Primary Particle

Shigeyuki Tajima

Duke/NCCU and TUNL

Slides courtesy of SLAC GEANT4 Team

Special thanks to Makoto Asai
Contents

- G4VUserPrimaryGeneratorAction
- Primary vertex and primary particle
- Built-in primary particle generators
  - Particle gun
  - Interfaces to HEPEVT and HEPMC
  - General particle source
- Exotic primary particle
Primary particle generation
User classes

- Initialization classes
  - Use G4RunManager::SetUserInitialization() to define.
  - Invoked at the initialization
    - G4VUserDetectorConstruction
    - G4VUserPhysicsList

- Action classes
  - Use G4RunManager::SetUserAction() to define.
  - Invoked during an event loop
    - G4VUserPrimaryGeneratorAction
    - G4UserRunAction
    - G4UserEventAction
    - G4UserStackingAction
    - G4UserTrackingAction
    - G4UserSteppingAction

- main()
  - Geant4 does not provide main().

Note: classes written in yellow are mandatory.
G4VUserPrimaryGeneratorAction

- This class is one of mandatory user classes to control the generation of primaries.
  - This class itself should NOT generate primaries but invoke `GeneratePrimaryVertex()` method of primary generator(s) to make primaries.
- Constructor
  - Instantiate primary generator(s)
  - Set default values to it(them)
- `GeneratePrimaries()` method
  - Randomize particle-by-particle value(s)
  - Set these values to primary generator(s)
    - Never use hard-coded UI commands
  - Invoke `GeneratePrimaryVertex()` method of primary generator(s)
Example: Novice N01

Source Code from ExN01PrimaryGeneratorAction.cc.
The class derived from G4VUserPrimaryGeneratorAction

```cpp
56 void ExN01PrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
57 {
58     G4int i = anEvent->GetEventID() % 3;
59     G4ThreeVector v(1.0, 0.0, 0.0);
60     switch(i)
61     {
62         case 0:
63             break;
64         case 1:
65             v.setY(0.1);
66             break;
67         case 2:
68             v.setZ(0.1);
69             break;
70     }
71     particleGun->SetParticleMomentumDirection(v);
72     particleGun->GeneratePrimaryVertex(anEvent);
73 }
```
Primary vertex
and primary particle
Primary Vertices and Primary Particles

• Primary vertices and primary particles are stored in an event (G4Event) before it is processed.

• G4PrimaryVertex class (particle starting point in space and time, etc)

• G4PrimaryParticle class (initial momentum, particle polarization, etc)

• These classes do not depend on G4ParticleDefinition nor G4Track.

• Primary particles may not necessarily be particles which can be tracked.
Particle in Geant4

- Particle in general has the following three properties:
  - **Particle position, geometrical info**  
    => **G4Track** class (representing a particle to be tracked)
  - **Dynamic properties (momentum, energy, spin, etc)**  
    => **G4DynamicParticle** class (representing an individual particle)
  - **Static properties (rest mass, charge, life time, etc)**  
    => **G4ParticleDefinition** class

- All **G4DynamicParticle** objects of the same kind of particle share the same **G4ParticleDefinition**
Pre-assigned decay products

- Physics generator can assign a decay channel for each individual particle separately, while in Geant4 you cannot specify a decay channel for each particle.
  - Decay chain can be “pre-assigned”.
- 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.

G4PrimaryParticle

- \( B^- \)
- \( D^0 \)
- \( K^- \)

G4Track

- \( B^- \)
- \( D^0 \)
- \( K^- \)
- \( D^0 \)
- \( K^- \)

Pre-assigned decay products
Built-in primary particle generators

- Geant4 provides some concrete implementations of `G4VPrimaryGenerator`.
  - `G4ParticleGun`
  - `G4HEPEvtInterface`, `G4HEPMMCInterface`
  - `G4GeneralParticleSource`
G4ParticleGun

- Concrete implementations of G4VPrimaryGenerator
  - A good example for experiment-specific primary generator implementation
  - It shoots one primary particle of a certain energy from a certain point at a
certain time to a certain direction.
    - Various set methods are available
    - Intercoms commands are also available for setting initial values
- One of most frequently asked questions is:
  I want “particle shotgun”, “particle machinegun”, etc.
- Instead of implementing such a fancy weapon, in your implementation of
UserPrimaryGeneratorAction, you can
  - Shoot random numbers in arbitrary distribution
  - Use set methods of G4ParticleGun
  - Use G4ParticleGun as many times as you want
  - Use any other primary generators as many times as you want to make
overlapping events
void T01PrimaryGeneratorAction::
 GeneratePrimaries(G4Event* anEvent)
{
  G4ParticleDefinition* particle;
  G4int i = (int)(5.*G4UniformRand());
  switch(i)
  {
    case 0: particle = positron; break; ... }
  particleGun->SetParticleDefinition(particle);
  G4double pp =
    momentum+(G4UniformRand()-0.5)*sigmaMomentum;
  G4double mass = particle->GetPDGMass();
  G4double Ekin = sqrt(pp*pp+mass*mass)-mass;
  particleGun->SetParticleEnergy(Ekin);
  G4double angle = (G4UniformRand()-0.5)*sigmaAngle;
  particleGun->SetParticleMomentumDirection
    (G4ThreeVector(sin(angle),0.,cos(angle)));
  particleGun->GeneratePrimaryVertex(anEvent);
}

You can repeat this for generating more than one primary particles.
Interfaces to HEPEvt and HepMC

- Concrete implementations of G4VPrimaryGenerator
  - A good example for experiment-specific primary generator implementation
- G4HEPEvtInterface
  - Suitable to /HEPEVT/ common block, which many of (FORTRAN) HEP physics generators are compliant to.
  - ASCII file input
- G4HepMCInterface
  - An interface to HepMC class, which a few new (C++) HEP physics generators are compliant to.
  - ASCII file input or direct linking to a generator through HepMC.
G4GeneralParticleSource

- A concrete implementation of G4VPrimaryGenerator
  - Suitable especially to space applications

```cpp
MyPrimaryGeneratorAction::
    MyPrimaryGeneratorAction()
    { generator = new G4GeneralParticleSource; } 
void MyPrimaryGeneratorAction::
    Generate Primaries (G4Event* anEvent)
    { generator->Generate Primary Vertex (anEvent); } 
```

- Detailed description
  - http://reat.space.qinetiq.com/gps/