Fast Detector Simulation Using Lelaps
Overview

- Introduction
  - What does it do?
  - What is it?
  - History
- CEPack
  - Geometry and Materials
  - Tracking, Multiple Scattering and dE/dx
  - Electromagnetic and Hadronic Shower Parameterization
  - Decays and Gamma Conversions
- Using CEPack inside Geant4
- Standalone CEPack: Lelaps
- Performance
- Future
Introduction: Lelaps

Lelaps ("storm wind") was a dog with such speed that, once set upon a chase, he could not fail to catch his prey. Having forged him from bronze, Hephaestus gave him to Zeus, who in turn gave him to Athena, the goddess of the hunt. Athena gave Lelaps as a wedding present to Procris, daughter of Thespius, and the new bride of famous hunter Cephalus.

A time came when a fox created havoc for the shepherds in Thebes. The fox had the divine property that its speed was so great that it could not be caught. Procris sent Lelaps to catch the fox. But because both were divine creatures, a stalemate ensued, upon which Zeus turned both into stone. Feeling remorse, Zeus elevated Lelaps to the skies, where he now shines as the constellation Canis Major, with Sirius as the main star.
Introduction: Lelaps

Clarification:
No need to infer that Lelaps (the program) is a dog!

Rather, think of “stellar” performance!
Introduction: What does it do?

- It swims particles through detectors, taking into account magnetic fields, multiple scattering and dE/dx energy loss.
- It produces parameterized showers in EM and hadronic calorimeters.
- It converts gammas.
- It supports decays of certain short-lived particles (“V” decays).
- It does all this very fast.
Wired picture of an $ee \rightarrow ZZ$ event as simulated by Lelaps for the LCD Large Detector model.

White dots represent track hits. The blue showers are in the EM calorimeter, the red ones in the hadron calorimeter.
The Present – Lelaps and CEPack 3.x

Wired picture of the decay chain:

\[ \Omega^- \rightarrow \Xi^0 \pi^- \]

\[ \Xi^0 \rightarrow \Lambda \pi^0 \]

\[ \Lambda \rightarrow p \pi^- \]

\[ \pi^0 \rightarrow \gamma \gamma \]

as simulated by Lelaps for the LCD LD model.
The Present – Lelaps and CEPack 3.x

Wired picture of a gamma conversion as simulated by Lelaps for the LCD LD model.
Introduction: What is it?

- Lelaps consists of a set of C++ class libraries and a main program, which itself is called lelaps.
- The main library is called CEPack, which contains the actual simulation toolkit. It uses only one utility library, “vec”.
- Other utility libraries include: IStdHep, vlutil, plotpp, pl3.
- Main programs (“lelapses”) have been written for BaBar and for LCD (both LDMar01 and SDJan03 are implemented).
- CEPack can also be used in conjunction with Geant4 parameterized volumes. In this way it is integrated in BaBar’s Geant4-based simulation.
- The standalone version for LCD reads StdHep generator files and produces SIO output files that can be read by JAS and LCDWired.
History

1. 1997 CEPack 0.x: CAD-EGS pilot project to import geometries into EGS4 from standard 3D file format (VRML). CEPack only does geometry.
2. 1998 CEPack 1.0: Geant4 program with Parameterized Volumes for BaBar SVT and DCH.
   CEPack does transport with dE/dx and multiple scattering.
3. 1999 Integration into Bogus (CEPack 1.x)
4. 2000 BaBar fast simulation shelved
5. 2002 Lelaps stand-alone program for LCD (CEPack 2.x)
   BaBar revived fast simulation, CEPack 2.x used in Moose
6. 2003 Lelaps 3.x for LCD using CEPack 3.x. Upgrade BaBar to CEPack 3.x
   CEPack does parameterized shower simulation
   CEPack does gamma conversions and decays
Caution

- The code has evolved beyond the original design, and is in fact still evolving.
- Because of its history, there are places with a lack of consistency in naming conventions.
- The interfaces are still not frozen.

Therefore, don’t depend on the details of what follows. Things WILL change!
Geometry in CEPack

- Geometries are constructed using CENodes (which may contain a list of subnodes).
- A number of common CENodes are predefined
  - cylinders, cones, boxes, spheres
- Transformations may be applied to CENodes in order to position and orient them. Arbitrary affine transformations are allowed.
- CENodes need to provide methods for tracks entering and exiting, and for determining whether points are inside them or not.
- CENodes may be assigned a numeric id and subid.
- CENodes may implement a method to compute a subid from a location.
  - May be used to compute calorimeter segmentation.
- CENodes do not have to be 3D objects.
  - Several predefined CENodes consist of one or more 2D surfaces.
Most objects in the LD are cylinders.

There are some conical masks (not drawn).

TPC and barrel muon detector each contain a CENode with a set of concentric cylinder surfaces.

Muon endcaps use a CENode with a set of cylindrical slices.

Calorimeters use subid calculation for their segmentation.
Geometry in CEPack (continued)

- Example: create a lead pipe ("double-walled cylinder")
  - `CENode *ce = new CEDWCylinder(r1, r2, length, nlayers, radially);`
  - `ce->rotate_around_x(radians(90.0));` // in Z
  - `ce->translate(vec(0, 0, offset));` // if needed
  - `ce->material = CEEElement("Pb");`

- Set type of object (Mask, Tracker, EM or Hadron calorimeter) and whether to report hits:
  - `ce->setflags(CEN_MASK | CEN_DETECTOR);`
  - `ce->setid(666);`
  - `ce->set_samplingFraction(0.02);` // if a calorimeter
Materials in CEPack: Elements

- Three types of materials: CEEElement, CECompound and CEMixture.
- CEEElement: basic knowledge of all the elements is built in.
  - CEEElement Helium(“He”), Lead(“Pb”), Oxygen(“O2”);
  - Knows that helium and oxygen are gasses and lead a solid
- Can specify pressure and temperature for gasses, density for solids.
  - CEEElement Argon(“Ar”, 5.0, 298.15); // in atm and K
  - CEEElement LowDensityCarbon(“C”, 0.01); // in g/cm³
- May use common English name:
  - CEEElement Tungsten(“Tungsten”);
Materials in CEPack: Compounds

- Compounds specified by (simple) chemical formula and density (for solids and liquids) or pressure and temperature (gasses):
  - CECompound Na2O("Na2O", 2.27);
  - CECompound CaO("CaO", 3.34);
  - CECompound Al2O3("Al2O3", 3.97);
  - CECompound SiO2("SiO2", 2.196);
  - CECompound MgO("MgO", 3.6);
  - CECompound Epichlorohydrin("C3H5ClO", 1.18);

- May use floating point:
  - CECompound YBaCuO("YBa2Cu3O6.6", density);

- May give it a convenient name:
  - CECompound Epichlorohydrin("Epoxy", "C3H5ClO", 1.18);

- May not use parentheses in formula.
Mixtures can be created by adding defined materials either by volume or by weight:

- CEMixture G10("G10");
- G10.addByWeight(Na2O, 17.0);
- G10.addByWeight(CaO, 5.0);
- G10.addByWeight(Al2O3, 1.0);
- G10.addByWeight(SiO2, 73.0);
- G10.addByWeight(MgO, 4.0);
- G10.addByWeight(Epichlorohydrin, 54.0);

The “fractions” do not have to add up to 1 (or 100%) – with each addition, the fractions are renormalized.

Use CEGasMixture or CECondensedMixture for mixtures of gasses and solids in order to specify whether the result is a gas or solid.
Materials in CEPack (continued)

- All needed quantities are calculated automatically
  - Constants needed for multiple scattering and energy loss
  - Radiation lengths (Tsai, PDG)
  - Interaction lengths (from a fit to element data)
  - Other constants needed for shower parametrization
Tracking, Multiple Scattering and dE/dx in CEPack

- Tracking is performed by taking steps along a linear trajectory with endpoints on a helix, such that the sagitta stays below a certain (settable) maximum.
- CENodes have bounding spheres (or bounding cylinders).
  - When computing distances to CENodes, only “relevant” CENodes are considered.
- After each step, the amount of material traversed is checked: if enough material was traversed, multiple scattering and energy loss is performed and track parameters and list of relevant CENodes are recalculated.
- When an intersection occurs within a step, the fractional step is executed, the CENode is entered, and the remaining fraction of the step follows.
Multiple Scattering and dE/dx (continued)

- Multiple scattering is performed using the algorithm of Lynch and Dahl.
- Material is “saved up” along the track until there is enough.

- dE/dx is calculated using the methods by Sternheimer and Peierls.

- All constants precalculated by the material classes.
Electromagnetic Shower Parameterization in CE Pack

- Electromagnetic showers are parameterized using the algorithms of Grindhammer and Peters.
    (Paper is a 1993 conference contribution, submitted by request to the archive in 2000).
- Calorimeters are treated as homogeneous media, with longitudinal shower profile given by a gamma distribution (t in radiation lengths):

\[
\frac{1}{E} \frac{dE(t)}{dt} = f(t) = \frac{(\beta t)^{\alpha-1} \beta e^{-\beta t}}{\Gamma(\alpha)}
\]

- Coefficients \( \alpha \) and \( \beta \) depend on the material (Z) and energy.
- The profiles are fluctuated, and correlations between \( \alpha \) and \( \beta \) are taken into account.
Electromagnetic Shower Parameterization (continued)

- For each step of one radiation length, a radial profile is computed of the form:

\[
\frac{1}{dE(t)} \frac{dE(r,t)}{dr} = f(r) = p \frac{2rR_C}{(r^2 + R_C^2)^2} + (1-p) \frac{2rR_T}{(r^2 + R_T^2)^2}
\]

with \(R_C\) the median radius (in units of Molière radius) of core component of the shower, \(R_T\) the median radius of tail component and \(p\) the fraction of “core” in the shower.

- \(R_C, R_T\) and \(p\) are functions of \(Z\), shower depth \(t\) and \(E\).

- Energy \(dE(t)\) is divided into spots (another gamma distribution with parameters depending on \(t\)). Spots are thrown in \(r\) according to the profile above, uniformly in \(\phi\) and also uniformly between \(t\) and \(t+1\).

- Roughly, about 400 spots are generated per GeV of shower energy.

- Spots are reported as “hits”.
EM Shower Parameterization (continued)

CEPack simulation of BaBar EM calorimeter in Moose (courtesy of Dominique Mangeol). See also the BaBar web site (computing – simulation – fast simulation).
Hadronic Shower Parameterization in CEPack

- Hadronic showers are parameterized using code that is similar to the code for electromagnetic showers, with some modifications:
- The location where the shower starts is simulated using an exponential law with attenuation given by the interaction length.
- The longitudinal profile uses the Bock parameterization.
- A combination of two gamma distributions, one using radiation lengths and the other interaction lengths, is used.
- Bock parameterization does not specify radial profiles. For the moment we use a radial profile similar to Grindhammer & Peters (for EM showers) but with radiation lengths replaced by interaction lengths and faster spread with depth. The parameters still need to be fine-tuned.
Shower Parameterization in CEPack

- Parameterized shower simulation was compared to Geant4.
- In general pretty good agreement for EM showers. Hadronic showers agree pretty well longitudinally, but not as well radially.
- Hadronic shower parameterization has been tweeked since then.
Shower Parameterization in CEPack (continued)

Comparison of CEPack longitudinal profile (green) of a 10 GeV electron in an EM calorimeter with Geant4 (orange).
Decays and Gamma Conversions

- Decays of unstable particles and gamma conversions are performed as follows:
- One starts with a CEParticleList, to which a primary particle is added
- Lelaps loops over the particles in the list, turns them into CETracks and swims them through the geometry
- Tracks of decaying particles and converting gammas stop at the decay/conversion point
- For supported unstable particles, CEPack utility function picks decay mode/conversion and adds decay products to particle list
- Lelaps continues to loop over the particles in the list until there are none left
- Supported unstable particles are pi0, K0-short (K0-long treated as stable), Lambda, Sigma+/-/0, Xi-,0 and Omega- (“V” decays)
- Only decay modes > 2%.
Using CEPack inside Geant4

- To use CEPack inside Geant4, create a Geant4 parameterized model, e.g.:
  ```
  class BgsSvtParamModel : public G4VFastSimulationModel
  ```
- In its setup(), create the CENode corresponding to the CEPack simulation with all its subnodes.
- In Doit(), convert from G4FastTrack track to CETrack, and call track.swim()
- By subclassing CETrack, all hits are reported using CETRack report_hit() method. Convert hits to your favorite format.
- Update G4FastStep (or call KillPrimaryTrack()):
  ```
  fastStep.SetPrimaryTrackFinalPosition(
      G4ThreeVector(track.p.getx() * m, track.p.gety() * m, track.p.getz() * m), false);
  fastStep.SetPrimaryTrackFinalKineticEnergyAndDirection(
      (track.energy - track.mass) * GeV,
      G4ThreeVector(track.q.getx(), track.q.gety(), track.q.getz()), false);
  ```
CEPack and Lelaps

- Lelaps for LCD is a standalone program which uses CEPack.
- It sets up the CEPack geometry (both LDMar01 and SDJan03 are supported).
- It uses IStdHep ("StdHep light" – included in the distribution): a class library to to read generator event files in StdHep format. StdHep particles are converted to SIO particles.
- Loops over events, creating CETracks. When hits are reported, they are added to SIO hit lists. For calorimeter hits, the “spots” are first accumulated and turned into energy depositions in individual calorimeter cells and then added.
- Finally, the SIO event structure is written out.
Lelaps (LDMar01)
Lelaps (SDJan03)
Performance

- 100 $ee\rightarrow ZZ$ events* at 500 GeV c.m. energy in the LD on a 1.4 GHz Pentium 4 running Linux (Noric09) gives the following performance (all numbers converted to 1 GHz processor speed):

<table>
<thead>
<tr>
<th></th>
<th>Overall time (at 1 GHz)</th>
<th>Time/event (msec at 1 GHz)</th>
<th>Time/track (msec at 1 GHz)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tracking only (with dE/dx &amp; MS)</td>
<td>23.6 s</td>
<td>236 ms</td>
<td>3.95 ms</td>
</tr>
<tr>
<td>Tracking + EM showers</td>
<td>28.8 s</td>
<td>288 ms</td>
<td>4.82 ms</td>
</tr>
<tr>
<td>Tracking + EM and hadronic showers</td>
<td>39.4 s</td>
<td>394 ms</td>
<td>6.60 ms</td>
</tr>
<tr>
<td>Same + SIO output file</td>
<td>77.3 s</td>
<td>773 ms</td>
<td>12.9 ms</td>
</tr>
</tbody>
</table>

* panpy-ZZ-500-001001-gen-1.stdhep (5973 tracks)
(Latest version is actually faster)
Performance (continued)

- Tracking alone: 5 events per second (at 1 GHz).
  - This can probably be improved (it used to do better before the recent changes).
- Adding parameterized showering costs almost a factor 2.
  - No optimization attempted so far.
- SIO output file (10 MB, compressed) costs almost another factor of 2!
- Platform/machine dependent. Tracking only, time per event:
  - Noric09 (1.4 GHz, gcc 2.95.3) 0.236 sec at 1 GHz
  - Tersk08 (0.44 GHz, Solaris WS 6u1) 0.275 sec at 1 GHz
  - Windows (2 GHz, cygwin with gcc 3.2) 0.458 sec at 1 GHz
Future

- Lelaps and CEPack interfaces are not yet frozen!
- New features planned for CEPack
  - Combinatorial geometry
  - Shower continuation into next volume
- New features planned for Lelaps:
  - Read in geometry from some “standard” file format
  - Write LCIO rather than SIO (almost complete)
- Old features need to be tested more thoroughly
  - Tune hadronic showers
- Review of code, improvements in consistency
- Revisit optimization
The End