Geometry III: GDML & CAD

Mike Kelsey (SLAC)
Geant4 Tutorial Course
Overview

• Up to now, the course has concentrated on showing how to define materials and volumes using C++ code using the API.

• Sophisticated coding of the Detector Construction classes can allow the geometry to be modified at runtime (e.g. A01).

• This talk addresses the issue of completely defining the detector geometry at runtime by providing a file-based detector description.
GDML: Geometry Description Markup Language

• An XML-based language designed as an application-independent persistent format for describing the geometries of detectors.
• It implements "geometry trees" which correspond to the hierarchy of volumes a detector geometry can be composed of, allows individual solids to be positioned, as well as to describe the materials they are made of.
• Being pure XML, GDML can be universally used, and in particular it can be considered as the format for interchanging geometries among different applications.

• http://gdml.web.cern.ch/GDML/
Why XML?

• Simplicity
  – Rigid set of rules
  – Self-describing data validated against schema

• Extensibility
  – easily add custom features, data types

• Interoperability
  – OS, languages, applications

• Hierarchical structure ↔ OOP, detector/subdetector

• Open W3 standard, lingua franca for B2B

• Many tools for validating, parsing, translating

• Automatic code-generation for data-binding

• Plain text: easily edited, cvs versioning
GDML Overview

- Definitions
- Materials
- Solids
- Structures
- Setup
GDML Definitions

- Contains numerical values of constants, positions, rotations and scales that will be used later on in the geometry construction.
  - Uses CLHEP expressions

- Constants
  - `<constant name="length" value="6.25"/>

- Variables
  - `<variable name="x" value="6"/>
  - `<variable name="y" value="x/2"/>
  - Once defined, can be used anywhere later, e.g.
    - `<box name="my_box" x="x" y="y" z="x+y"/>`
GDML Definitions II

- **Positions**
  - `<position name="P1" x="25.0" y="50.0" z="75.0" unit="cm"/>

- **Rotations**
  - `<rotation name="RotateZ" z="30" unit="deg"/>

- **Scales**
  - `<scale name="my_reflection" x="-1" y="-1" z="1"/>

- **Matrices**
  - `<matrix name="m" coldim="3" values=" 0.4  9 126 8.5  7  21 34.6  7   9"/>

Geometry III - N. Graf(SLAC)
GDML Materials I

• Isotopes

  <isotope name="U235" Z="92" N="235">
    <atom type="A" value="235.04">
  </isotope>

• Compare to:

  G4Isotope* U235 = new G4Isotope("U235",92,235,235.04);
GDML Materials II

• Simple Elements
  <element Z="8" formula="O" name="Oxygen" >
    <atom value="16" />
  </element>

• User-defined isotopic abundances
  <element name="enriched_uranium" >
    <fraction ref="U235" n="0.9" />
    <fraction ref="U238" n="0.1" />
  </element>

• Compare to:
  G4Element* enU = new G4Element(“eU”, ”U”, 2);
  enU->AddIsotope(isoU235,90.*perCent);
  enU->AddIsotope(isoU238,10.*perCent);
• Materials can be defined in three ways:
  – Created directly from an element:
    
    ```xml
    <material name="Al" Z="13.0" >
      <D value="2.70" />
      <atom value="26.98" />
    </material>
    ```
  – Created from previously defined elements or materials by number of atoms (“molecule”)
    
    ```xml
    <material name="Water" formula="H2O">  
      <D value="1.0" />
      <composite n="2" ref="Hydrogen" />
      <composite n="1" ref="Oxygen" />
    </material>
    ```
• Created as a fractional mixture of previously defined elements or materials, ("compound"): 

```
<material formula="air" name="Air" >
  <D value="0.00129" />
  <fraction n="0.7" ref="Nitrogen" />
  <fraction n="0.3" ref="Oxygen" />
</material>
```
GDML Solids

- Collection of all solid definitions which are used in the geometry description:

<table>
<thead>
<tr>
<th>Box</th>
<th>Trapezoid (x&amp;y vary along z)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cone Segment</td>
<td>General Trapezoid</td>
</tr>
<tr>
<td>Ellipsoid</td>
<td>Tube with Hyperbolic Profile</td>
</tr>
<tr>
<td>Elliptical Tube</td>
<td>Cut Tube</td>
</tr>
<tr>
<td>Elliptical Cone</td>
<td>Tube Segment</td>
</tr>
<tr>
<td>Orb</td>
<td>Twisted Box</td>
</tr>
<tr>
<td>Paraboloid</td>
<td>Twisted Trapezoid</td>
</tr>
<tr>
<td>Parallelepiped</td>
<td>Twisted General Trapezoid</td>
</tr>
<tr>
<td>Polycone</td>
<td>Twisted Tube Segment</td>
</tr>
<tr>
<td>Polyhedron</td>
<td>Extruded Solid</td>
</tr>
<tr>
<td>Sphere</td>
<td>Tesselated Solid</td>
</tr>
<tr>
<td>Torus Segment</td>
<td>Tetrahedron</td>
</tr>
</tbody>
</table>
GDML Boolean Solids

- The boolean operations union, subtraction and intersection are also supported, e.g.

\[
<\text{box name="box}_1\text{" x="1" y="5" z="20"} /> \\
<\text{box name="box}_2\text{" x="4" y="4.5" z="18"} />
\]

\[
<\text{union name="union"}> \\
<\text{first ref="box}_1\text{"} /> \\
<\text{second ref="box}_2\text{"} /> \\
<\text{positionref ref="union\_position"} /> \\
<\text{rotationref ref="union\_rotation"} />
\]

\[
</\text{union}> 
\]
GDML Structure

- This is where volumes are created from solids and materials previously defined. Both logical and physical volumes are defined here.

```xml
<volume name="World">
    <materialref ref="Air"/>
    <solidref ref="WorldBox"/>
    <physvol>
        <volumeref ref="vol0"/>
        <positionref ref="center"/>
        <rotationref ref="identity"/>
    </physvol>
</volume>
```
GDML Structure II

• Replicated and Parameterised volumes are also supported.
  – See the documentation for examples.
GDML Setup

• The top volume is defined here.

<setup name="Test1" version="1.0">
  <world ref="World"/>
</setup>

• It is possible define multiple geometry setups choosing different volumes as world volumes.
• Geometry description can be split over multiple files, allowing more granular and/or distributed development.
Connection to Geant4

• GDML files can be directly imported into Geant4 geometry, using the GDML plug-in facility.
  – `#include "G4GDMLParser.hh"`

• Generally you will want to put the following lines into your DetectorConstruction class:

• In the Class Constructor:
  – `G4GDMLParser parser;`

• In the Construct method:
  – `parser.Read("geometryFile.gdml");`

• To access the World volume:
  ```
  G4VphysicalVolume* W = parser.GetWorldVolume();
  ```
Getting Started

• In order to build the Geant4 module for GDML, one needs to have:
  – The XercesC parser pre-installed (presently GDML uses either XercesC 2.8.0 or 3.0.0 versions);
    • [http://xerces.apache.org/xerces-c/download.cgi](http://xerces.apache.org/xerces-c/download.cgi)
  – The following environment variables set at the time the Geant4 libraries get built:
    • XERCESROOT, specifying the path where the XercesC parser library and headers are installed in the system;
    • G4LIB_BUILD_GDML set to "1".

• Three examples available in:
  `$G4INSTALL/examples/extended/persistency/gdml`
Visualizing GDML Files using root

- Assuming that there is a file called test.gdml in the current directory, a script similar to the following will load a plain GDML file into ROOT (http://root.cern.ch)
  - TGeoManager::Import("test.gdml");
  - gGeoManager->GetTopVolume()->Draw("ogl");

- This command should display the ROOT OpenGL viewer with the geometry from the test.gdml file.

- One can customize this by displaying only a particular volume, specified by name, using the following command:
  - gGeoManager->GetVolume("VolumeName")->Draw("ogl");
GDML Extensions

- GDML only provides a description of the detector geometry (volumes, materials and their hierarchical and geometrical positioning).
- Much more is required for most applications to fully describe the system.
  - Fields, regions, limits, sensitive detectors, etc.
- Example G03 provides a skeleton for extending gdml (uses visualization as an example).
- The American Linear Collider Physics Group has developed an xml format which includes most of the elements needed to define an HEP detector.
  - Complete textual detector description, no C++ coding.
LCDD

- Adopted GDML as base geometry definition, then extended it to incorporate missing detector elements

GDML

- expressions (CLHEP)
- materials
- solids
- volume definitions
- geometry hierarchy
LCDD

- Adopted GDML as base geometry definition, then extended it to incorporate missing detector elements

LCDD
- detector information
- identifiers
- sensitive detectors
- regions
- physics limits & cuts
- visualization
- magnetic fields

GDML
- expressions (CLHEP)
- materials
- solids
- volume definitions
- geometry hierarchy
LCDD Structure

- `<lcdd>`: LCDD Root Element
  - `<header>`: Information about the Detector
  - `<iddict>`: Identifier Specifications
  - `<sensitive_detectors>`: Detector Readouts
  - `<limits>`: Physics Limits
  - `<regions>`: Regions (sets of volumes)
  - `<display>`: Visualization Attributes
  - `<gdml>`: GDML Root Element
    - `<define>`: Constants, Positions, Rotations
    - `<materials>`: Material Definitions
    - `<solids>`: Solid Definitions
    - `<structure>`: Volume Hierarchy
  - `<fields>`: Magnetic Field

</lcdd>
CAD

• Clients often have 3D engineering drawings of their setup and would like to simply incorporate those into their Geant4 simulation.

• Difficulties include:
  – Proprietary, undocumented or changing formats
  – Often no connection to materials
  – Mismatch in level of detail required to machine a part and to simulate the response of the part to particles.

• Exchange formats
  – Some standard CAD output formats exist, e.g. STEP & IGES, but these are surface-oriented formats and do not contain material information.
  – GDML is one candidate for standard Geant input.
CAD to Geant4

• FastRad is a free application (but requires a license and has a limit on the part complexity) which can import STEP files, associate materials, and export GDML files. http://www.fastrad.net/

• ST-Viewer from STEP Tools (was free until recently, now part of ST Developer http://www.steptools.com) can import a variety of CAD formats and export a STEP file with an associated material file. Example G02 can import these files and build geometry.

• Neither an endorsement of, nor an advertisement for, these third-party software packages.
Summary

- Runtime geometry definition using text files:
  - opens up the user-base to non-C++ coders
  - provides an avenue to connect to CAD geometries or to other databases (construction, survey, …) as geometry input.
    - No cad2g4 command (yet), but some solutions exist.
  - provides an avenue to export the Geant4-based geometry for downstream clients (e.g. event reconstruction, display).

- GDML plugin fully supports the whole set of G4 solids, including boolean solids, parameterisations, and replications.

- Use of xml allows some control over GDML detector description (can be validated against schema), but also allows extensions to be added.