



Primary Generators

Gianfranco Paternò
INFN – Ferrara division

XI International Geant4 School, Pavia - Jan 14 – 19, 2024



Outline

- Primary vertex and primary particle
- G4VPrimaryGenerator instantiated via the `GeneratePrimaryVertex()`
 - The particle gun
 - Interfaces to HEPEVT and HEPMC
 - General Particle Source (or GPS)
- Particle gun or GPS?



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

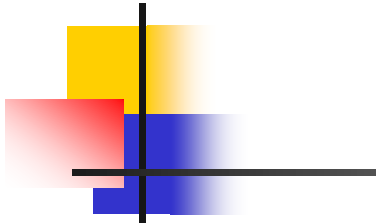
Local: an **instance** of each action class exists **for each thread**.

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



G4VUserPrimaryGeneratorAction

- It is one of the **mandatory** user classes and it controls the **generation** of **primary particles**
 - This class does not directly generate primaries but invokes the **GeneratePrimaryVertex()** **method** of a **generator** to create the initial state
 - It **registers** the primary particle(s) to the **G4Event** object
- It has **GeneratePrimaries(G4Event*)** method which is **purely virtual**, so it **must** be implemented in the user class



```
26 //
27 // $Id: G4VUserPrimaryGeneratorAction.hh,v 1.5 2006/06/29 21:13:38 gunter Exp $
28 // GEANT4 tag $Name: geant4-09-03-patch-02 $
29 //
30
31 #ifndef G4VUserPrimaryGeneratorAction_h
32 #define G4VUserPrimaryGeneratorAction_h 1
33
34 class G4Event;
35
36 // class description:
37 //
38 // This is the abstract base class of the user's mandatory action class
39 // for primary vertex/particle generation. This class has only one pure
40 // virtual method GeneratePrimaries() which is invoked from G4RunManager
41 // during the event loop.
42 // Note that this class is NOT intended for generating primary vertex/particle
43 // by itself. This class should
44 // - have one or more G4VPrimaryGenerator concrete classes such as G4ParticleGun
45 // - set/change properties of generator(s)
46 // - pass G4Event object so that the generator(s) can generate primaries.
47 //
48
49 class G4VUserPrimaryGeneratorAction
50 {
51 public:
52     G4VUserPrimaryGeneratorAction();
53     virtual ~G4VUserPrimaryGeneratorAction();
54
55 public:
56     virtual void GeneratePrimaries(G4Event* anEvent) = 0;
57 };
58
59 #endif
```



Outline

- Primary vertex and primary particle
- **G4VPrimaryGenerator instantiated via the GeneratePrimaryVertex()**
- The particle gun
- Interfaces to HEPEVT and HEPMC
- General Particle Source (or GPS)
- Particle gun or GPS?



G4VPrimaryGenerator

- **G4VPrimaryGenerator** is the **base class** for particle generators, that are called by `GeneratePrimaries(G4Event*)` to produce an **initial state**
 - **Notice:** you may have **many particles** from one vertex, or even **many vertices** in the initial state
- **Derived** class from **G4VPrimaryGenerator** **must implement** the purely virtual method **GeneratePrimaryVertex()**
- Geant4 provides **three concrete classes** derived by **G4VPrimaryGenerator**
 - G4ParticleGun
 - G4HEPEvtInterface
 - G4GeneralParticleSource



G4ParticleGun

- (Simplest) **concrete implementation** of **G4VPrimaryGenerator**
 - It can be used for experiment-specific **primary generator** implementation
- It shoots **one primary particle** of a given energy from a given point at a given time to a given direction
- Various **“Set” methods** are available (see `../source/event/include/G4ParticleGun.hh`)

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


G4VUserPrimaryGeneratorAction: the usual recipe

- Constructor
 - **Instantiate** primary generator (i.e. **G4ParticleGun()**)
particleGun = new G4ParticleGun();
 - (Optional, but advisable): set the **default** values
particleGun -> SetParticleEnergy(1.0*GeV);
- **GeneratePrimaries()** **mandatory** method
 - **Randomize** particle-by-particle value, if required
 - **Set** these values to the primary generator
 - **Invoke GeneratePrimaryVertex()** method of primary generator
 - **particleGun->GeneratePrimaryVertex()**

A "real-life" myPrimaryGenerator: constructor & destructor

```
myPrimaryGenerator::myPrimaryGenerator ()
: G4VUserPrimaryGeneratorAction(), fParticleGun(0)
{
    fParticleGun = new G4ParticleGun(); } Instantiate
                                     } concrete generator
    // set defaults
    fParticleGun->SetParticleDefinition(
        G4Gamma::Definition());
    fParticleGun->
        SetParticleMomentumDirection(G4ThreeVector(0.,0.,1.));
    fParticleGun->SetParticleEnergy(6.*MeV);
}

myPrimaryGenerator::~~myPrimaryGenerator ()
{
    delete fParticleGun; } Clean it up in the destructor
}
```



A "real-life" myPrimaryGenerator: **GeneratePrimaries(G4Event*)**

```
myPrimaryGenerator::GeneratePrimaries(G4Event* evt)
{
  // Randomize event-per-event
  G4double cost = -1.0 + G4UniformRand()*2.0;
  G4double phi = G4UniformRand()*twopi;
  } Sample direction
  } isotropically

  G4double sinT = sqrt(1-cost*cost);
  G4ThreeVector direction(sinT*sin(phi), sinT*cos(phi), cost);

  G4double ene = G4UniformRand()*6*MeV;
  } Sample energy
  } (flat distr.)

  fParticleGun->SetParticleDirection(direction);
  fParticleGun->SetParticleEnergy(ene);
  }

  fParticleGun->GeneratePrimaryVertex(evt);
  } Shoot event
}
```

G4ParticleGun

- Commands can be also given **interactively** by **user interface**
 - But **cannot** do **randomization** in this case
- Allows to change **primary parameters** **between** one run and an other
 - Notice: parameters from the UI could be **overwritten** in **GeneratePrimaries()**

```
/gun/energy 10 MeV
```

```
/gun/particle mu+
```

```
/gun/direction 0 0 -1
```

} Change settings

```
/run/beamOn 100
```

← Start first run

```
/gun/particle mu-
```

```
/gun/position 10 10 -100 cm
```

} Change settings

```
/run/beamOn 100
```

← Start second run



Outline

- Primary vertex and primary particle
- Built-in primary particle generators
 - The particle gun
 - **Interfaces to HEPEVT and HEPMC**
 - General Particle Source (or GPS)
- Particle gun or GPS?



G4HEPEvtInterface

- 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** (born at the LEP time) used by almost all event generators
- It generates only the kinematics of the initial state, so does **the interaction point must be still set by the user**



Outline

- Primary vertex and primary particle
- Built-in primary particle generators
 - The particle gun
 - Interfaces to HEPEVT and HEPMC
 - **General Particle Source (or GPS)**
- Particle gun or GPS?



G4GeneralParticleSource()

- `source/event/include/G4GeneralParticleSource.hh`
- Concrete implementation of `G4VPrimaryGenerator`
`class G4GeneralParticleSource : public G4VPrimaryGenerator`
- Is designed to replace the `G4ParticleGun` class
- It is designed 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, or within a volume
 - Momentum direction and kinetic energy of the primary particle can also be randomized
- Distribution defined by **UI commands**

Methods for setting (default) features of a GPS

PrimaryGeneratorAction::PrimaryGeneratorAction()

//defining a GPS

```
fGPS = new G4GeneralParticleSource();
```

```
//Primary particles
```

```
G4ParticleDefinition* particle = G4ParticleTable::GetParticleTable()->FindParticle("e-");
```

```
fGPS->SetParticleDefinition(particle);
```

//Spatial distribution

```
G4SPSPosDistribution* vPosDist = fGPS->GetCurrentSource()->GetPosDist();
```

```
vPosDist->SetPosDisType("Plane");
```

```
vPosDist->SetPosDisShape("Circle");
```

```
vPosDist->SetRadius(50.*mm);
```

```
vPosDist->SetCentreCoords(G4ThreeVector(0.,0.,0.));
```

//Angular distribution

```
G4SPSANGDistribution* vAngDist = fGPS->GetCurrentSource()->GetAngDist();
```

```
vAngDist->SetParticleMomentumDirection(G4ThreeVector(0., 0., 1.));
```

//Energy distribution

```
G4SPSEneDistribution* vEneDist = fGPS->GetCurrentSource()->GetEneDist();
```

```
vEneDist->SetEnergyDisType("Mono");
```

```
vEneDist->SetMonoEnergy(400.*keV);
```

Useful macro commands for GPS

```
#source type
#/gps/pos/type Plane
#/gps/pos/shape Circle           #circular source
#/gps/pos/radius 0.5 mm
/gps/pos/type Plane
/gps/pos/shape Ellipse          #elliptical source
/gps/pos/halfx 40 mm
/gps/pos/halfy 30 mm
#/gps/pos/type Volume           #Volume or Surface
#/gps/pos/shape Sphere          #spherical source
#/gps/pos/radius 0.5 mm
#/gps/pos/type Volume
#/gps/pos/shape Cylinder        #cylindrical source
#/gps/pos/radius 0.5 mm
#/gps/pos/halfz 0.00015 mm
#/gps/pos/type Point           #point-source

#position
/gps/pos/centre 0. 0. -5. mm

#direction
#divergetnt beam
#/gps/ang/type iso
#/gps/ang/maxtheta 0.000179 rad
#/gps/ang/rot1 -1 0 0
#parallel beam
/gps/direction 1. 1. 1.         #beam 45 deg tilted

#particle
/gps/particle gamma             #proton,neutron,e,e+,mu+,pi0,...

#energy
/gps/ene/type Gauss
/gps/ene/mono 3. MeV
/gps/ene/sigma 0.015 MeV
```

#full list of GPS commands:

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

#useful GPS examples:

http://hurel.hanyang.ac.kr/Geant4/Geant4_GPS/reat.space.qinetiq.com/gps/examples/examples.html



ParticleGun vs. GPS

■ G4ParticleGun

- **Simple** and native
- Shoots **one track** at a time
- **Easy** to handle

■ G4GeneralParticleSource

- **Powerful**
- Controlled by **UI commands**
 - G4GeneralParticleSourceMessenger.hh
 - Almost impossible to do with the naïve Set methods
- Capability of shooting particles from a **surface** or 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



When do you need your own derived class of **G4VPrimaryGenerator**

- In some cases, what is provided by Geant4 **does not fit** specific needs: need to write a **derived class** from **G4VPrimaryGenerator**
 - Must implement the virtual method **GeneratePrimaryVertex(G4Event* evt)**
 - Generate **vertices (G4PrimaryVertex)** and attach **particles** to each of them (**G4PrimaryParticle**)
 - Add vertices to the event **evt->AddPrimaryVertex()**
- Needed when:
 - You need to **interface** to a **non-HEPEvt external generator**
 - neutrino interaction, Higgs decay, non-standard interactions
 - **Many particles** from one vertex, or **many vertices**
 - double beta decay
 - **Time difference** between primary tracks



Examples

- **examples/extended/analysis/A01/src/A01PrimaryGeneratorAction.cc** is a good example to start with
- Examples also exist for **GPS**
examples/extended/eventgenerator/exgps
- And for **HEPEvtInterface**
example/extended/runAndEvent/RE01/src/RE01PrimaryGeneratorAction.cc

Reading a phase-space file

```
#include "G4ParticleDefinition.hh"

#include "Randomize.hh"

#include "G4GeneralParticleSource.hh"
#include "G4ParticleGun.hh"

#include "G4PhysicalConstants.hh"
#include "G4SystemOfUnits.hh"
#include "globals.hh"

//...oooOO00Oooo.....oooOO00Oooo.....oooOO00Oooo.....oooOO00Oooo.....

namespace {G4Mutex PrimaryGeneratorActionMutex = G4MUTEX_INITIALIZER;}
FileReader* PrimaryGeneratorAction::fFileReader = 0;

//...oooOO00Oooo.....oooOO00Oooo.....oooOO00Oooo.....oooOO00Oooo.....

PrimaryGeneratorAction::PrimaryGeneratorAction():
fReadFromFile(0)
{
    //G4cout << "### PrimaryGeneratorAction instantiated ###" << G4endl;

    //instantiating the messenger
    fMessenger = new PrimaryGeneratorActionMessenger(this);

    //defining a Particle Gun (to be used with the file reader)
    fGun = new G4ParticleGun();

    //defining a GPS (it is used in batch mode if fReadFromFile=0)
    fGPS = new G4GeneralParticleSource();

    //set default values (for the runs from the GUI)
```

PrimaryGeneratorAction.cc

Reading a phase-space file

```
//.....ooo0000ooo.....ooo0000ooo.....ooo0000ooo.....ooo0000ooo.....  
void PrimaryGeneratorAction::SetFileName(G4String vFileName) {fFileName = vFileName;}  
//.....ooo0000ooo.....ooo0000ooo.....ooo0000ooo.....ooo0000ooo.....  
void PrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)  
{  
    G4AutoLock lock(&PrimaryGeneratorActionMutex);  
  
    if (!fFileReader) {  
        fFileReader = new FileReader(fFileName);  
  
        if (fReadFromFile) {  
            G4cout << G4endl << "Reading " << fFileName << " ..." << G4endl;  
            fFileReader->StoreEvents();  
            G4cout << "File correctly read!" << G4endl << G4endl;  
        }  
    }  
  
    if (fReadFromFile) {  
        fGun->SetParticleDefinition(G4ParticleTable::GetParticleTable()  
            ->FindParticle(fFileReader->GetAnEventParticle(anEvent->GetEventID())));  
        fGun->SetParticlePosition(fFileReader->GetAnEventPosition(anEvent->GetEventID()));  
        fGun->SetParticleMomentumDirection(fFileReader->GetAnEventMomentum(anEvent->GetEventID()));  
        fGun->SetParticleEnergy(fFileReader->GetAnEventEnergy(anEvent->GetEventID()));  
        fGun->GeneratePrimaryVertex(anEvent);  
    } else {  
        fGPS->GeneratePrimaryVertex(anEvent);  
    }  
}  
//.....ooo0000ooo.....ooo0000ooo.....ooo0000ooo.....ooo0000ooo.....
```

PrimaryGeneratorAction.cc

Reading a phase-space file

```
//.....ooo0000ooo.....ooo0000ooo.....ooo0000ooo.....ooo0000ooo.....  
  
class FileReader  
{  
public:  
    FileReader(G4String);  
    FileReader();  
    ~FileReader();  
  
public:  
    void SetFileName(G4String);  
    void StoreEvents();  
  
    G4String GetAnEventParticle(G4int);  
    G4ThreeVector GetAnEventPosition(G4int);  
    G4ThreeVector GetAnEventMomentum(G4int);  
    G4double GetAnEventEnergy(G4int);  
    G4int GetNumberOfEvents();  
  
private:  
    G4String fFileName;  
    std::ifstream inputFile;  
    std::vector<G4String> evListPart;  
    std::vector<G4ThreeVector> evListPos;  
    std::vector<G4ThreeVector> evListMom;  
    std::vector<G4double> evListEnergy;  
};  
  
//.....ooo0000ooo.....ooo0000ooo.....ooo0000ooo.....ooo0000ooo.....
```

FileReader.hh

Reading a phase-space file

```
//...ooo00000ooo.....ooo00000ooo.....ooo00000ooo.....ooo00000ooo....  
  
#include "FileReader.hh"  
  
#include "G4SystemOfUnits.hh"  
#include "G4ParticleTable.hh"  
  
//...ooo00000ooo.....ooo00000ooo.....ooo00000ooo.....ooo00000ooo....  
  
FileReader::FileReader(G4String fileName)  
{  
    fFileName = fileName;  
    inputFile.open(fFileName.data());  
}  
  
//...ooo00000ooo.....ooo00000ooo.....ooo00000ooo.....ooo00000ooo....  
  
FileReader::~FileReader()  
{  
    inputFile.close();  
}  
  
//...ooo00000ooo.....ooo00000ooo.....ooo00000ooo.....ooo00000ooo....  
  
void FileReader::SetFileName(G4String vFileName)  
{  
    fFileName=vFileName;  
}  
  
//...ooo00000ooo.....ooo00000ooo.....ooo00000ooo.....ooo00000ooo....  
  
void FileReader::StoreEvents()  
{
```

FileReader.cc

Reading a phase-space file

```
void FileReader::StoreEvents()
{
    if (evListPos.size() == 0) {
        G4String particle = "geantino";
        G4double x = 0.;
        G4double y = 0.;
        G4double z = 0.;
        G4double px = 0.;
        G4double py = 0.;
        G4double pz = 0.;
        G4double p = 0.;
        G4double m = 0.;
        G4double E = 0.;
        G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable();
        while (inputFile.good()) {
            if (inputFile.good()) inputFile >> particle;
            if (inputFile.good()) inputFile >> x;
            if (inputFile.good()) inputFile >> y;
            if (inputFile.good()) inputFile >> px;
            if (inputFile.good()) inputFile >> py;
            if (inputFile.good()) inputFile >> pz;
            p = sqrt(px*px+py*py+pz*pz);
            m = particleTable->FindParticle(particle)->GetPDGMass();
            E = sqrt(p*p + m*m)*MeV;
            if (E <= 0) {E = 1.*eV;}
            evListPart.push_back(particle);
            evListPos.push_back(G4ThreeVector(x*cm, y*cm, z*cm));
            evListMom.push_back(G4ThreeVector(px*MeV, py*MeV, pz*MeV));
            evListEnergy.push_back(E);
        }
        G4cout << "the file contains " << evListPart.size() << " particles" << G4endl;
    }
}
```

FileReader.cc

Hands-on session

Task2

- Task2a: Geant4 Particle Gun
- Task2b: Geant4 General Particle Source
- <http://geant4.lns.infn.it/pavia2024/task2/index.html>