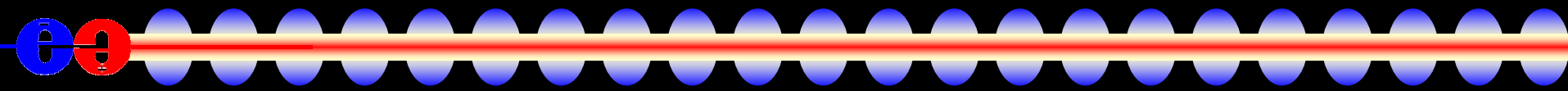
Defining Your Detector

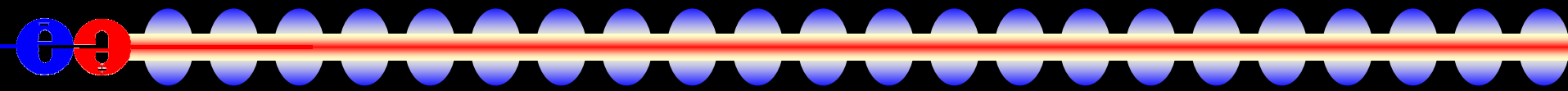
Norman Graf / Tony Johnson
SLAC
May 10, 2005

Geant4 Detector Response Simulation



- Defining detectors at runtime using a single, common executable should enable many detector variants to be simulated and compared.
- Historically, we have limited the allowed subdetector geometries to a few simplified shapes and assumed topologies for flexibility. (detparms)
- Can now do this for arbitrary detector elements using lcdd, built on top of GDML.
- Would like to bind simulation with reconstruction!
  - lcdg4 & hep.lcd : detparms xml file and .ini files, resp.

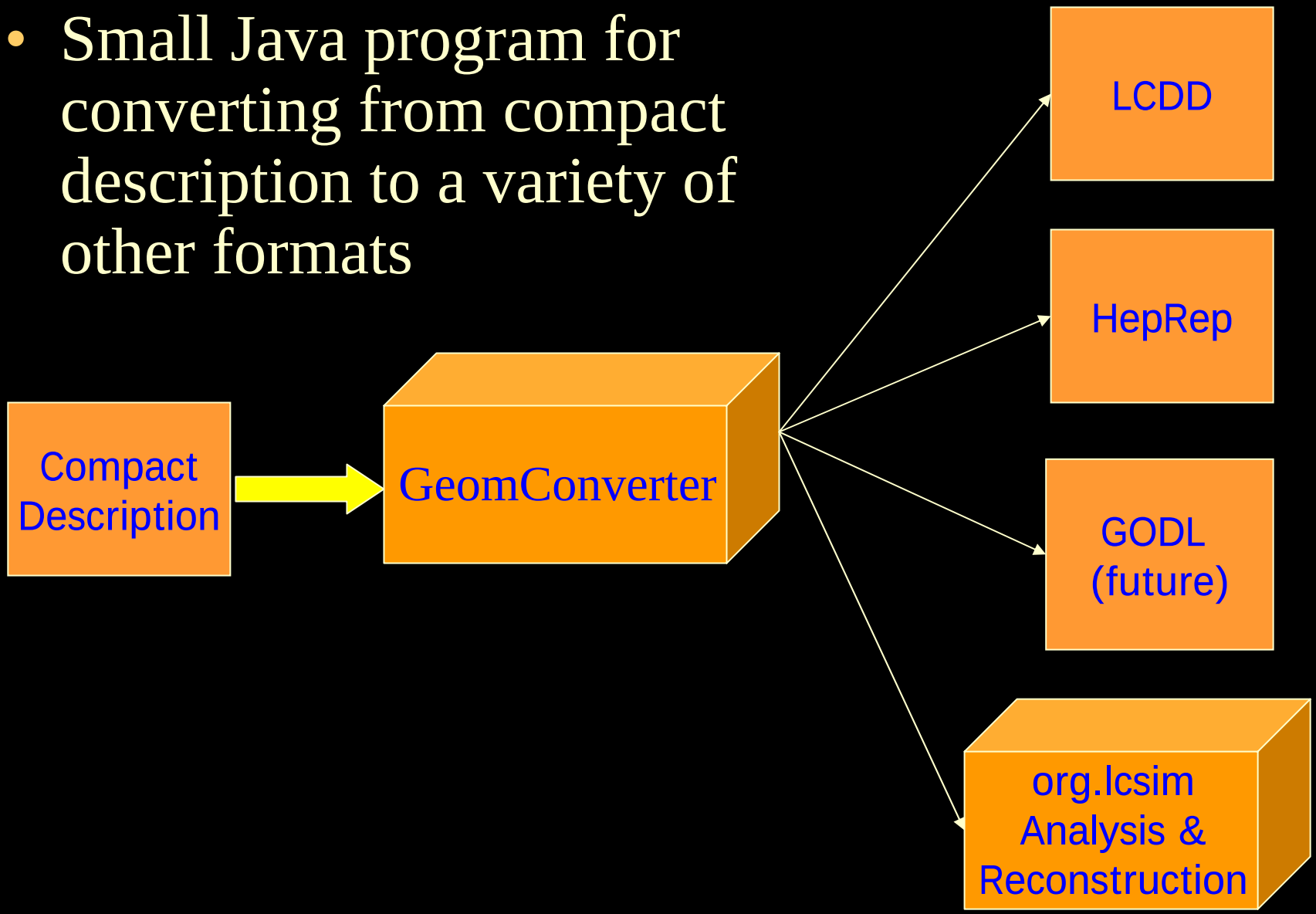
Why another geometry format?



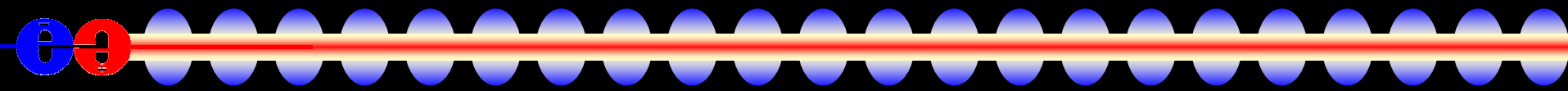
- LCDD is great, handles any geometry, but
  - Files are large, since entry for every G4 volume
  - Simple change (e.g. # layers) may require many changes to LCDD file
  - Not right level of detail for reconstruction
- Compact format is less generic, but
  - Files are much shorter and easier to edit
  - Can handle any likely geometry/segmentation
    - May require additional “drivers” to be implemented in Java
  - Maintains XML advantages cited by Jeremy
  - LCDD can be generated from compact format
- Goal:
  - Rapid prototyping of detector geometries
  - Ability to provide description of new (or existing) detectors for reconstruction (org.lcsim)

GeomConverter

- Small Java program for converting from compact description to a variety of other formats

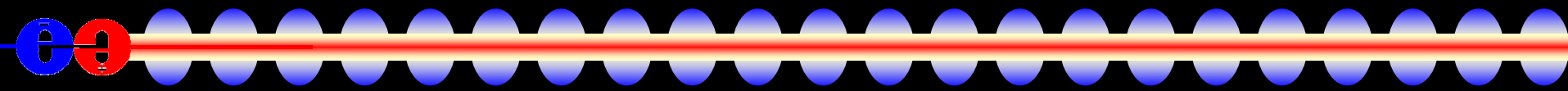


Compact Elements



- <lccdd>
  - <info>
  - <define/>
  - <materials/>
  - <detectors/>
  - <readouts/>
  - <fields/>
- </lccdd>

<*info*>



```
<info name="sdjan03"
        author="Jeremy McCormick"
        version="1.0"
        timestamp="2004-12-13T12:00:53"
        url="http://www.lcsim.org/detector/sdjan03">
  <comment>
   Test of the compact format for sdjan03 detector.
  </comment>
 </info>
```

<define>

```
<define>
  <constant name="cm" value="10"/>
  <!-- world -->
  <constant name="world_side" value="15000" />
  <constant name="world_x" value="world_side" />
  <constant name="world_y" value="world_side" />
  <constant name="world_z" value="world_side" />

  <!-- tracking region -->
  <constant name="tracking_region_radius" value="127.0*cm"/>
  <constant name="tracking_region_zmax" value="168.0*cm"/>

  <constant name="vertex_inner_r" value="1.2*cm"/>
  <constant name="vertex_delta_r" value="1.2*cm"/>
  <constant name="vertex_outer_z" value="12.5*cm"/>
</define>
```

- A few items are required (world_*, tracking_region_*), rest are user defined.

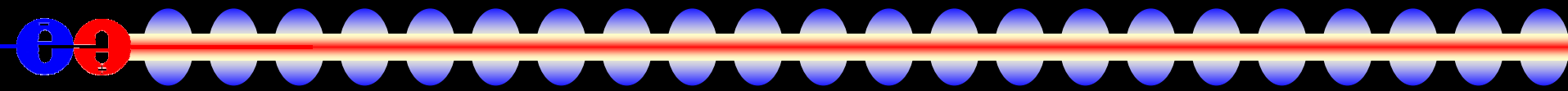
<*materials*>

```
<materials>
    <element name="Silicon_e" formula="Si" Z="14.">
        <atom value="28.09" />
    </element>

    <material name="Polystyrene">
        <D value="1.032" unit="g/cm3" />
        <composite n="19" ref="Carbon_e"/>
        <composite n="21" ref="Hydrogen_e" />
    </material>
</materials>
```

- Currently all materials used must be defined, but plan to have “standard” material file, and define only “special” materials

<*detectors*>



```
<detectors>
   <detector id="2" name="EMBarrel" type="CylindricalCalorimeter"
                                    readout="EcalBarrHits">
      <dimensions inner_r = "127.0*cm" outer_z = "184.0*cm" />
      <layer repeat="30">
        <slice material = "Tungsten" width = "0.25*cm" />
        <slice material = "G10" width = "0.068*cm" />
        <slice material = "Silicon" width = "0.032*cm" sensitive = "yes" />
        <slice material = "Air" width = "0.025*cm" />
      </layer>
   </detector>
</detectors>
```

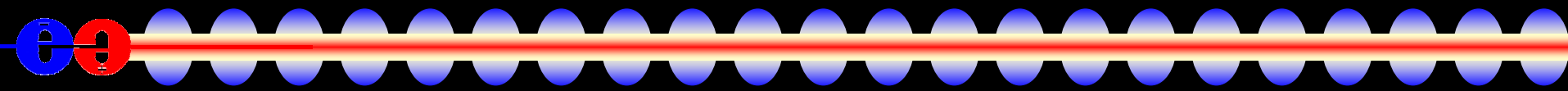
- Contents of detector element depends on “type”, types are extensible, see discussion later.

<*readouts*>

```
<readouts>
   <readout name="EcalBarrHits">
         <segmentation type="ProjectiveCylinder" thetaBins="1000"
                                                 phiBins="2000"/>
         <id>layer:7,system:3,barrel:3,theta:32:11,phi:11</id>
      </readout>
</readouts>
```

- Contents of segmentation element depends on “type”, types are extensible, see discussion later.
- Support projective Barrel and Endcaps, finishing cartesian planar and fixed-z, phi cylindrical.
- IDDecoder in org.lcsim reconstruction uses same information to convert global ⇔ local

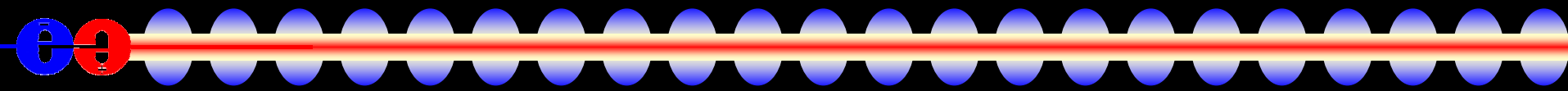
<fields>



```
<fields>
  <field type="Solenoid" name="GlobalSolenoid"
         inner_field="5.0"
         outer_field="-0.6"
         zmax="1000"
         outer_radius="144*cm+(2+1)*34*cm"/>
 </fields>
```

- Contents of field element depends on “type”, types are extensible, see discussion later.

GeomConverter Implementation

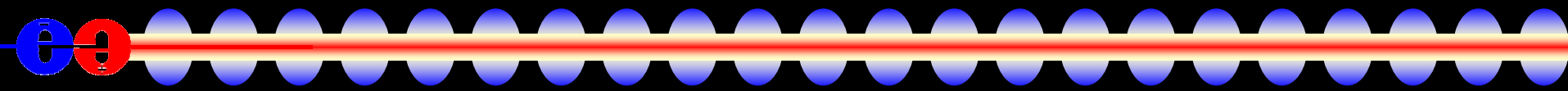


- GeomConverter provides basic functionality for reading file.
  - Plugin modules (Java classes) provide capability of generating different types of output.
  - Plugin drivers (Java classes) provide capability of supporting different types of fields, segmentations, detector shapes.
    - GeomConverter comes with a small set of generic classes for common cases (cylinders, polygonal, etc).
    - Specialized classes can be developed if necessary for strangely shaped detectors.

Dead Material

- Currently dead material can be specified as detector with no sensitive volumes.
- In future will allow dead-material to be specified using full GDML markup, included into LCDD file during generation.
  - Suitable for defining complex shapes such as masking which is normally only relevant for simulation but not reconstruction.

Getting GeomConverter

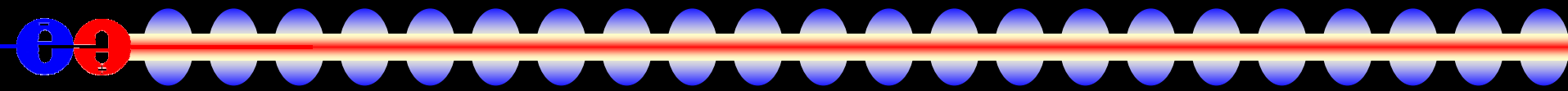


- Web Page:
  - http://www.lcsim.org/software/geomconverter
- CVS:
  - :pserver:anonymous@cvs.freehep.org:/cvs/lcd
  - module GeomConverter
  - After checkout use “maven” to build.
- GeomConverter integrates with org.lcsim reconstruction framework, see tomorrow’s talk.

Detector Repository

- Standard detector descriptions are available in the LCDetectors package
- CVS:
  - :pserver:anonymous@cvs.freehep.org:/cvs/lcd
  - module LCDetectors
- Currently have sdjan03, sdfeb05, sidmay05*
- Plan to support GLD and LDC.
- Also have a template for new designs.

Building Geometry



>setenv CVSROOT
:pserver:anonymous@cvs.freehep.org:/cvs/lcd

>cvs login (hit enter when prompted for password)

> cvs co GeomConverter

> cvs co LCDetectors

> cd GeomConverter

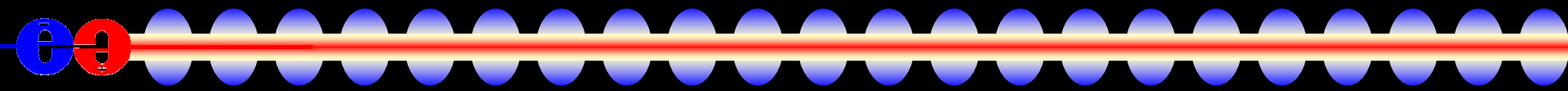
> maven

> maven run

-Drun.class="org.lcsim.geometry.compact.converter.lcdd.Main"

-Dargs="../LCDetectors/detectors/sidmay05/compact.xml
sidmay05.lcdd"

Summary



- Compact detector description provides not only a simpler definition of the detector, but also a binding for the visualization and the reconstruction.
- First release of GeomConverter now available
  - Tested with sdjan03, sdfeb05, sidmay05*
    - Able to generate full LCDD description for SLIC
    - Able to generate HepRep for display with WIRED
  - Encourage others to define variants or other concepts.
- Will continue to enhance in parallel with org.lcsim reconstruction package.