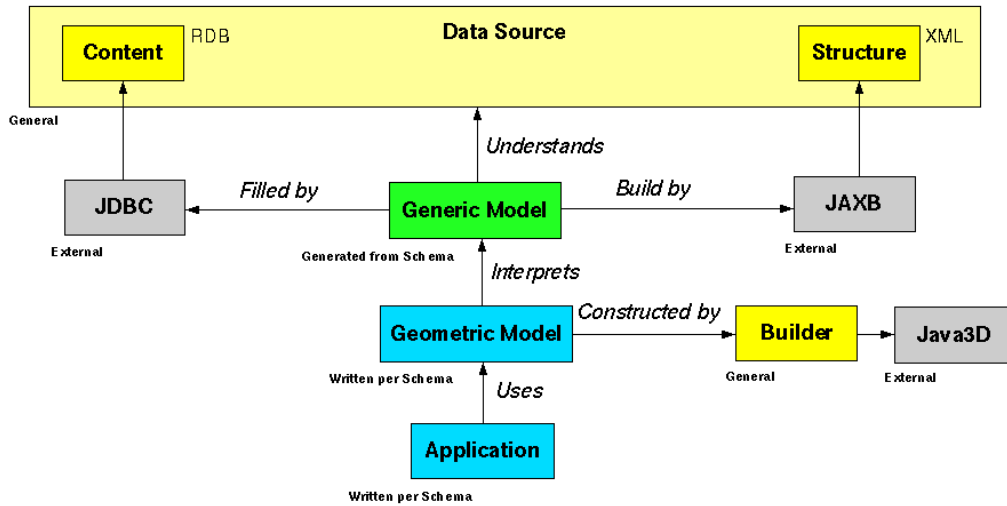


FIG. 2: GraXML Toolkit Global Architecture.

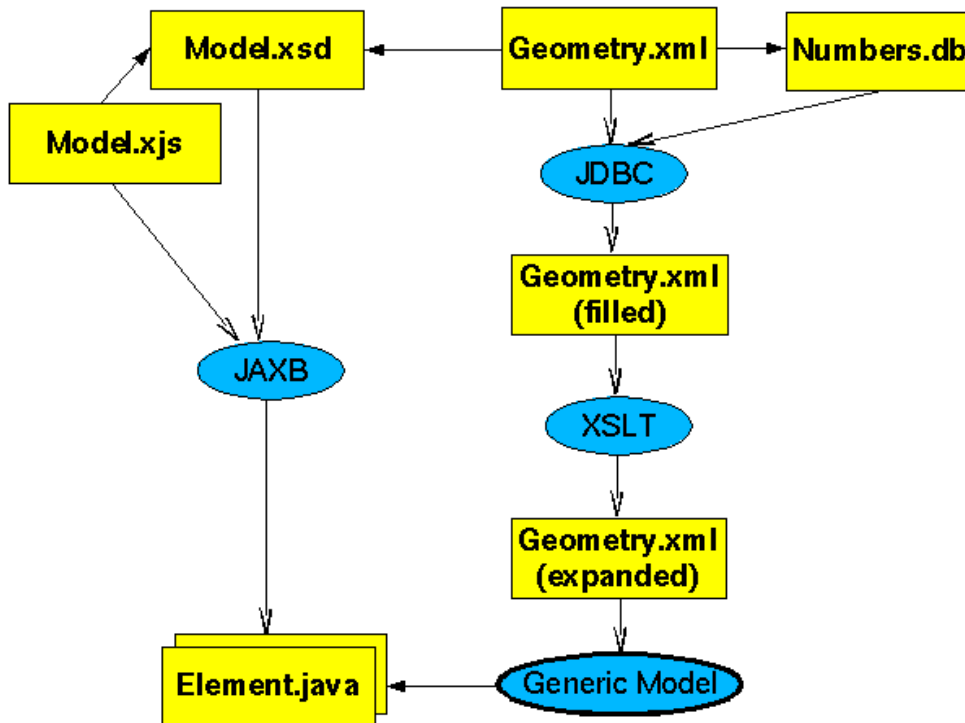


FIG. 3: Generic Model Architecture.

### B. Geometric Model

The Java3D [9] SceneGraph is build from the Generic Model according to several Build Options:

- Presence of **Graphical** visual attributes and functionality.
- **Level of Optimization** specifies how much will be created SceneGraph optimized. This mainly

influences the strategy employed to share representations (SharedGroups) for identical or similar structures. Repeated structures are discovered during SceneGraph building and reused as SharedGroups. Higher Levels of Optimization allow creation of very compact Geometric Models. The interactivity (and calibration capabilities) of such optimized models are limited as many volumes don't exist as individual en-

```

< array name = "a" values = "1; 2; 3; 4; 5; 6; 7; 8; 9; 10" / >
< table name = "t" >
  < row values = "1; 2; 3; 4; 5" / >
  < row values = "6; 7; 8; 9; 10" / >
< /table >
< var name = "a0" value = "1" / >
< var name = "b" value = "a0 * 2" / >
< var name = "c" value = "a[2] * a[3]" / >
...
< box XYZ = "5.5; a[5]; t[2, 3]" name = "abox" / >

```

Listing III: Examples of formulas in AGDD XML v.6.

tities and can't then be individually manipulated. A more optimized SceneGraph is smaller and faster but allows only limited interactivity or calibration.

- **Level of Quality** influences how closely are 3D objects represented. Its consequences are for example the level of approximation of curved surfaces or using of visual enhancements like anti-aliasing. The effects of Level of Quality are restricted by a chosen Level of Optimization.
- **Level of Interactivity** specifies how interactive Display will be and how much objects can be calibrated. A higher Level of Interactivity allows higher possibilities to change objects properties at run-time. The effects of Level of Interactivity are restricted by the chosen Level of Optimization and Quality. Depending on the Level of Optimization, SceneGraph elements (Shapes, Groups) can be modified at run time to provide a dynamic functionality like:

- **Calibration**

- **Graphics Operations:**

- \* Modification of **Real Objects** (their shape, place, orientation, ...),
- \* Modification of the Objects **Visual Characteristics** (visibility, color, transparency, ...).

- **Representations** to be used to represent Generic Objects and their properties.

Additional Geometric Model functionalities are generally added as special nodes inserted into a SceneGraph as is described in Figure 5.

As SceneGraphs (or their subgraphs) are compiled for speed, Builder Options can't be changed later (when Scene Graph is active and used).

All Geant4 CSG Solids (and some others) have been implemented as standard Java3D Shapes with constructors equivalent to the Geant4 constructors.

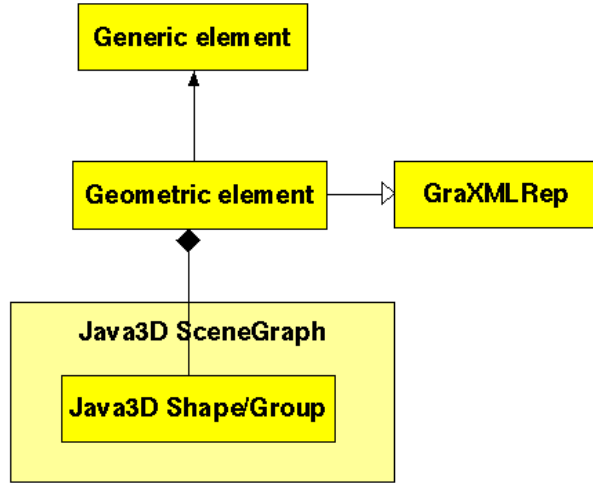


FIG. 4: Geometric Model Architecture.

Those solids are usually specializations of more generic Shapes.

New Shapes (Helix, ...) or special Shapes (Outline, ...) are also included.

All those Shapes have been contributed to the Free-HEP [10] library.

### III. DATA SOURCES

GraXML can get data in various ways, preferably in the XML format. The system is easily extendable. GraXML currently supports following data sources:

- **XML**

- **Detector Description**

- \* **AGDD v4** [3] is the original explicit ATLAS Generic Detector Description.
- \* **AliDD** is AGDD with additional elements, used in ALICE, but not in ATLAS.
- \* **AGDD v6** is AGDD with arithmetic formulas and connection to a relational database.
- \* **GDML** [5] (prototype) is the Geant4 format of geometric description.

- **Events**

- \* **AtlasEvent** [6] is the XML format of ATLAS Event data.
- \* **AtlantisEvent** is the XML format of ATLAS Event data used by Atlantis [19] Event Display.

- **API**

- **Java** programs can be used to directly create both Generic and Geometric Models.

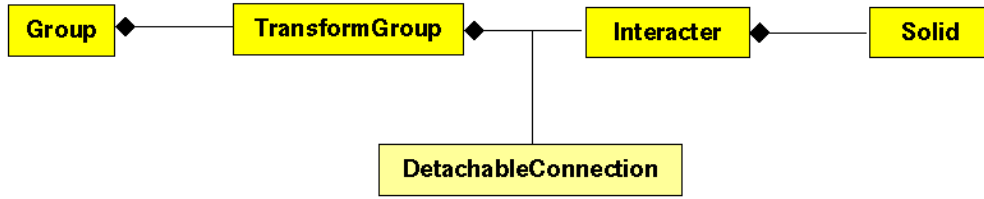


FIG. 5: SceneGraph extensions.

- C++ programs can be interfaced using proxies created by JACE [11].

Many of supported XML Schemas are supported also by other programs, like Atlantis [19] or PersInt [20].

#### IV. APPLICATIONS

Various applications have been built on top of the GraXML Toolkit. The most important (and most widely used) is the GraXML 3D Display.

##### A. 3D Display

The GraXML Display offers the full functionality for browsing and manipulating 3D data. All the major User Requirements functionality for the 3D display are available:

- Standard 3D visual operations, like Rotation, Translation, Zooming, Scaling, Sheering and Skewing. Those operations can be anizotropic, if applicable. All those operations can be applied either to the whole Scene (Global Operations) or to the selected (picked) objects (Local Operations). The Local Operation allows, for example, to move one object away from others. Operations are available via a Drag mouse action, using a keyboard or through the menu. The Go-to operation is also available, it moves the observer closer to the selected object (or away from it).
- Changing of visual properties of an object. The color of an object, its transparency properties and its polygon status (solid - wireframe - vertexframe) can be changed interactively. An operation can be applied either to the selected (picked) objects or the the whole SceneGraph.
- Both Parallel and Perspective projections.
- Cutting the 3D Scene from front and from back.
- Selective switching of the objects on and off from the hierarchical Tree Navigator. The switching works also in the optimized mode, the result

may be, however, surprising due to hidden relations.

- Snapshot saving into the jpg file.
- Context sensitive help and Context sensitive information about the Tree Navigator elements.
- Optimized or fully Interactive navigation. The Optimized mode uses much smaller memory footprint and is therefore useful for the larger 3D Scenes (more than tens of thousands of simple objects). Interactive features are limited in this mode.
- Quality level. Higher quality levels offer features like better antialiasing or smoother objects, but requires significantly more CPU and memory.
- Output to VRML [16]. The VRML output can be also translated into a POV [18] Raytracing format file, which can then be used to create photographic-quality pictures (incl. reflections, ...).
- Context sensitive introspection and actions allowing inspecting and calling features of the real objects connected to the 3D objects. This means that, for example, a user can re-fit a Track by interacting with its graphical representation. This functionality, however, requires a collaboration of a back-end (reconstruction) framework.
- Full Java scripting interface. While complete Java environment is available (including access to all GraXML objects), the intended use of the scripting interface is to set various options, open files and clean the window. An example of a GraXML Display script is shown in the Listing IV.

Several snapshots of running GraXML Display are shown in Figures 6, 7, 8, 9, 10.

##### B. Non-Graphical Applications

The non-graphical Applications built on top of GraXML Toolkit are Exporters, Converters and Importers:

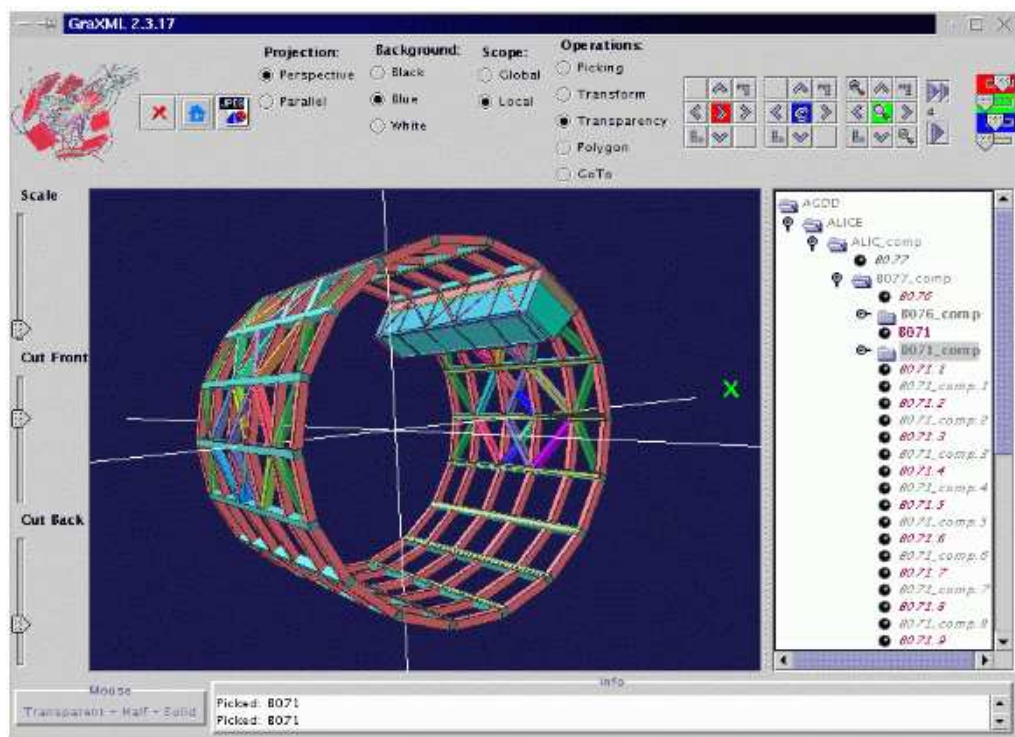


FIG. 6: ALICE Frame in the GraXML Display (the XML file has been generated from Geant4).

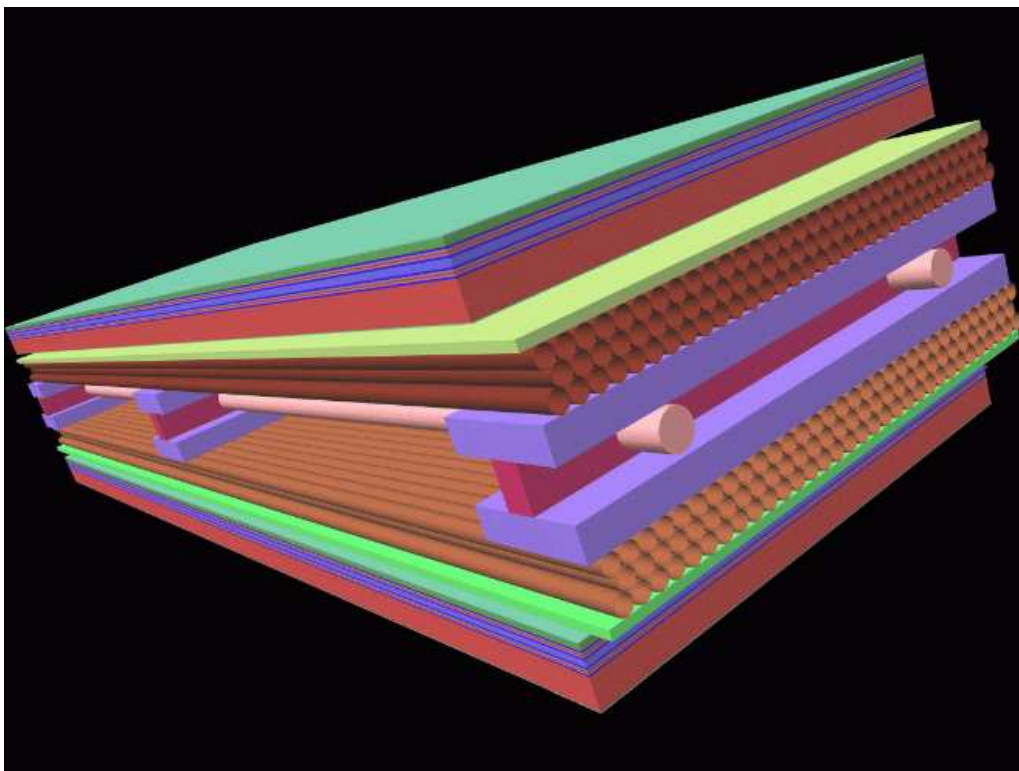


FIG. 7: ATLAS Muon Chamber.



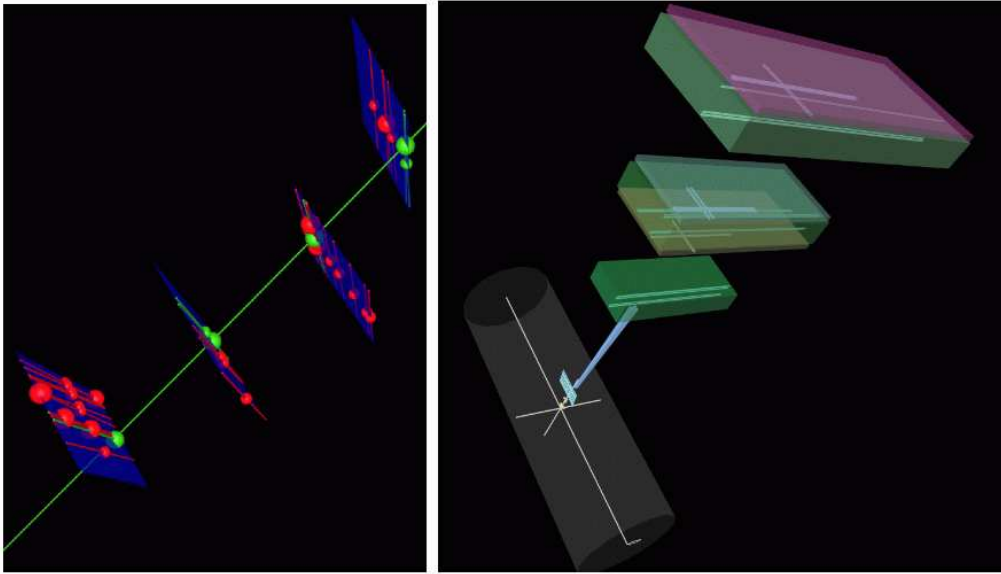


FIG. 8: Track in ATLAS Inner Detector and Muon Detector - Photo-realistic View and Transparent Volumes.

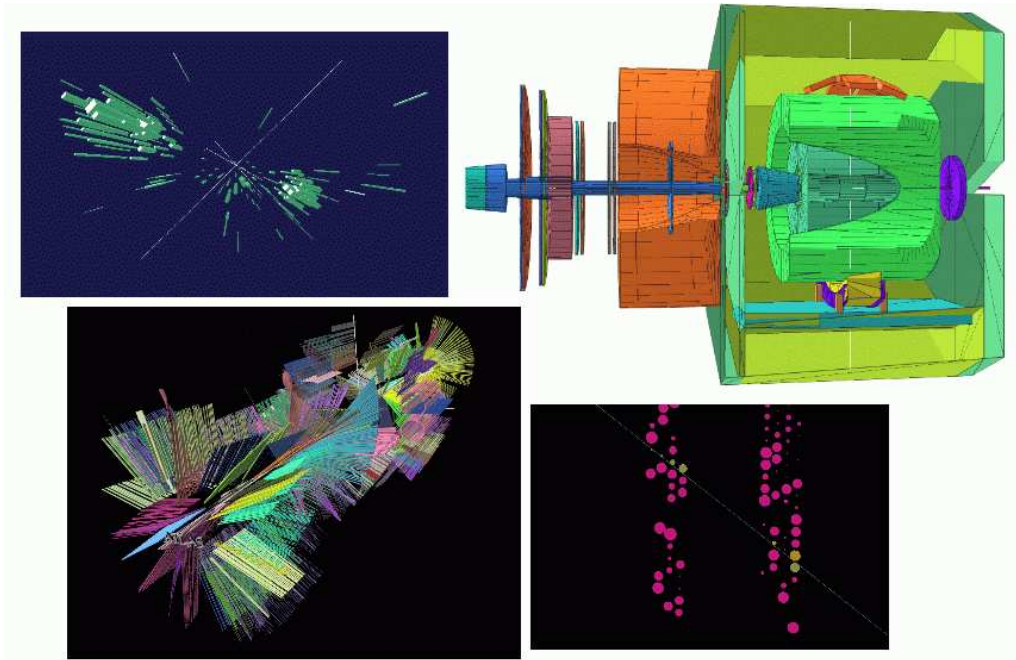


FIG. 9: 1) ATLAS Event in Tile Calorimeter. 2) Semi-cutted, semi-transparent ALICE. 3) ATLAS Event in TRT. 4) ATLAS Track in Muon Chamber.

- **Exporters** for exporting geometric data into other Applications or file formats.

- Into VRML and X3D [17] format to be used by any VRML browser or in the 3D Cave.
- Into TXT format for easy debugging.

- **Converters** for converting between various supported data formats.

- Between AGDD v4 and v6 via XSLT stylesheets.
- Between AtlasEvent and AtlantisEvent via a simple application.

- **Importers** for importing from other application.

- Geant4 to AGDD via a simple C++ application provided in Virtual MC [8] built in

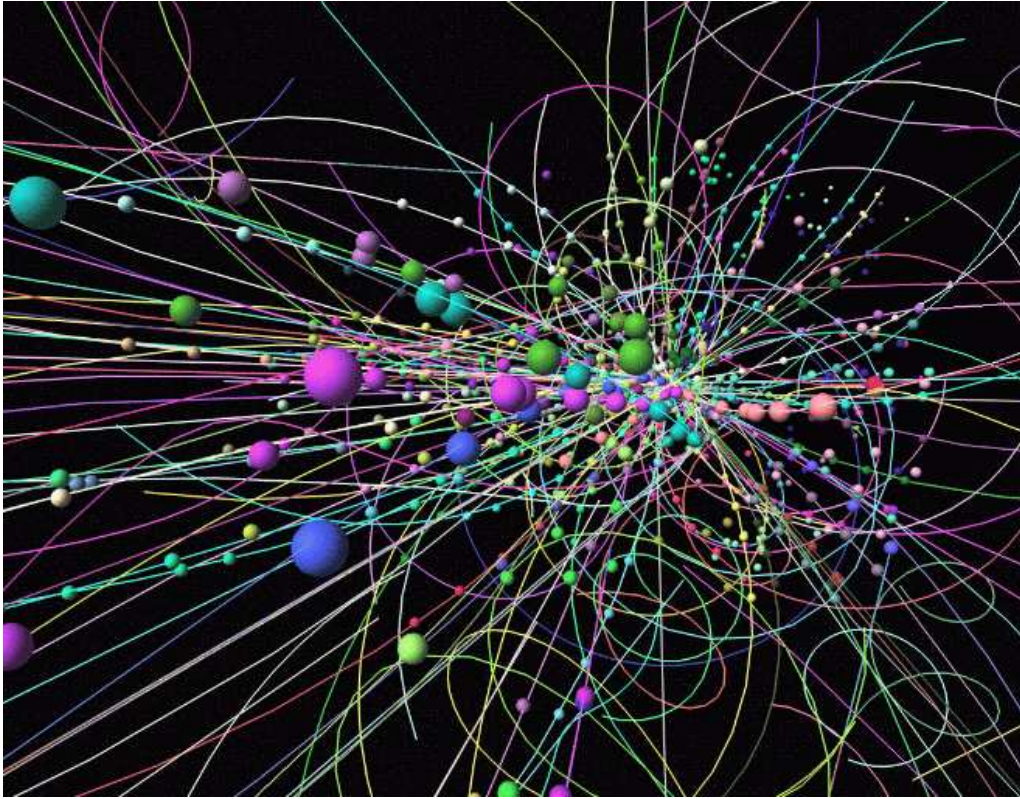


FIG. 10: Artistic view of the ATLAS Event in the Inner Detector.

```
Configuration.setQuality(9);
SelectedColor.setPalette(SelectedColor.ATLANTIS);
TruthTrack.setPtCut(5.0);
Hit.asSphere();
Hit.colorFromKine();
w.show("test.xml");
j3d.snapshot("Picture.jpg");
```

Listing IV: Example of simple script for GraXML.

ALICE.

## V. SUMMARY

The GraXML toolkit provides a flexible foundation for modeling of 3D spacial data in HEP. Most of the required functionality is provided directly by Java3D, the rest is implemented on top. Applications built on GraXML can run in a Graphical or non-Graphical mode.

- 
- [1] GraXML Home:  
<http://hrivnac.home.cern.ch/hrivnac/Activities/Packages/GraXML>
  - [2] More details talk about GraXML:  
<http://hrivnac.home.cern.ch/hrivnac/Activities/2001/September/GraXML>
  - [3] ATLAS Generic Detector Description (AGDD):  
[http://ATLAS.web.cern.ch/ATLAS/GROUPS/DATABASE/detector\\_description](http://ATLAS.web.cern.ch/ATLAS/GROUPS/DATABASE/detector_description)
  - [4] Java Framework for AGDD Processing:  
<http://hrivnac.home.cern.ch/hrivnac/Activities/Packages/JAGDD>
  - [5] Geometry Description Markup Language (GDML):  
<http://gdml.web.cern.ch/gdml>
  - [6] Graphics ATLAS Event:  
<http://hrivnac.home.cern.ch/hrivnac/Activities/Packages/GraphicsAtlasEvent>
  - [7] NOVA:



- <http://atlassw1.phy.bnl.gov/NOVA/index.php3>
- [8] Geant4 Geometry Converter:  
<http://root.cern.ch/root/vmc/XML.html>
  - [9] Java3D:  
<http://www.j3d.org>
  - [10] FreeHEP library:  
<http://java.freehep.org>
  - [11] JACE:  
<http://reyelts.dyndns.org:8080/jace/release/docs/index.html>
  - [12] XML Transformations (XSLT):  
<http://www.w3.org/TR/xslt>
  - [13] XSQL:  
<http://xsql.sourceforge.net>
  - [14] JDBC:  
<http://java.sun.com/products/jdbc>
  - [15] Java XML Binding (JAXB):  
<http://java.sun.com/xml/jaxb>
  - [16] Web3D Consortium:  
<http://www.web3d.org>
  - [17] Extensible 3D (X3D):  
<http://www.web3d.org/x3d>
  - [18] Persistence of Vision Raytracer (POV):  
<http://www.povray.org>
  - [19] Atlantis Event Display:  
<http://atlantis.web.cern.ch/atlantis>
  - [20] PersInt Detector and Event Display:  
<http://atlas.web.cern.ch/ATLAS/GROUPS/MUON/persint.html>