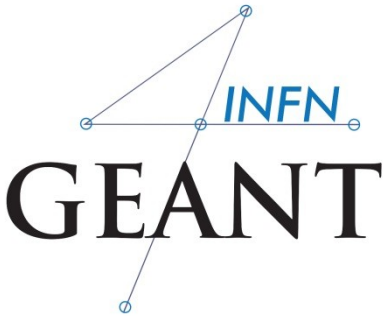




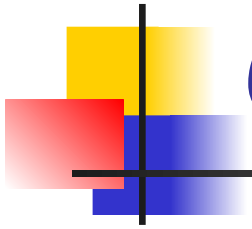



# *Generation of a primary event*

*Alexei Sytov*

*INFN – Ferrara Division*

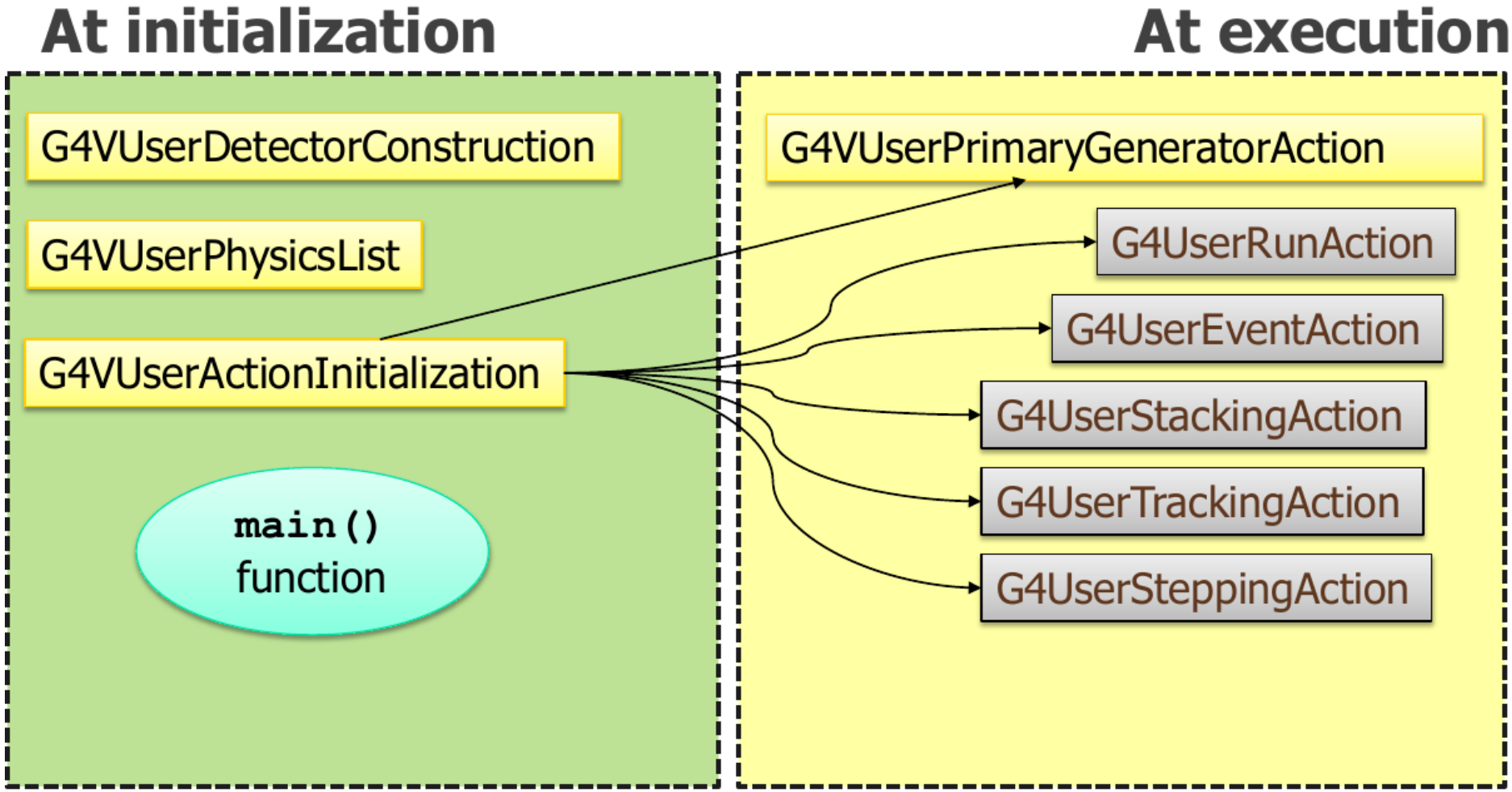
*Geant4 Course*
*XXII Seminar on software for*
*nuclear, subuclear and applied physics*
*Alghero, June 8th- 13th, 2024*



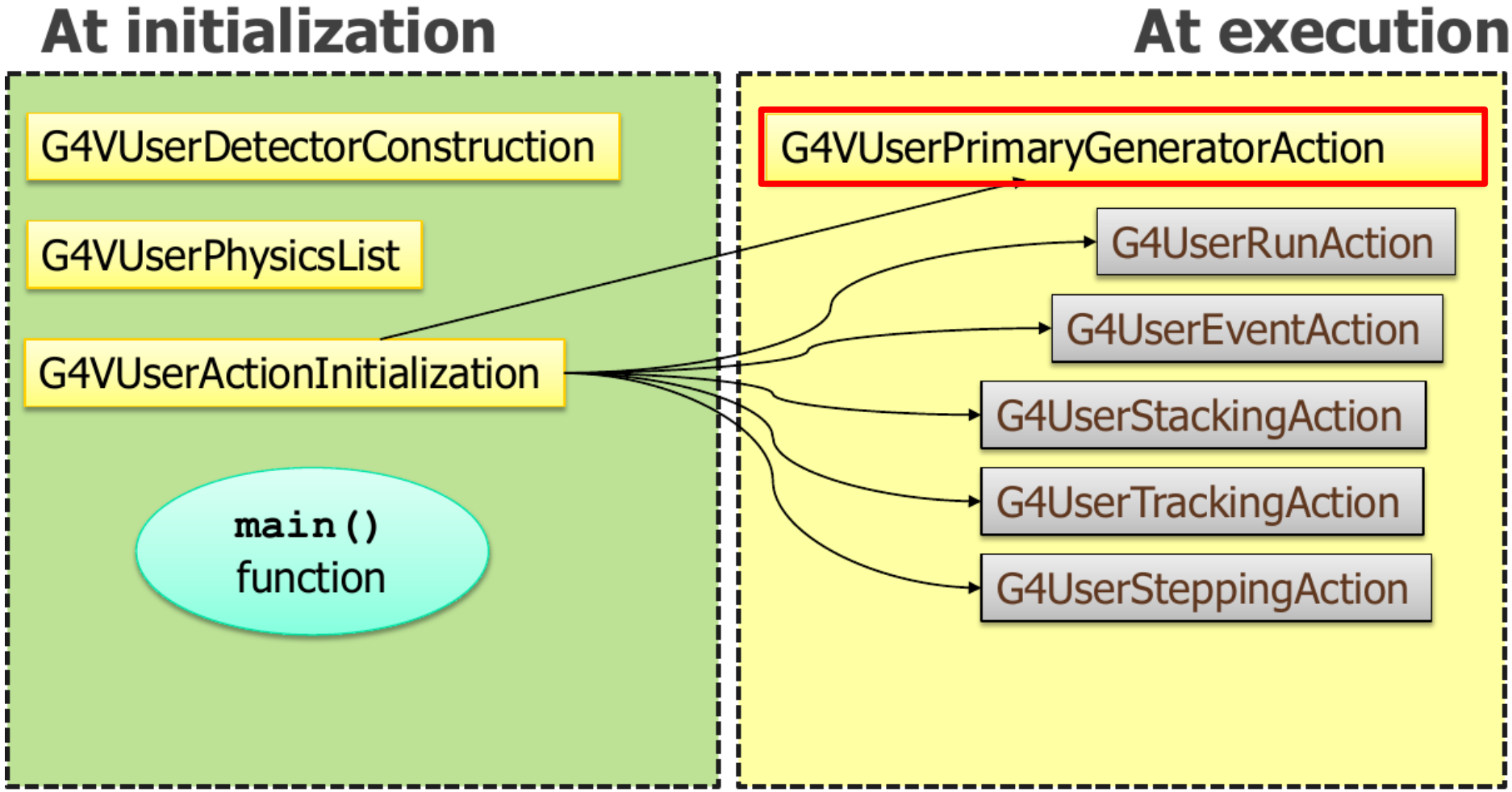
# *Outline*

- *VUserPrimaryGeneratorAction and PrimaryGeneratorActionG4*
- *Particle gun or GPS?*
- *The particle gun*
- *General Particle Source (or GPS)*

# User Classes



# User Classes



# PrimaryGeneratorAction inherits G4VUserPrimaryGeneratorAction

```
1  #ifndef PRIMARY_GENERATOR_ACTION_HH
2  #define PRIMARY_GENERATOR_ACTION_HH
3
4  #include <G4VUserPrimaryGeneratorAction.hh>
5
6  class G4ParticleGun;
7  //class G4GeneralParticleSource;
8  class G4Event;
9
10 class PrimaryGeneratorAction : public G4VUserPrimaryGeneratorAction
11 {
12   public:
13     PrimaryGeneratorAction();
14     ~PrimaryGeneratorAction() override;
15
16     // method from the base class
17     void GeneratePrimaries(G4Event*) override;
18
19     // method to access particle gun
20     //const G4ParticleGun* GetParticleGun() const { return fParticleGun; }
21
22   private:
23     G4ParticleGun* fGun{nullptr};
24   //G4GeneralParticleSource* fGPS{nullptr};
25 };
26
27 //....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo
28
29 #endif
30
```

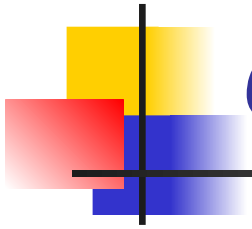
*Alternative to #include; works if you need only pointers; faster compilation*

*key function in which an event is generated*

*constructor and destructor*

*override: a good practice to let the compiler know that you override virtual methods, otherwise silent errors may appear*

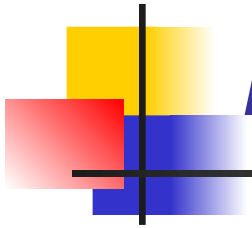
*Declaration of Geant4 particle gun or general particle source (depends what you prefer)*



# 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:*
  - ***Has `GeneratePrimaries()` method** using either G4ParticleGun or G4GeneralParticleSource*
  - *It registers the primary particle(s) to the **`G4Event`***

***It is possible to attach several primaries to the same event.***



# ParticleGun vs. GPS

- Both
  - Derive from G4VPrimaryGenerator class
  - Possess GeneratePrimaryVertex(G4Event*) method to generate the primary particles
- G4ParticleGun
  - Suitable for hardcoded particle distribution within PrimaryGeneratorAction
- G4GeneralParticleSource (GPS)
  - Suitable for usage of standard macro commands (no hardcoding)

# PrimaryGeneratorAction

```
PrimaryGeneratorAction::PrimaryGeneratorAction()
{
    // Task 2b.1: Comment out the particle gun creation and instatiate a GPS instead
    fGun = new G4ParticleGun();
    //fGPS = new G4GeneralParticleSource();
```

*Create G4VPrimaryGenerator (ParticleGun or GPS)*

*Here you may setup your ParticleGun (see exercises)*

```
PrimaryGeneratorAction::~PrimaryGeneratorAction()
{
    // Task 2b.2: Delete the GPS instead of the gun
    delete fGun;
    //delete fGPS;
}
```

*delete G4VPrimaryGenerator (ParticleGun or GPS)*

*GPS is set up from macro*

```
void PrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
{
  //...

  fGun->GeneratePrimaryVertex(anEvent);
  //fGPS->GeneratePrimaryVertex(anEvent);
}
```

*Here you may setup the particle distribution and use it in ParticleGun (see exercise)*

*Generate your primary*

# 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);
```

# A "real-life" myPrimaryGenerator: constructor & destructor

```
myPrimaryGenerator::myPrimaryGenerator ()
: G4VUserPrimaryGeneratorAction(), fParticleGun(0)
{
  fParticleGun  = new G4ParticleGun();

  // set defaults
  fParticleGun->SetParticleDefinition(
       G4Gamma::Definition());
  fParticleGun->
SetParticleMomentumDirection(G4ThreeVector(0.,0.,1.));
  fParticleGun->SetParticleEnergy(6.*MeV);
}
myPrimaryGenerator::~myPrimaryGenerator ()
{
  delete fParticleGun;
}
```

*Instantiate concrete generator*

*Clean it up in the destructor*

# A "real-life" myPrimaryGenerator: GeneratePrimaries(G4Event*)

```
myPrimaryGenerator::GeneratePrimaries(G4Event* evt)
{
  // Randomize event-per-event
  G4double cosT = -1.0 + G4UniformRand()*2.0;
  G4double phi = G4UniformRand()*twopi;

  G4double sinT = sqrt(1-cosT*cosT);
  G4ThreeVector direction(sinT*sin(phi),sinT*cos(phi),cosT);

  G4double ene = G4UniformRand()*6*MeV;

  fParticleGun->SetParticleDirection(direction);
  fParticleGun->SetParticleEnergy(ene);

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

*Sample direction isotropically*

*Sample energy (flat distr.)*

*Shoot event*

# G4ParticleGun

- *Commands can be also given interactively by* ***user interface***
  - *But cannot do randomization in this case*
- *Allows to change primary parameters* ***between*** *one run and an other*
  - *Notice: parameters from the UI could be overwritten in* ***`GeneratePrimaries()`***

```
/gun/energy 10 MeV
/gun/particle mu+
/gun/direction 0 0 -1
/run/beamOn 100
/gun/particle ion
/gun/ion 55 137
/gun/position 10 10 -100 cm
/run/beamOn 100
```

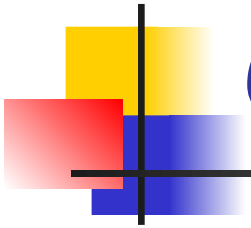
*Change settings*

*Start first run*

*Change settings*

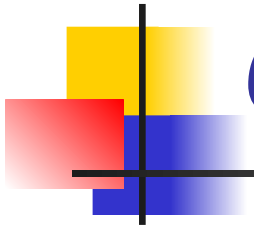
*Generate* $^{137}Cs$

*Start second run*



# G4GeneralParticleSource()

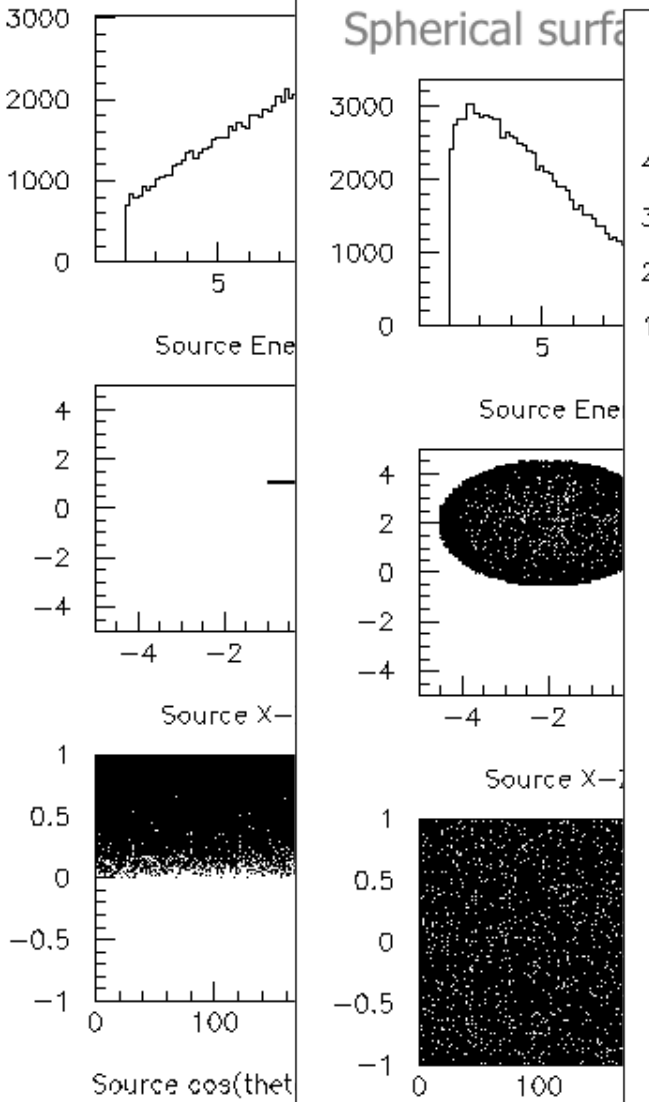
- **`source/event/include/G4GeneralParticleSource.hh`**
- *Concrete implementation of* **`G4VPrimaryGenerator`**
  **`class G4GeneralParticleSource : public G4VPrimaryGenerator`**
- *Is designed to replace the G4ParticleGun class*
- *It is designed to allow specification of multiple particle sources 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, or within a volume*
  - *Momentum direction and kinetic energy of the primary particle can also be randomized*
- *Distribution defined by* ***UI commands***



# G4GeneralParticleSource

- *On line manual:*
  - *Section 2.7 of the Geant4 Application Developer Manual*
- */gps main commands*
  - ***/gps/pos/type*** *(planar, point, etc.)*
  - ***/gps/ang/type*** *(iso, planar wave, etc.)*
  - ***/gps/energy/type*** *(monoenergetic, linear, User defined)*
  - *……*

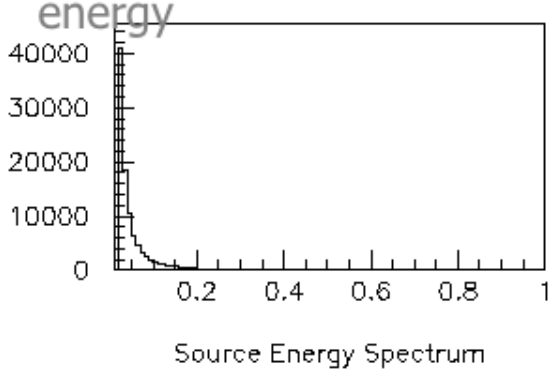
Square plane

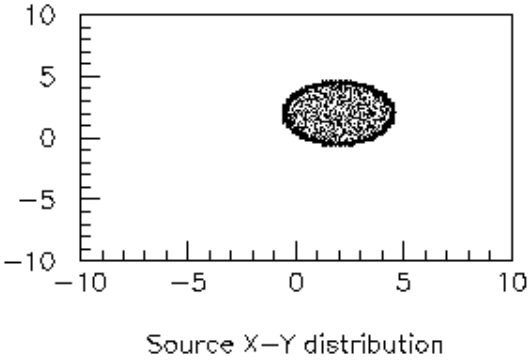


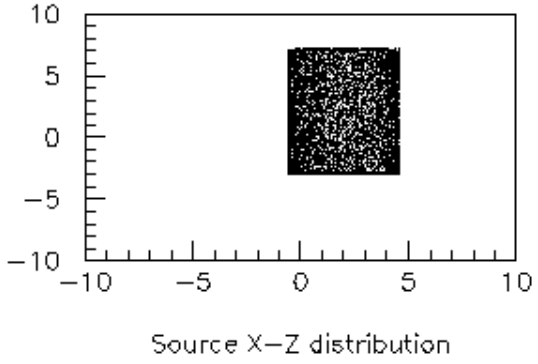

Spherical surfa

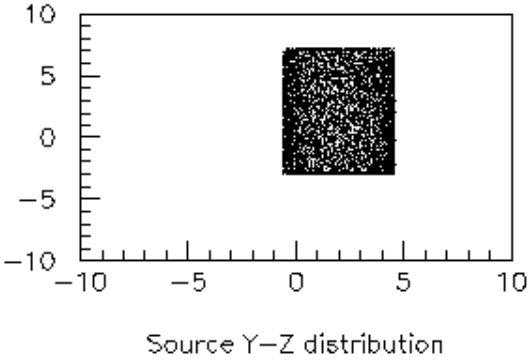


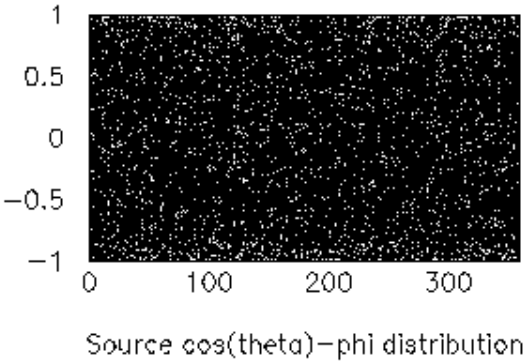

## Cylindrical surface, cosine-law radiation, Cosmic diffuse energy

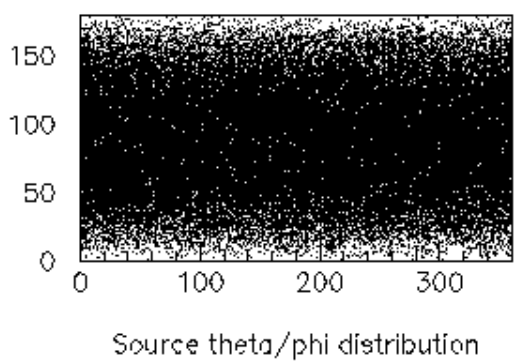


# GPS documentation



## 2.7. Geant4 General Particle Source

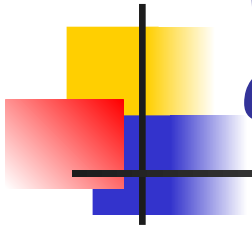
### 2.7.1. Introduction

The `G4GeneralParticleSource` (GPS) is part of the Geant4 toolkit for Monte-Carlo, high-energy particle transport. Specifically, it allows the specifications of the spectral, spatial and angular distribution of the primary source particles. An overview of the GPS class structure is presented here. Section 2.7.2 covers the configuration of GPS for a user application, and Section 2.7.3 describes the macro command interface. Section 2.7.4 gives an example input file to guide the first time user.

| Spectrum | Abbreviation | Functional Form | User Parameters |
|---|---|---|---|
| mono-energetic | Mono | $I \propto \delta(E-E_0)$ | Energy $E_0$ |
| linear | Lin | $I \propto I_0 + m \times E$ | |
| exponential | Exp | $I \propto \exp(-E/E_0)$ | |
| power-law | Pow | $I \propto E^{\alpha}$ | |
| Gaussian | Gauss | $I = (2\pi\sigma)^{-½} \exp[-(E-E_0)^2 / \sigma^2]$ | |
| bremsstrahlung | Brem | $I = \int 2E^2 [ h^2c^2 (\exp(-E/kT$ $1)]^{-1}$ | |
| black body | Bbody | $I \propto (kT)^{-½} E \exp(-E/kT)$ | |
| cosmic diffuse gamma ray | Cdg | $I \propto [ (E/E_b)^{\alpha 1} + (E/E_b)^{\alpha 2} ]$ | |

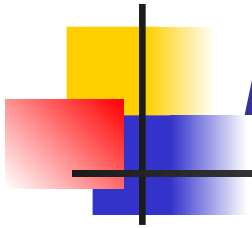
#### 2.7.3.3. Source position and structure

| Command | Arguments | Description and restrictions |
|---|---|---|
| /gps/pos/type | dist | Sets the source positional distribution type: *Point* [default], *Plane, Beam, Surface, Volume*. |
| /gps/pos/shape | shape | Sets the source shape type, after `/gps/pos/type` has been used. For a Plane this can be *Circle, Annulus, Ellipse, Square, Rectangle*. For both Surface or Volume sources this can be *Sphere, Ellipsoid, Cylinder, Para* (parallelpiped). |
| /gps/pos/centre | X Y Z unit | Sets the centre co-ordinates (X,Y,Z) of the source [default (0,0,0) cm]. The units can only be micron, mm, cm, m or km. |
| /gps/pos/rot1 | $R1_x$ $R1_y$ $R1_z$ | Defines the first (x' direction) vector R1 [default (1,0,0)], which does not need to be a unit vector, and is used together with `/gps/pos/rot2` to create the rotation matrix of the shape defined with `/gps/shape`. |
| /gps/pos/rot2 | $R2_x$ $R2_y$ $R2_z$ | Defines the second vector R2 in the xy plane [default (0,1,0)], which does not need to be a unit vector, and is used tohgether with `/gps/pos/rot1` to create the rotation matrix of the shape defined with `/gps/shape`. |
| /gps/pos/halfx | len unit | Sets the half-length in x [default 0 cm] of the source. The units can only be micron, mm, cm, m or km. |



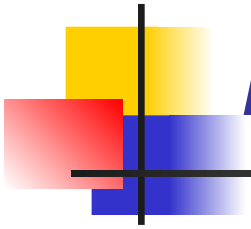
# *When do you need your own derived class of* ***G4VPrimaryGenerator***

- *In some cases, what it provided by Geant4* ***does not fit*** *specific needs: need to write a derived class from* ***G4VPrimaryGenerator***
  - *Must implement the virtual method* ***GeneratePrimaryVertex(G4Event* evt)***
  - *Generate vertices* ***(G4PrimaryVertex)*** *and attach particles to each of them* ***(G4PrimaryParticle)***
  - *Add vertices to the event* ***evt->AddPrimaryVertex()***
- *Needed when:*
  - *You need to interface to a non-HEPEvt* ***external generator***
    - *neutrino interaction, Higgs decay, non-standard interactions*
  - *Many particles from one vertex, or many vertices*
    - *double beta decay*
  - *Time difference between primary tracks*



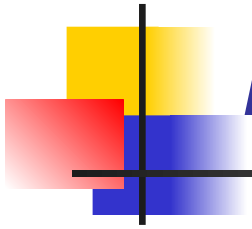
# Examples

- **`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`**



# Bonus: 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 the interaction point must be still set by the user

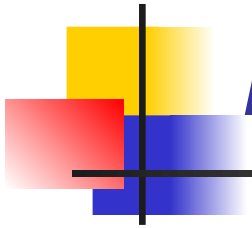


# *Bonus 2: G4Mutex*

- *Reading from (as well as writing into) the same file is not safe in multithreading!*
- *Solution: use **G4Mutex***
- ***G4Mutex** allows you to lock specific lines of your code in a sequential mode, so you threads will not conflict:*

```
namespace { G4Mutex stuffMutex =
G4MUTEX_INITIALIZER; } //in beginning of your class

stuffMutex.lock();
// … your code in a sequential mode
stuffMutex.unlock();
```



# *Hands-on session*

- *Task2*
  - *G4ParticleGun and Geant4 GPS*

- ***http://geant4.lngs.infn.it/alghero2025/task2***