Generation of a Primary event

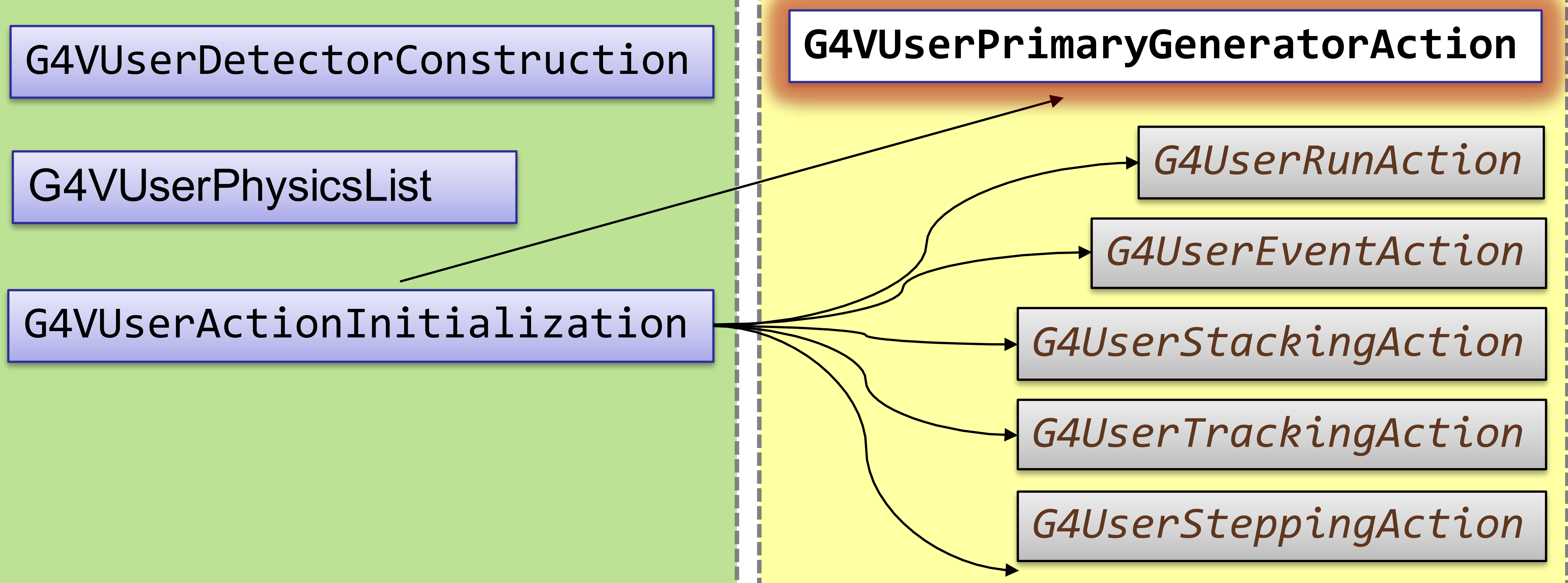
Giada Petringa (LNS-INFN)

The 2nd Geant4 School in China
Shandong University
Qingdao

User Classes

At initialization

At execution



Global: only one instance exists in memory, shared by all threads.

Thread-local: an instance of each action class exists for each thread.

The Primary is a mandatory action class

- **The PrimaryGeneratorAction.cc class file is an 'Action' that must be defined**
- **The initialisation classes**

  -Use: `G4RunManager::SetUserInitialization()` to define;

  - Invoked at the initialisation:
  G4VUserDetectorConstruction
  G4VUserPhysicsList

- **Action classes**

  -`G4RunManager::SetUserAction()` to define;

  -Invoked during an event loop

  ✓G4VUserPrimaryGeneratorAction

  ✓G4UserRunAction

  ✓G4UserStackingAction

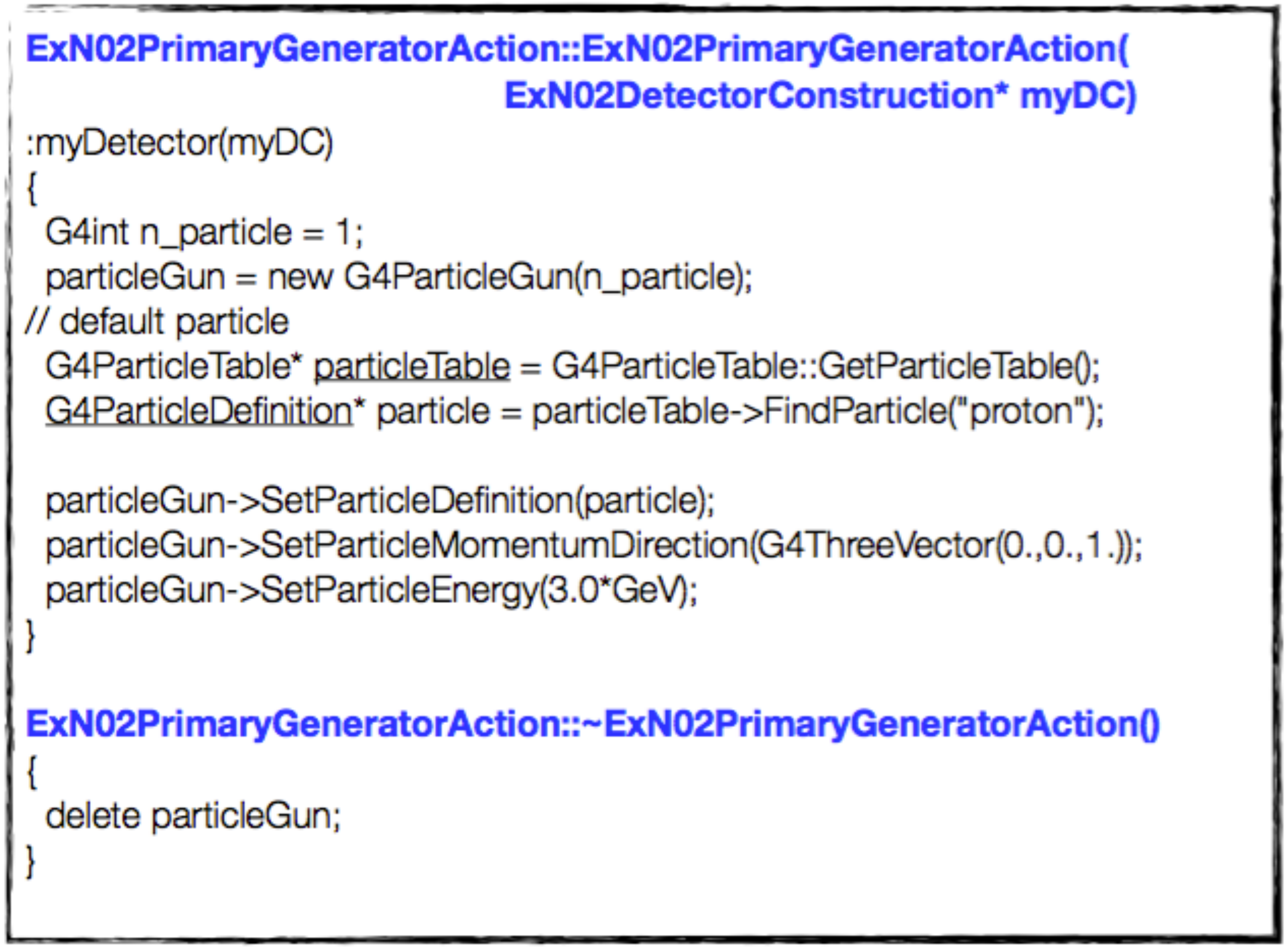
  ✓G4UserTrackingAction

  ✓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 = new G4ParticleGun(n_particle);`
  - Set the default values
    `particleGun -> 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



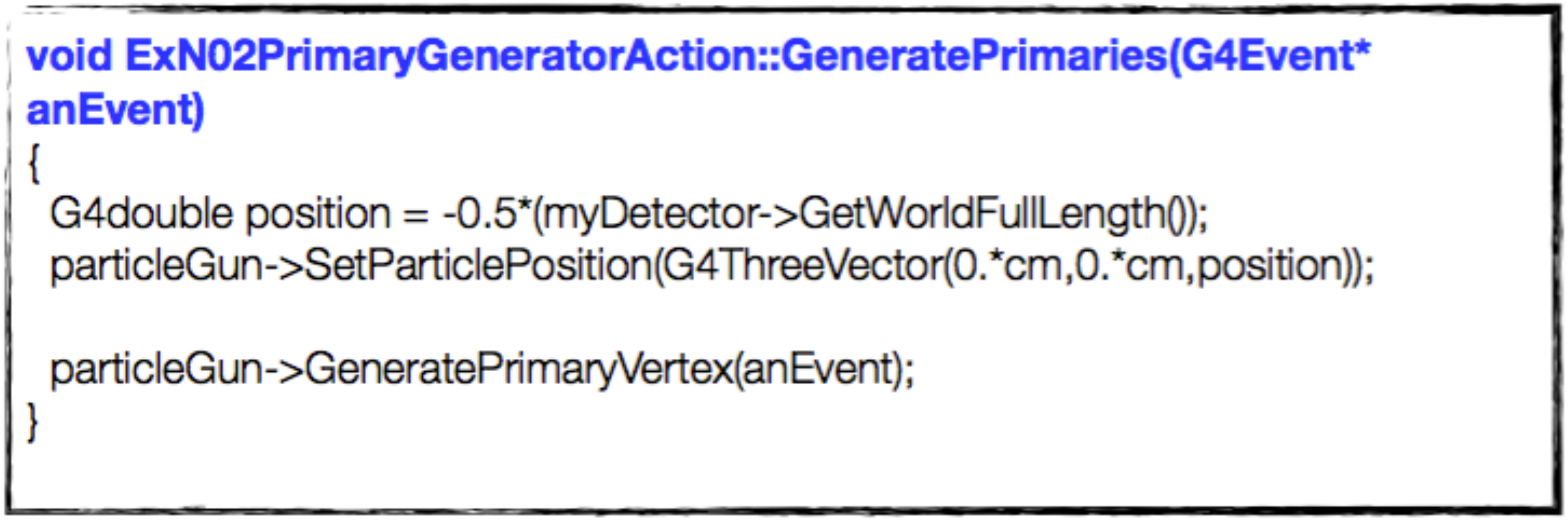
```
ExN02PrimaryGeneratorAction::ExN02PrimaryGeneratorAction(
                                ExN02DetectorConstruction* myDC)
:myDetector(myDC)
{
  G4int n_particle = 1;
  particleGun = new G4ParticleGun(n_particle);
// default particle
  G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable();
  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 distructor

…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 event

- 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

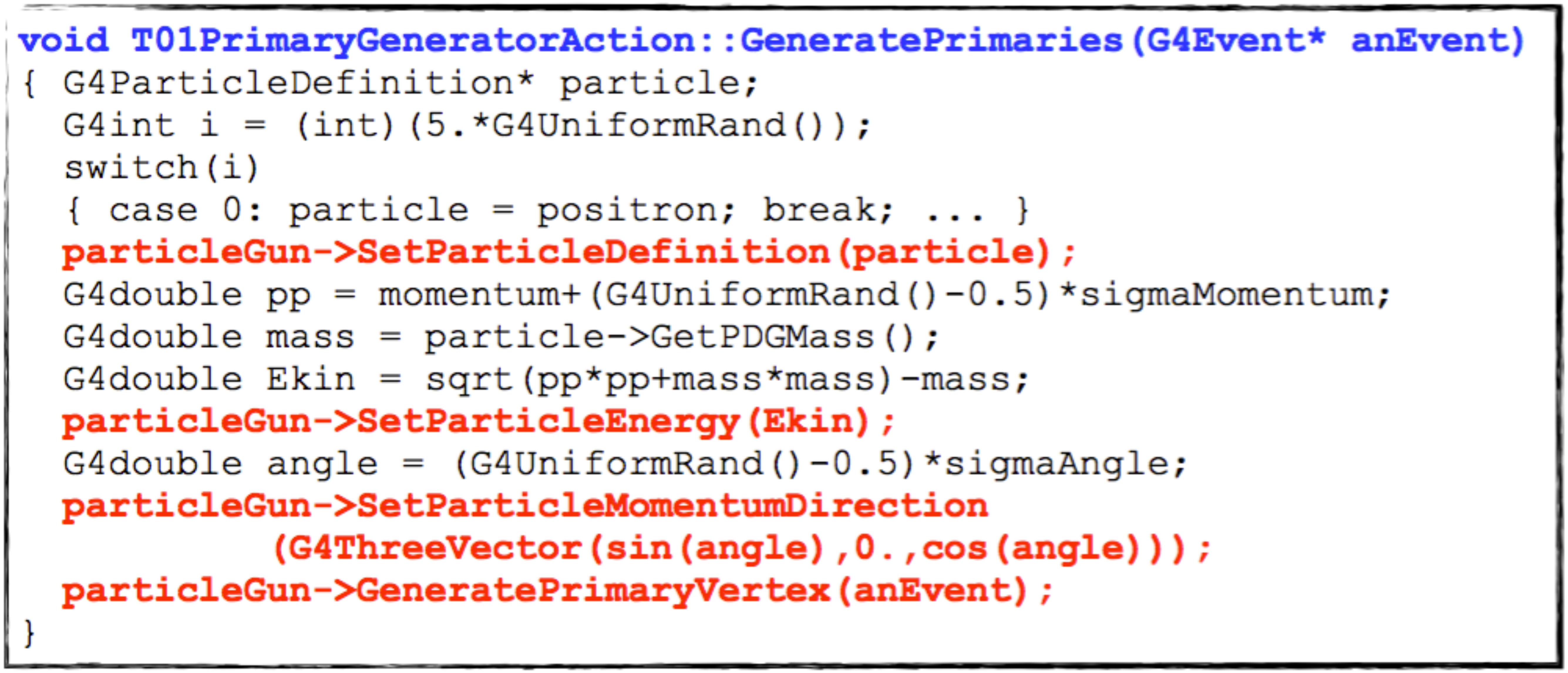
G4ParticleGun()

```
particleGun = new G4ParticleGun ();
```

- Concrete implementation of G4VPrimaryGenerator
- 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 (see ../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()

```
fGenerateParticleSource = new G4GenerateParticleSoure ();
```

- ../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.)

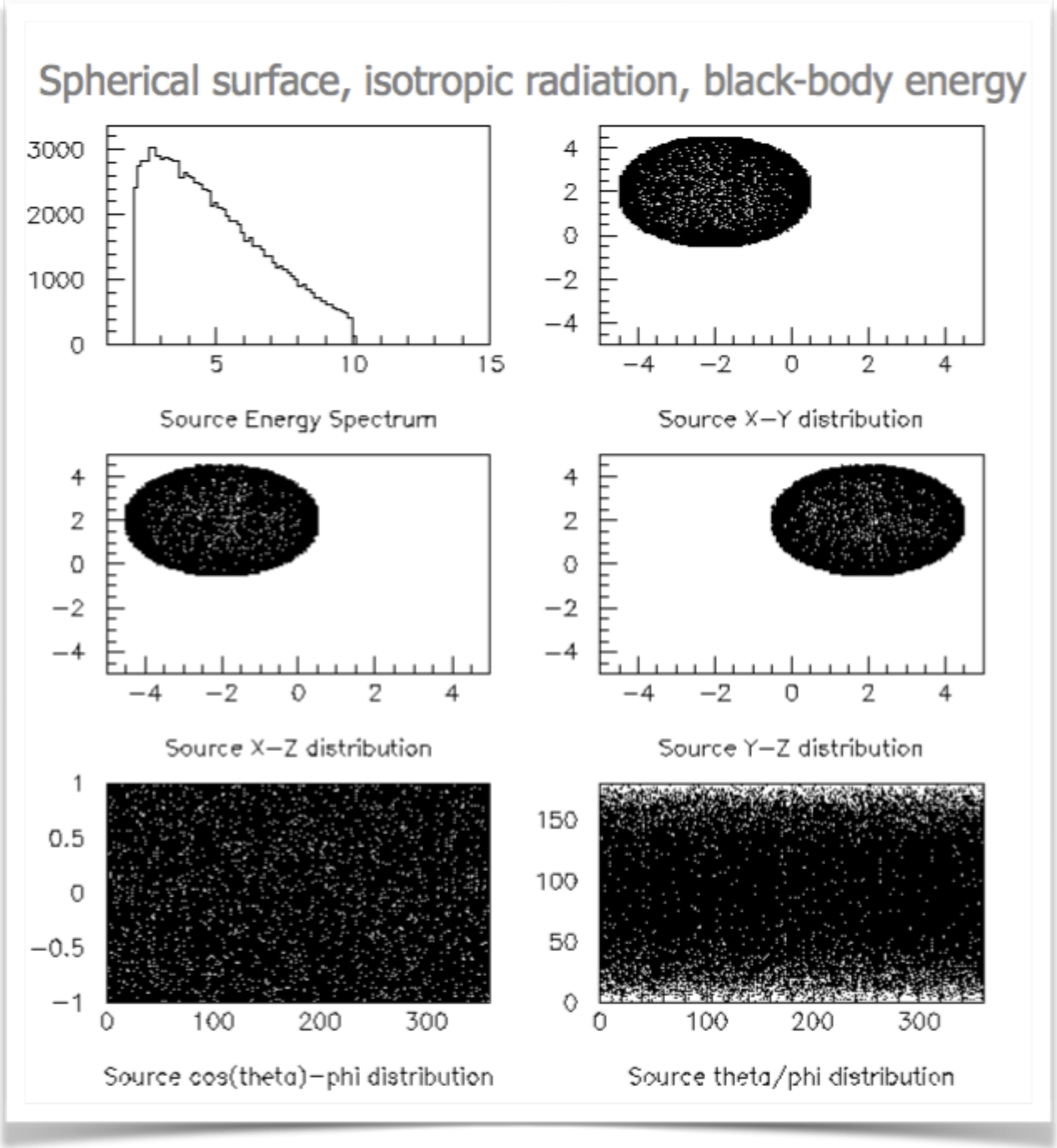
  -gps/ang/type (iso, planar wave, etc.)

  -gps/energy/type (monoenergetic, linear, User defined)

  -..............

G4GeneralParticleSource()

- Source 1: point-like source, 100 MeV proton, along z
  - /gps/pos/type point
  - /gps/particle proton
  - /gps/energy 100 MeV
  - /gps/direction 0 0 1
- Source 2: plane source (2x2 cm), 100 MeV proton, along z
  - /gps/pos/type plane
  - /gps/pos/shape square
  - /gps/pos/centre x y z
  - /gps/pos/Halfx
  - /gps/pos/Halfy
- Source 3: gaussian-like (sigmax and sigmay = 2cm), 100 MeV proton, along z
  - /gps/pos/shape Circle
  - /gps/pos/centre x y z
  - /gps/pos/sigmax 2 cm



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 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

example/extended/…..

- ✓ **GPS**
  /eventgenerator/exgps

- ✓ **HEPEvInterface**
  /runAndEvent/RE02/srcRE01PrimaryGeneratorAction.cc

Next task

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

Exercise 2b.1: Instantiate the GeneralParticleSource

```
PrimaryGeneratorAction::PrimaryGeneratorAction()
{
    // Task 2b.1: Comment out the particle gun creation and instatiate a GPS instead
    fGPS = new ...();

    // Task 2b.1: Set the same properties for the GPS (removing previous lines)
    fGPS->SetParticleDefinition(...);
    fGPS->GetCurrentSource()->GetEneDist()->SetMonoEnergy(...);
    fGPS->GetCurrentSource()->GetAngDist()->SetParticleMomentumDirection(G4ThreeVector(...));
    fGPS->GetCurrentSource()->GetPosDist()->SetCentreCoords(G4ThreeVector(...));
}
```

Exercise 2b.2: Changing GPS parameter from macro commands

Exercise 2b.3: Creating a complicated GPS source with macro commands

Next task

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

Exercise 2a.1: Instantiate and customize the Particle Gun

```
// complete particle name, energy and momentum
G4ParticleDefinition* myParticle;
myParticle = G4ParticleTable::GetParticleTable()->FindParticle("...");
fGun->SetParticleDefinition(myParticle);
fGun->SetParticleEnergy(...);
fGun->SetParticleMomentumDirection(G4ThreeVector(...));
```

Exercise 2a.2: Change parameters of the particle gun

```
void PrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
{
    // Task 2a.2: Include the position randomization
    // Definition of the new original coordinates
    G4double x0  = ..., y0  = ..., z0  = ...;

    // Definition of the spatial extent of the uniform source
    G4double dx0 = ..., dy0 = ..., dz0 = ...;

   // Start the randomization of the initial coordinates using the G4UniformRand() function
   x0 += dx0*(G4UniformRand()-0.5);
   y0 += dy0*(G4UniformRand()-0.5);
   z0 += dz0*(G4UniformRand()-0.5);

   ...
   fGun->Set...
   fGun->GeneratePrimaryVertex(anEvent);
}
```

INFN
GEANT

...It's all!