

Primary Particle Generation

<http://cern.ch/geant4>

Contents

- Primary vertex and primary particle
- Primary generator
 - What is
 - What is available
- G4VUserPrimaryGeneratorAction
 - How to use it

Primary Vertices and Primary Particles

- Primary vertices and primary particles must be stored in an event (G4Event) before it is processed. Need instances of:
 - G4PrimaryVertex
 - particle starting point in space, time
 - can add user information as well
 - G4PrimaryParticle
 - initial momentum, polarization of particle to be propagated
 - also PDG code
 - linked list of daughters for decay chains
- These classes do not depend on G4ParticleDefinition or G4Track
- Primary particles may **not** necessarily be particles which can be tracked by Geant

Primary Generator

- All primary generators must be derived from `G4VPrimaryGenerator` and implement its single pure virtual method `GeneratePrimaryVertex()`
 - this is where the primary vertex and primary particle are added to the event
- Geant4 provides some implementations of `G4VPrimaryGenerator`:
 - `G4HEPEvtInterface`
 - `G4HEPMCInterface`
 - `G4GeneralParticleSource`
 - `G4ParticleGun`

Interfaces to HEP Evt and HepMC

- Concrete implementations of G4VPrimaryGenerator
 - good examples of experiment-specific primary generator implementation
- G4HEPEvtInterface
 - interface to /HEPEVT/ common block which many FORTRAN HEP physics generators (e.g. Pythia-Jetset) are compliant with
 - uses ASCII file input
 - see novice example N04
- G4HepMCInterface
 - interface to HepMC class, which a few new (C++) HEP physics generators are compliant with
 - uses ASCII file input or direct linking to generator through HepMC

General Particle Source

- G4GeneralParticleSource is another concrete implementation of G4VPrimaryGenerator
- Primary vertex is randomly chosen on surface of a given volume
 - useful for radioactive sources
 - pre-defined energy spectra (power law, etc.) available
- Capable of event biasing (variance reduction)
 - can enhance certain particle types
 - bias the vertex point distribution
 - bias energy and/or direction
- Especially suitable for space and medical applications
 - see advanced examples: x-ray telescope, underground physics

Particle Gun

- G4ParticleGun is an implementation of G4VPrimaryGenerator which is used to simulate a beam of particles
 - see novice example N02
- It shoots a primary particle of a certain energy and direction from a given point at a given time
 - methods available to customize your particle gun
 - set particle type
 - set energy, momentum
 - set polarization
 - set charge
 - number of particles shot at one time

G4VUserPrimaryGeneratorAction (1)

- This is one of the mandatory user action classes
 - used to control the generation of primaries
 - this class itself should not generate primaries – instead use `GeneratePrimaryVertex()` method of primary generator
- Geant4 developers are frequently asked to implement “particle shot guns” or “particle machine guns”
 - such fancy weapons can easily be implemented by the user with
 - set methods in `G4ParticleGun`
 - repeated use of `G4ParticleGun` in a single event
 - random sampling of particle type and direction
 - additional primary generators (described earlier)

G4VUserPrimaryGeneratorAction (2)

- Constructor
 - instantiate primary generator(s)
 - set default items (particle, energy, etc.)
- GeneratePrimaries() method:
 - randomize particle-by-particle values
 - assign them to primary generator(s)
 - invoke GeneratePrimaryVertex() method of primary generator(s)

G4VUserPrimaryGeneratorAction (3)

```
void MyPrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
{
    G4ParticleDefinition* particle;
    G4int i = (int)(5.*G4UniformRand());
    switch(i) {
        case 0: particle = positron; break;
        .....
    }
    gun->SetParticleDefinition(particle);
}
```

G4VUserPrimaryGeneratorAction (4)

```
G4double pp = momentum + (G4UniformRand() - 0.5)*sigmaMomentum;  
G4double mass = particle->GetPDGMass();  
G4double kineticE = sqrt(pp*pp + mass*mass) - mass;  
gun->SetParticleEnergy(kineticE);  
G4double angle = (G4UniformRand() - 0.5) * sigmaAngle;  
gun->SetParticleMomentumDirection  
    (G4ThreeVector(sin(angle), 0., cos(angle)));  
  
gun->GeneratePrimaryVertex(anEvent);  
}
```

- Repeat the above as many times as desired per event