Generation of a primary event

III International Geant4 and GPU programming school

9-13 November 2015, Catania

Antonella Tramontana

Data

Outlines

- Introduction
- Built-in primary particle generators
  - Particle gun
  - Interface to HEPEVT and HEPCM
  - General particle source

Outlines

- Introduction
- Built-in primary particle generators
  - Particle gun
  - Interface to HEPEVT and HEPCM
  - General particle source

User Classes

Initialization classes

Use G4RunManager::SetUserInitialization() to define
Invoked at the initialization

- ➡ G4VUserDetectorConstruction
- ➡ G4VUserPhysicsList

Action classes

Use G4RunManager::SetUserAction() to define
Invoked during an event loop

- ➡ G4VUserPrimaryGeneratorAction
- ➡ G4UserRunAction
- ➡ G4UserEventAction
- ➡ G4UserStackingAction
- ➡ G4UserTrackingAction
- ➡ G4UserSteppingAction
- ➡ G4VUserPhysicsList

Classes written in red are **mandatory**.
For these classes only virtual interfaces are provided -> users **MUST** implement their concrete classes

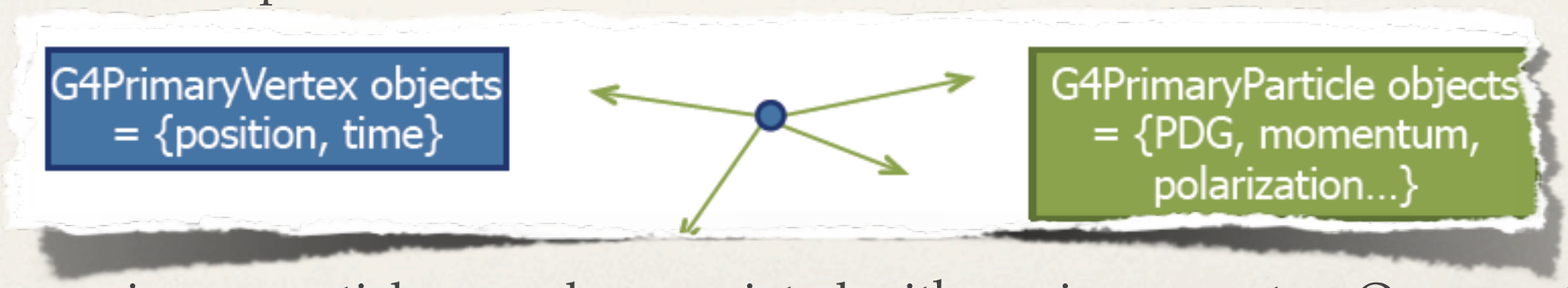
Primary vertex and primary particle

G4VUserPrimaryGeneratorAction is the abstract base class to control the generation of primary vertex/particle

....What is a primary particle/vertex?

- Primary particle means particle with which you start an event.
  - ➡ e. g. particles made by primary p-p collision, alpha emitted form radioactive material, gamma-ray from treatment head, etc..
  - ➡ Primary particle has a particle ID, momentum and optionally polarization.
  - ➡ Then Geant4 tracks these primary particles in your geometry with physics interaction and generated secondaries, detector responses and /or scorers

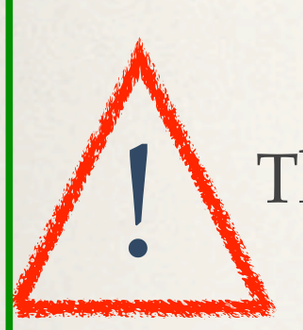
- Primary vertex has position and time.



One or more primary particles may be associated with a primary vertex. One event may have one or more primary vertices.

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 to create the initial state

It sends 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

Where? → geant4.10.01.p01-install/include/Geant4

```
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
```

G4VUserPrimaryGeneratorAction

Constructor:

- ➡ Instantiate primary generator(s) (i.e. **G4ParticleGun()**)

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

- ➡ (Optional, but advisable): set the default values

```
particleGun -> SetParticleEnergy(1.0*GeV);
```

GeneratePrimaries() mandatory method:

- ➡ Invoked at the beginning of each event
- ➡ Randomize particle-by-particle value(s)
- ➡ Set these values to the primary generator(s)
- ➡ Invoke GeneratePrimaryVertex() method of primary generator

```
particleGun->GeneratePrimaryVertex()
```

The concrete implementation

Constructor : Invoked only once

.cc

```
MyPrimaryGeneratorAction::MyPrimaryGeneratorAction()
: G4VUserPrimaryGeneratorAction(),fParticleGun(0)
{
  G4int n_particle = 1;
  fParticleGun  = new G4ParticleGun(n_particle);

  // default particle kinematic
  G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable();
  G4String particleName;
  G4ParticleDefinition* particle
    = particleTable->FindParticle(particleName="gamma");
  fParticleGun->SetParticleDefinition(particle);
  fParticleGun->SetParticleMomentumDirection(G4ThreeVector(0.,0.,1.));
  fParticleGun->SetParticleEnergy(6.*MeV);
}
//....oooOOOOOooo........oooOOOOOooo........
MyPrimaryGeneratorAction::~MyPrimaryGeneratorAction()
{
  delete fParticleGun;
}
```

.hh

```
  private:
    G4ParticleGun*  fParticleGun; // pointer a to G4 gun class
```

The concrete implementation

GeneratePrimaries:
Invoked once per each event

```
void MyPrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
{

  G4double size = 0.8*cm;
  G4double z0 = 5*cm;
  G4double x0 = size *(G4UniformRand()-0.5);
  G4double y0 = size * (G4UniformRand()-0.5);
  G4double z0 = size * (G4UniformRand()-0.5)+z0;

  fParticleGun->SetParticlePosition(G4ThreeVector(x0,y0,z0));

  fParticleGun->GeneratePrimaryVertex(anEvent);
}
```

Outlines

- Introduction

- Built-in primary particle generators

  - Particle gun

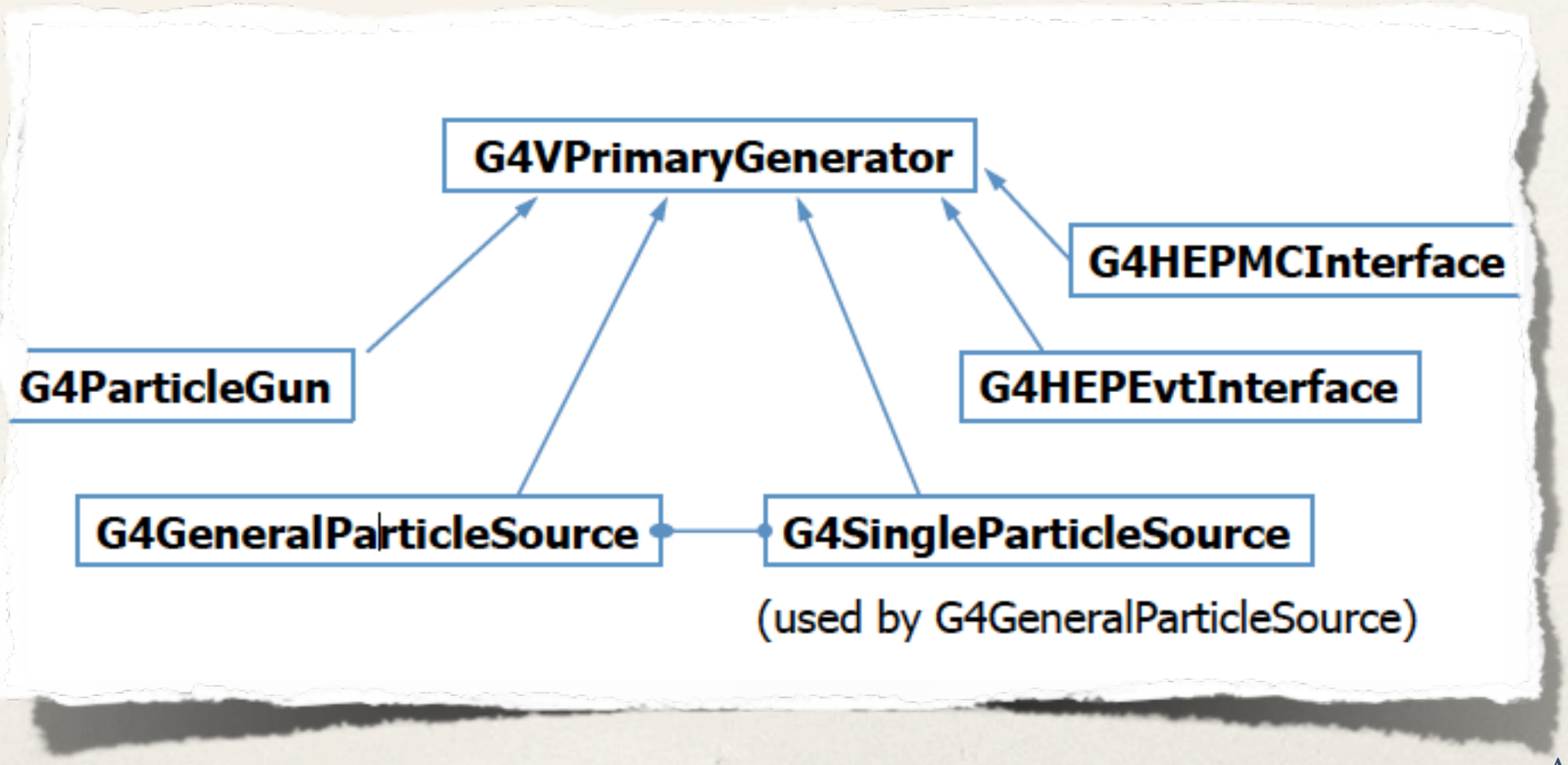
  - Interface to HEPEVT and HEPCM

  - General particle source

Built-in concrete classes of G4VPrimaryGenerators

Geant4 provides three G4VPrimaryGenerators

- ➡ G4ParticleGun
- ➡ G4HEPEvtInterface & G4HepMCInterface
- ➡ G4GeneralParticleSource



G4ParticleGun

➡ Concrete implementation of `G4VPrimaryGenerator`

➡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 SetParticleDefinition(G4ParticleDefinition*)
void SetParticleMomentum(G4ParticleMomentum)
void SetParticleMomentumDirection(G4ThreeVector)
void SetParticleEnergy(G4double)
void SetParticleTime(G4double)
void SetParticlePosition(G4ThreeVector)
void SetParticlePolarization(G4ThreeVector)
void SetNumberOfParticles(G4int)
```

➡ Intercoms commands are also available for setting initial values

What to do and where to do

1. In the **constructor** of your UserPrimaryGeneratorAction
   - Instantiate `G4ParticleGun`
   - Set default values by Set methods of G4ParticleGun:

     Particle type, kinetic energy, position and direction
2. 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
3. In your **macro file** or from your interactive terminal session
   - Set values for a run

The concrete implementation of 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.

Outlines

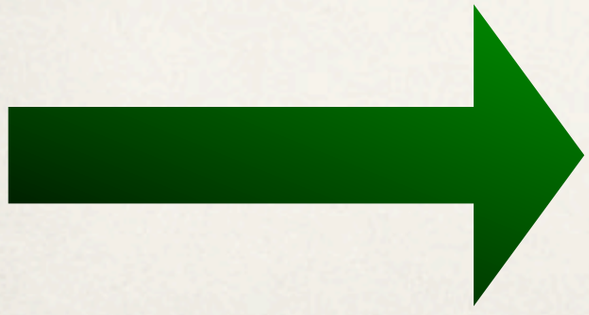
- Introduction
- Built-in primary particle generators
  - Particle gun
  - Interface to HEPEVT and HEPCM
  - General particle source

Interface to HEPEvt?

Almost all event generators in use are written in FORTRAN

....but Geant4 does not link with any **external** **FORTRAN** code

However 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.

Interfaces to HEPEvt and HepMC

Concrete implementations of `G4VPrimaryGenerator`

Primary

G4HEPEvtInterface

- Suitable to /HEPEVT/ common block, which many of Fortran generators are compliant to
- ASCII file input

G4HepMCInterface

– An interface to HepMC class, which a few new (C++) HEP physics generators are compliant to.
– ASCII file input or direct linking to a generator through HepMC.

As example see: examples/extended/runAndEvent/RE01/src/ RE01PrimaryGeneratorAction.cc

For more details see: https://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/ForApplicationDeveloper/html/ch03s06.html

Outlines

- Introduction
- Built-in primary particle generators
  - Particle gun
  - Interface to HEPEVT and HEPCM
  - General particle source

G4GeneralParticleSource

- Concrete implementation of `G4VPrimaryGenerator`

  Suitable especially to space and medical applications
- 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
- Some beam features can be randomly defined
- Distributions are defined by UI commands

The concrete implementation of G4GeneralParticleSource

```
MyPrimaryGeneratorAction::
MyPrimaryGeneratorAction()
{
generator = new G4GeneralParticleSource;
}

void MyPrimaryGeneratorAction::
GeneratePrimaries(G4Event* anEvent)
{
 generator->GeneratePrimaryVertex(anEvent);
}
```

Very simple!

Summary of gps features

- Primary vertex can be randomly positioned with several options

  Emission from point, plane,…
- Angular emission

  Several distributions; isotropic, cosine-law, focused, …

  With some additional parameters (min/max-theta, min/max-phi,…)
- Kinetic energy of the primary particle can also be randomized.

  Common options (e.g. mono-energetic, power-law), some extra shapes (e.g. black-body) or user defined
- Multiple sources

  With user defined relative intensity

```
# beam #1
   # default intensity is 1,
   # now change to 5.
   /gps/source/intensity 5.
.......
# beam #2
   # 2x the intensity of beam #1
   /gps/source/add 10
......
```

Position distributions /gps/pos/...

Point

E.g.
```
/gps/pos/type Point
/gps/pos/centre 0. 0. 0. cm
```

Beam

E.g.
```
/gps/pos/type Beam
/gps/pos/shape Circle
/gps/pos/radius 1. mm
/gps/pos/sigma_r 2. mm
```

Plane

Shape: Circle, Annulus, Ellipsoid, Square or Rectangle

E.g.
```
/gps/pos/type Plane
/gps/pos/shape Rectangle
/gps/pos/halfx 50 cm
/gps/pos/halfy 70 cm
```

Surface or Volume

Shape: Sphere, Ellipsoid, Cylinder or Para

Surface: zenith automatically oriented as normal to surface at point

E.g.
```
/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/radius 1. m
```

Angular distribution /gps/ang/...

- Isotropic (**iso**)
- Cosine-law (**cos**)
  See next slides for more information

- Planar wave (**planar**)
  Standard emission in one direction (it's also implicitly set by **`/gps/direction x y z`**)
- Accelerator beam
  1-d or 2-d gaussian emission, **beam1d** or **beam2d**
- Focusing to a point (**focused**)
- User-defined (**user**)

Energy distributions /gps/ene/...

Kinetic energy of the primary particle can also be randomized, with several predefined options:

- Common options (e.g. mono-energetic, power-law, exponential, gaussian, etc)
  mono-energetic (**Mono**)
  linear (**Lin**)
  power-law (**Pow**)
  exponential (**Exp**)
  gaussian (**Gauss**)

- Some extra predefined spectral shapes (bremsstrahlung, black-body, cosmic diffuse gamma ray,…)
  bremsstrahlung (**Brem**)
  black-body (**Bbody**)
  cosmic diffuse gamma ray (**Cdg**)

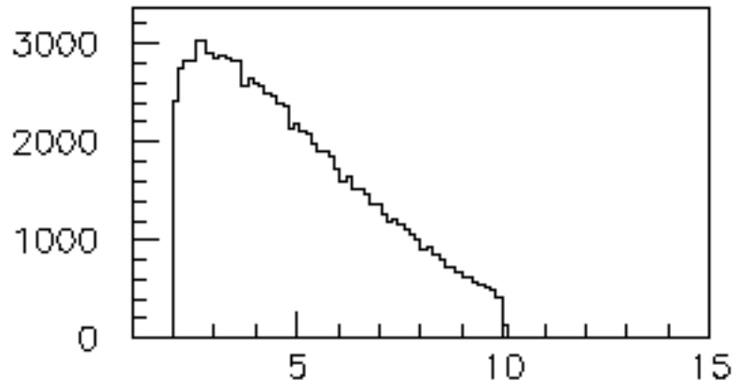
- User defined
  user-defined histogram (**User**)
  arbitrary point-wise spectrum (**Arb**) and
  user-defined energy per nucleon histogram (**Epn**)

Example(1)

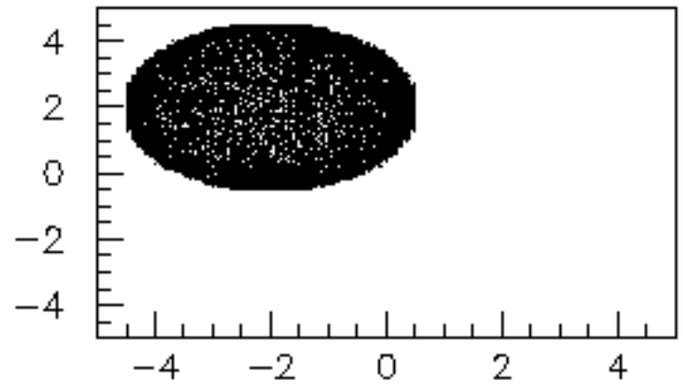
Macro

```
/gps/particle geantino

/gps/pos/type Surface
/gps/pos/shape Sphere
/gps/pos/centre -2. 2. 2. cm
/gps/pos/radius 2.5 cm

/gps/ang/type iso

/gps/ene/type Bbody
/gps/ene/min 2. MeV
/gps/ene/max 10. MeV
/gps/ene/temp 2e10
/gps/ene/calculate
```

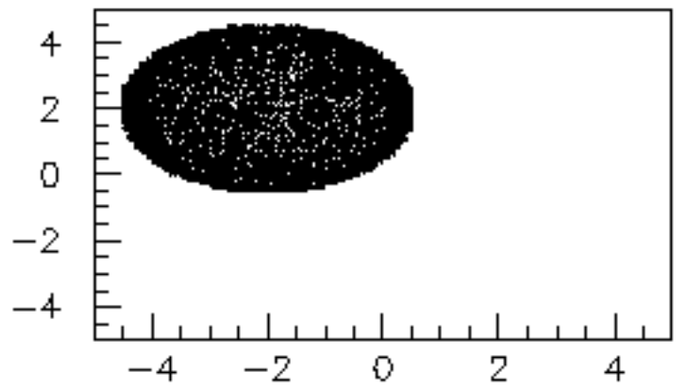
Spherical surface, isotropic radiation, black-body energy spectrum



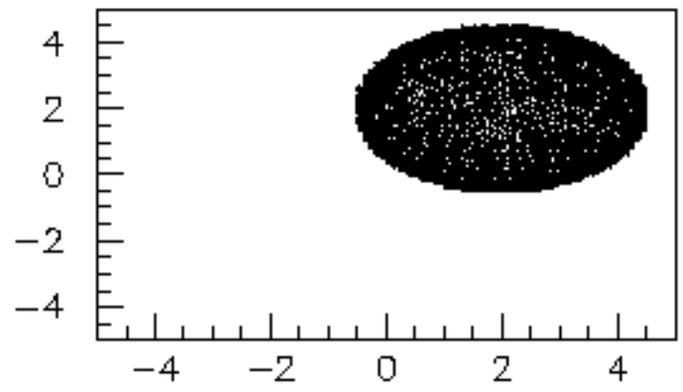
Source Energy Spectrum



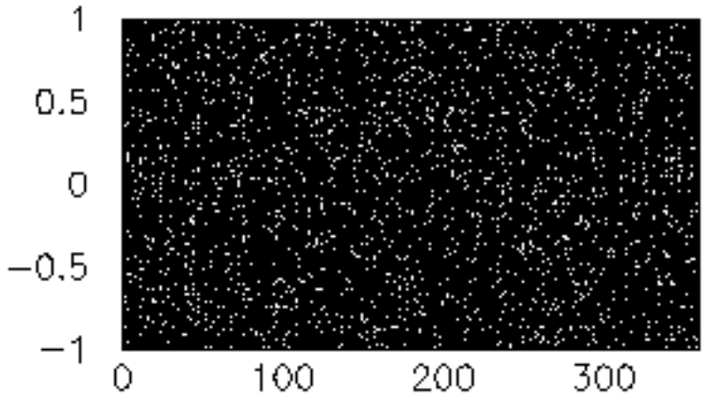
Source X−Y distribution



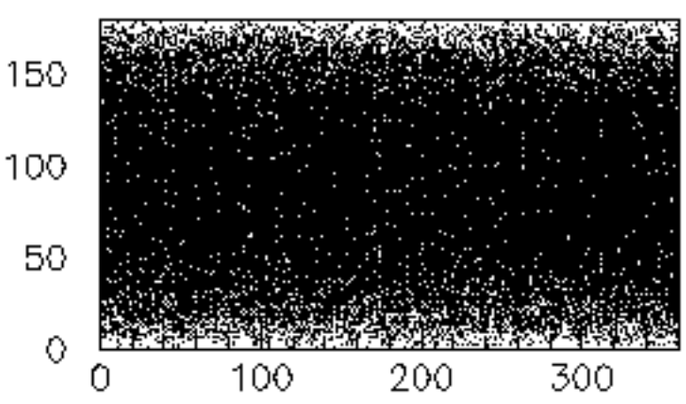
Source X−Z distribution



Source Y−Z distribution



Source cos(theta)−phi distribution



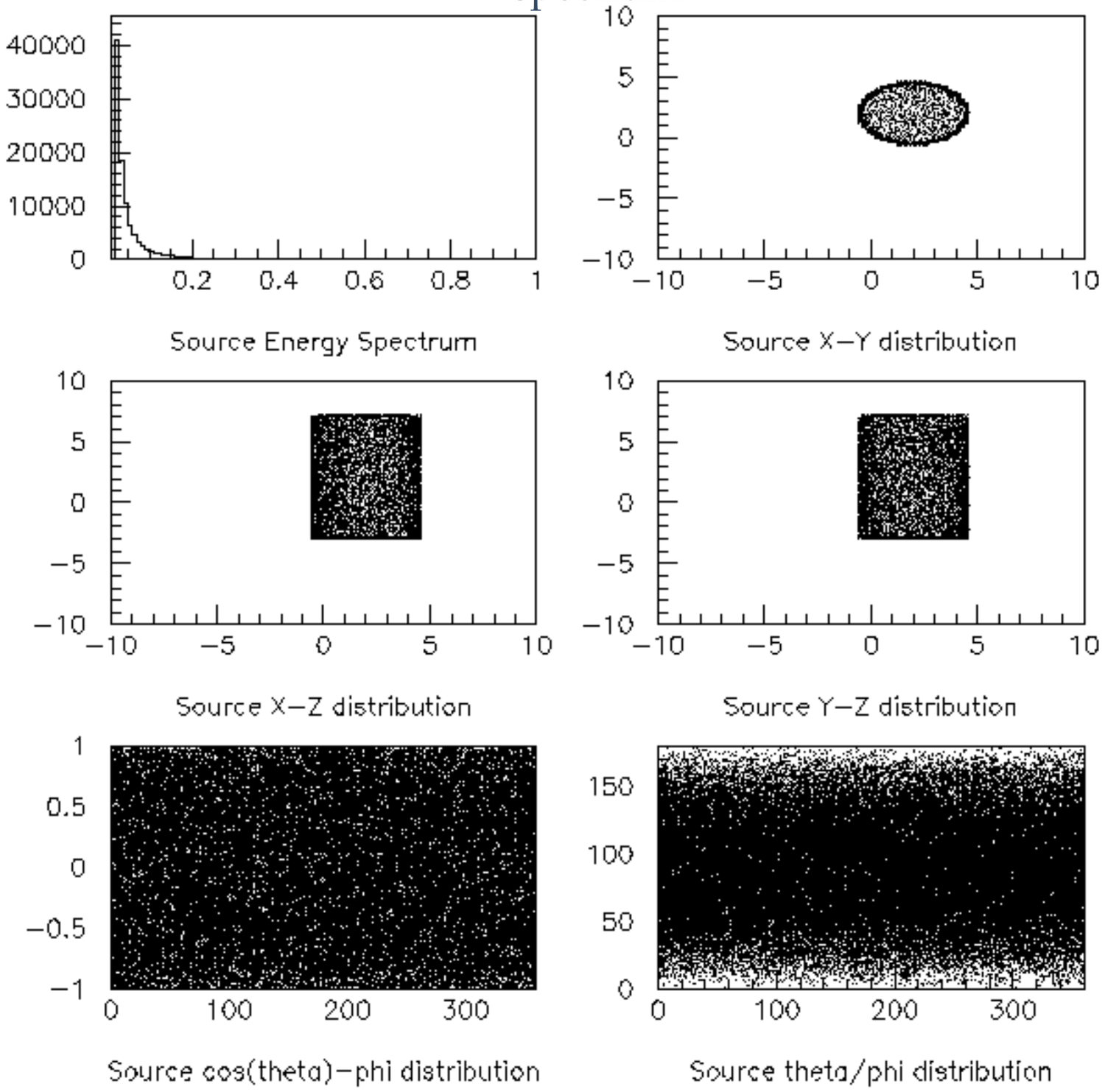
Source theta/phi distribution

Example(2)

Macro

```
/gps/particle gamma

/gps/pos/type Surface
/gps/pos/shape Cylinder
/gps/pos/centre 2. 2. 2. cm
/gps/pos/radius 2.5 cm
/gps/pos/halfz 5. cm

/gps/ang/type cos

/gps/ene/type Cdg
/gps/ene/min 20. keV
/gps/ene/max 1. MeV
/gps/ene/calculate
```

Cylindrical surface, cosine-law radiation, cosmic diffuse energy spectrum



Examples & Online Manual

- Examples also exists for GPS

examples/extended/eventgenerator/exgps

- On line manual: http://geant4.web.cern.ch/geant4/

UserDocumentation/UsersGuides/ForApplicationDeveloper/html/

ch02s07.html#sect.G4GPS.Commands

- Very nice and useful link: http://hurel.hanyang.ac.kr/Geant4/

Geant4_GPS/reat.space.qinetiq.com/gps/examples/examples.html

Particle Gun vs General Particle Source

Particle Gun

- Simple and naïve
– Shoot one track at a time
– Easy to handle.

General Particle Source

– Powerful
– Controlled by UI commands.
– Capability of shooting particles from a surface of a volume.
– Capability of randomizing kinetic energy, position and/or
direction following a user-specified distribution (histogram).

If you need to shoot primary particles from a surface of a volume, either outward or inward, GPS is the choice.
If you need a complicated distribution, not flat or simple Gaussian, GPS is the choice.
Otherwise, use Particle Gun.

From Particle Gun to General Particle Source

Example of isotropic emission in UserPrimaryGenerator code:
examples/advanced/human_phantom/src/G4HumanPhantomPrimaryGeneratorAction.cc

Gun Particle

```
G4double a,b,c;
G4double n;
do {
  a = (G4UniformRand()-0.5)/0.5;
  b = (G4UniformRand()-0.5)/0.5;
  c = (G4UniformRand()-0.5)/0.5;
  n = a*a+b*b+c*c;
} while (n > 1 || n == 0.0);
n = std::sqrt(n);
a /= n;
b /= n;
c /= n;
G4ThreeVector direction(a,b,c);
particleGun->SetParticleMomentumDirection(direction);
```

GPS

```
/gps/ang/type iso
```

Hands-on section

Task 2 - Primary Particles and User Interface

- Particle gun
- General Particle source

Antonella Tramontana

Two beam source definition, Gaussian profile plane, cosine-law direction, linear energy spectrum

Two-beam source definition
(multiple sources)

Gaussian profile

Can be focused / defocused

Macro

```
# beam #1
    # default intensity is 1,
    # now change to 5.
    /gps/source/intensity 5.

    /gps/particle proton
    /gps/pos/type Beam

    # the incident surface is
    # in the y-z plane
    /gps/pos/rot1 0 1 0
    /gps/pos/rot2 0 0 1

    # the beam spot is centered
    # at the origin and is
    # of 1d gaussian shape
    # with a 1 mm central plateau
    /gps/pos/shape Circle
    /gps/pos/centre 0. 0. 0. mm
    /gps/pos/radius 1. mm
    /gps/pos/sigma_r .2 mm

    # the beam is travelling
    # along the X_axis
    # with 5 degrees dispersion
    /gps/ang/rot1 0 0 1
    /gps/ang/rot2 0 1 0
    /gps/ang/type beam1d
    /gps/ang/sigma_r 5. deg

    # the beam energy is in
    # gaussian profile centered
    # at 400 MeV
    /gps/ene/type Gauss
    /gps/ene/mono 400 MeV
    /gps/ene/sigma 50. MeV

# beam #2
    # 2x the instensity of beam #1
    /gps/source/add 10.

    # this is a electron beam
    ...
```