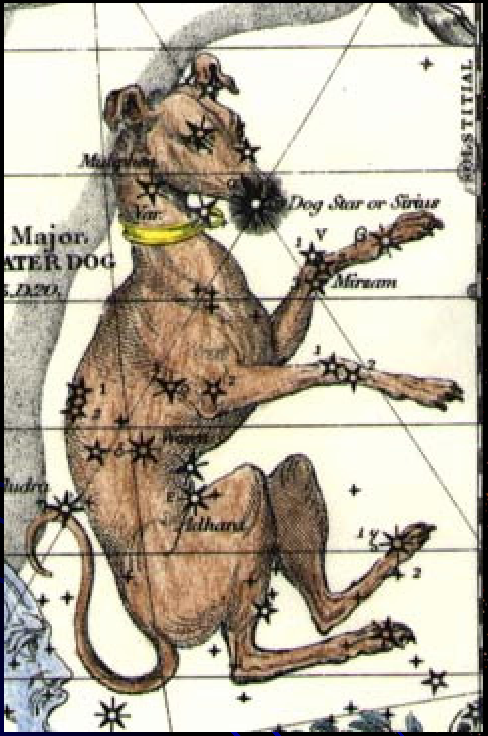
Ληλαψ 3

Fast Detector Simulation Using Lelaps

Overview

- Introduction
  - What does it do and what is it?
- History
  - CEPack Tool Kit
  - Lelaps standalone
- 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.

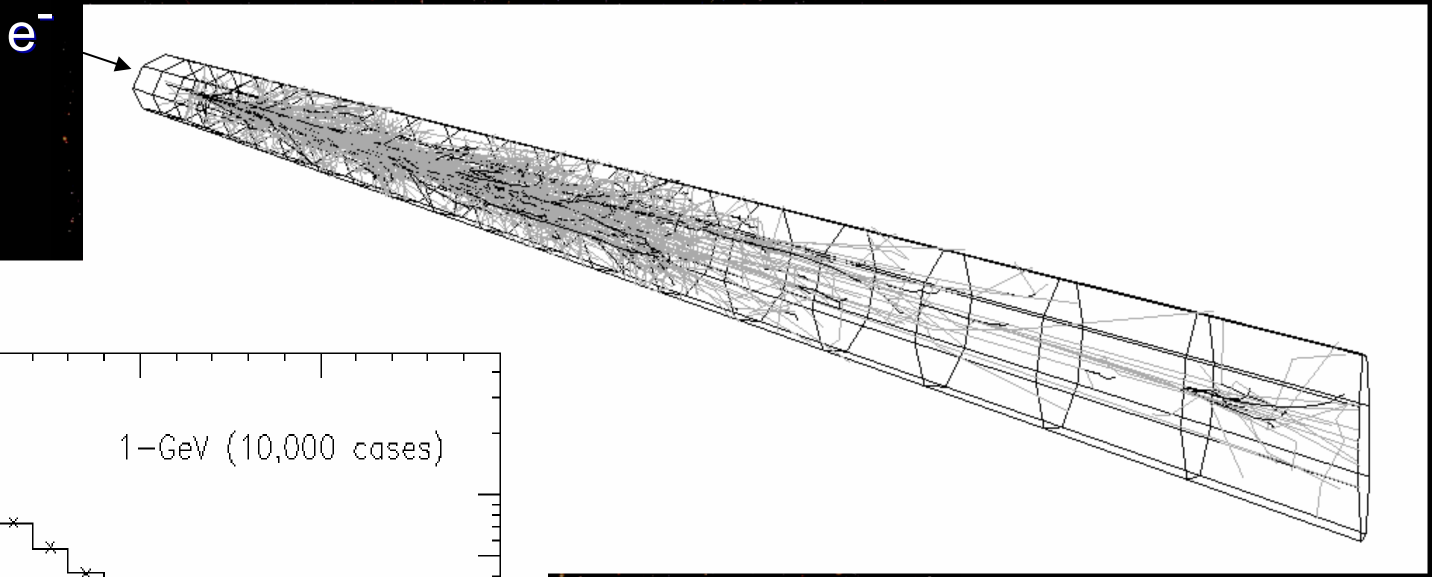
Introduction: What is it?

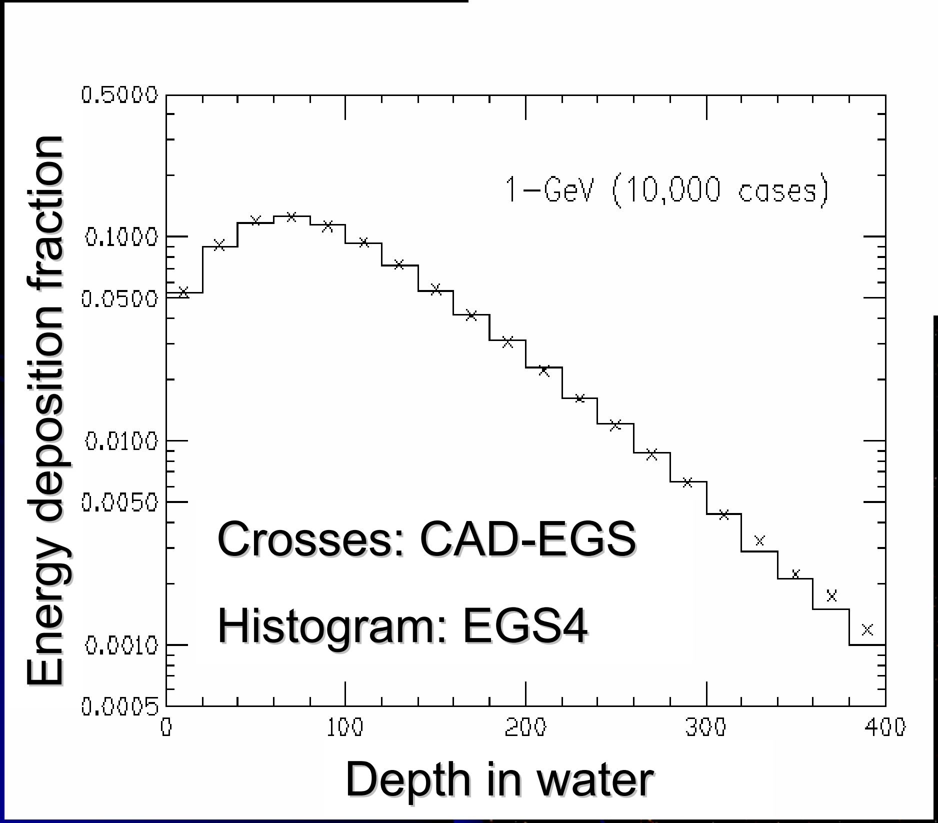
- Lelaps consists of a set of C++ class libraries and a main program, which itself is called lelaps.
- The main library is called CEPack, which contains the actual simulation toolkit. It uses only one utility library, “vec”.
- Other utility libraries include: lStdHep, vlutil, plotpp, pl3.
- Main programs (“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 – 1997: CAD-EGS

- 1997 CEPack 0.x: CAD-EGS pilot project to import geometries into EGS4 from standard 3D file format (VRML). CEPack only does geometry.
  1. "The CAD-EGS Project: Using CAD geometries in EGS4", W.G.J. Langeveld, W.R. Nelson, SLAC TN 97-001, July 1997.
  2. "The CAD-EGS Project: Using CAD geometries in EGS4", W.G.J. Langeveld, J.C. Liu, W.R. Nelson, Proceedings of the First International Workshop on EGS4, August 26--29, 1997, KEK, Tsukuba, Japan, page 75.

History – 1997: CAD-EGS





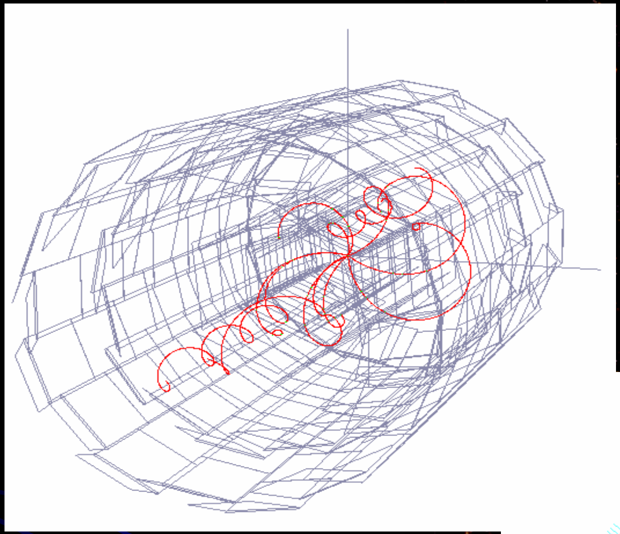
Crannel benchmark: 1 GeV electron beam in a 4 m long water cylinder, radius 12 cm, sampled in 20 segments.

History – 1998: CEPack 1.x and BaBar

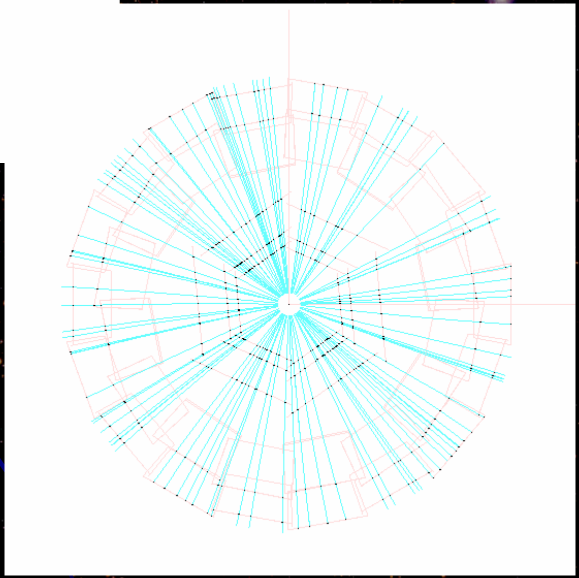
- 1997 CEPack 0.x: CAD-EGS pilot project to import geometries into EGS4 from standard 3D file format (VRML). CEPack only does geometry.
  1. "The CAD-EGS Project: Using CAD geometries in EGS4", W.G.J. Langeveld, W.R. Nelson, SLAC TN 97-001, July 1997.
  2. "The CAD-EGS Project: Using CAD geometries in EGS4", W.G.J. Langeveld, J.C. Liu, W.R. Nelson, Proceedings of the First International Workshop on EGS4, August 26--29, 1997, KEK, Tsukuba, Japan, page 75.
- 1998 CEPack 1.0: Geant4 program with Parameterized Volumes for BaBar SVT and DCH.

  CEPack does transport with dE/dx and multiple scattering.
- 1999 Integration into Bogus (CEPack 1.x)

History – 1998: CEPack 1.x and BaBar



BaBar Silicon Vertex Detector as modeled by CEPack. All 320 wafers are individually represented.



100 one-GeV muons, end view. Black dots represent hits.

(BaBar Drift Chamber was added later.)

History – 2002: Lelaps and LCD

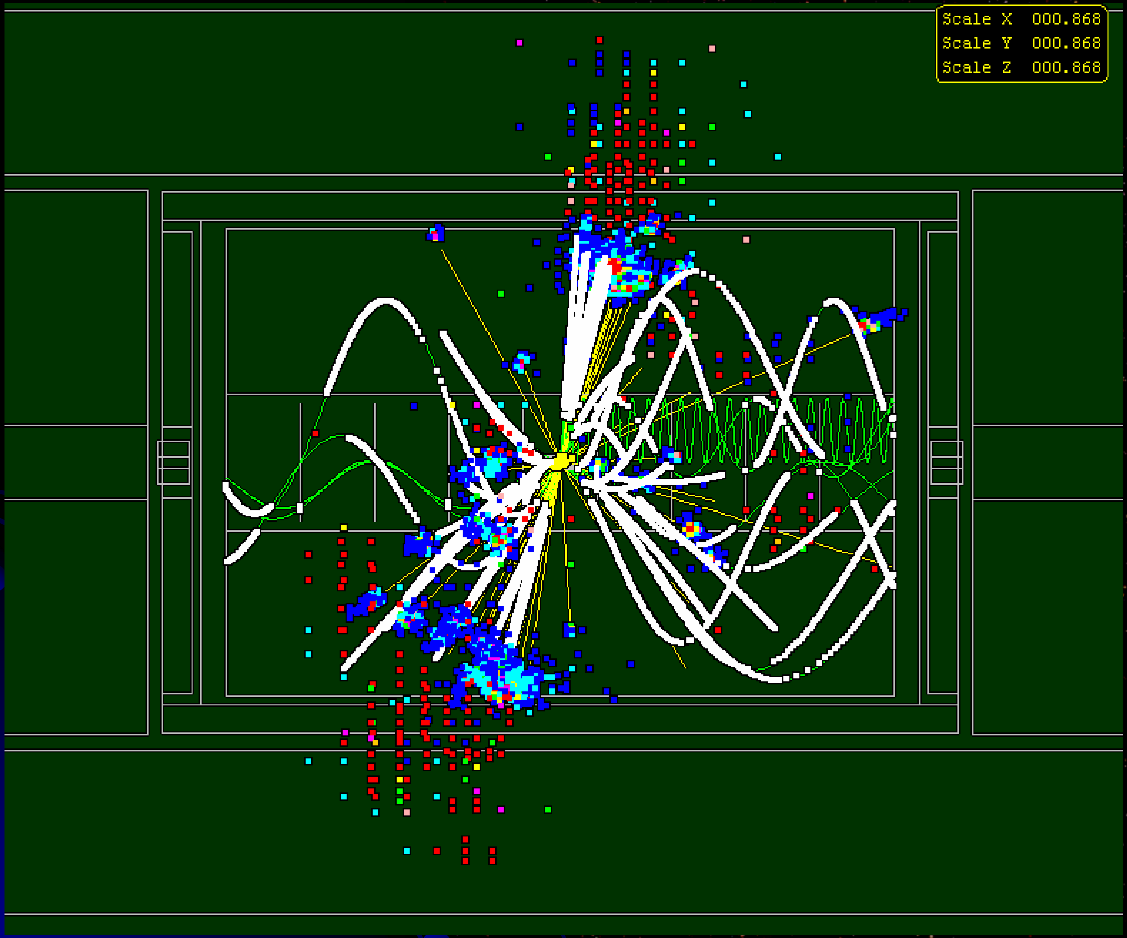
- 1997 CEPack 0.x: CAD-EGS pilot project to import geometries into EGS4 from standard 3D file format (VRML). CEPack only does geometry.
  1. "The CAD-EGS Project: Using CAD geometries in EGS4", W.G.J. Langeveld, W.R. Nelson, SLAC TN 97-001, July 1997.
  2. "The CAD-EGS Project: Using CAD geometries in EGS4", W.G.J. Langeveld, J.C. Liu, W.R. Nelson, Proceedings of the First International Workshop on EGS4, August 26--29, 1997, KEK, Tsukuba, Japan, page 75.
- 1998 CEPack 1.0: Geant4 program with Parameterized Volumes for BaBar SVT and DCH.

  CEPack does transport with dE/dx and multiple scattering.
- 1999 Integration into Bogus (CEPack 1.x)
- 2000 BaBar fast simulation shelved
- 2002 Lelaps stand-alone program for LCD (CEPack 2.x)

  BaBar revived fast simulation, CEPack 2.x used in Moose
- 2003 Lelaps 3.x for LCD using CEPack 3.x. Upgrade BaBar to CEPack 3.x

  CEPack does parameterized shower simulation.

History – 2002: Lelaps and LCD



Wired picture of an ee→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

- 1997 CEPack 0.x: CAD-EGS pilot project to import geometries into EGS4 from standard 3D file format (VRML). CEPack only does geometry.
  1. "The CAD-EGS Project: Using CAD geometries in EGS4", W.G.J. Langeveld, W.R. Nelson, SLAC TN 97-001, July 1997.
  2. "The CAD-EGS Project: Using CAD geometries in EGS4", W.G.J. Langeveld, J.C. Liu, W.R. Nelson, Proceedings of the First International Workshop on EGS4, August 26--29, 1997, KEK, Tsukuba, Japan, page 75.
- 1998 CEPack 1.0: Geant4 program with Parameterized Volumes for BaBar SVT and DCH.

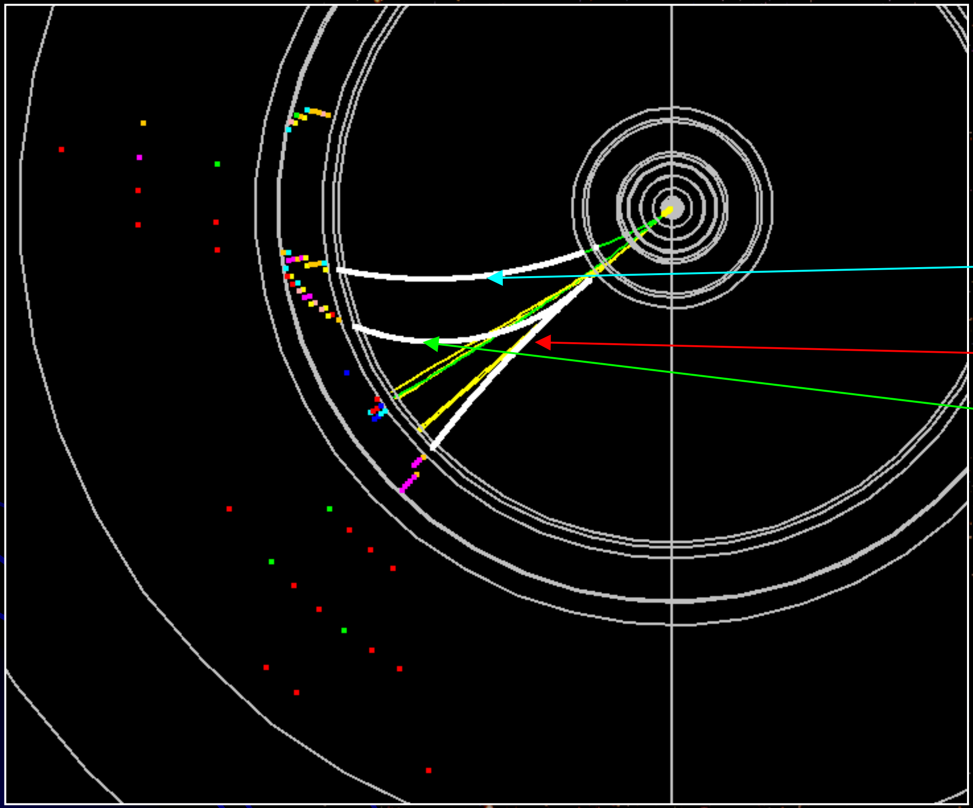
  CEPack does transport with dE/dx and multiple scattering.
- 1999 Integration into Bogus (CEPack 1.x)
- 2000 BaBar fast simulation shelved
- 2002 Lelaps stand-alone program for LCD (CEPack 2.x)

  BaBar revived fast simulation, CEPack 2.x used in Moose
- 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

The Present – Lelaps and CEPack 3.x



Wired picture of the decay chain:

$\Omega^- \rightarrow \Xi^0 \pi^-$

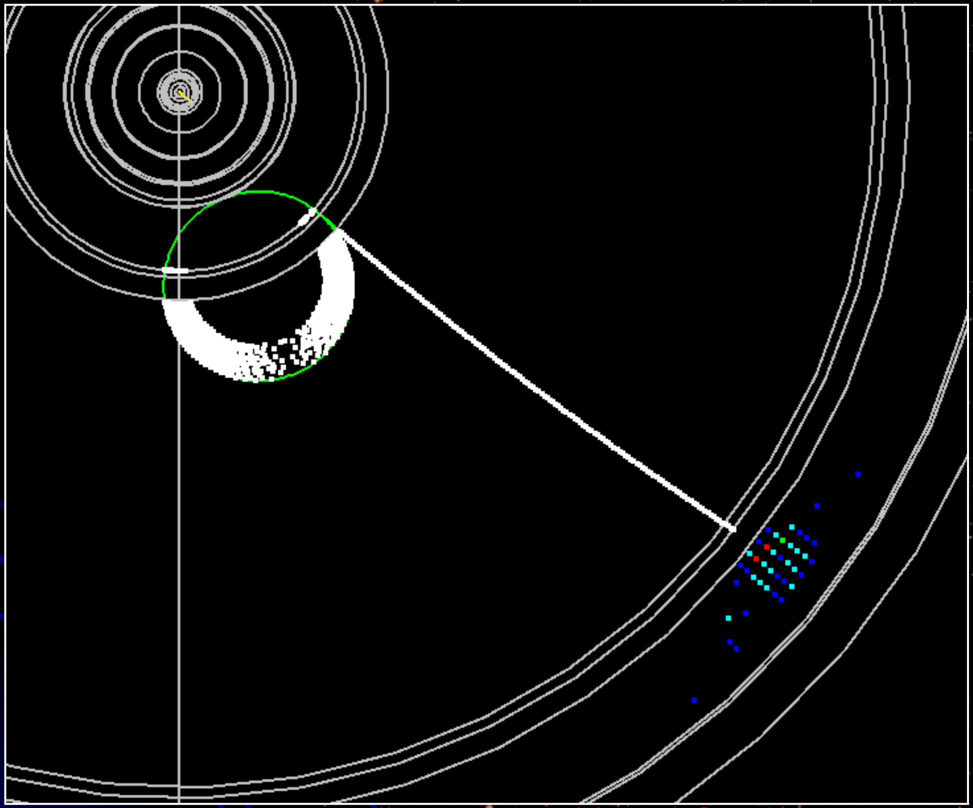
$\Xi^0 \rightarrow \Lambda \pi^0$

$\Lambda \rightarrow p \pi^-$

$\pi^0 \rightarrow \gamma \gamma$

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.

Caution

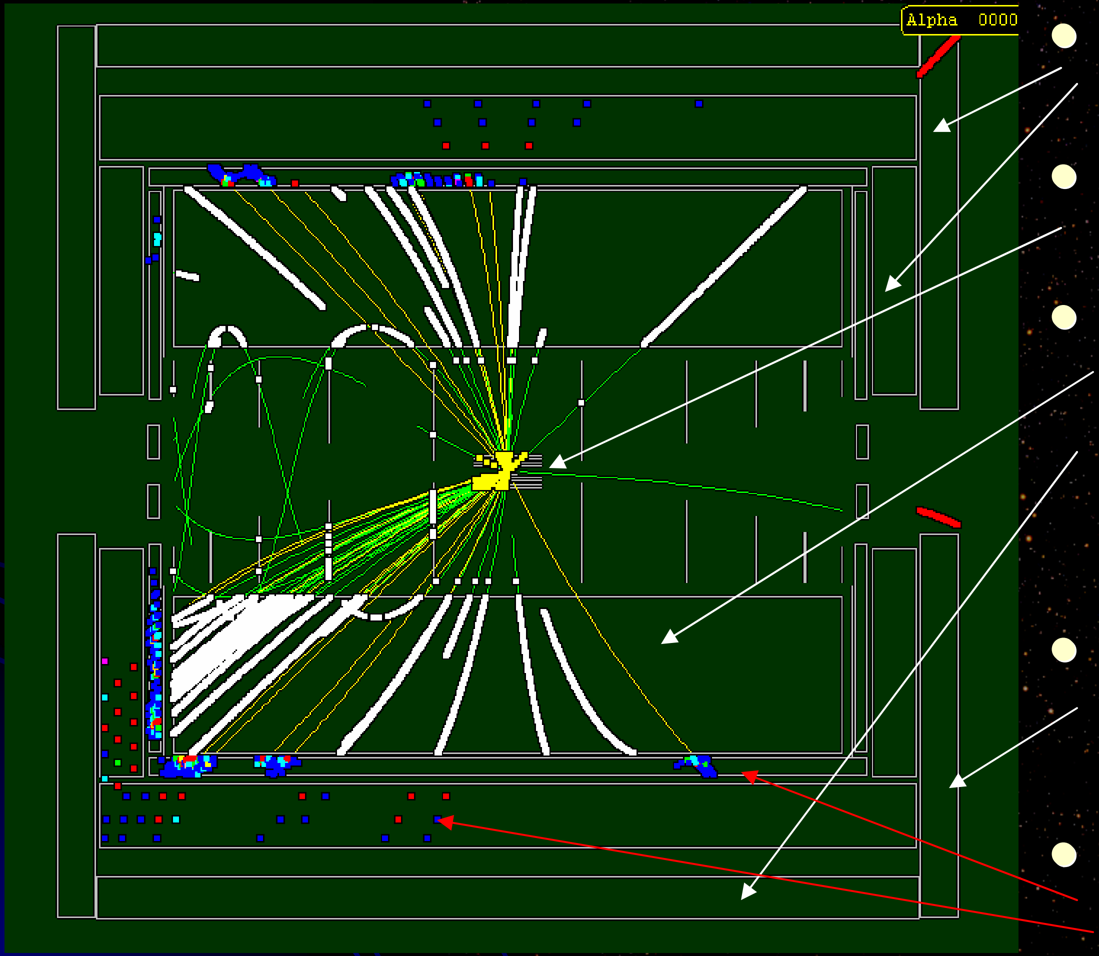
- The code has evolved beyond the original design, and is in fact still evolving.
- Because of its history, there are places with a lack of consistency in naming conventions.
- The interfaces are still not frozen.

- 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.

Geometry in CEPack: LD



- 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: 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 = CEElement("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: CEElement, CECompound and CEMixture.
- CEElement: basic knowledge of all the elements is built in.
  - `CEElement 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.
  - `CEElement Argon(“Ar”, 5.0, 298.15);    // in atm and K`
  - `CEElement LowDensityCarbon(“C”, 0.01);// in g/cm`^{3}
- May use common English name:
  - `CEElement 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.

Materials in CEPack: Mixtures

- 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

- 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

Matprop

- Lelaps distribution comes with a little program called **`matprop`**:

  -
    ```
    matprop U                    # for elements (gasses at NTP)
    ```

  -
    ```
    matprop SiO2/2.32            # for solids (and liquids) specify
                                 # density (g/cc)
    ```

  -
    ```
    matprop CO2/1/298.15         # for gasses specify pressure (atm)
                                 # and temperature (K)
    ```

  -
    ```
    matprop O2/STP               # for gasses at STP (0 C, 1 atm)
    ```

  -
    ```
    matprop Ar/NTP               # for gasses at NTP (25 C, 1 atm)
    ```

  -
    ```
    matprop H2//                 # for gasses at NTP
    ```
- For a mixture (e.g. Air at 20 C) use (-g to indicate gas):

  -
    ```
    matprop -g O2/1/293.15 20.946 N2/1/293.15 78.084
               Ar/1/293.15 0.934
    ```
- **`Just type matprop to get a list of options`**
- **`Prints out lots of material properties`**

Matprop: Example

- `matprop SiO2/2.32`

```
=== MatProp Version 1.0 (02.06.2003) ===
(CEPack version 3.18 (08.12.2003) by W.G.J. Langeveld, SLAC)

CEMaterial: === SiO2 (solid or liquid at density 2.32 g/cc) ===
   ----- Composition: -----
       100% SiO2
   ----- Properties: -----
            Z = 10.37   A = 20.03   <Z/A> = 0.4993   mol. wgt = 60.08 g/mol

            Average density =       2.32 g/cc      Density at STP =        2.32 g/cc
            Ion. potential  =      131.6 eV        Plasma energy  =          31 eV

            Radiation length   =      27.05 g/cm2 or       11.66 cm
            Interaction length =      97.78 g/cm2 or       42.15 cm

            Moliere radius     =      12.31 g/cm2 or       5.305 cm
            Critical energy    =       46.6 MeV
```

Matprop: Example (cont'd)

```
----- Sternheimer/Peierls coefficients: -----
          A =     0.07669    B =         17.2    Cbar =        3.892    Xa =      0.8451
          a =      0.1353    m =            3    X0   =          0.2    X1 =           3
 ----- Sample dEdx (MeV) in 1 g/cm^2 (0.431034 cm) material -----
 p (MeV)    :         100        1000       1e+04       1e+05
 - - - - - + - - - - - - - - - - - - - - - - - - - - - - -
 dE/dx e    :    0.008045    0.008824    0.009585     0.01035
 dE/dx mu   :     0.01165     0.00773    0.009296     0.01031
 dE/dx p    :      0.2616      0.0106    0.007842     0.00956
           +
 betagamma :         0.3           1           3          10         100        1000
 - - - - - + - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
 dE/dx e    :     0.04614     0.01014    0.006579    0.006762    0.007793    0.008602
 dE/dx mu   :     0.05174     0.01117    0.007346    0.007764    0.009324     0.01033
 dE/dx p    :     0.05178     0.01118    0.007356    0.007792    0.009514     0.01087
```

Tracking 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.

Tracking: CETracks

- In order to swim tracks through the geometry, first create a CEParticle:

  -
    ```
    vec position(0, 0, 0), direction(1, 0, 0);
    double energy = 10.0; // GeV
    int PDG_id = 11;      // electron
    CEParticle particle(position, direction, energy,
                        PDG_id);
    ```
- Now create a CETrack:

  -
    ```
    CETrack *track = new CETrack(particle, world,
                                 magnetic_field);
    ```
- Finally, swim the particle:

  -
    ```
    track->swim();
    ```
- Inherit from CETrack and implement report_hit() method to keep track of hits.

Tracking: MyCETrack

```
class MyCETrack : public CETrack {
public:
//
// Constructor
//
   MyCETrack(const CEParticle &particle, CENode *inside_node,
             CENode *magnetic_field = 0,
             double start_time = 0.0) :
        CETrack(particle, inside_node, magnetic_field,
                start_time)
           { return; };
//
// Destructor
//
   virtual ~MyCETrack() { return; };
//
// Intersection routine
//
   virtual void report_hit(CEHit &);
};
```

Tracking: report_hit

```
void MyCETrack::report_hit(const CEHit &hit)
{
   printf("Hit energy is %g GeV\n", hit.energy);
   return;
}
```

Multiple Scattering and dE/dx

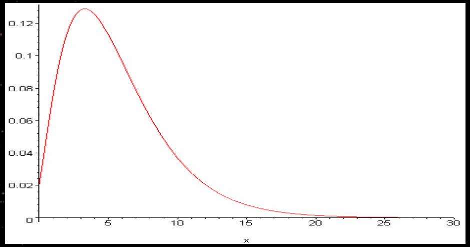
- Multiple scattering is performed using the algorithm of Lynch and Dahl.
  - Gerald R. Lynch and Orin I. Dahl, Nucl. Instr. And Meth. B58 (1991) 6.
- Material is “saved up” along the track until there is enough.
- dE/dx is calculated using the methods by Sternheimer and Peierls.
  - R.M. Sternheimer and R.F. Peierls, Phys. Rev. B3 (1971) 36810
- All constants precalculated by the material classes.

Electromagnetic Shower Parameterization

- Electromagnetic showers are parameterized using the algorithms of Grindhammer and Peters.
  - G. Grindhammer and S. Peters, arXiv:hep-ex/0001020v1 (2000)

    (Paper is a 1993 conference contribution, submitted by request to the archive in 2000).
- Calorimeters are treated as homegeneous media, with longitudinal shower profile given by a gamma distribution (t in radiation lengths):

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

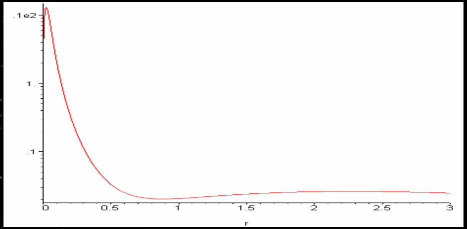


- Coefficients α and β depend on the material (Z) and energy.
- The profiles are fluctuated, and correlations between α and β are taken into account.

EM Shower Parameterization: Radial Profile

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

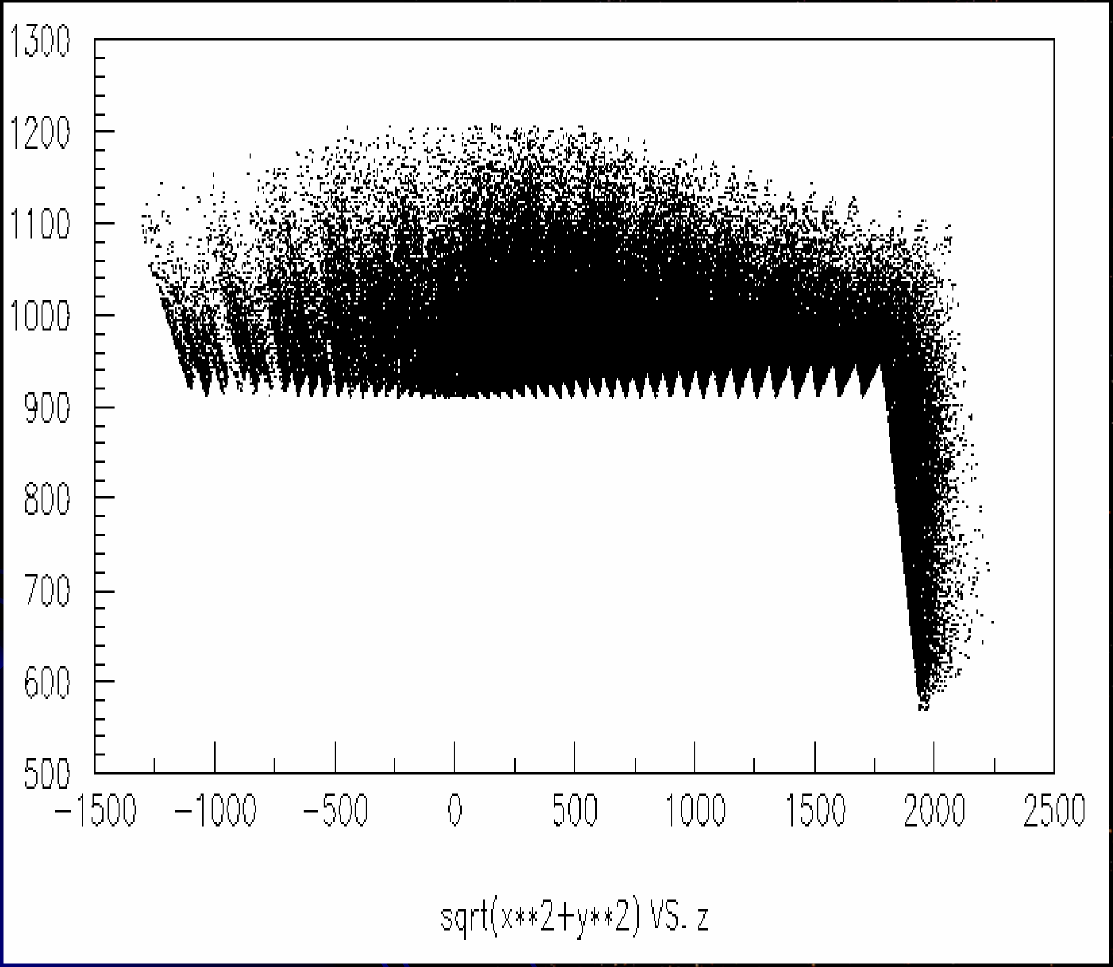
$$\frac{1}{dE(t)}\frac{dE(r,t)}{dr} = f(r) = p\frac{2rR_C}{(r^2+R_C^2)^2} + (1-p)\frac{2rR_T}{(r^2+R_T^2)^2}$$



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 φ 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: BaBar CsI



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

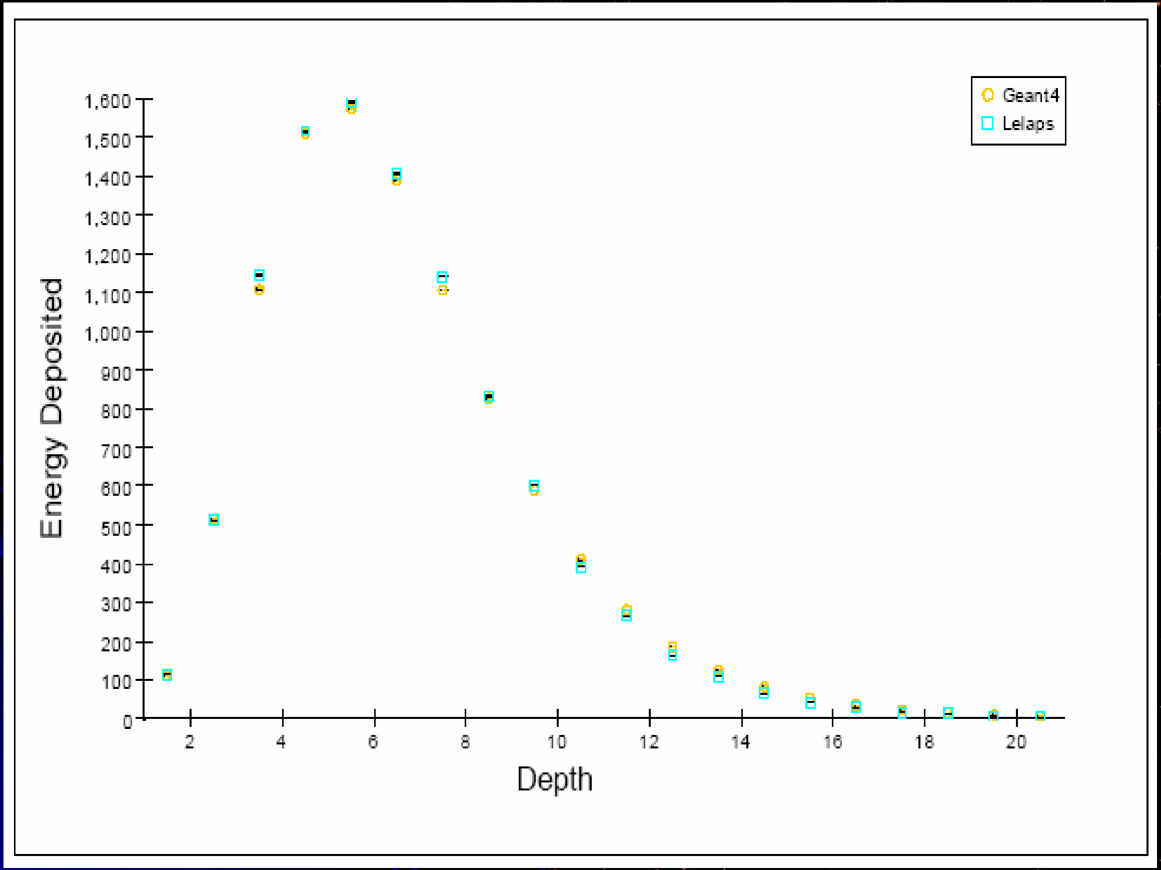
Hadronic Shower Parameterization

- 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 longitudunal profile uses the Bock parameterization.
  - R.K. Bock, T. Hansl-Kozanecka and T.P. Shah, Nucl. Instr. And Meth. 186 (1981) 533.
- 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 Compared to Geant4

- Parameterized shower simulation was compared to Geant4.
  - "Parameterized Shower Simulation in Lelaps: A Comparison with GEANT4", Daniel Birt, Amy Nicholson, Willy Langeveld, Dennis Wright, SLAC-TN-03-005, Aug 2003.
- 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.

EM Shower Parameterization and Geant4



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%.

Decays and Gamma Conversions: Example

```
CEParticleList pl;
CEParticle primary;
pl.add(primary);

for (int i = 0; i < pl.size(); i++) {
   CETrack * track = new MyCETrack(pl[i], world,
                                    magnetic_field);

   track->swim();

   int exit_status = track->exitStatus();
   if (exit_status == …) ….

   int ntracks_added = track->decay_info(pl);

   delete track;
}
```

Using CEPack inside Geant4

- To use CEPack inside Geant4, create a Geant4 parameterized model, e.g.:

```
class BgsSvtParamModel : public G4VFastSimulationModel
```

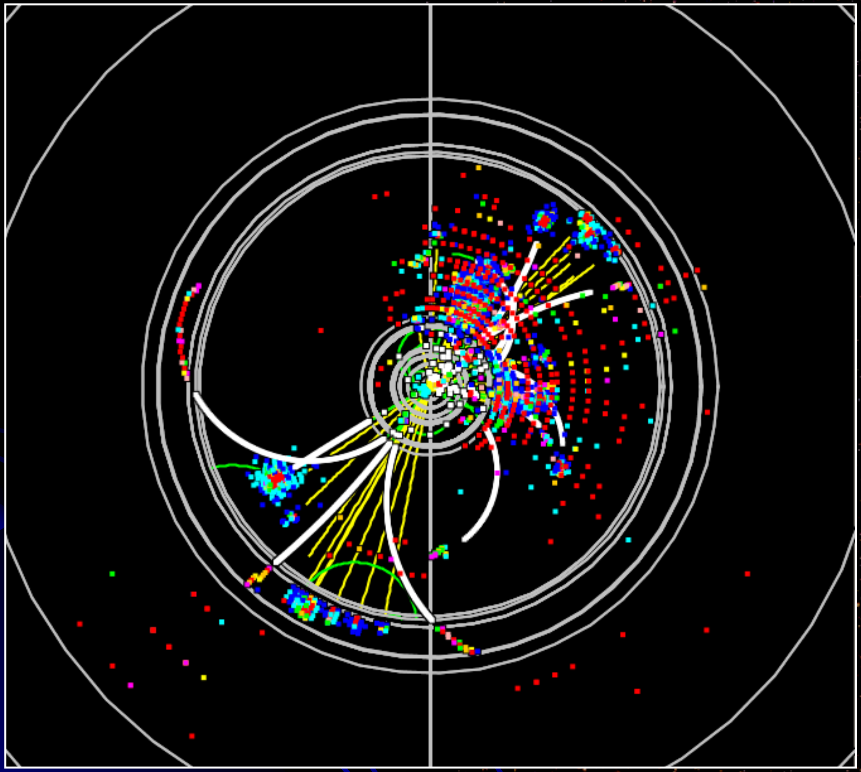
- In its `setup()`, create the CENode corresponding to the CEPack simulation with all its subnodes.
- In `Doit()`, convert from G4FastTrack track to CETrack, and call `track.swim()`
- By subclassing CETrack, all hits are reported using CETRack `report_hit()` method. Convert hits to your favorite format.
- Update `G4FastStep` (or call `KillPrimaryTrack()`):

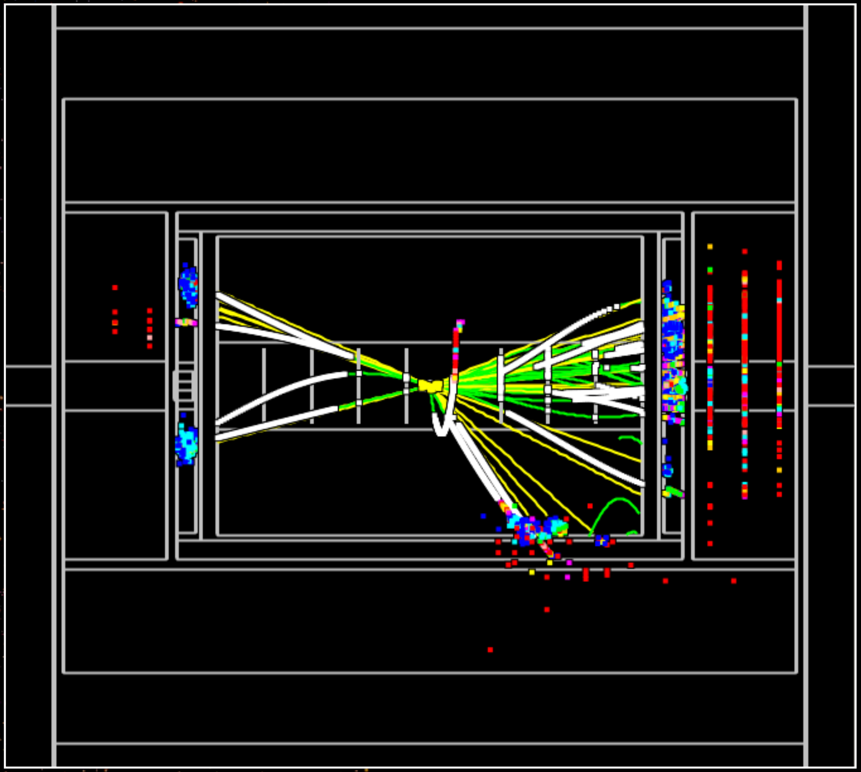
```
fastStep.SetPrimaryTrackFinalPosition(
   G4ThreeVector(track.p.getx() * m, track.p.gety() * m,
   track.p.getz() * m), false);
fastStep.SetPrimaryTrackFinalKineticEnergyAndDirection(
   (track.energy - track.mass) * GeV,
   G4ThreeVector(track.q.getx(), track.q.gety(),
   track.q.getz()), false);
```

CEPack and Lelaps

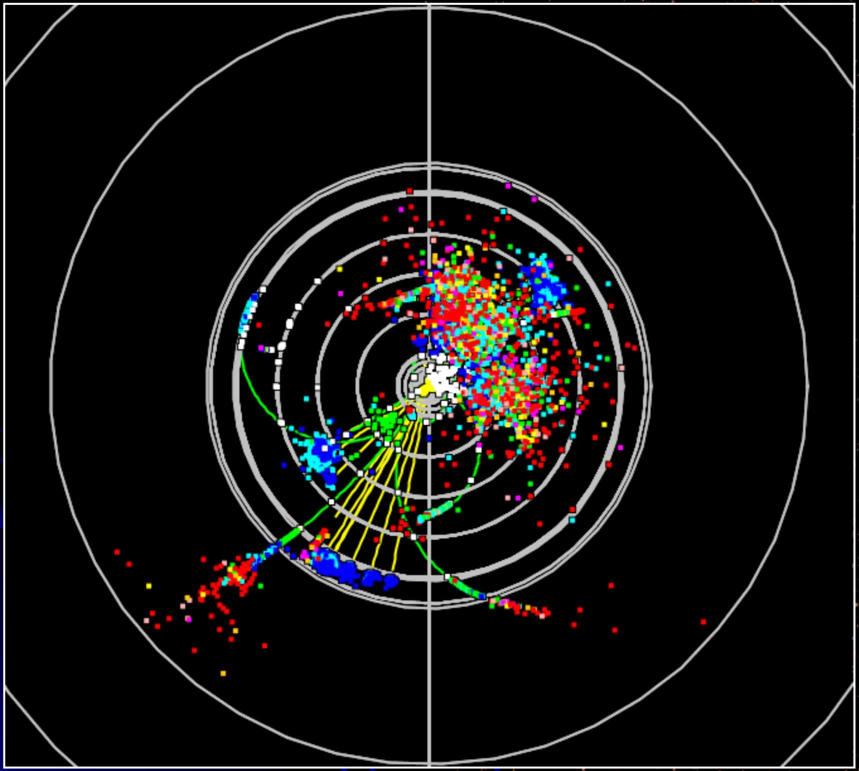
- Lelaps for LCD is a standalone program which uses CEPack.
- It sets up the CEPack geometry (both LDMar01 and SDJan03 are supported).
- It uses lStdHep ("StdHep light" – included in the distribution): a class library to to read generator event files in StdHep format. StdHep particles are converted to 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.

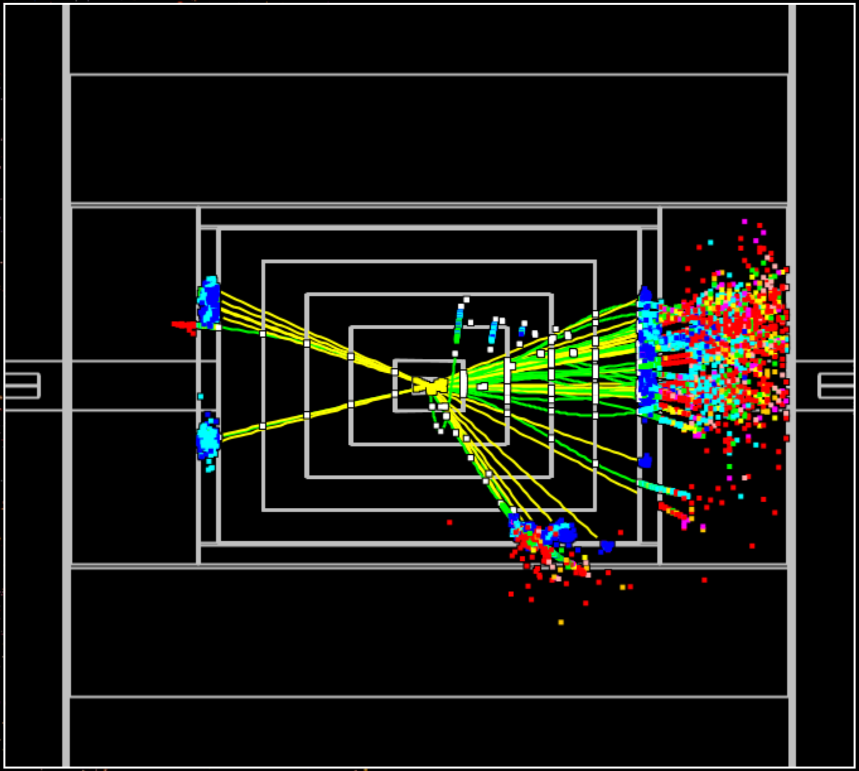
Lelaps (LDMar01)





Lelaps (SDJan03)





Lelaps: Usage

- Most common usage:
  - `lelaps -o foo.sio -E bar1.stdhep bar2.stdhep` …
  - Defaults to SD. Use –L switch for LD.
- Also has built-in simple particle “gun”:
  - `lelaps -o foo.sio  -i 11 -m 10 -n 4 -N 1000`

    generates 1000 events with 4 electrons/event with 10 GeV each random in 4π
- Options for generation in x,y plane (2π) or along x, y, z directions
- Options for turning off energy loss, multiple scattering and/or showering in calorimeters
- Options for turning on decays and/or conversions and tracking the secondaries (by default these are currently turned off)

Lelaps: Usage

- `lelaps -h`

```
[-c]                      continue
[-C]                      C[c]onversions, D[d]ecays
      e.g. -Cc            - do conversions, no decays
           -CC            - do conversions and track e+e-
           -CCD           - do both, track products
[-d]                      debug
[-D <track>]              track to debug
[-e]                      no energy loss
[-E <filename ...>]       stdhep event filename(s)
                          Must use a space after -E!
[-h]                      This message
[-i <particle_id>]        particle id
[-L]                      Do LDMar01 (default SDJan03)
[-m <momentum>>]          momentum in GeV
[-n <ntracks>]            # of tracks/event
[-N <nevents>]            # of events
[-o <filename>            SIO output filename
[-p]                      plot all
[-P <track #>]            which track to plot
```

Lelaps: Usage

[-r <# calls>]	# of pre-calls to drand48()
[-s]	show seed
[-S <type>]	no showering of type Any, Hadronic, E.M.
[-t]	trace
[-x]	no multiple scattering
[-z <seed>]	seed for ran2()
[-2]	generate in 2pi
[-1 <{x|y|z}>]	generate along one direction

Performance

- 100 ee→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):

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

* panpy-ZZ-500-001001-gen-1.stdhep (5973 tracks)

(Latest version is actually faster)

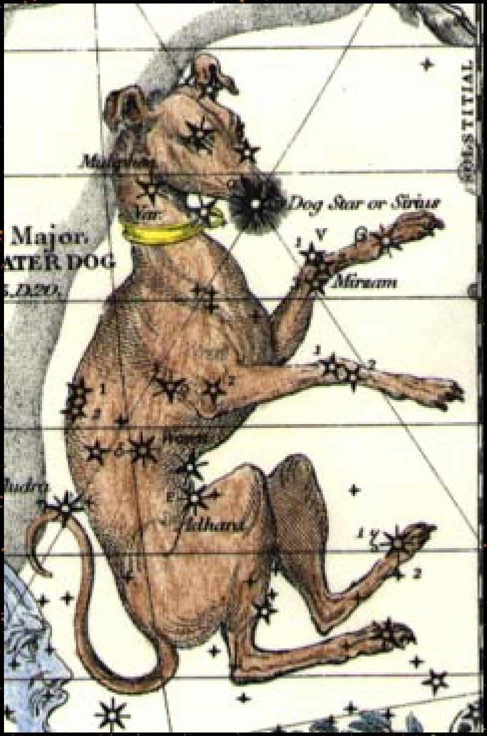
Performance: Analysis

- 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 a almost a factor 2.
  - No optimization attempted so far.
- SIO output file (10 MB, compressed) costs a lot, 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