

Generation of a Primary Event

Davide Chiappara
University of Padova (UNIPD)
Istituto Nazionale di Fisica Nucleare (INFN)



User Classes

Initialisation classes

Invoked at the initialization

- G4VUserDetectorConstruction
- G4VUserPhysicsList

Global: **only one instance** of them exists in memory, shared by all threads (**readonly**). Managed only by the **master** thread.

Action classes

Invoked during the execution loop

- G4VUserActionInitialization

- G4VUserPrimaryGeneratorAction
- G4UserRunAction (*)
- G4UserEventAction
- G4UserTrackingAction
- G4UserStackingAction
- G4UserSteppingAction

Local: an **instance** of each action class exists **for each thread**.

(*) Two RunAction's allowed: one for master and one for threads

Primary as a mandatory action class

The **PrimaryGeneratorAction.cc** class file is an **Action** that must be defined. Actions are:

- **Initialization classes:**
 - Use:
`G4RunManager::SetUserInitialization()` to define
 - Invoked at the initialization
`G4VUserDetectorConstruction;`
`G4VUserPhysicsList;`
- **Loop Actions:**
 - `G4RunManager::SetUserAction()` to define
 - Invoked during an event loop
`G4VUserPrimaryGeneratorAction;`
`G4UserRunAction;`
`G4UserStackingAction;`
`G4UserTrackingAction;`
`G4UserSteppingAction;`

G4VUserPrimaryGeneratorAction

Is one of the **mandatory user classes** and it controls the generation of primary particles

- This class does not generate primaries but invokes the **GeneratePrimaryVertex()** method to make the primary;
- It sends the primary particles to the **G4Event** object;

Constructor

- Instantiate primary generator (i.e. **G4ParticleGun()**)
particleGun gun = new **G4ParticleGun**(n_particle);
- Set the default values
gun->**SetParticleEnergy**(1.0 * GeV);

GeneratePrimaries() method

- Randomise particle-by-particle value;
- Set these values to primary generator;
- Invoke **GeneratePrimaryVertex()** method of primary generator;

...its concrete implementation

```
ExN02PrimaryGeneratorAction::ExN02PrimaryGeneratorAction(  
    ExN02DetectorConstruction* myDC)
```

```
:myDetector(myDC)  
{  
    G4int n_particle = 1;  
    particleGun = new G4ParticleGun(n_particle);  
    // default particle  
    G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable();  
    G4ParticleDefinition* particle = particleTable->FindParticle("proton");  
  
    particleGun->SetParticleDefinition(particle);  
    particleGun->SetParticleMomentumDirection(G4ThreeVector(0.,0.,1.));  
    particleGun->SetParticleEnergy(3.0*GeV);  
}
```

```
ExN02PrimaryGeneratorAction::~ExN02PrimaryGeneratorAction()
```

```
{  
    delete particleGun;  
}
```

Class constructor

Class destructor

...its concrete implementation

Generate primaries

```
void ExN02PrimaryGeneratorAction::GeneratePrimaries(G4Event*  
anEvent)  
{  
  G4double position = -0.5*(myDetector->GetWorldFullLength());  
  particleGun->SetParticlePosition(G4ThreeVector(0.*cm,0.*cm,position));  
  
  particleGun->GeneratePrimaryVertex(anEvent);  
}
```

G4VPrimaryGenerator

GeneratePrimaries(G4Event* aEvent) is the mandatory method

Geant4 provides three **G4VPrimaryGenerators**:

- G4ParticleGun
- G4HEPEvtInterface
- G4GeneralParticleSource

G4HEPEvInterface

- Concrete implementation of **G4VPrimaryGenerator**
- Almost all event generators in use are written in **FORTRAN** but Geant4 **does not link** with any external **FORTRAN** code
- Geant4 provides an ASCII file interface for such event generators
- **G4HEPEvtInterface** reads an ASCII file produced by an Event generator and reproduce the **G4PrimaryParticle** objects.
- In particular it reads the /HEPEVT/ fortran block used by almost all event generators
- It does not give a place for the primary particle so the interaction point **must be still set** by the User

G4ParticleGun

```
G4ParticleGun* pG = new G4ParticleGun();
```

- Concrete implementation of **G4VPrimaryGenerator**
- It shoots a **certain number** of primary particles of a **certain energy** from a **certain point** at a **certain time** toward a **certain direction**
- Various “**Set**” methods are available (see <https://geant4.kek.jp/lxr/source//event/include/G4ParticleGun.hh>)

```
void SetParticleEnergy(G4double aKineticEnergy);  
void SetParticleMomentum(G4double aMomentum);  
void SetParticlePosition(G4ThreeVector aPosition);  
void SetNumberOfParticles(G4int aHistoryNumber);
```

G4ParticleGun

```
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

G4GeneralParticleSource

```
G4GeneralParticleSource* GPS = new G4GeneralParticleSource();
```

- <https://geant4.kek.jp/lxr/source//event/include/G4GeneralParticleSource.hh>
- Concrete implementation of **G4VPrimaryGenerator**
class **G4GeneralParticleSource** : public G4VPrimaryGenerator
- Is designed to **replace** the **G4ParticleGun** class
- It is design to allow specification of **multiple particle sources** each with **independent** definition of particle type, position, direction and energy distribution
- Primary vertex can be **randomly chosen** on the surface of a certain volume
- **Momentum direction** and **kinetic energy** of the primary particle can also be randomized
- Distribution defined by **UI commands**

G4GeneralParticleSource

Online manual:

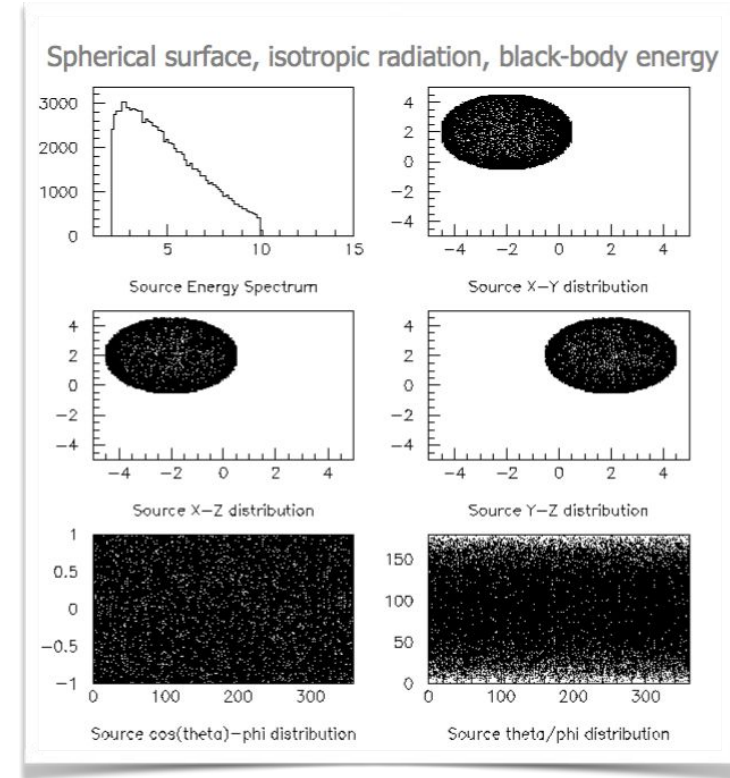
<http://geant4-userdoc.web.cern.ch/geant4-userdoc/UsersGuides/ForApplicationDeveloper/html/GettingStarted/generalParticleSource.html?highlight=gps#macro-commands>

/gps main commands:

- `/gps/pos/type` (Point, Planar, Beam, ...)
- `/gps/ang/type` (iso, planar, cos, ...)
- `/gps/ene/type` (Mono, Lin, Pow, ...)
- Many more!

G4GeneralParticleSource

- **Source I: point-like source, 100 MeV proton, along Z**
 - `/gps/pos/type` Point
 - `/gps/particle` proton
 - `/gps/ene/mono` 100 MeV
 - `/gps/direction` 0. 0. 1.
- **Source II: plane source (2x2 cm²), 100 MeV proton, along Z**
 - `/gps/pos/type` Plane
 - `/gps/pos/shape` Square
 - `/gps/pos/halfx` 1. * cm
 - `/gps/pos/halfy` 1. * cm
 - `/gps/particle` proton
 - `/gps/ene/mono` 100 MeV
 - `/gps/direction` 0. 0. 1.
- **Source III: gaussian-like (σ_x and $\sigma_y = 2$ cm), 100 MeV proton, along Z**
 - `/gps/pos/type` Beam
 - `/gps/pos/sigma_x` 2.0 * cm
 - `/gps/pos/sigma_y` 2.0 * cm
 - `/gps/particle` proton
 - `/gps/ene/mono` 100 MeV
 - `/gps/direction` 0. 0. 1.



ParticleGun vs GPS

- **Particle Gun**

- Simple and native
- Shoot one track at a time
- Easily to handle

- **General Particle Source**

- Powerful
- **Controlled by UI commands** (`G4GeneralParticleSourceMessenger.hh`)

✓ Almost impossible to control with set methods

- capability of shooting particles from a surface of a volume
- Capability of randomizing kinetic energy, position, direction following a user-specified distribution (histogram)

● If you need to shoot primary particles from a surface of a complicated volume (outward or inward), GPS is the choice

● If you need a complicated distribution, GPS is the choice

Examples

example/extended/.....

- **GPS**

[/eventgenerator/exgps](#)

- **HEPEvInterface**

[/runAndEvent/RE02/srcRE01PrimaryGeneratorAction.cc](#)

Task 2

Link to the Tasks:

<http://geant4.lns.infn.it/alghero2023/task2>

Task 1 - Primary generators

Exercises

- **Task 2a Geant4 Particle Gun**
- **Task 2b Geant4 General Particle Source**