

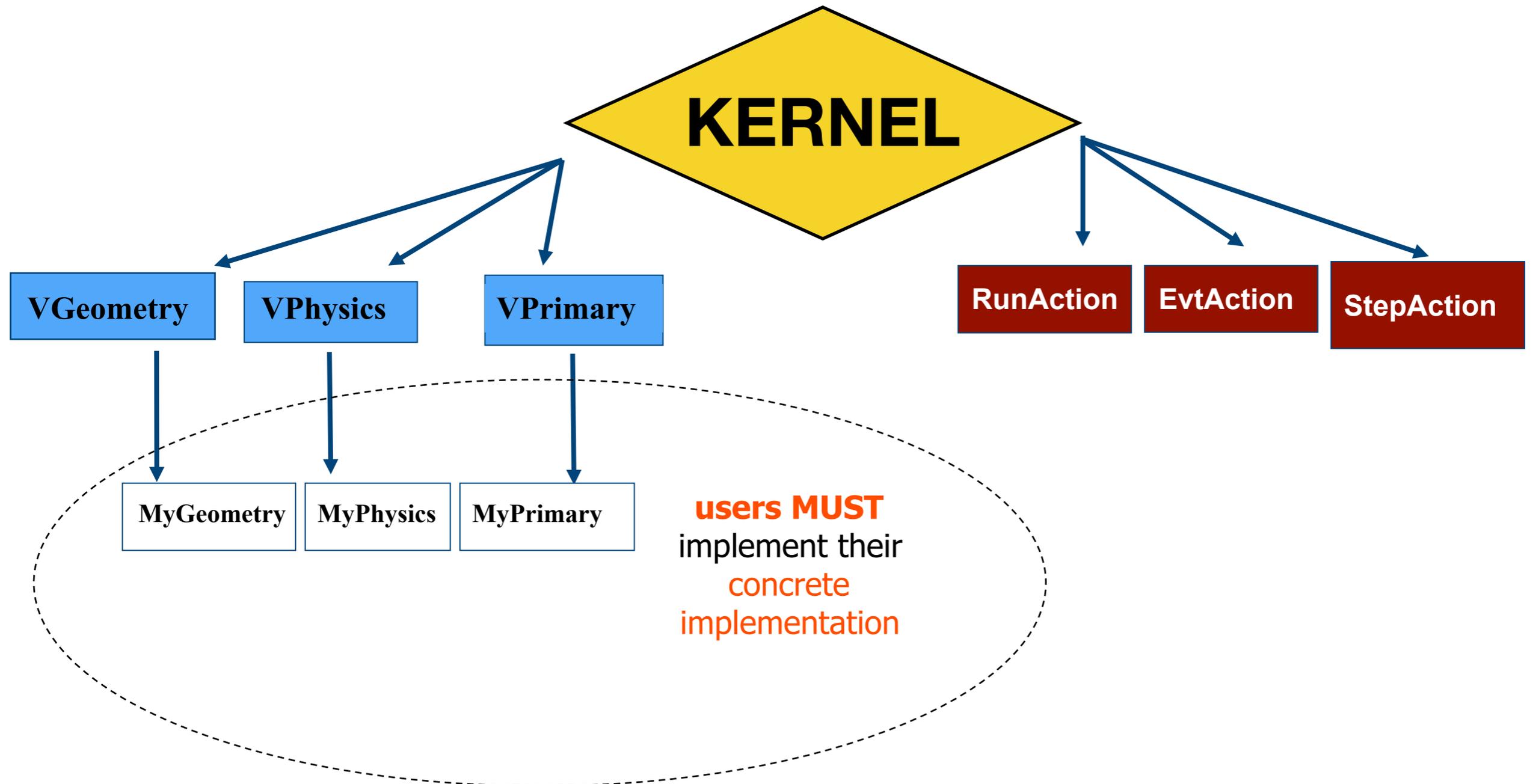
Generation of a primary event



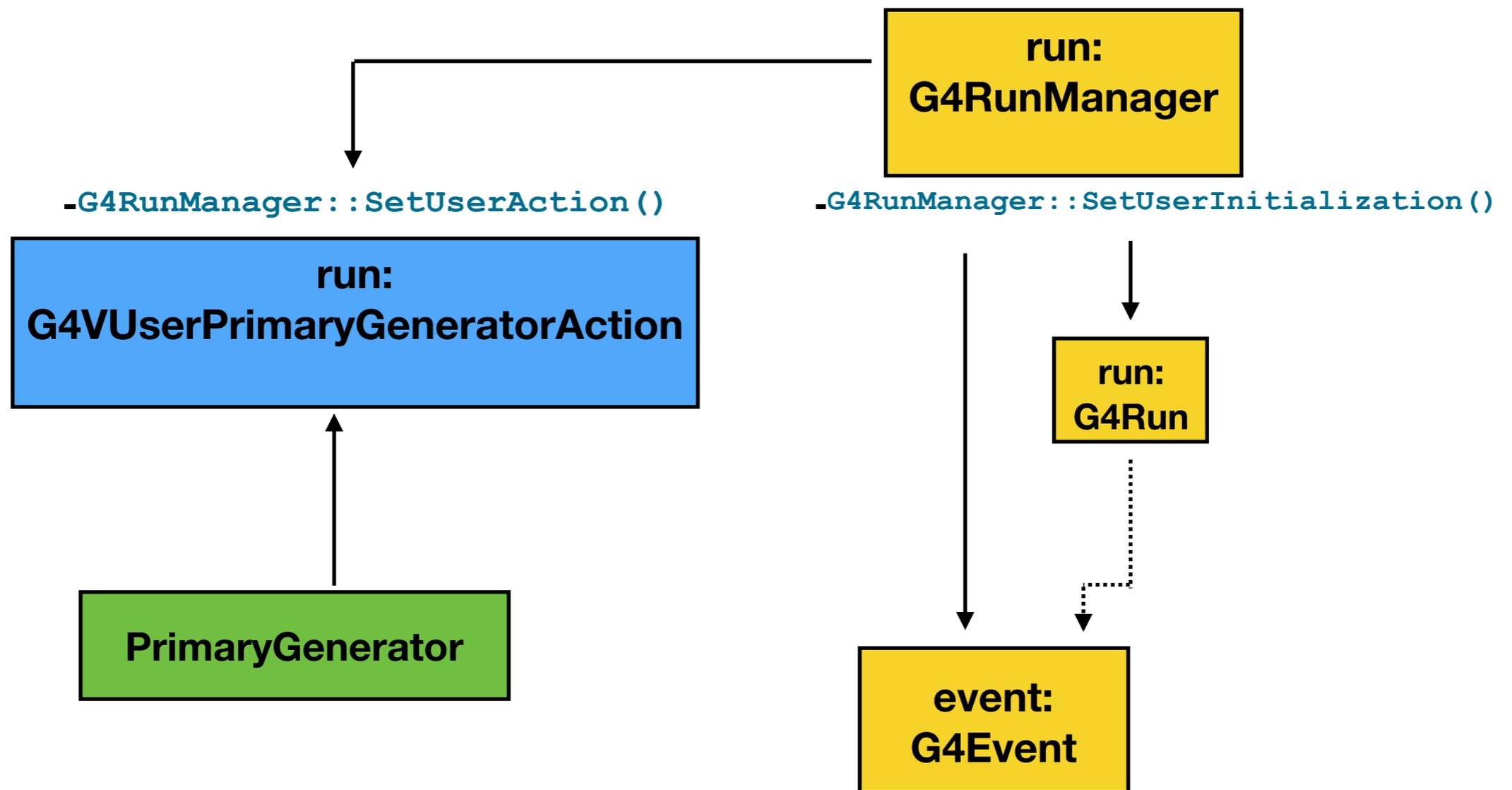
Giada Petringa (LNS-INFN)

XIV Seminar on Software for Nuclear, Subnuclear and Applied Physics
June 5, 2017 - June 9, 2017 Alghero (Italy)

Logical structure of a Geant4 application

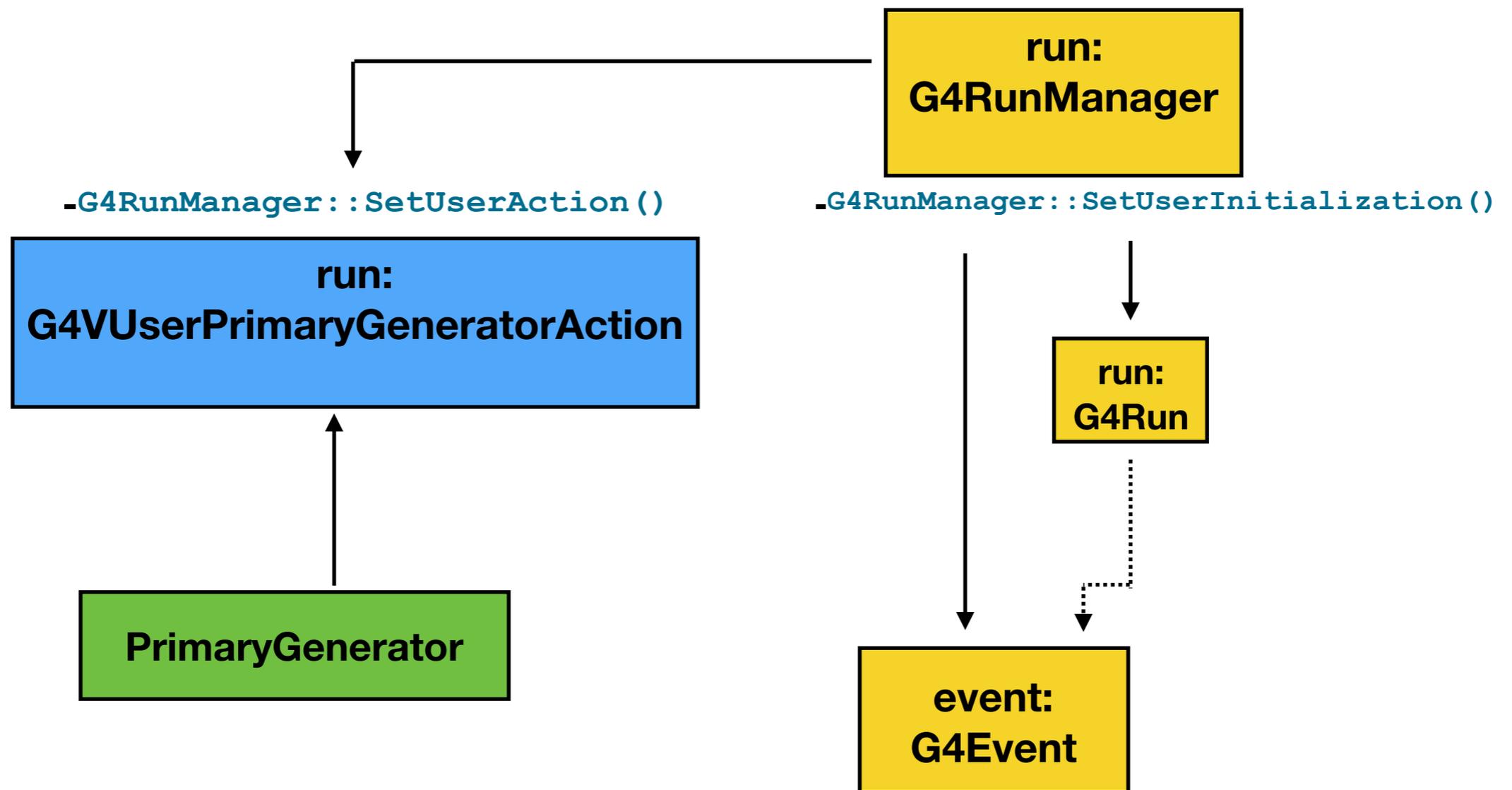


Logical structure of a Geant4 application



Logical structure of a Geant4 application

- **GeneratePrimaries()** method





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





G4VUserPrimaryGeneratorAction

Where? → geant4.10.3.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
```



.... 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 destructor



.... 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



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



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 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**



G4GeneralParticleSource()

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

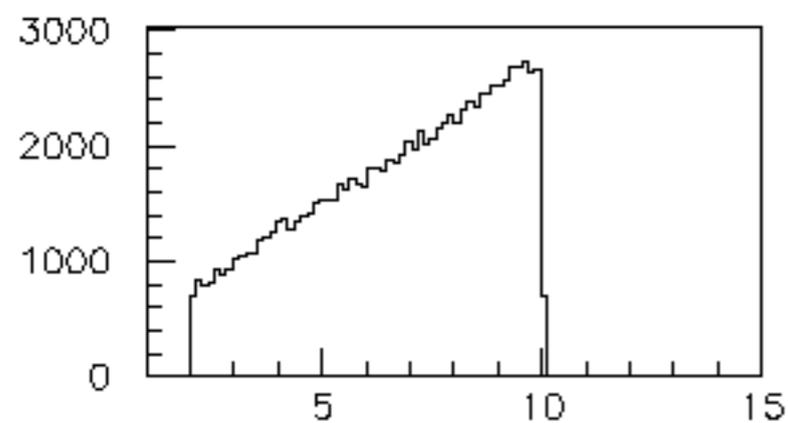
- **../source/event/include/G4GeneralParticleSource.hh**
- **Concrete implementation of G4VPrimaryGenerator**
class G4GeneralParticleSource : public G4VPrimaryGenerator
- **Is designed to replace the G4ParticleGun class**
- **It is designed to allow specification of multiple particle sources each with 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**



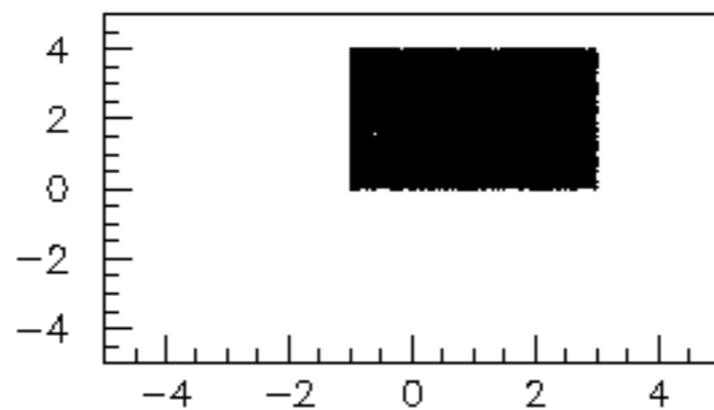
G4GeneralParticleSource

- **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)**
 - **.....**

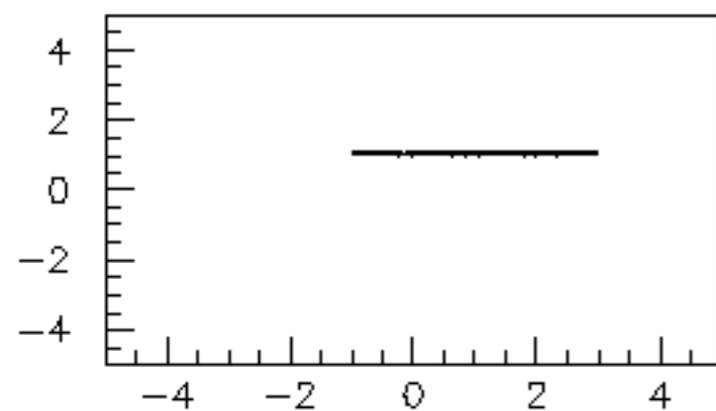
Square plane, cosine-law direction, linear energy



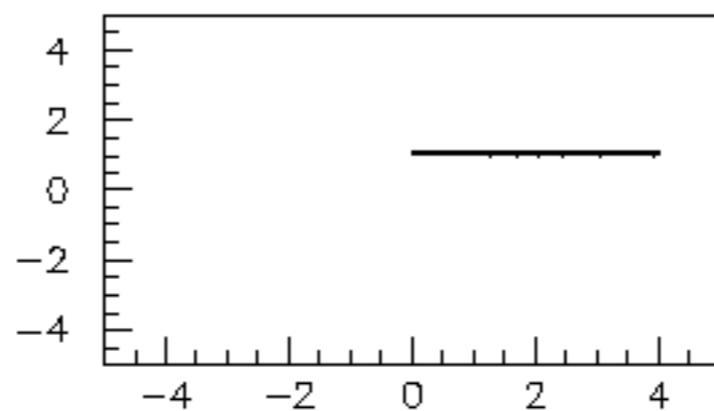
Source Energy Spectrum



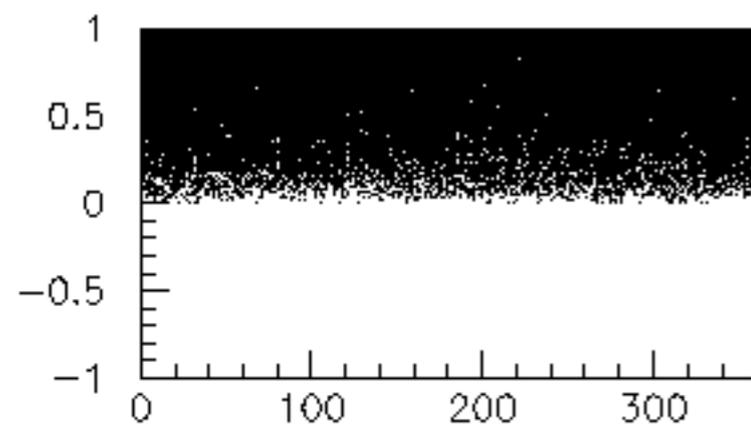
Source X-Y distribution



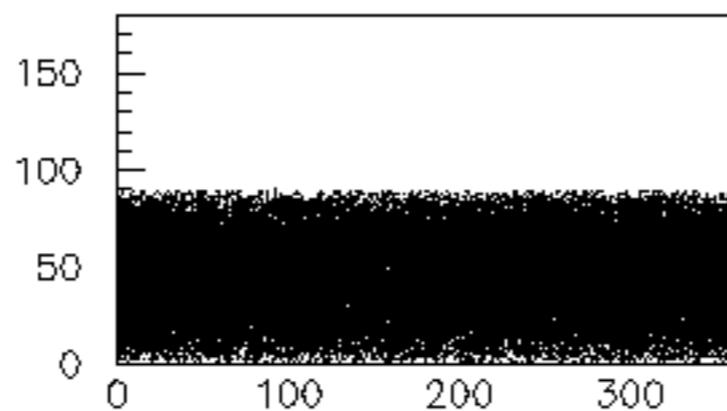
Source X-Z distribution



Source Y-Z distribution

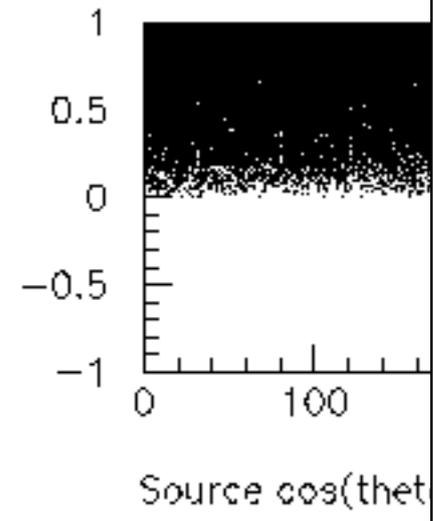
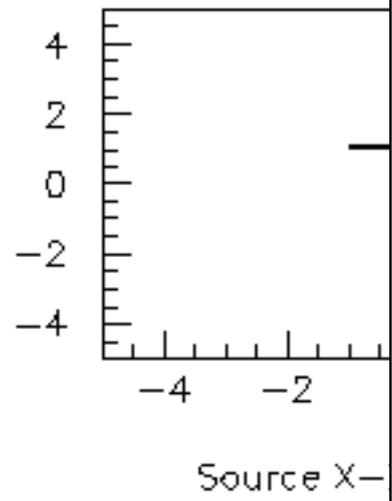
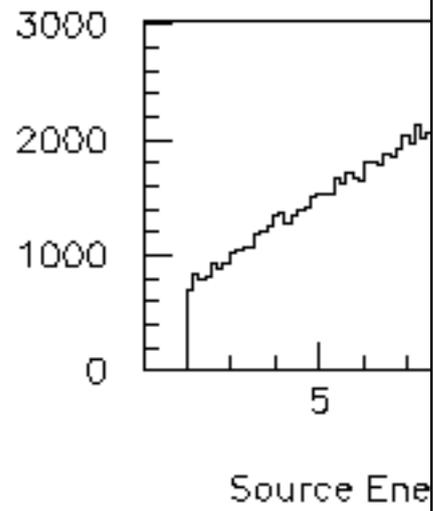


Source $\cos(\theta)$ - ϕ distribution

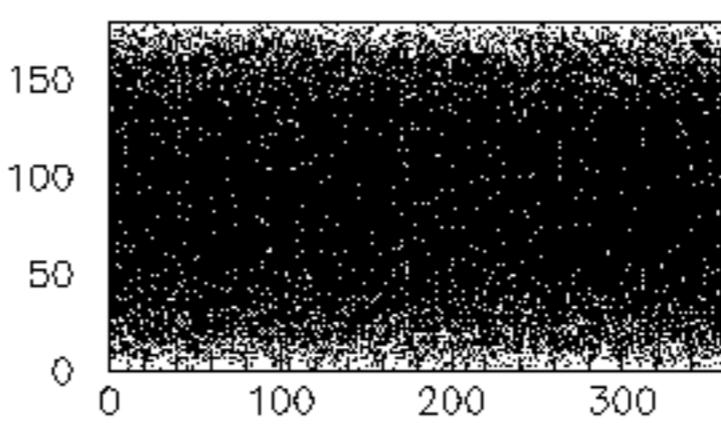
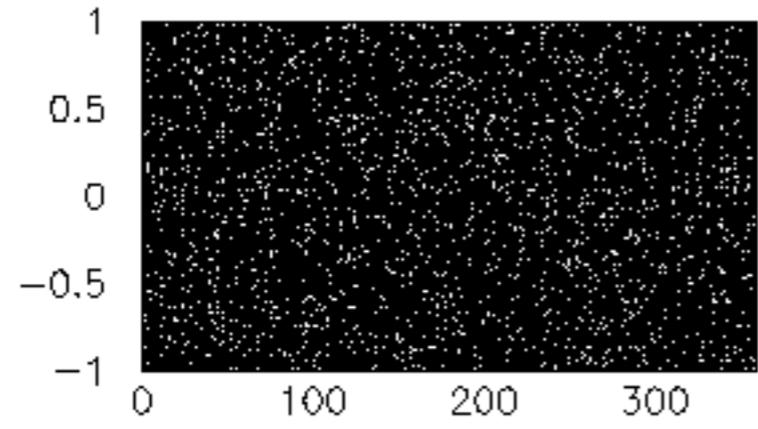
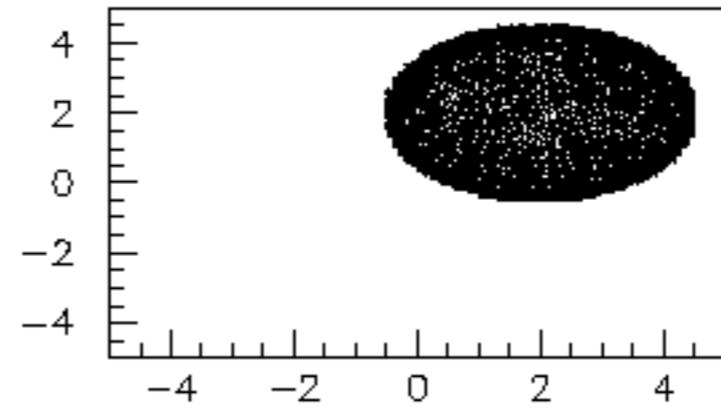
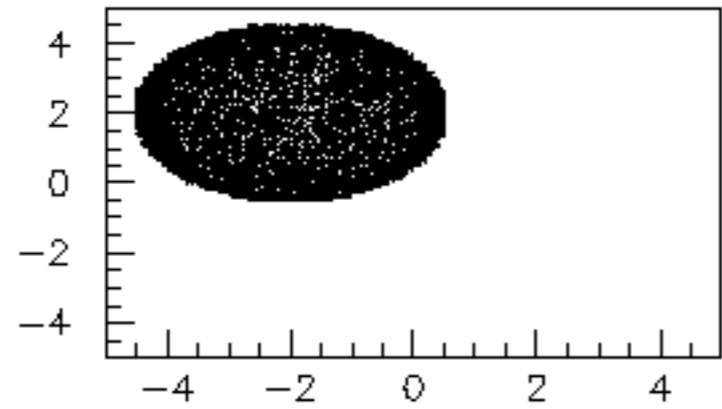
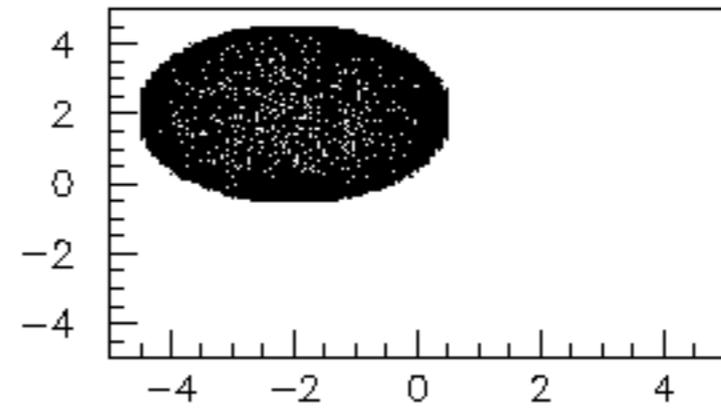
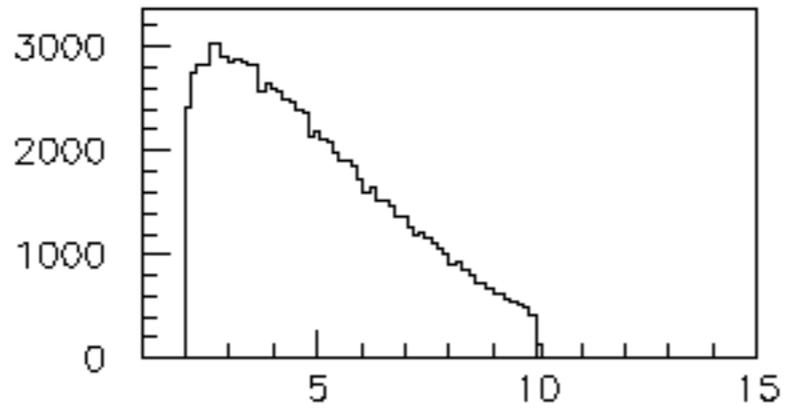


Source θ/ϕ distribution

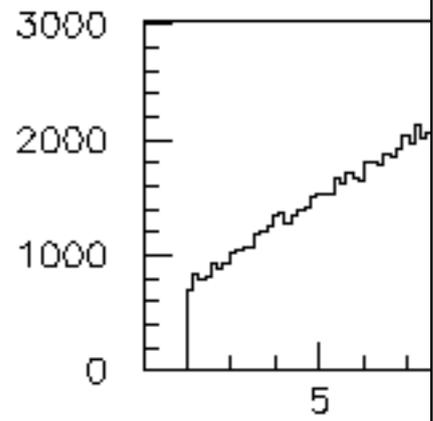
Square plane cosine law direction linear energy



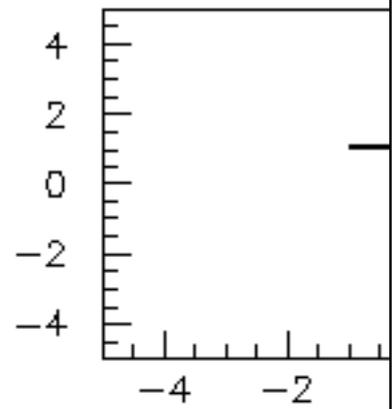
Spherical surface, isotropic radiation, black-body energy



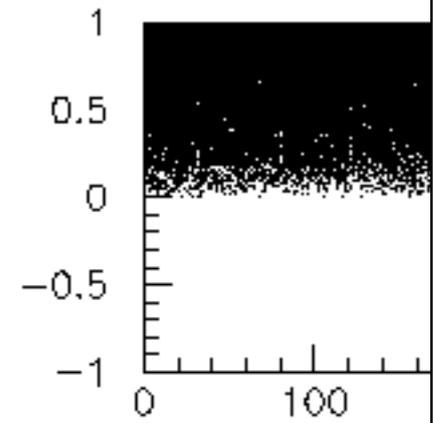
Square plane cosine law direction linear energy



Source Energy

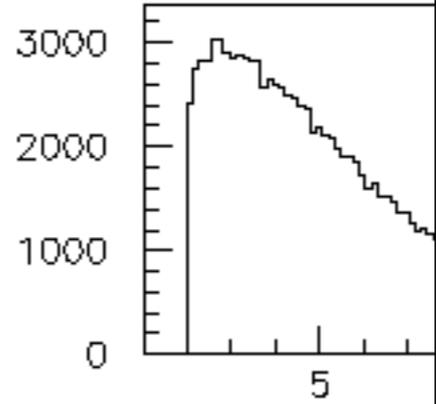


Source X-Y

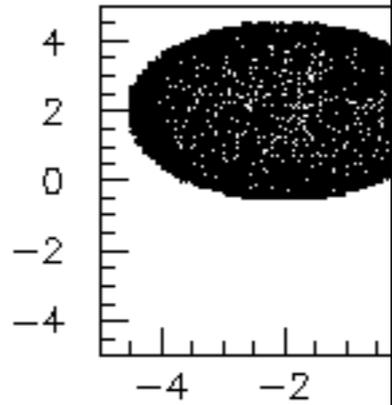


Source cos(theta)

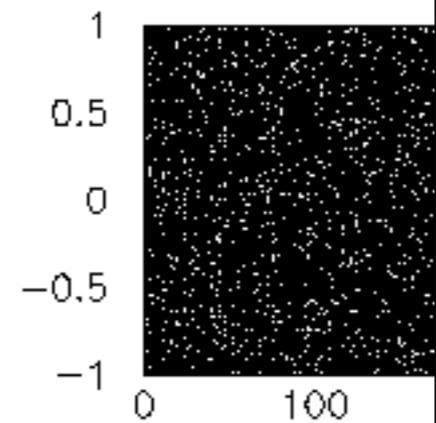
Spherical surface



Source Energy

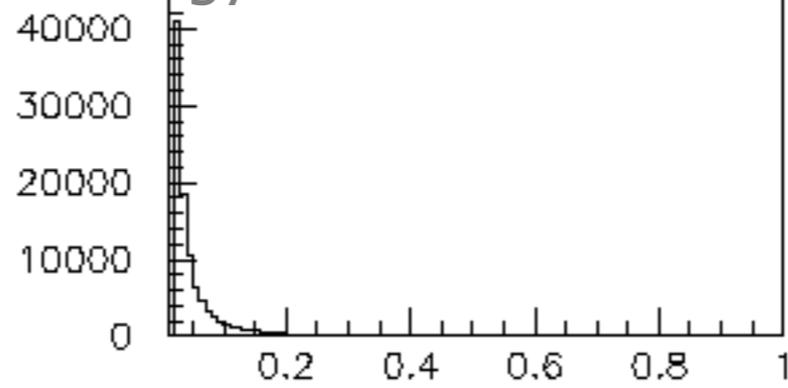


Source X-Y

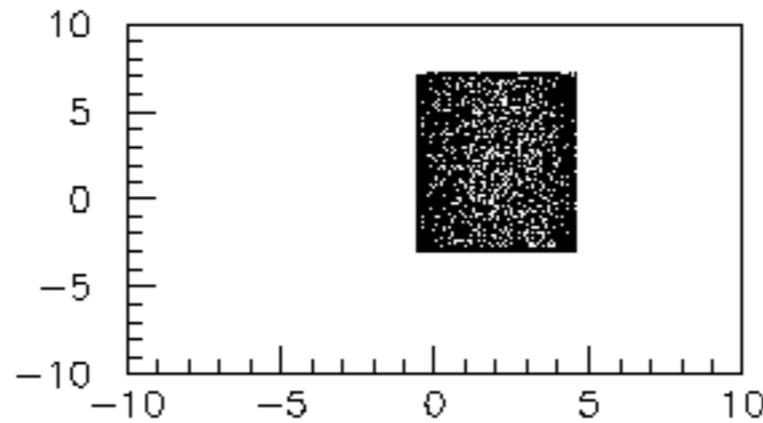


Source cos(theta)

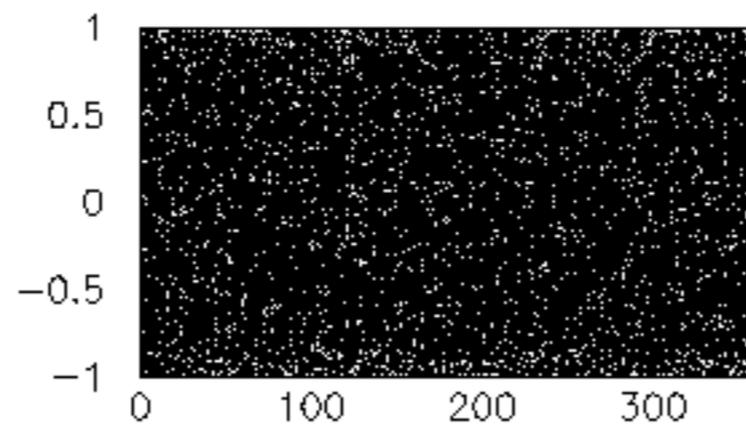
Cylindrical surface, cosine-law radiation, Cosmic diffuse energy



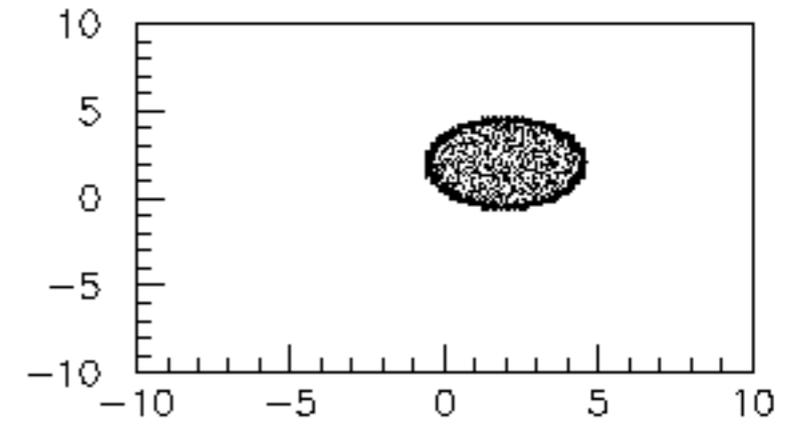
Source Energy Spectrum



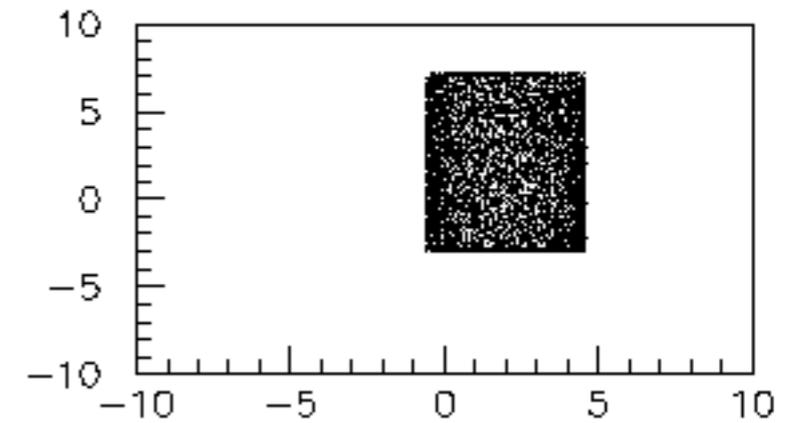
Source X-Z distribution



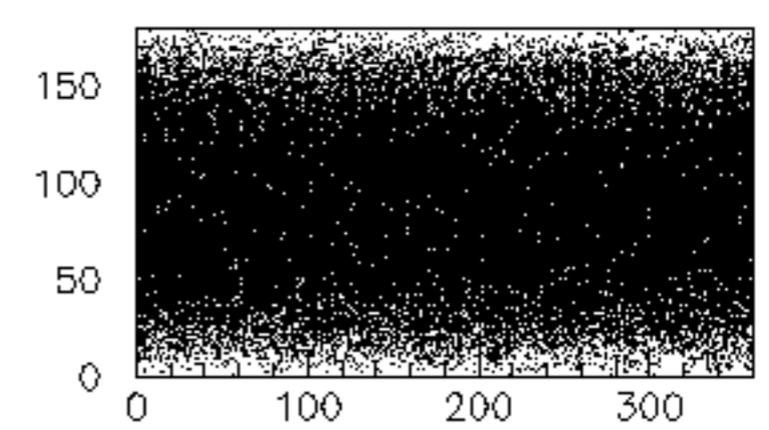
Source cos(theta)-phi distribution



Source X-Y distribution



Source Y-Z distribution



Source theta/phi distribution



Example of gps commands

- 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

Particle Gun 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)

Particle Gun 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

- Examples also exists for GPS
`examples/extended/
eventgenerator/exgps`
- And for HEPEvtInterface
`example/extended/runAndEvent/
RE01/src/RE01PrimaryGeneratorAction.cc`

Thank you for your attention!