Geant4 Tutorial

SciNeGHE – Trieste 2010

An hands-on course based on Geant4 with emphasis on high energy astroparticle physics.

Lectures will cover all aspects of Geant4 from basic installation through advanced topics and will be interspersed with examples that build a progressively more complex application.

The course should be of interest both to complete novices and to those who already have some basic familiarity with Geant4. Participants are expected to have a reasonable knowledge of C++.

Based on Lectures by the G4 collaboration

Generate primary event

- Derive your concrete class from G4VUserPrimaryGeneratorAction abstract base class.
- Pass a G4Event object to one or more primary generator concrete class objects which generate primary vertices and primary particles.
- Geant4 provides several generators in addition to the G4VPrimaryParticlegenerator base class.
  - G4ParticleGun
  - G4GeneralParticleSource
    - Define radioactivity

Lecture – 3

Particle Generation

Contents

- G4VUserPrimaryGeneratorAction
- Primary vertex and primary particle
- Built-in primary particle generators
  - Particle gun
  - General particle source

Primary particle generation

G4VUserPrimaryGeneratorAction

- This class is one of mandatory user classes to control the generation of primaries.
  - This class itself should NOT generate primaries but invoke **`GeneratePrimaryVertex()`** method of primary generator(s) to make primaries.
- Constructor
  - Instantiate primary generator(s)
  - Set default values to it(them)
- GeneratePrimaries() method
  - Randomize particle-by-particle value(s)
  - Set these values to primary generator(s)
    - Never use hard-coded UI commands
  - Invoke **`GeneratePrimaryVertex()`** method of primary generator(s)

Primary vertex and primary particle

Primary vertices and particles

- Primary vertices and primary particles are stored in G4Event in advance to processing an event.
  - G4PrimaryVertex and G4PrimaryParticle classes
    - These classes don't have any dependency to G4ParticleDefinition nor G4Track.
  - Capability of bookkeeping decay chains
    - Primary particles may not necessarily be particles which can be tracked by Geant4.
- Geant4 provides some concrete implementations of G4VPrimaryGenerator.
  - G4ParticleGun
  - G4GeneralParticleSource

Built-in primary particle generators

G4ParticleGun

- Concrete implementations of G4VPrimaryGenerator
  - A good example for experiment-specific primary generator implementation
- 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
  - Intercoms commands are also available for setting initial values
- One of most frequently asked questions is :

  I want “particle shotgun”, “particle machinegun”, etc.
- Instead of implementing such a fancy weapon, in your implementation of UserPrimaryGeneratorAction, you can
  - Shoot random numbers in arbitrary distribution
  - Use set methods of G4ParticleGun
  - Use G4ParticleGun as many times as you want
  - Use any other primary generators as many times as you want to make overlapping events

G4VUserPrimaryGeneratorAction

```
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

- A concrete implementation of G4VPrimaryGenerator
  - Suitable especially to space applications

```
MyPrimaryGeneratorAction::
        MyPrimaryGeneratorAction()
{ generator = new G4GeneralParticleSource; }
void MyPrimaryGeneratorAction::
        GeneratePrimaries(G4Event* anEvent)
{ generator->GeneratePrimaryVertex(anEvent); }
```

- Detailed description

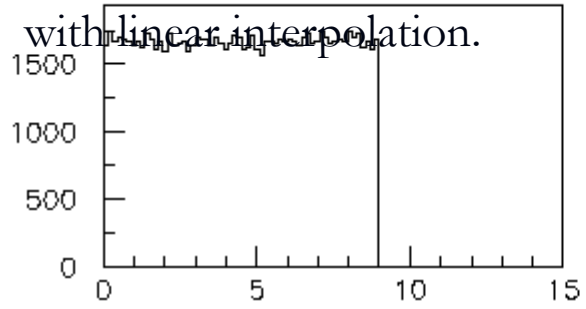
  http://reat.space.qinetiq.com/gps/

Square p

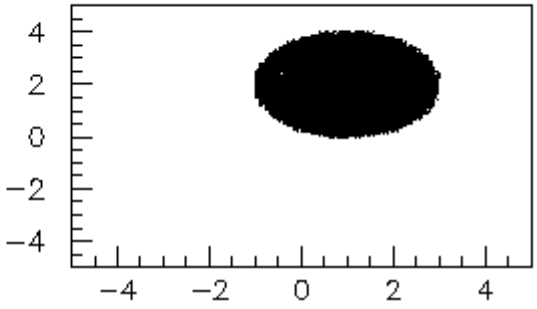
Spheric

Cylindric

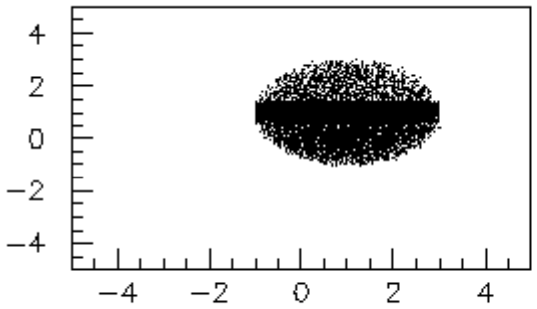
Spherical volume with z biasing, isotropic radiation with theta and phi biasing, integral arbitrary point-wise energy distribution with linear interpolation.



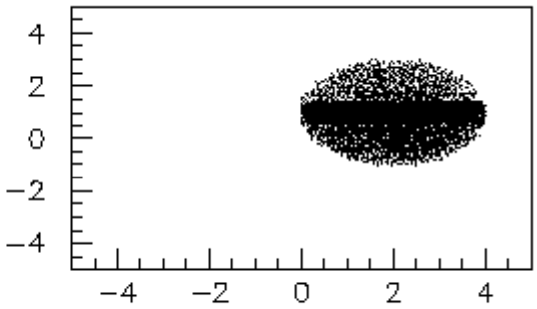
Source Energy Spectrum



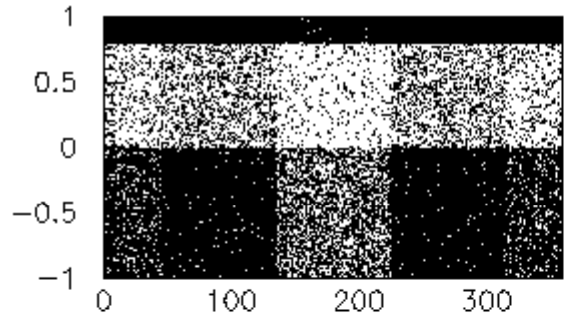
Source X–Y distribution



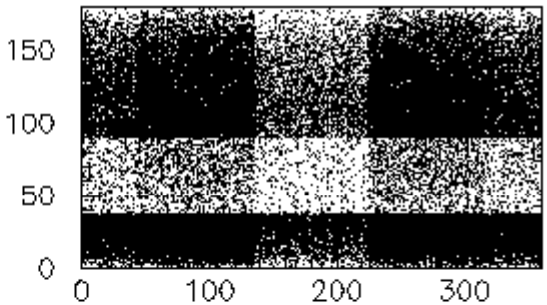
Source X–Z distribution



Source Y–Z distribution



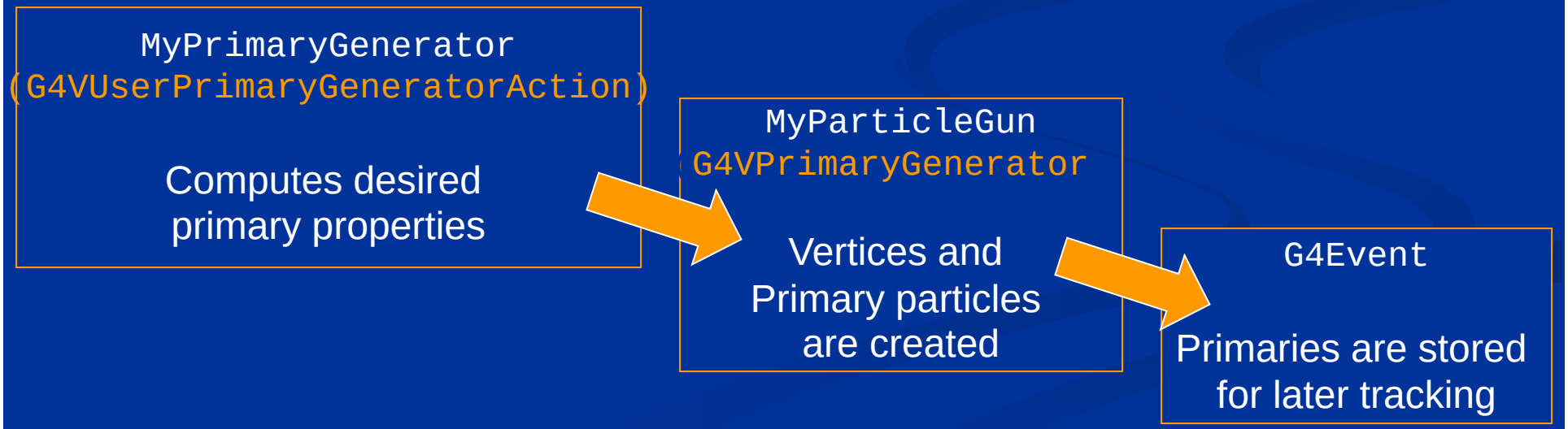
Source cos(theta)–phi distribution



Source theta/phi distribution

Primary vertices and primary particles

- Primary vertices and primary particles are stored in G4Event in advance to processing an event.
  - G4PrimaryVertex and G4PrimaryParticle classes
  - These classes don't have any dependency to G4ParticleDefinition nor G4Track.
  - They will become "primary tracks" only at Begin-of-Event phase and put into a "stack"



G4ParticleGun

- Concrete implementations of G4VPrimaryGenerator
  - A good example for experiment-specific primary generator implementation
- It shoots one primary particle of a certain energy from a certain point at a certain time to a certain direction.
  - Various C++ set methods are available
- Intercoms commands are also available for setting initial values

/gun/List	List available particles
/gun/particle	Set particle to be generated
/gun/direction	Set momentum direction
/gun/energy	Set kinetic energy
/gun/momentum	Set momentum
/gun/momentumAmp	Set absolute value of momentum
/gun/position	Set starting position of the particle
/gun/time	Set initial time of the particle
/gun/polarization	Set polarization
/gun/number	Set number of particles to be generated (per event)
/gun/ion	Set properties of ion to be generated [usage] /gun/ion Z A Q

Complex sources with G4ParticleGun (1)

- G4ParticleGun is basic, but it can be used from inside UserPrimaryGeneratorAction to model complex source types / distributions:
  - Generate the desired distributions (by shooting random numbers)
  - Use (C++) set methods of G4ParticleGun
  - Use G4ParticleGun as many times as you want
  - Use any other primary generators as many times as you want to make overlapping events

Complex sources with G4ParticleGun (2)

- Example of user PrimaryGeneratorAction using G4ParticleGun

```
void T01PrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent){
  G4ParticleDefinition* particle;
  G4int i = (int)(5.*G4UniformRand());
  switch(i){
    case 0:
      particle = positron;
      break;
    case 1:
      ...
  }
  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 (GPS)

- An advanced concrete implementation of G4VPrimaryGenerator
- Offers as pre-defined many common (and not so common) options
  - Position, angular and energy distributions
  - Multiple sources, with user defined relative intensity
- Capability of event biasing (variance reduction).
- All features can be used via C++ or command line (or macro) UI

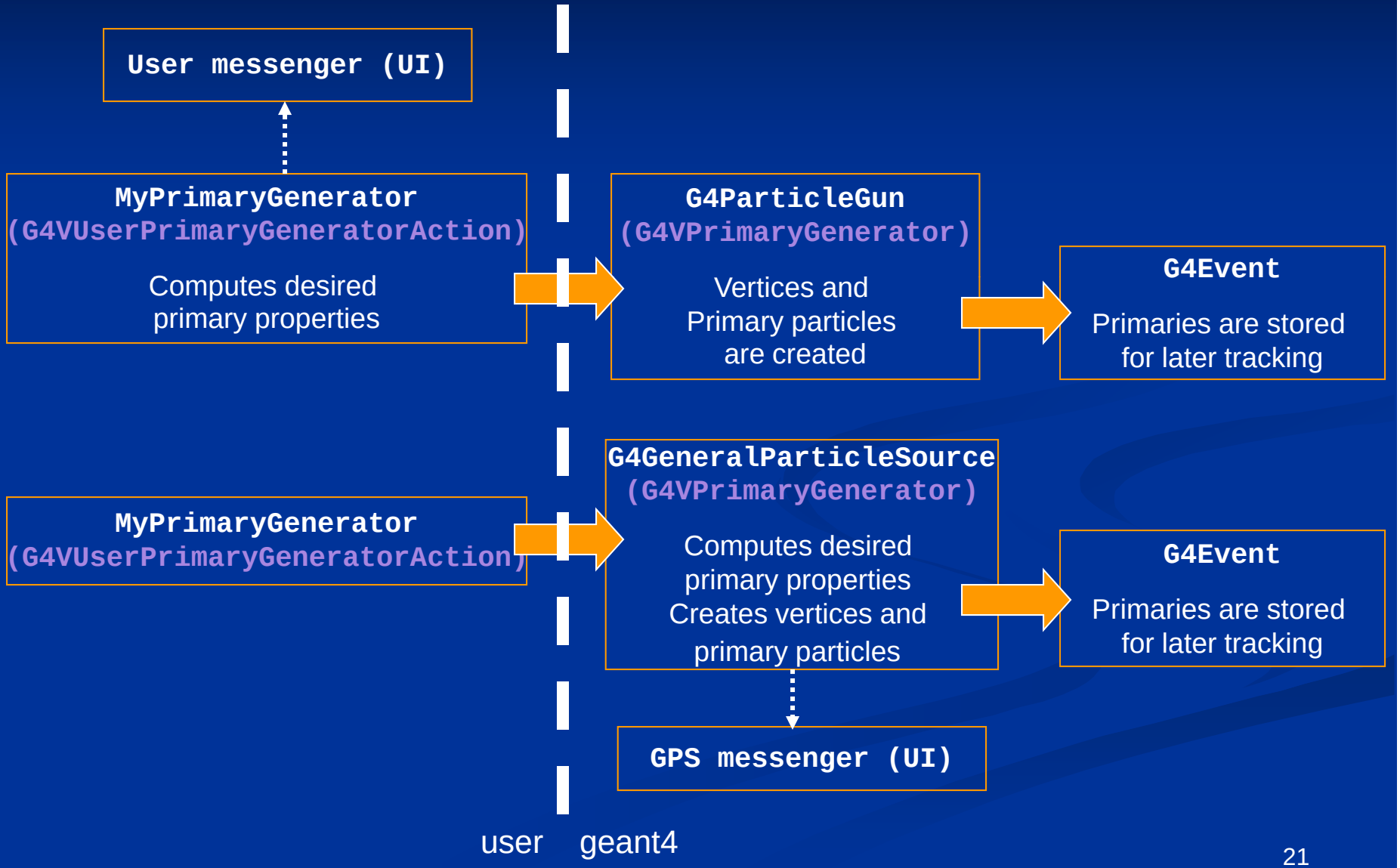
What is GPS?

- The General Particle Source (GPS) offers as pre-defined many common options for particle generation (energy, angular and spatial distributions)
  - GPS is a concrete implementation of G4VPrimaryGenerator (as G4ParticleGun but more advanced)
  - G4 class name: G4GeneralParticleSource (in the event category)
- User cases: space radiation environment, medical physics, accelerator (fixed target)
- First development (2000) University of Southampton (ESA contract), maintained and upgraded now mainly by QinetiQ and ESA
- Extensive up-to-date documentation at http://reat.space.qinetiq.com/gps

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
- Capability of event biasing (variance reduction).
  - By enhancing particle type, distribution of vertex point, energy and/or direction

GPS vs G4ParticleGun



UserPrimaryGeneratorAction class

- Can be extremely simple:

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

MyPrimaryGeneratorAction::~MyPrimaryGeneratorAction() {
   delete m_particleGun;
}

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

- All user instructions given via macro UI commands

GPS control: scripting UI

- All features can be used via C++ or command line (or macro) UI
- Example of isotropic emission in UserPrimaryGenerator C++ code:

examples/advanced/human_phantom/src/G4HumanPhantomPrimaryGeneratorAction.cc

```
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);
```

- Equivalent GPS (script)

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

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
  ```

Position distributions /gps/pos/… (2)

- Some shared UI commands

- **/gps/pos/centre**
- **/gps/pos/halfx | y | z**
- **/gps/pos/radius**
- **/gps/pos/inner_radius**
- **/gps/pos/sigmar**
- **/gps/pos/sigmax | y**
- **/gps/pos/rot1**
- **/gps/pos/rot2**

- When usinig Volume type, one can limit the emission from within a certain volume in the “mass” geometry

  /gps/pos/confine your_physical_volume_name

Angular distributions `/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`**)

GPS
Example 7

- Vertex on cylinder surface
- Cosine-law emission
  (to mimic isotropic source in space)
- Pre-defined spectrum
  (Cosmic Diffuse Gamma)

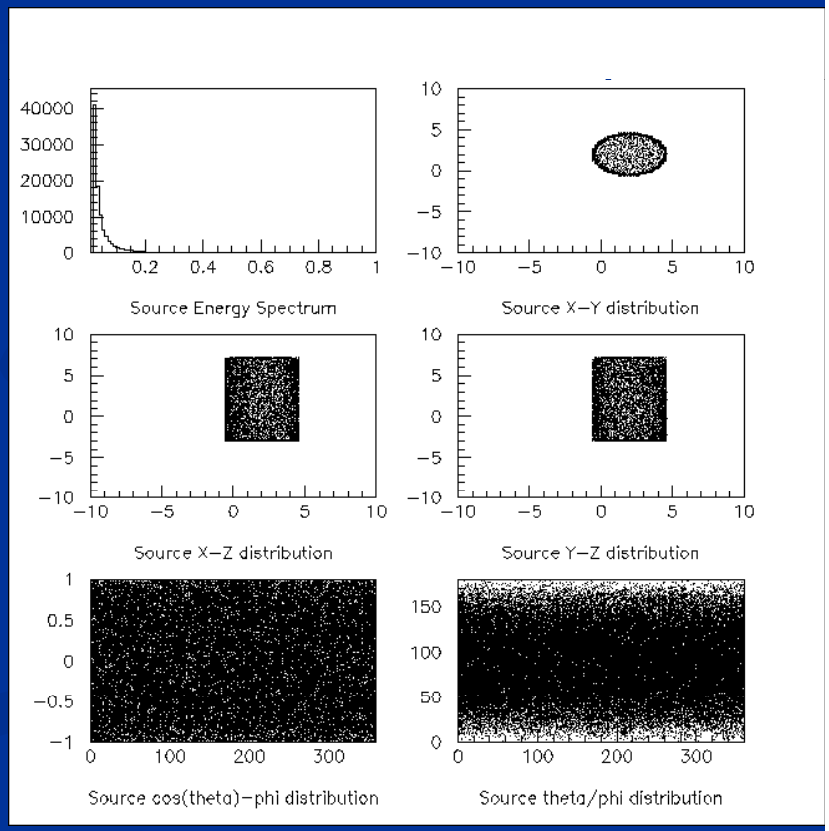
- Macro

```
/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
```



GPS
Example 24

- Vertex in sphere volume
  with z biasing
- Isotropic radiation
  with theta and phi biasing
- Integral arbitrary point-wise energy distribution with linear interpolation.

- Macro

```
/gps/particle geantino
/gps/pos/type Volume
/gps/pos/shape Sphere
/gps/pos/centre 1. 2. 1. cm
/gps/pos/radius 2. Cm

/gps/ang/type iso

/gps/ene/type Arb
/gps/ene/diffspec 0
/gps/hist/type arb
/gps/hist/point 0.0 11.
/gps/hist/point 1.0 10.
/gps/hist/point 2.0 9.
/gps/hist/point 3.0 8.
/gps/hist/point 4.0 7.
/gps/hist/point 7.0 4.
/gps/hist/point 8.0 3.
/gps/hist/point 9.0 2.
/gps/hist/point 10.0 1.
/gps/hist/point 11.0 0.
/gps/hist/inter Lin

/gps/hist/type biasz
/gps/hist/point 0. 0.
/gps/hist/point 0.4 0.5
/gps/hist/point 0.6 1.
/gps/hist/point 1. 0.2

/gps/hist/type biast
/gps/hist/point 0. 0.
/gps/hist/point 0.1 1.
/gps/hist/point 0.5 0.1
/gps/hist/point 1. 1.

/gps/hist/type biasp
/gps/hist/point 0. 0.
/gps/hist/point 0.125 1.
/gps/hist/point 0.375 4.
/gps/hist/point 0.625 1.
/gps/hist/point 0.875 4.
/gps/hist/point 1. 1.
```