Geant4e Track Extrapolation in the (Super)Belle Experiment

Leo Piilonen, Virginia Tech
Nobu Katayama, KEK
on behalf of the Belle Collaboration

CHEP 2009
geant4e, a part of geant4, is used for covariance propagation of charged tracks during event reconstruction

GEANT4E:
Error propagation for track reconstruction inside the GEANT4 framework

Pedro Arce (CIEMAT)

CHEP 2006, Mumbai, 13-17th February 2006
In (Super)Belle, use `geant4e` for track propagation and muon identification during event reconstruction; only forward propagation is done.
**geant4 model of the (Super)Belle detector:**

- complete subdetector geometry
- non-uniform solenoidal magnetic field (~1.5 T)
- common geometry for geant4 and geant4e
Charged track extrapolation using `geant4e`:

- For each of 5 hypotheses $e, \mu, \pi, K, p \ldots$
- Swim each track from outer edge of drift chamber to calorimeter face [or muon detector face for $\pi$]
- Store position, momentum and covariance matrix at entrance/exit of selected volumes
Muon identification using geant4e:

✔ Swim track from outer edge of drift chamber through muon detector (with Kalman fitting to matching hits, if any)
We have two usage modes of `geant4e`:

- for real events: standalone
- for simulated events: in combination with `geant4`, since we do simulation and reconstruction in one pass

But `geant4e`, as distributed, cannot coexist with `geant4`:
- distinct particle lists
- distinct physics processes
- conflicting usage of common detector geometry
- distinct states when calling `RunManager`
- distinct user actions (`SteppingAction` etc)

We have resolved these issues ...
PhysicsList is a concrete implementation of G4VUserPhysicsList, and must define:

- ConstructParticle()
- ConstructProcess()
- SetCuts()

geant4 and geant4e require different PhysicsLists.

Lots of overhead to change PhysicsList when switching between geant4 and geant4e, so avoid this!
Define a combined PhysicsList

- ConstructParticle() defines gamma e+ e− mu+ mu− pi+ pi− pi0 kaon+ kaon− kaon0 kaon0L kaon0S proton anti_proton neutron anti_neutron geantino chargedgeantino opticalphoton etc., as well as g4e_e+ g4e_e− g4e_gamma g4e_mu+ g4e_mu− g4e_proton g4e_pi+ g4e_pi− g4e_kaon+ g4e_kaon− with PIDcode = 1000000000 + stdPIDcode
Define a combined PhysicsList (cont’d)

• For standard particles, ConstructProcess() does AddTransportation(), ConstructDecayProcess(), ConstructEMProcess(), ConstructHadronicProcess(), and ConstructOpticalPhotonProcess(), as appropriate

• For “g4e” particles, ConstructProcess() does only AddTransportation() and ConstructEMProcess(); the latter defines ionization energy loss as the sole physics process for charged particles.
Particles and Physics Processes, cont’d:

- Define a combined PhysicsList (cont’d)
  - For standard particles, SetCuts() does SetCutsWithDefault() using default = 1.0*mm
  - For g4e particles, SetCuts() does SetCutsWithDefault() using default = 1.0E9*cm
SteppingManager in geant4 calls user code to process steps through “sensitive” detector volumes and record hits therein.

This behaviour is undesirable in the geant4e context.

For “g4e” particles, ConstructEMProcess() adds a new NoHits() process:

```cpp
G4ParticleChange particleChange;
G4VParticleChange* NoHits::PostStepDoIt( const G4Track& track, const G4Step& step )
{
    particleChange.Initialize( track );
    particleChange.ProposeSteppingControl( AvoidHitInvocation );
    return &particleChange;
}
```
geant4e “Target” Geometry:

- Beyond the standard detector geometry, geant4e prescribes a “target” surface: geant4e terminates the track propagation when the track crosses this surface.
- The available surfaces are not adequate for our needs.

- Duplicate then modify G4ErrorCylSurfaceTarget so that it includes the cylinder endcaps.
Distinct Run States and User Actions:

During job initialization, detect presence of `geant4` by non-empty `G4PhysicalVolumeStore`. If co-existing, do `G4StateManager::GetStateManager() -> SetNewState(G4State_Idle)` after `InitGeant4e()`, then save `UserTrackingAction` and `UserSteppingAction`.

During processing of one event:

```cpp
if ( geant4e is running with geant4 ) {
    hide UserTrackingAction and UserSteppingAction;
}
extrapolate all tracks in the event using “g4e” particles;
if ( geant4e is running with geant4 ) {
    restore UserTrackingAction and UserSteppingAction;
}
```
Distinct Run States, cont’d:

- Duplicate and modify `G4ErrorPropagationNavigator` so that it exhibits the `geant4e` behaviour during track propagation
  ```
g4edata != 0 &&
  g4edata->GetState() == G4ErrorState_Propagating
or the `geant4` behaviour otherwise.
  ```

`G4**Navigator` has two methods — `ComputeStep` and `ComputeSafety` — to determine distance to volume boundary. While `geant4e` is active, the distance to the “target” surface is included in these calculations.
Conclusion:

In the (Super)Belle software library, we have succeeded in implementing `geant4e` for track propagation and muon identification during event reconstruction, either standalone or in conjunction with `geant4` event simulation:

- merged particle list including “g4e” particles
- distinct physics processes for “g4e” particles
- no hit invocation in sensitive volumes for `geant4e`
- distinct states and user actions during event processing