Generation of a primary event

Lesson Outline

- Introduction to some classes
- Primary Generation Action class
- Implementation
- Generators
- Further details/ examples
- Task

Classes

Global - only exists in memory for 1 instance, shared by all threads

At Initialisation:

- G4UserDetectorConstruction
- G4VUserPhysicsList
- G4VUserActionInitialization

To define use
G4RunManager::SetUserInitialization()
Invoked at initialisation

Thread-local - instance of each action class exists for each thread

At Execution:

- G4VUserPrimaryGeneratorAction
- G4UserRunAction
- G4UserEventAction
- G4UserStackingAction
- G4UserTrackingActio
- G4UserSteppingAction

To define use
G4RunManager::SetUserAction()
Invoked during an event loop

Primary Generator Action

At Execution:

G4VUserPrimaryGeneratorAction

- Mandatory user class
- Doesn't generate primaries
- Invokes GeneratePrimaryVertex() (method to make the primary)
- Sends the primary particles to G4Event object

Primary vertex and the primary particle are added to a GEANT4 Event

Implementation in the src file

```
ExP02PrimaryGeneratorAction::ExP02PrimaryGeneratorAction()
                            G4VUserPrimaryGeneratorAction(),
fParticleGun(0)
{
G4int n_particle = 1;
fParticleGun = new G4ParticleGun(n_particle);
G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable();
G4String particleName;
fParticleGun->SetParticleDefinition(particleTable-
>FindParticle(particleName="geantino"));
fParticleGun->SetParticleEnergy(1.0*GeV);
fParticleGun->SetParticlePosition(G4ThreeVector(0.0, 0.0, 0.0));
}
ExP02PrimaryGeneratorAction::~ExP02PrimaryGeneratorAction()
{
delete fParticleGun;
}
void ExP02PrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
{
G4ThreeVector v(1.0,0.1,0.1);
fParticleGun->SetParticleMomentumDirection(v);
fParticleGun->GeneratePrimaryVertex(anEvent);
}
```

Class constructor

Initiation of primary generator -> G4ParticleGun()

Setting of default values

Class destructor

GeneratePrimaries()
- Randomises particle-by-particle values
- Sets values to primary generator

Invokes GeneratePrimaryVertex() -> method of primary generator

Generators

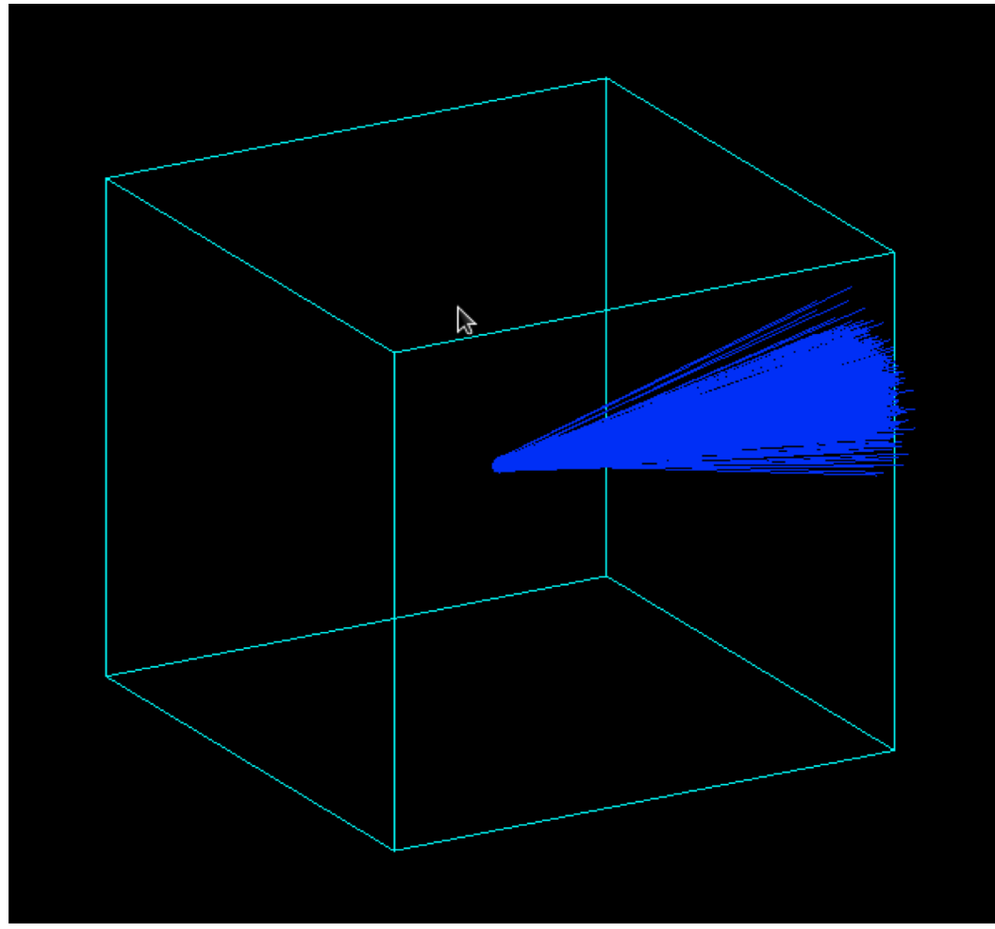
G4VPrimaryGenerator

- Introduction to some classes GeneratePrimaries(G4Event*aEvent) (mandatory event)
- Geant4 provides 3 G4VPrimaryGenerators: (all concrete implementation

G4ParticleGun

G4HEPEvtInterface

G4GeneralParticleSource



G4ParticleGun

- Shoots one primary particle of a certain energy from a certain point at a certain time to a certain direction
- Various 'Set' methods available (/source/event/include/G4ParticleGun.hh)
  - Void SetParticleEnergy (G4Double aKineticEnergy)
  - Void SetParticleMomentum (G4double aMomentum)
- Methods can be repeated for generating more than one primary particle

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

Implementation example

```
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);
}
```

Can be repeated for generating more than one primary particles

G4HEPEvtInterface

- GEANT4 provides an ASCII file interface (unlike usual FORTRAN code) for event generators
- G4HEPEvtInterface reads this ASCII file produced by an Event generator to reproduce the G4PrimaryParticle objects (in particular the /HEPEVT/ fortran block)
- Does not place for the primary particle so the interaction point must be set by the User

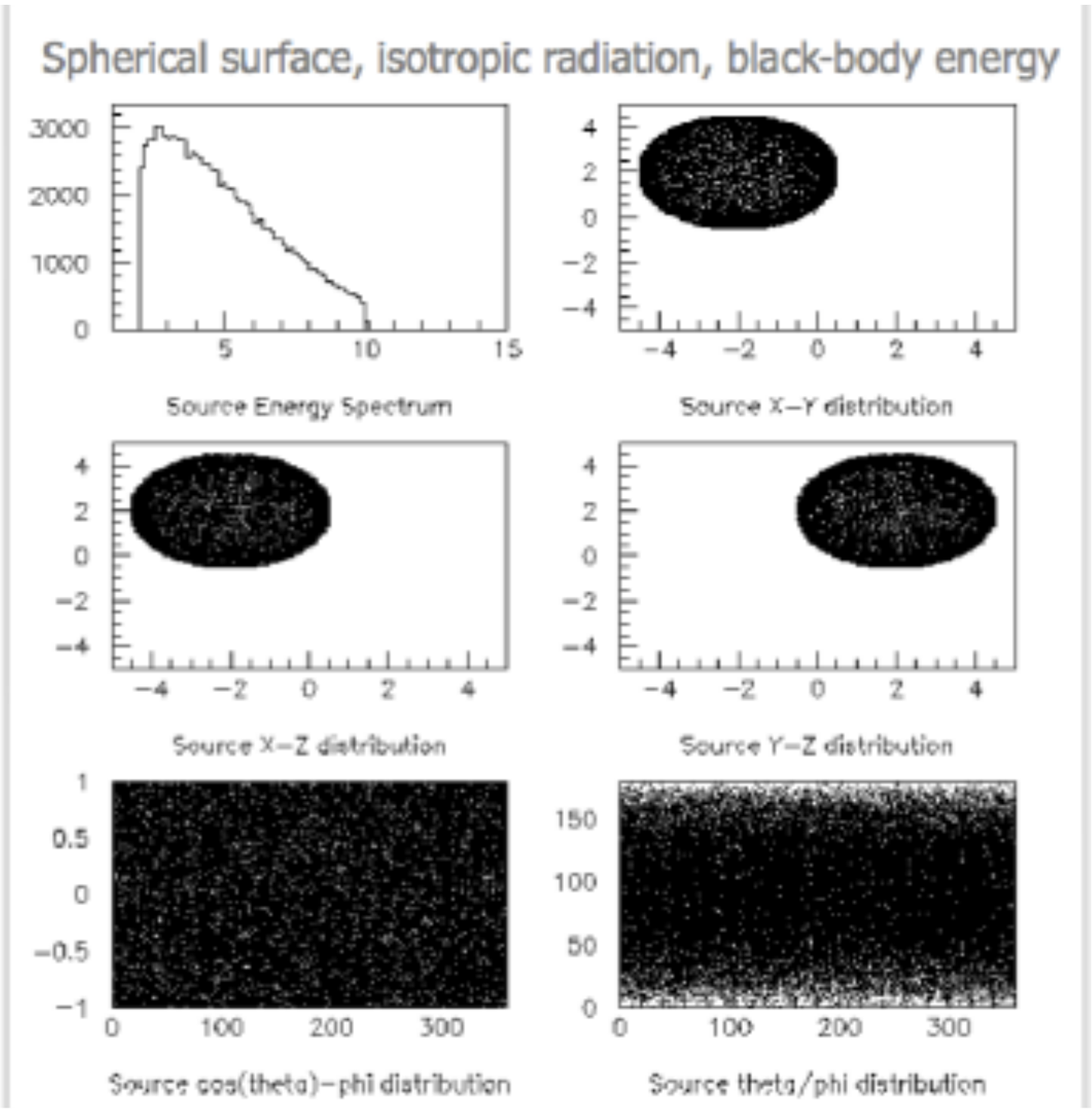
G4GeneralParticleSource

- Designed to replace G4ParticleGun class
- Allows specification of multiple particle sources each with independent definition of particle type, position, direction and energy distribution
- Primary vertex can be chosen on the surface of a certain volume (randomly)
- Momentum, direction and kinetic energy can also be randomised
- Distribution defined by UI commands

```
fGenerateParticleSource = new G4GenerateParticleSource();
```

…/source/event/include/G4GeneralParticleSource.hh

Implementation example



- Source: point like, 100 MeV proton along z

```
/gps/pos/type point

/gps/particle proton

/gps/energy 100 MeV

/gps/direction 0 0 1
```

- Source: plane source(2x2), 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
```

Comparison

Particle Gun	General Particle Source	HEP event interface
Simple and native	Powerful	Doesn't give place of primary particle
Shoots one track at a time	Controlled by UI commands	Interaction point must be set by user
Easy to handle	Capability of shooting particles from a surface of a **volume** and of randomising kinetic energy, position, direction, following (complicated) user specified **distribution**	

Further details

- Online manual:

(http://reat.space.qinetiq.com/gps/)

Thank you