Generation of a primary event

Luciano Pandola

INFN-LNGS

Queen's University, Belfast (UK), January 22, 2013

Based on a presentation by G.A.P. Cirrone (INFN-LNS)

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

G4RunManager::
SetUserInitialization()

Action classes

Invoked during the execution loop

- G4VUserPrimaryGeneratorAction
- G4UserRunAction
- G4UserEventAction
- G4UserTrackingAction
- G4UserStackingAction
- G4UserSteppingAction

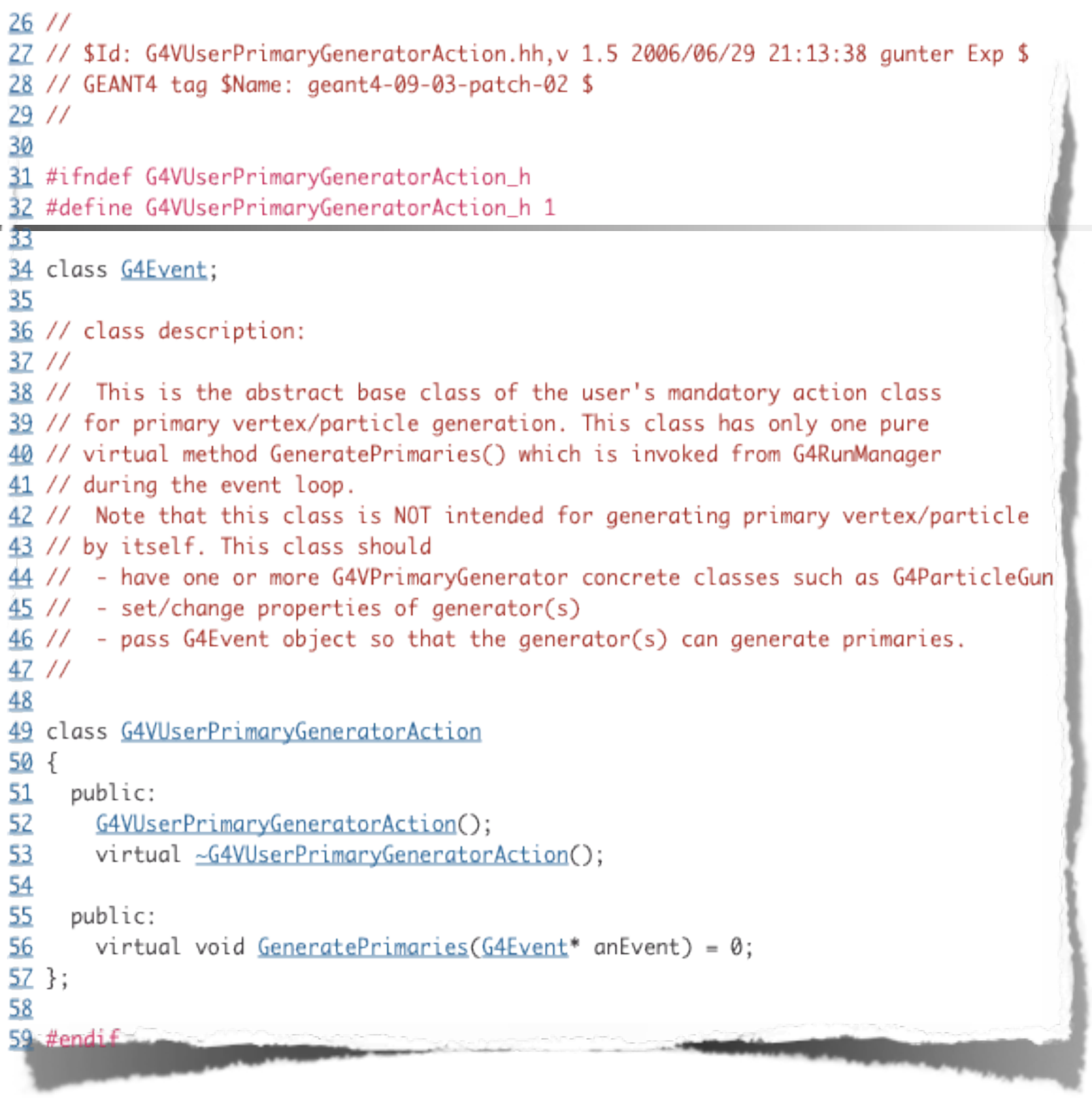
G4RunManager::
SetUserAction()

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

G4VUserPrimaryGeneratorAction: the usual recipe

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



```
//
// $Id: G4VUserPrimaryGeneratorAction.hh,v 1.5 2006/06/29 21:13:38 gunter Exp $
// GEANT4 tag $Name: geant4-09-03-patch-02 $
//

#ifndef G4VUserPrimaryGeneratorAction_h
#define G4VUserPrimaryGeneratorAction_h 1

class G4Event;

// class description:
//
//  This is the abstract base class of the user's mandatory action class
// for primary vertex/particle generation. This class has only one pure
// virtual method GeneratePrimaries() which is invoked from G4RunManager
// during the event loop.
//  Note that this class is NOT intended for generating primary vertex/particle
// by itself. This class should
//  - have one or more G4VPrimaryGenerator concrete classes such as G4ParticleGun
//  - set/change properties of generator(s)
//  - pass G4Event object so that the generator(s) can generate primaries.
//

class G4VUserPrimaryGeneratorAction
{
  public:
    G4VUserPrimaryGeneratorAction();
    virtual ~G4VUserPrimaryGeneratorAction();

  public:
    virtual void GeneratePrimaries(G4Event* anEvent) = 0;
};

#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);
```

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

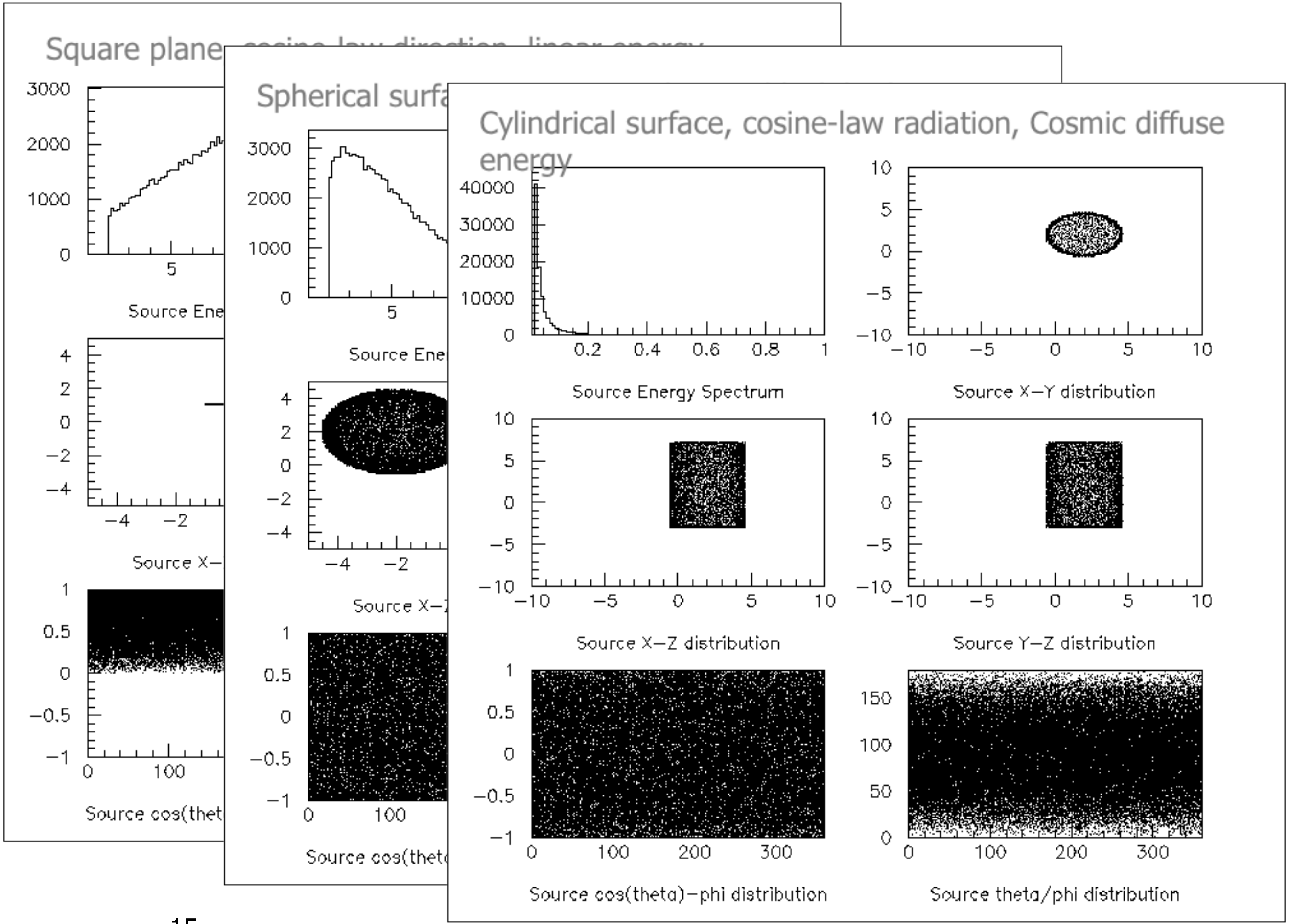
- Primary vertex and primary particle
- Built-in primary particle generators
  - The particle gun
    - Interfaces to HEPEVT and HEPMC
  - General Particle Source (or GPS)
- Particle gun or GPS?

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
  - Momentum direction and kinetic energy of the primary particle can also be randomised
- Distribution defined by **UI commands**

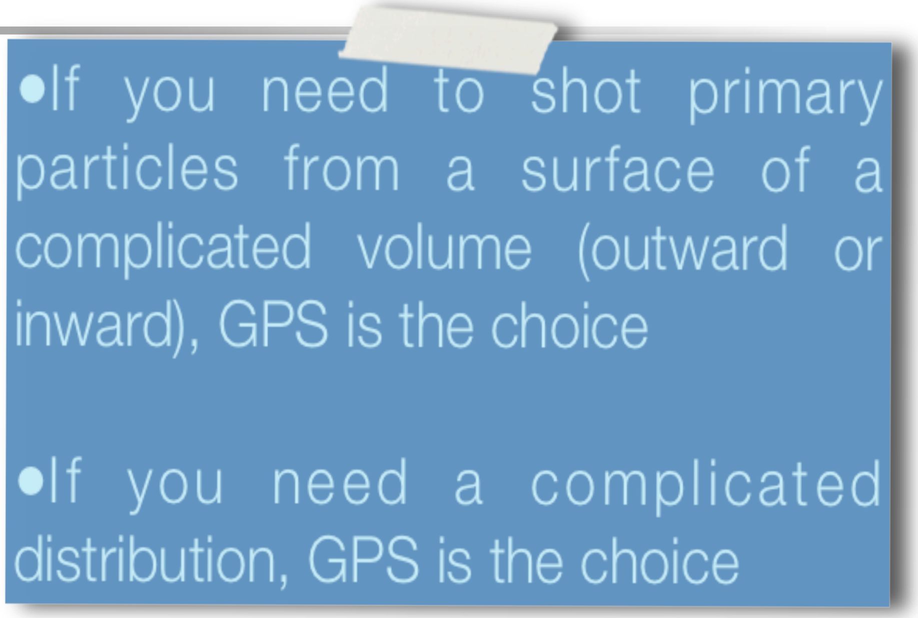
G4GeneralParticleSource

- On line manual: **`http://reat.space.qinetiq.com/gps/`**
- /gps main command
  - **`/gps/pos/type`** (planar, point, etc.)
  - **`/gps/ang/type`** (iso, planar wave, etc.)
  - /**`gps/energy/type`** (monoenergetic, linear, User defined)
  - ....


Square plane, cosine-law direction, linear energy
Spherical surfa
Cylindrical surface, cosine-law radiation, Cosmic diffuse energy
Source Energy Spectrum
Source X−Y distribution
Source X−Z distribution
Source Y−Z distribution
Source cos(theta)−phi distribution
Source theta/phi distribution

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 shot 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

- **`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`**

A summary: what to do and where to do

- In the constructor of your UserPrimaryGeneratorAction
  - Instantiate **`G4ParticleGun`**
  - Set default values by Set methods of G4ParticleGun:
    - Particle type, kinetic energy, position and direction
- In your macro file or from your interactive terminal session
  - Set values for a run
- In the **`GeneratePrimaries()`** method
  - Shoot random numbers and prepare the values of
    - kinetic energy, position, direction ....
  - Use set methods of G4ParticleGun to set such values
  - Then invoke **`GeneratePrimaryVertex()`** method of G4ParticleGun
  - If you need more than one primary track per event, loop over randomisation and **`GeneratePrimaryVertex()`**