slic
A Geant4-based detector simulation package

Norman Graf, Jeremy McCormick (SLAC)
Geant4 Space Users’ Workshop
August 20, 2010
Mission Statement

- Provide full detector response simulation capabilities for Linear Collider physics program.
- Need flexibility for new detector geometries and readout technologies.
- The system should be flexible, powerful, yet simple to install and maintain.
- Limited resources demand efficient solutions, focused effort.
Full Simulation History

- Provide static binary to run full detector simulations using runtime xml detector descriptions.
  - in-house lcdparm xml format (1998)
  - collaboration with R. Chytracek on GDML (2000)
- GISMO (C++ GEISHA + EGS, lcdparm) 1998
- LCDRoot (Geant4 + Root, lcdparm) 1999
- LCDG4 (Geant4 + sio, lcdparm) 2002
- LCS (Geant4 + lcio, lcdparm) 2004
- slic (Geant4 + lcio, GDML) 2005
Boeing F-22A Raptor
5th Generation Fighter
Full Detector Response Simulation

- Use Geant4 toolkit to describe interaction of particles with matter and fields.
- Thin layer of non-G4 C++ provides access to:
  - Event Generator particle input
  - Detector Geometry description input
  - Detector Hits output
- Geometries fully described at run-time!
  - In principle, as fully detailed as desired.
  - Uses lcdd, an extension of GDML.
- Solution is applicable beyond LC problem domain.
Geometry Definition

• Goal was to free the end user from having to write any C++ code or be expert in Geant4 to define the detector.

• All of the detector properties should be definable at runtime with an easy-to-use format.

• Selected xml, and extended the existing GDML format for pure geometry description.
Why XML?

- Simplicity: Rigid set of rules
- Extensibility: easily add custom features, data types
- Interoperability: OS, languages, applications
- Self-describing data, validate against schema
- 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
ADOPTED GDML AS BASE GEOMETRY DEFINITION, THEN EXTENDED IT TO INCORPORATE MISSING DETECTOR ELEMENTS.

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

GDML
- expressions (CLHEP)
- materials
- solids
- volume definitions
- geometry hierarchy
lcdd Features

- **Regions**: production cuts
- **Physics limits**: track length, step length, etc.
- **Visualization**: color, level of detail, wireframe/solid
- **Sensitive detectors**
  - calorimeter, optical calorimeter, tracker
  - segmentation
- **IDs**
  - volume identifiers (physical volume id)
- **Magnetic fields**
  - dipole, solenoid, field map
- **Utilities**
  - information on Geant4 stores
  - GDML load/dump
“Compact” Description

• The lcdd file is very descriptive, but therefore also very verbose.

• Can be written by hand, but prone to human error.
  – Also, just specific to the simulation and not easily accessible to reconstruction and visualization.

• Developed a “compact” detector description which encapsulates the basic properties of a detector and which is further processed by code to produce the input specific to different clients.
Compact Detector Description

- A number of generally useful detector types (at least for HEP collider detectors) have been developed, such as:
  - Sampling calorimeters
  - TPCs
  - Silicon trackers (microstrip as well as pixel)
  - Generic geometrical support structures
- Can also incorporate GDML snippets
  - Allows inclusion of more complicated volumes derived for instance from engineering (CAD) drawings.
<table>
<thead>
<tr>
<th>SUBDETECTOR</th>
<th>SEGMENTATION</th>
<th>Barrel or Endcap?</th>
<th>Tracker or Cal?</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>CylindricalBarrelCalorimeter</td>
<td>NonprojectiveCylinder,</td>
<td>B</td>
<td>C</td>
<td>simple calorimeter barrel</td>
</tr>
<tr>
<td></td>
<td>ProjectiveCylinder</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CylindricalEndcapCalorimeter</td>
<td>ProjectiveZPlane,</td>
<td>E</td>
<td>C</td>
<td>simple calorimeter endcap</td>
</tr>
<tr>
<td></td>
<td>GridXYZ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DiskTracker</td>
<td>NA</td>
<td>E</td>
<td>T</td>
<td>simple tracker endcap</td>
</tr>
<tr>
<td>MultiLayerTracker</td>
<td>NA</td>
<td>B</td>
<td>T</td>
<td>simple tracker barrel</td>
</tr>
<tr>
<td>PolyhedraBarrelCalorimeter</td>
<td>CartesianGridXY</td>
<td>B</td>
<td>C</td>
<td>planar calorimeter barrel</td>
</tr>
<tr>
<td>PolyhedraEndcapCalorimeter2</td>
<td>CartesianGridXZ</td>
<td>E</td>
<td>C</td>
<td>planar calorimeter endcap</td>
</tr>
<tr>
<td>SiTrackerBarrel</td>
<td>NA</td>
<td>B</td>
<td>T</td>
<td>detailed Si tracker modules</td>
</tr>
<tr>
<td>SiTrackerEndcap2</td>
<td>NA</td>
<td>E</td>
<td>T</td>
<td>detailed Si tracker endcap</td>
</tr>
<tr>
<td>TestBeamCalorimeter</td>
<td>GridXYZ, GlobalGridXY</td>
<td>-</td>
<td>C</td>
<td>calorimeter test beam box layers</td>
</tr>
<tr>
<td>TestBeamTracker</td>
<td>NA</td>
<td>-</td>
<td>T</td>
<td>tracker test beam box layers</td>
</tr>
</tbody>
</table>
Materials

- Chemical Elements
  - name, formula, Z, g/mol

- Compounds
  - name, density
  - mass fractions - element or compound fractions totaling 1.0
    OR
  - number of molecules or atoms

GeomConverter

```xml
<element name="O" formula="O", Z="8.0">
    <atom type="A" unit="g/mol" value="15.9994"/>
</element>

<material name="Oxygen">
    <D type="density" unit="g/cm3" value="1.141"/>
    <composite n="1" ref="O"/>
</material>

<!-- Silicon not shown -->

<material name="Quartz">
    <D type="density" value="2.2" unit="g/cm3"/>
    <composite n="1" ref="Si"/>
    <composite n="2" ref="O"/>
</material>
```

```
org.lcsim.detector.material
org.lcsim.material

resources/org/lcsim/material
    elements.xml
    materials.xml
```
### Physics Limits

<table>
<thead>
<tr>
<th>LIMIT</th>
<th>UNIT</th>
<th>DESCRIPTION</th>
<th>DEFAULT</th>
</tr>
</thead>
<tbody>
<tr>
<td>step_length_max</td>
<td>mm</td>
<td>maximum step length before forcing new step</td>
<td>double max</td>
</tr>
<tr>
<td>track_length_max</td>
<td>mm</td>
<td>maximum track length before track is killed</td>
<td>double max</td>
</tr>
<tr>
<td>time_max</td>
<td>ns</td>
<td>maximum track time before track is killed</td>
<td>double max</td>
</tr>
<tr>
<td>ekin_min</td>
<td>meV</td>
<td>minimum remaining kinetic energy before track is killed</td>
<td>0.0</td>
</tr>
<tr>
<td>range_min</td>
<td>mm</td>
<td>minimum remaining range before track is killed</td>
<td>0.0</td>
</tr>
</tbody>
</table>

#### Example XML

```xml
<limits>
  <limitset name="MyLimits">
    <limit name="step_length_max" value="1.0" unit="mm" />
    <limit name="track_length_max" value="9999.0" unit="m"/>
    <limit name="ekin_min" value="1.0" unit="MeV"/>
  </limitset>
</limits>

<detector … >
  <layer limits="MyLimits" … />
</detector>
```
Settings

- color
  - alpha
  - RGB
- drawing style
  - wireframe
  - solid
- show daughters?
- visible?

SiD (Geant4)

(Display)

<display>
  <vis name="MyVis"
    alpha="1.0"
    r="0.8" g="0.1" b="0.1"
    showDaughters="false"
    visible="true" />
</display>

<detector vis="MyVis" ... />

SiD – Si Tracker (Wired4)
**GeomConverter**

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

This is simply a convenience. A LCDD file can be created in many ways, e.g. from survey database, CAD models, or by hand.
Compact Description - Example

<detector
  id="3"
  name="HADBarrel"
  type="CylindricalBarrelCalorimeter"
  readout="HcalBarrHits"
  vis="HADVis">
  <dimensions inner_r = "141.0*cm" outer_z = "294*cm" />
  <layer repeat="40">
    <slice material="Steel235" thickness="2.0*cm"/>
    <slice material="RPCGasDefault" thickness="0.12*cm" sensitive="yes" region="RPCGasRegion"/>
  </layer>
</detector>
xml: Defining a Tracker Module

<module name="VtxBarrelModuleInner">
  <module_envelope width="9.8" length="63.0 * 2" thickness="0.6"/>
  <module_component width="7.6" length="125.0" thickness="0.26"
    material="CarbonFiber" sensitive="false">
    <position z="-0.08"/>
  </module_component>
  <module_component width="7.6" length="125.0" thickness="0.05"
    material="Epoxy" sensitive="false">
    <position z="0.075"/>
  </module_component>
  <module_component width="9.6" length="125.0" thickness="0.1"
    material="Silicon" sensitive="true">
    <position z="0.150"/>
  </module_component>
</module>
xml: Placing the modules

```
<layer module="VtxBarrelModuleInner" id="1">
  <barrel_envelope inner_r="13.0" outer_r="17.0" z_length="63 * 2"/>
  <rphi_layout phi_tilt="0.0" nphi="12" phi0="0.2618" rc="15.05" dr="-1.15"/>
  <z_layout dr="0.0" z0="0.0" nz="1"/>
</layer>

<layer module="VtxBarrelModuleOuter" id="2">
  <barrel_envelope inner_r="21.0" outer_r="25.0" z_length="63 * 2"/>
  <rphi_layout phi_tilt="0.0" nphi="12" phi0="0.2618" rc="23.03" dr="-1.13"/>
  <z_layout dr="0.0" z0="0.0" nz="1"/>
</layer>

<layer module="VtxBarrelModuleOuter" id="3">
  <barrel_envelope inner_r="34.0" outer_r="38.0" z_length="63 * 2"/>
  <rphi_layout phi_tilt="0.0" nphi="18" phi0="0.0" rc="35.79" dr="-0.89"/>
  <z_layout dr="0.0" z0="0.0" nz="1"/>
</layer>

<layer module="VtxBarrelModuleOuter" id="4">
  <barrel_envelope inner_r="46.6" outer_r="50.6" z_length="63 * 2"/>
  <rphi_layout phi_tilt="0.0" nphi="24" phi0="0.1309" rc="47.5" dr="0.81"/>
  <z_layout dr="0.0" z0="0.0" nz="1"/>
</layer>

<layer module="VtxBarrelModuleOuter" id="5">
  <barrel_envelope inner_r="59.0" outer_r="63.0" z_length="63 * 2"/>
  <rphi_layout phi_tilt="0.0" nphi="30" phi0="0.0" rc="59.9" dr="0.77"/>
  <z_layout dr="0.0" z0="0.0" nz="1"/>
</layer>
```
A Barrel Vertex Detector
Example Vertex Detector

CAD Drawing

GEANT Volumes

LCIO Hits
Barrel Outer Tracker

Composed of overlapping silicon wafers.
Complete Silicon Tracker
Generic Hits Problem Statement

- We wish to define a generic output hit format for full simulations of the response of detector elements to physics events.
- Want to preserve the “true” Monte Carlo track information for later comparisons.
- Want to defer digitization as much as possible to allow various resolutions, readout technologies, etc. to be efficiently studied.
Types of Hits

• “Tracker” Hits
  – Position sensitive.
  – Particle unperturbed by measurement.
  – Save “ideal” hit information.

• “Calorimeter” Hits
  – Energy sensitive.
  – Enormous number of particles in shower precludes saving of each “ideal” hit.
  – Quantization necessary at simulation level.
Tracker Hit

- MC Track handle
- Encoded detector ID (detector dependent)
- Hit position in sensitive volume
- Track momentum at hit position.
- Energy deposited in sensitive volume.
- Time of track's crossing.
- Path length in sensitive volume.

Sufficient information to do hit digitization.
Calorimeter Hit

- Encoded detector ID (detector dependent)
- MC IDs for tracks contributing to this cell.
- Energy deposited.
- Time of energy deposition.
- Repeated for each energy contribution.
- Support recently added for optical calorimeters
  - Can store Cerenkov and scintillation light.
LCIO

- Persistency framework for LC simulations.
- Currently uses SIO: Simple Input Output
  - on-the-fly data compression
  - random access
  - C++, Java, python (and FORTRAN!) implementations available
- Changes in IO engine designed for (e.g. root).
- Extensible event data model
  - Generic Tracker and Calorimeter Hits.
  - Monte Carlo particle hierarchy.
**slic: The Executable**

- Provide static executable on Linux, Windows, Mac.
  - or build your own with SimDist build kit
- Commandline or G4 macro control.
- Only dependence is local detector description file.
  - Trivial grid/cloud usage (no database call-backs, etc.)
- Event input via stdhep, particle gun, …
- Detector input via GDML, lcdd
- Response output via LCIO using generic hits or Geant4 scoring via macros.
slic: Getting the executable

- Download executables from [http://www.lcsim.org/dist/slic](http://www.lcsim.org/dist/slic)
- Or build your own
  > cvs -d :pserver:anonymous@cvs.freehep.org:/cvs/lcd co SimDist
  > cd SimDist
  > chmod +x build.sh
  > ./build.sh
- Tested on some Linuxes (rhel, sl), Mac OSX and Windows/cygwin
Detector Full Simulation

- MC Event (stdhep, gps)
- Geometry (lcdd)
- Compact Geometry Description (xml)

GEANT4

slic

Raw Event (slcio) or scoring info

Reconstruction, Analysis, Visualization, …
Detector Variants

• Runtime XML format allows variations in detector geometries to be easily set up and studied:
  – Sampling calorimeters:
    • absorber materials, dimensions
    • Readout technologies, e.g. RPC, scintillator
    • Layering (radii, number, composition)
    • Readout segmentation (size, projective vs. nonprojective)
  – Total absorption crystal calorimeters
    • Optical properties
  – Tracking detector technologies & topologies
    • TPC, silicon microstrip, silicon pixels
ILC Full Detector Concepts

SiD

GLD

LDC
slic & lcdd: Summary

- Provides a complete and flexible detector simulation package capable of simulating arbitrarily complex detectors with runtime detector description.
- Used by ILC, CLIC and Muon Collider detector community for simultaneous and iterative evolution of different detector concepts and their variations.
- Being used for ATLAS upgrade tracking studies.
- Has been applied to CPT simulations.
- Could be used by other communities (astro, medical) for rapid prototyping or full simulation.
Additional Information

• ILC Detector Simulation [http://www.lcsim.org](http://www.lcsim.org)
• ILC Forum [http://forum.linear collider.org](http://forum.linear collider.org)
• SLIC [http://www.lcsim.org/software/slic](http://www.lcsim.org/software/slic)
• LCDD [http://www.lcsim.org/software/lcdd](http://www.lcsim.org/software/lcdd)
• LCIO [http://lcio.desy.de](http://lcio.desy.de)
• Wiki [http://confluence.slac.stanford.edu/display/ilc/Home](http://confluence.slac.stanford.edu/display/ilc/Home)