Primary Particle

(based on slides by Makoto Asai)

Geant4 School at IFIN-HH, Bucharest
15 November 2016
Dennis Wright (SLAC)

SLAC NATIONAL ACCELERATOR LABORATORY

Outline

- Primary Particle Generation
- Built-in primary particle generators
  - particle gun
  - interface to HEPEVT
  - general particle source
- Primary vertex and primary particle
- Using particle gun and general particle source

Primary Particle Generation

- G4VUserPrimaryGeneratorAction
  - one of three mandatory classes users must derive from
  - used to set or change the properties of particle generators
  - concrete classes of this should not generate primaries directly
    - instead invoke GeneratePrimaryVertex() of generator class
  - may have more than one concrete class

- Implementation
  - instantiate generator(s) in constructor
  - implement GeneratePrimaries(G4Event* evt) method which is invoked during the event loop
    - use this method to pass info to generator through event pointer
  - in main() register derived class to run manager

Primary Particle Generation

- Things to do in G4VUserPrimaryGeneratorAction::GeneratePrimaries()
  - set generator defaults
  - initialize particle positions, energies, types
  - randomize the above
  - invoke GeneratePrimaryVertex() of generator class
  - don’t use hard-coded UI commands

    (UI->ApplyCommand(“…”);   )
    - very slow

Generators

- Generators
  - provide initial vertex of particle
    - position, energy, momentum, particle type, multiplicity
  - write this information into the event, which generator action passes to run manager
  - must derive from G4VPrimaryGenerator and implement GeneratePrimaryVertex()

- Generators cannot
  - randomize primaries
    - must do this in generator action class
  - at same vertex, generate particles with different properties
    - must invoke GeneratePrimaryVertex() more than once per event

Generators

- Geant4 provides some ready-built generators
  - G4ParticleGun
    - position, energy, momentum, particle type, multiplicity
  - G4GeneralParticleSource
    - many, many options for initial particles, spectra
    - useful for space physics, radioactive decay
    - documentation in section 2.7 of Application Developers' Guide
  - G4SingleParticleSource
    - extended version of G4ParticleGun
    - used by General Particle Source
  - G4HEPEvtInterface
    - conforms to /HEPEVT/ common block (standard for many Fortran event generators)
    - requires ASCII file input (4-vectors from HEP generator code)

Primary Vertex and Primary Particles

- Primary vertexes and primary particles are stored in G4Event in advance of event processing
  - by GeneratePrimaries(), GeneratePrimaryVertex()
- G4PrimaryParticle class
  - contains particle definition, initial energy, momentum, etc.
- G4PrimaryVertex class
  - contains primary particle, initial position, time, etc.

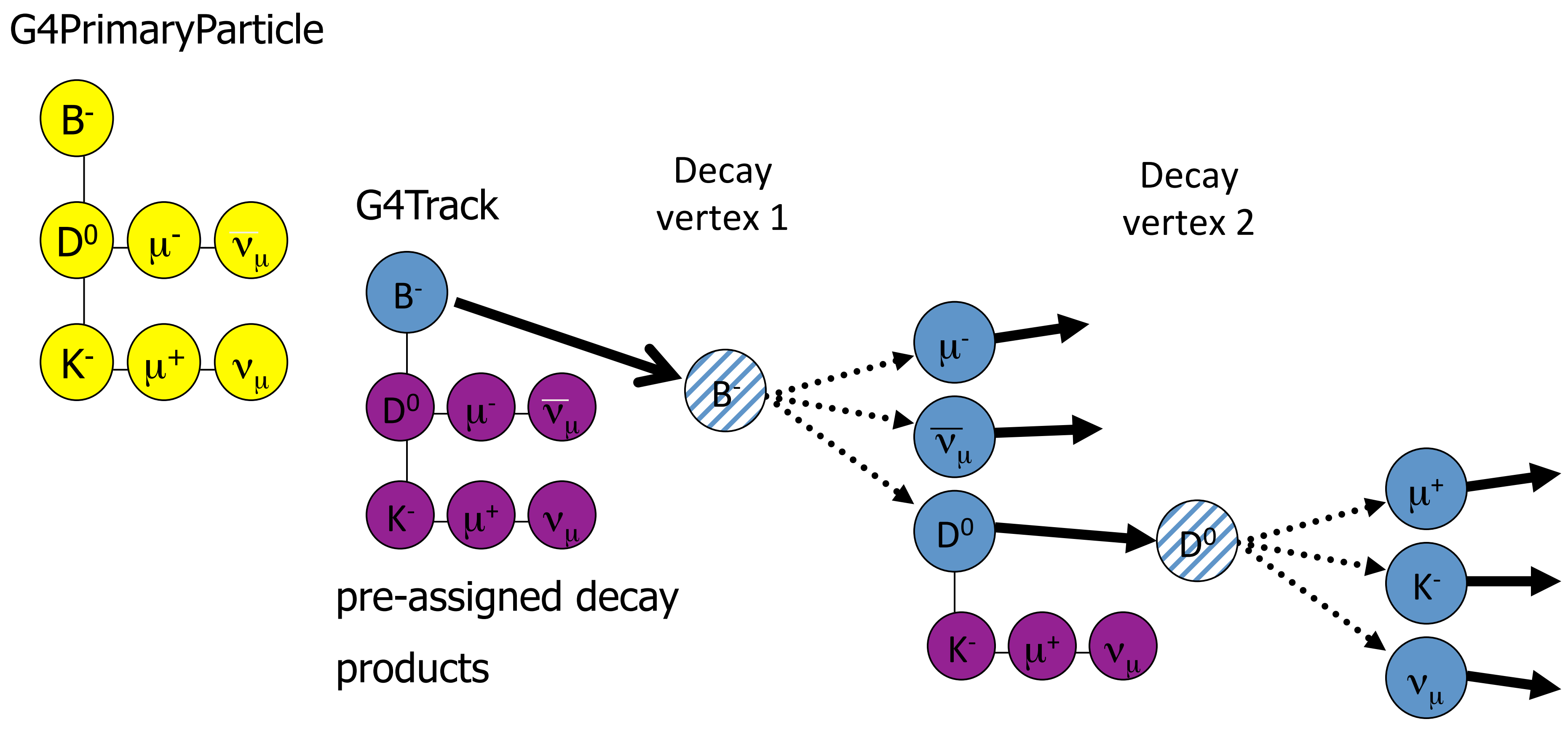
- Bookkeeping of decay chains
  - primaries need not be particles which can be tracked by Geant4 (W, quark, exotics, etc.)
    - but methods must be provided for handling them

Pre-assigned Decay Products

- Physics generators can assign decay channels for each individual particle separately, while in Geant4 you cannot specify a decay channel for each particle
  - it is assigned randomly according to the branching ratio
  - but, a decay chain can be "pre-assigned"

- A parent particle in the form of a G4Track object travels in the detector, bringing along with it "pre-assigned" decay daughters as objects of G4DynamicParticle
  - at decay point, daughters from pre-assigned channel become the secondaries, instead of randomly selecting a decay channel
  - decay time of the parent can be pre-assigned as well

Pre-assigned Decay Products



Using G4ParticleGun

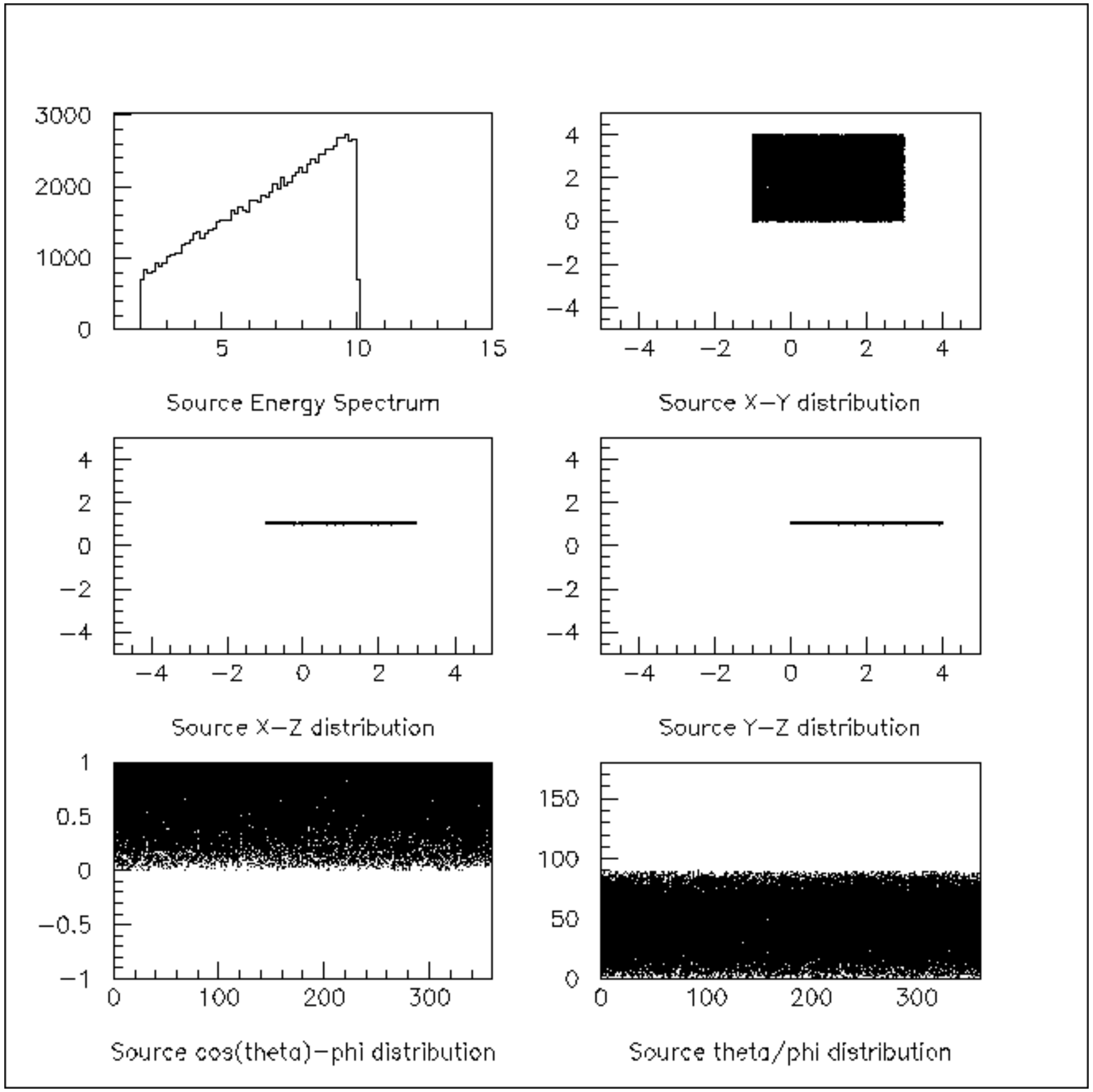
```
void T01PrimaryGeneratorAction::
         GeneratePrimaries(G4Event* anEvent) {
 G4ParticleDefinition* particle;
 G4int i = (int)(5.*G4UniformRand());
 switch(i)
  { case 0: particle = positron; break; ... }
 particleGun->SetParticleDefinition(particle);
 G4double pp = momentum +
    (G4UniformRand()-0.5)*sigmaMomentum;
 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 particle

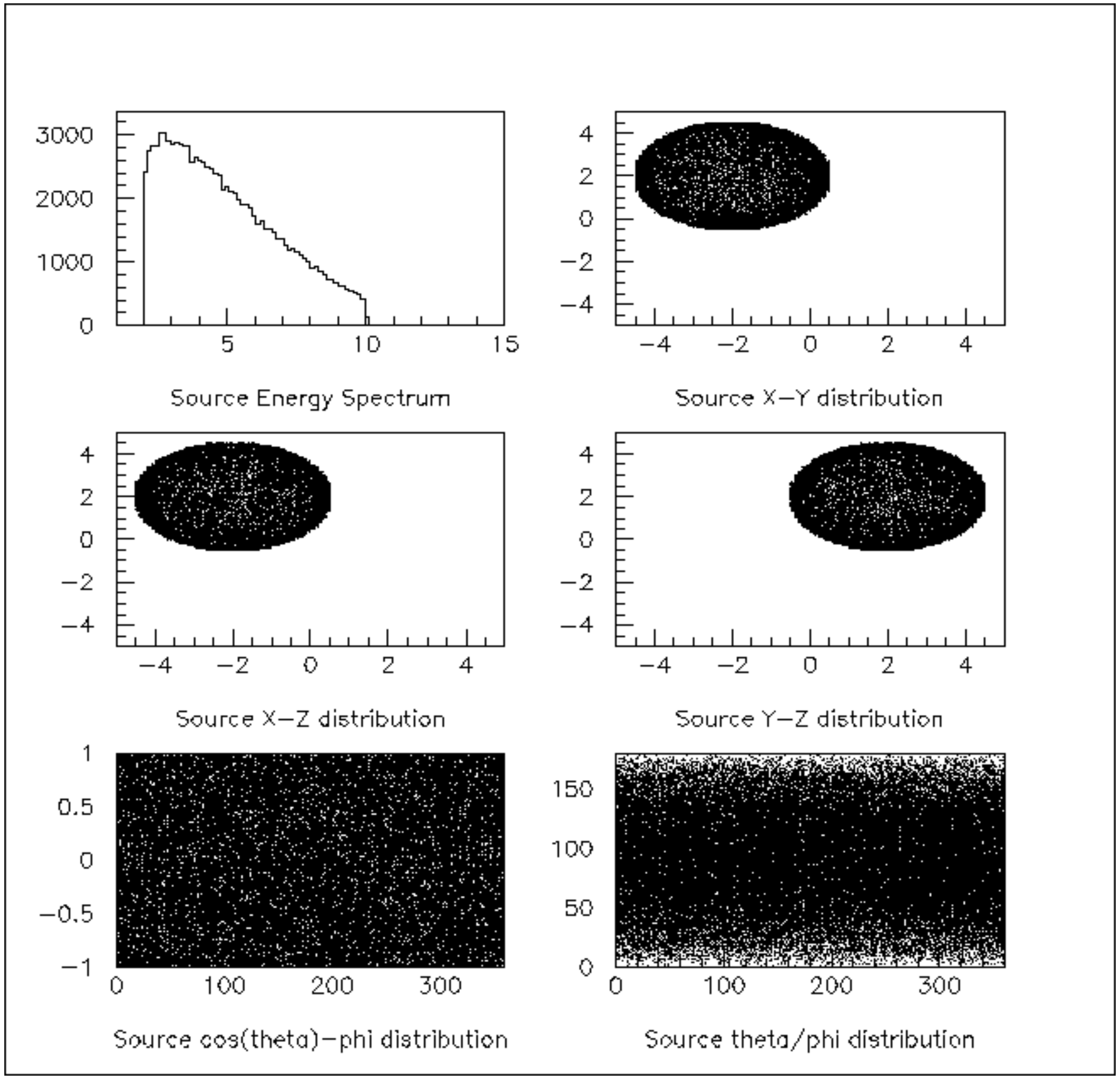
Using G4GeneralParticleSource

- 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 randomized
- Distributions can be set by UI commands
- Capable of event biasing (variance reduction)
  - by enhancing particle type, distribution of vertex point, energy and/or direction
- A few examples follow

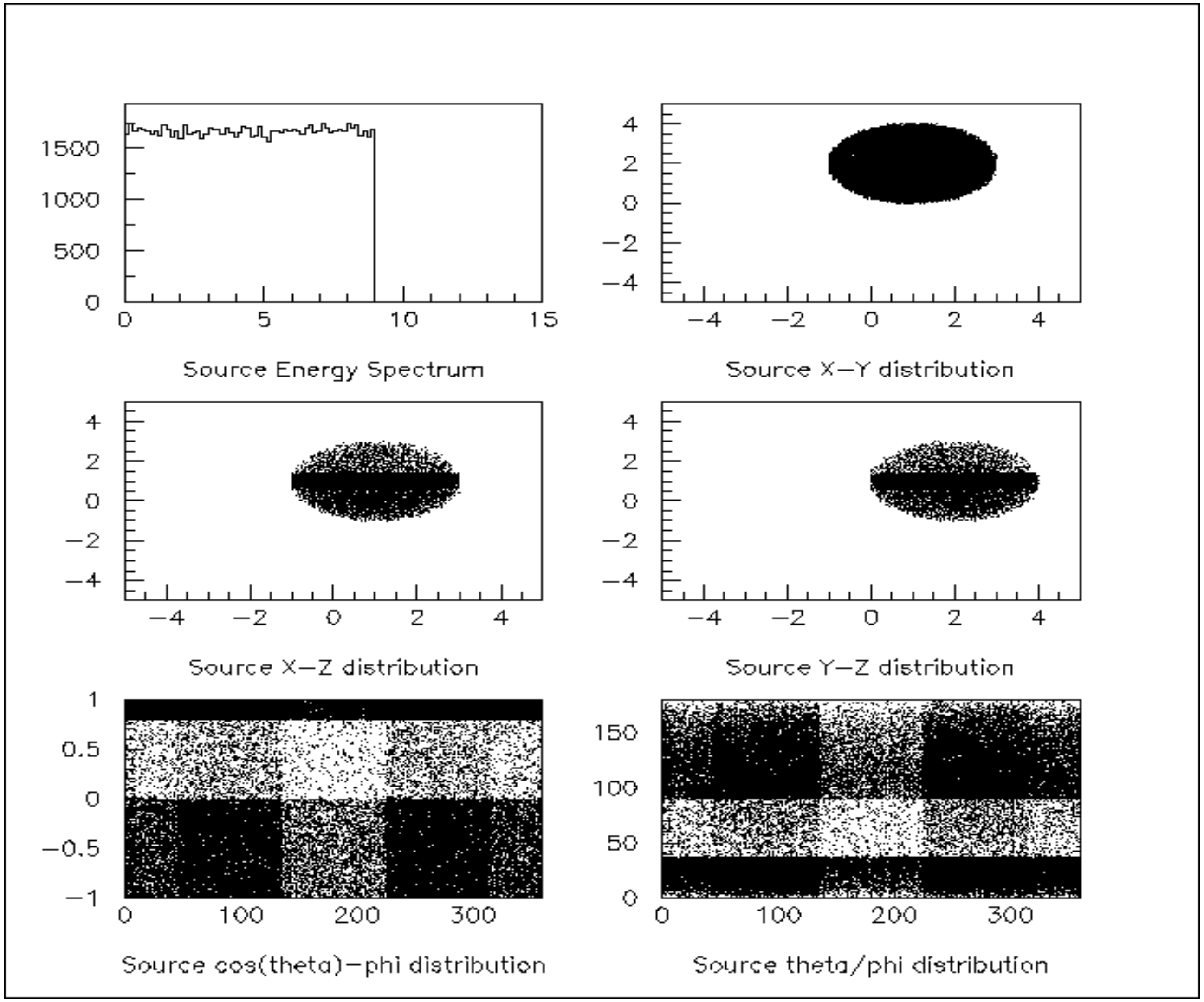
Square plane source, linearly increasing energy distribution, cosine law direction



Spherical surface source, black body energy spectrum, isotropic direction



Spherical volume source, flat energy spectrum, z-, phi- and theta-biasing of isotropic direction



Particle Gun vs. General Particle Source

- Particle Gun
  - simple and naïve
  - shoot one track at a time
  - easy to handle.
    - use set methods to alternate track-by-track or event-by-event values
- General Particle Source
  - powerful
  - controlled by UI commands
    - almost impossible to control through set methods
  - capable of shooting particles from a surface or a volume
  - can randomize kinetic energy, position and/or direction following a user-specified distribution (histogram)

- Use GPS if you need
  - primary particles from a surface or a volume, outward or inward
  - a complicated distribution, not flat or simple Gaussian
- Otherwise, use Particle Gun

Summary

- User must derive class from G4VUserPrimaryGeneratorAction
  - sets the characteristics of the generator
  - register it to run manager
- Generators must be derived from G4VPrimaryGenerator
  - and implement GeneratePrimaryVertex()
  - this is where you shoot the particle
- G4PrimaryParticle contains the particle type, energy, etc.
- G4PrimaryVertex contains the primary particle, initial position, time
- Some pre-built generators are available
  – General Particle Source for complex sources
  – G4ParticleGun for simple ones