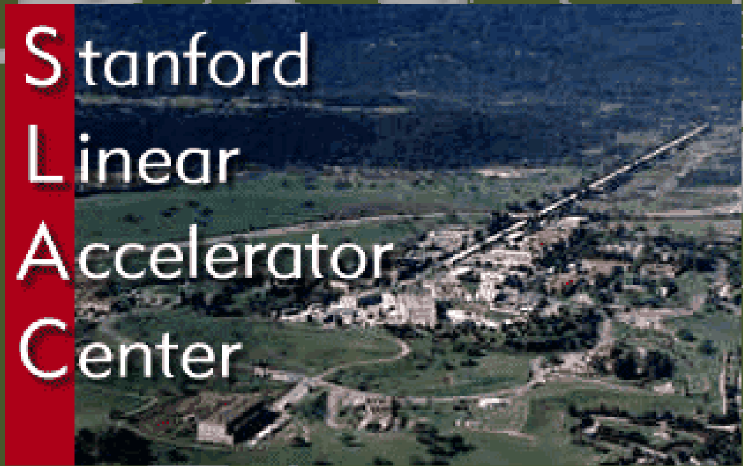


Physics III

Makoto Asai (SLAC)

Geant4 Tutorial Course

Geant 4

Outline

- Production Thresholds and Cuts per Region
- Optical processes
- The Decay Process
  - pre-assigned decay products
  - exotic particle

Production Thresholds and Cuts per Region

Threshold for Secondary Production (1)

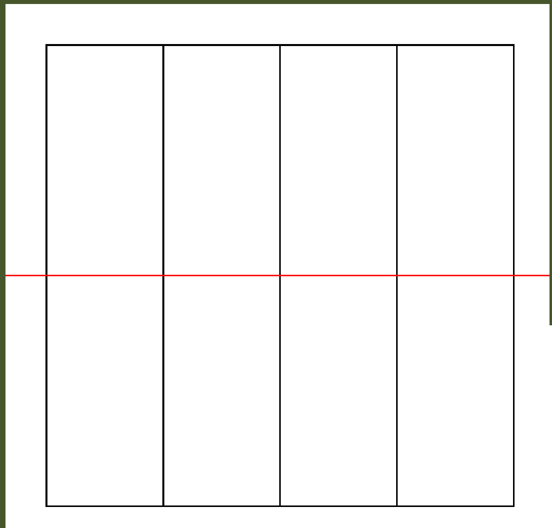
- Every simulation developer must answer the question: how low can you go?
  - at what energy do I stop tracking particles?
- This is a balancing act
  - need to go low enough to get the physics you're interested in
  - can't go too low because some processes have infrared divergence causing CPU time to skyrocket
- The traditional Monte Carlo solution is to impose an absolute cutoff in energy
  - particles are stopped when this energy is reached
  - remaining energy is dumped at that point
- But, such a cut may cause imprecise stopping location and deposition of energy
  - There is also a particle dependence
    - range of 10 keV γ in Si is a few cm
    - range of 10 keV e- in Si is a few microns
  - And a material dependence
    - suppose you have a detector made of alternating sheets of Pb and plastic scintillator
    - if the cut-off is OK for Pb, it will likely be wrong for the scintillator which does the actual energy deposition measurement

Threshold for Secondary Production (2)

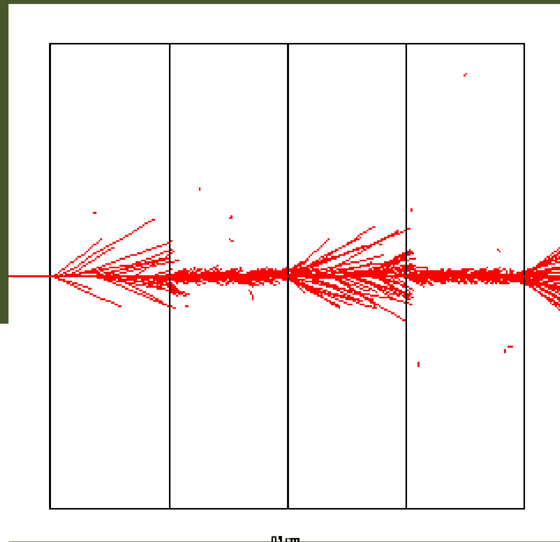
- Geant4 solution: impose a production threshold
  - this threshold is a distance, not an energy
  - the primary particle loses energy by producing secondary electrons or gammas
  - if primary no longer has enough energy to produce secondaries which travel at least 1mm, two things happen:
    - discrete energy loss ceases (no more secondaries produced)
    - the primary is tracked down to zero energy using continuous energy loss
- Stopping location is therefore correct
- Only one value of production threshold distance is needed for all materials because it corresponds to different energies depending on material.
- Geant4 recommends the default value of 1mm
  - user needs to decide the best value
  - this will depend on the size of sensitive elements within the simulated detector, and on available CPU
- This value is set in the SetCuts() method of your physics list.
- Instead of "secondary production threshold distance" it is more convenient to simply say "cuts"
  - but please remember that this does not mean that any particle is actually stopped before it runs out of energy

Production Threshold vs. Energy Cut

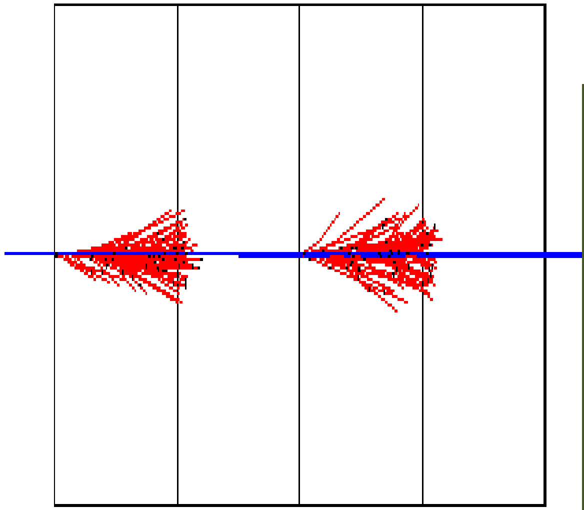
500 MeV p in LAr-Pb sampling calorimeter



Cut = 2 MeV



Cut = 450 keV



Production range =1.5 mm

Cuts per Region

- In a complex detector there may be many different types of sub-detectors involving
  - finely segmented volumes, very sensitive materials, large undivided volumes and/or inert materials
- The same value of the secondary production threshold may not be appropriate for all of these
  - user must define regions of similar sensitivity and granularity and assign a different set of production thresholds (cuts) for each
- Warning: this feature is for users who are
  - simulating the most complex detectors, and
  - experienced at simulating EM showers in matter
- A default region is created automatically for the world volume
  - it has the cuts which you set in SetCuts() in your physics list
  - these will be used everywhere except for user-defined regions
- To define a special region with different cuts, user must
  - create a G4ProductionCuts object
  - initialize it with the new cuts
  - assign it to a region which has already been created

Cuts per Region

```
void MyPhysicsList::SetCuts()
{
  defaultCutValue = 1.0 * mm;
  SetCutValue(defaultCutValue, "gamma");
  SetCutValue(defaultCutValue, "e-");
  SetCutValue(defaultCutValue, "e+");

  // Get the region
  G4Region* aRegion =
    G4RegionStore::GetInstance()->GetRegion("myRegion");
  // Define cuts object for the new region and set values
  G4ProductionCuts* cuts = new G4ProductionCuts;
  cuts->SetProductionCut(0.01*mm); // same cut for gamma, e+, e-
  // Assign cuts to region
  aRegion->SetProductionCuts(cuts);
}
```

Default cuts to be used at the world default region and regions which do not have specified cuts.

The region must be instantiated and set to logical volume(s) beforehand.

Optical processes

Optical Photons

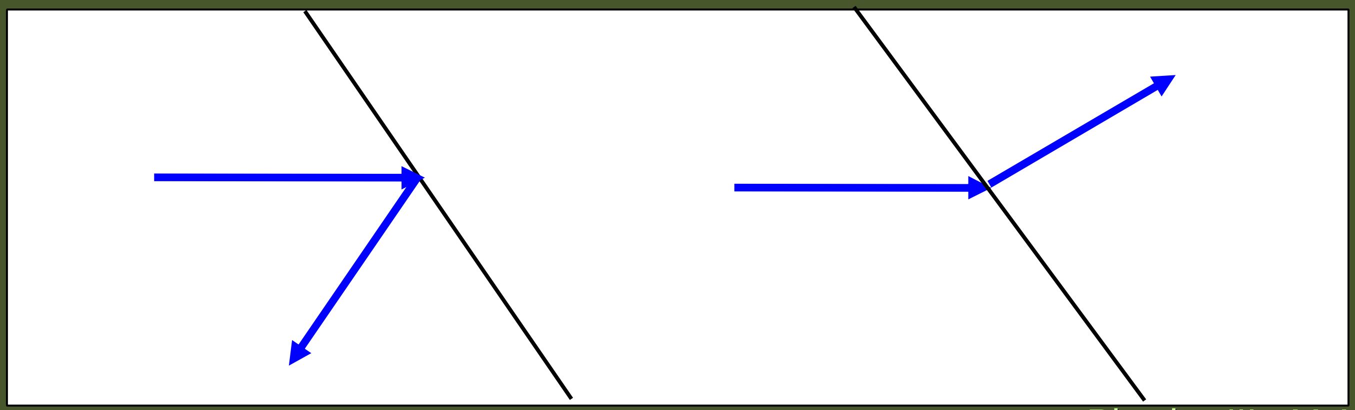
- Technically, should belong to electromagnetic category, but:
  - optical photon wavelength is >> atomic spacing
  - treated as waves
    - no smooth transition between gamma and optical particle classes
- Optical photons are produced by the following Geant4 processes:
  - G4Cerenkov, G4Scintillation and G4TransitionRadiation
    - Warning: these processes generate optical photons without energy conservation
- Optical photons undergo:
  - Rayleigh scattering
  - bulk absorption
  - refraction and reflection at medium boundaries
  - wavelength shifting
- Geant4 keeps track of polarization
  - but not overall phase -> no interference
- Optical properties can be specified in G4Material
  - reflectivity, transmission efficiency, dielectric constants, surface properties
- Photon spectrum properties also defined in G4Material
  - scintillation yield, time structure (fast, slow components)

Absorption and Rayleigh Scattering

- G4OpAbsorption
  - uses photon attenuation length from material properties to get mean free path
  - photon is simply killed after a selected path length
- G4OpRayleigh
  - elastic scattering including polarization of initial and final photons
  - builds it own private physics table (for mean free path) using G4MaterialTable
  - may only be used for optical photons

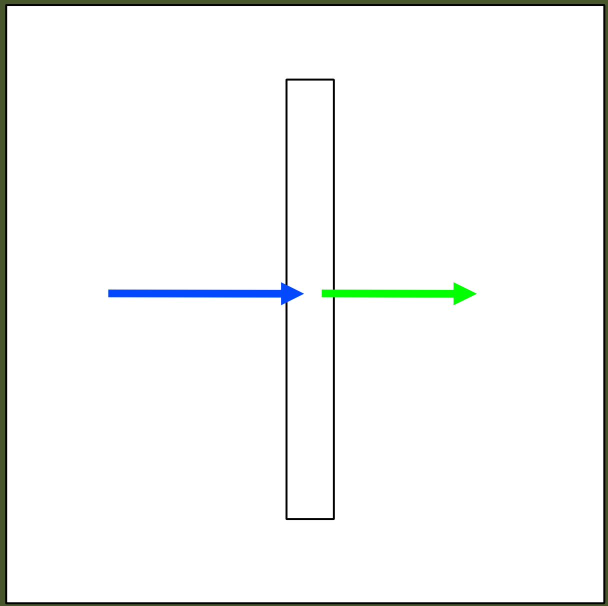
Boundary Interactions

- Handled by G4OpBoundaryProcess
  - refraction
  - reflection
- Geant4 demands particle-like behaviour for tracking:
  - thus, no "splitting"
  - event with both refraction and reflection must be simulated by at least two events
- User must supply surface properties using G4OpticalSurfaceModel
- Boundary properties
  - dielectric-dielectric
  - dielectric-metal
  - dielectric-black material
- Surface properties:
  - polished
  - ground
  - front- or back-painted, ...

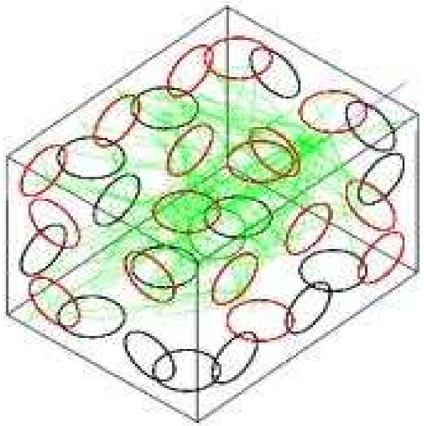


Wavelength Shifting

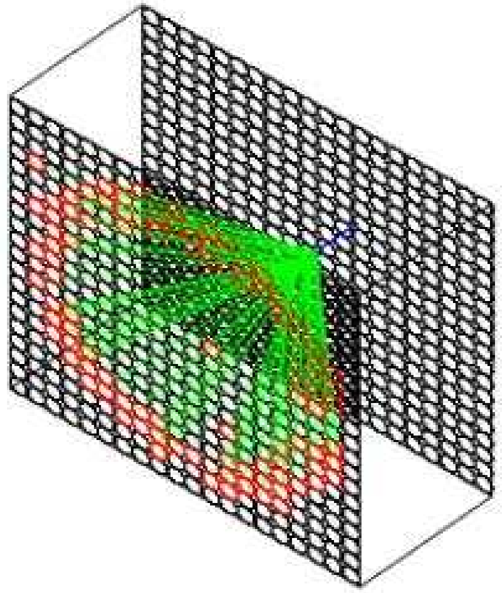
- Handled by G4OpWLS
  - initial photon is killed, one with new wavelength is created
  - builds it own physics table for mean free path
- User must supply:
  - absorption length as function of photon energy
  - emission spectra parameters as function of energy
  - time delay between absorption and re-emission



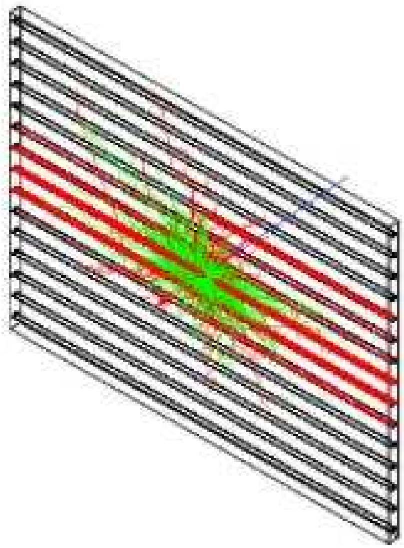
/examples/extended/optical/LXe



Scintillation



Cerenkov



WaveLengthShifting

Decay process

The Decay Process

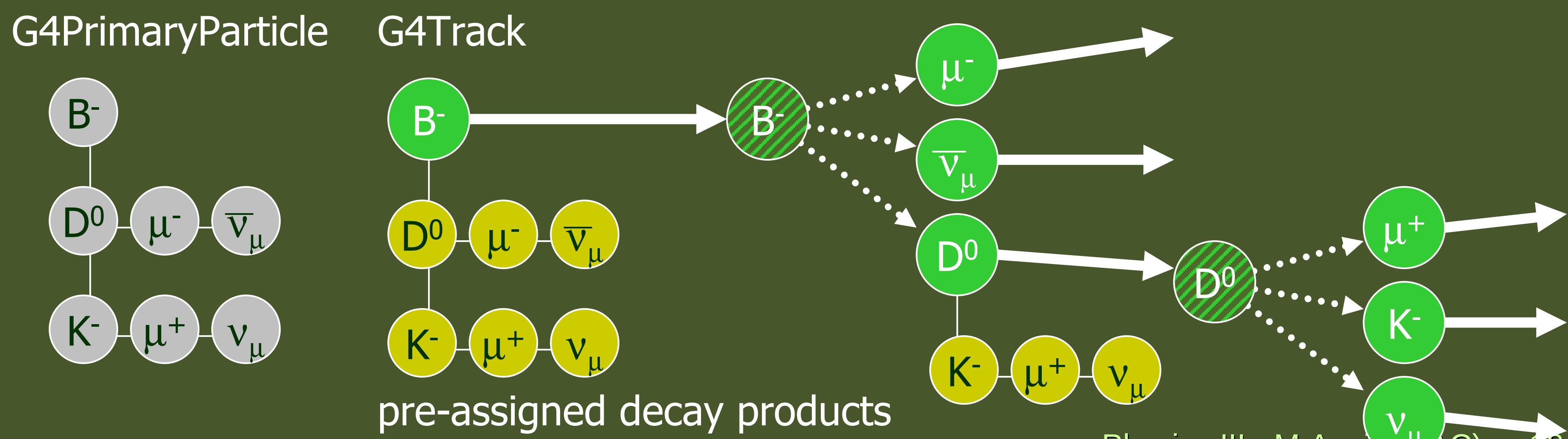
- Derived from G4VRestDiscreteProcess, i.e. decay can happen in-flight or at rest
- Same decay process for all eligible unstable, long-lived particles
  - decay process retrieves BR and decay modes from decay table stored in each particle type
- Different from other physical processes:
  - mean free path for most processes: $\lambda = N\rho\sigma / A$
  - for decay in-flight: $\lambda = \gamma\beta c\tau$
- Available decay modes
  - Phase space:
    - 2-body e.g. $\pi^0 \to \gamma\gamma$, $\Lambda \to p\,\pi^-$
    - 3-body e.g. $K^0_L \to \pi^0\,\pi^+\,\pi^-$
    - many body
  - Dalitz: $P^0 \to \gamma\, l^+\, l^-$
  - Muon decay
    - V – A, no radiative corrections, mono-energetic neutrinos
  - Leptonic tau decay
    - like muon decay
  - Semi-leptonic K decay: $K \to \pi\, l\, \nu$

Specialized Decay Processes

- G4DecayWithSpin
  - produces Michel positron spectrum with 1^{st} order radiative corrections
  - initial muon spin is required
  - propagates spin in magnetic field (precession) over remainder of muon lifetime
- G4UnknownDecay
  - only for “unknown” particles (Higgs, SUSY, etc.)
  - discrete process – only in-flight decays allowed
  - pre-assigned decay channels must be supplied by user or generator

Pre-assigned decay products

- Geant4 provides decay modes for long-lived particles, but decay modes for short-lived (e.g. heavy flavour) particles are not provided by Geant4
  - decay process can invoke an external decay handler (G4VExtDecayer)
  - Or, user must "pre-assign" proper lifetime and decay products to the parent G4PrimaryParticle.
- A parent particle in the form of G4Track object travels in the detector, bringing "pre-assigned" decay daughters as objects of G4DynamicParticle.
  - When the parent track comes to the decay point, pre-assigned daughters become to secondary tracks, instead of randomly selecting a decay channel defined to the particle type.
  - Decay time of the parent can be pre-assigned as well.



Importing "exotic" particles

- "Exotic" particle means a type of particle that Geant4 physics processes do not know how to deal with and would never generate as a secondary.
  - It is thus not provided as a class in particle category of Geant4 distribution.
  - E.g. Higgs, W/Z boson, SUSY particle, r-hadron, monopole, black hole, etc.
- "Exotic" particle also includes a type of particle that should not be seen outside of a hadron.
  - It is used inside Geant4 processes, but it should not be treated as a track.
  - E.g. quark, gluon.
- Such exotic particle can be imported as a G4PrimaryParticle object.
  - It should have pre-assigned decay products (if it decays), since Geant4 does not know how it decays.
- There are two kinds of exotic particles from the view point of Geant4. We have to deal them separately.
  - Particles that immediately decay without traveling finite distance.
  - Particles that travel a distance meaningful to Geant4 tracking.

Exotic particle that decays immediately

- As a default, Geant4 ignores such exotic particle and takes its pre-assigned decay products as primaries.
  - Anyway, such a particle should not travel through your geometry.
- In case you want to see it as a primary track (so that it has a unique track ID and it is recorded as a trajectory), use G4UnknownParticle.
  - G4UnknownParticle must be defined in your physics list with G4UnknownDecay process attached.
  - G4UnknownDecay process immediately enforces such particle to decay in its first step naively using pre-assigned decay products.
- Once G4UnknownParticle is defined in your physics list, G4PrimaryTransformer converts whatever the exotic particle to a G4Track object of Unknown.
  - If you want to limit this conversion to be applied only to some kinds of exotic particle types, create your own PrimaryTransformer to override a method.

    ```
    G4ParticleDefinition* GetDefinition(G4PrimaryParticle*)
    ```
  - If non-null pointer is returned, this primary particle is converted into G4Track (or G4DynamicParticle for pre-assigned decay product).
  - If null is returned, its pre-assigned decay daughters will be treated as primaries.
  - Your PrimaryTransformer class must be assigned to G4RunManagerKernel.

Exotic particle that travels

- As a default, Geant4 cannot deal with such a particle. Geant4 does not know what to do. You have to do the followings to import such exotic particle.
- Implement ParticleDefinition concrete class to represent (a family of) exotic particle(s).
  - Typically one concrete class for each category and each charge state.
    - MyRHadronZero, MyRHadronPlus, etc.
    - BMesonStarPlus, BMesonStarMinus, etc.
  - PDG code in ParticleDefinition object for such exotic particle must be 0, and the mass could be arbitrary value. G4DynamicParticle::GetPDGcode() and G4DynamicParticle::GetMass() will return correct values for each individual track.
- Assign reasonable processes to it.
  - G4Transportation, G4Decay (don't use G4UnknownDecay), EM processes, hadronic processes(?)
- create your own PrimaryTransformer to override a method.

```
G4ParticleDefinition* GetDefinition(G4PrimaryParticle*)
```

  - By this method, return proper ParticleDefinition object.