Detector Simulation Primary particles

Witek Pokorski
Alberto Ribon
CERN

10-11.02.2014

Primary particles

This lecture is entirely based on the talk from Geant4 tutorial by Giovanni Santin

Ecole Geant4, Annecy 2008

Outline

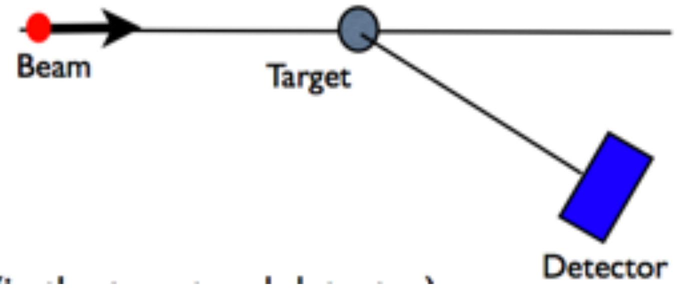
General concepts

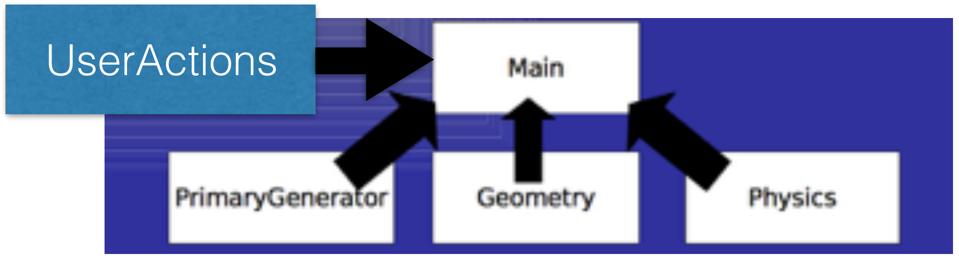
- G4VUserPrimaryGeneratorAction class
- Primary vertex and primary particle

Built-in primary particle generators

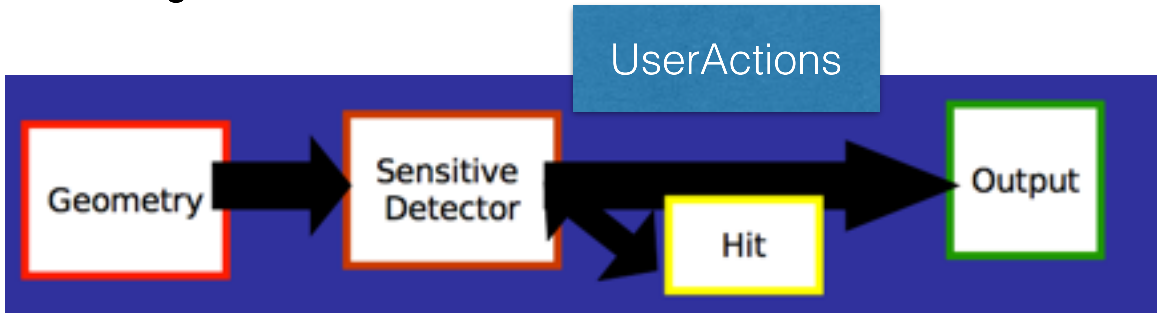
- G4ParticleGun
- Interfaces to HEPEVT and HEPMC
- General Particle Source (GPS)

What you need to make simulation?
(slide from Introduction)

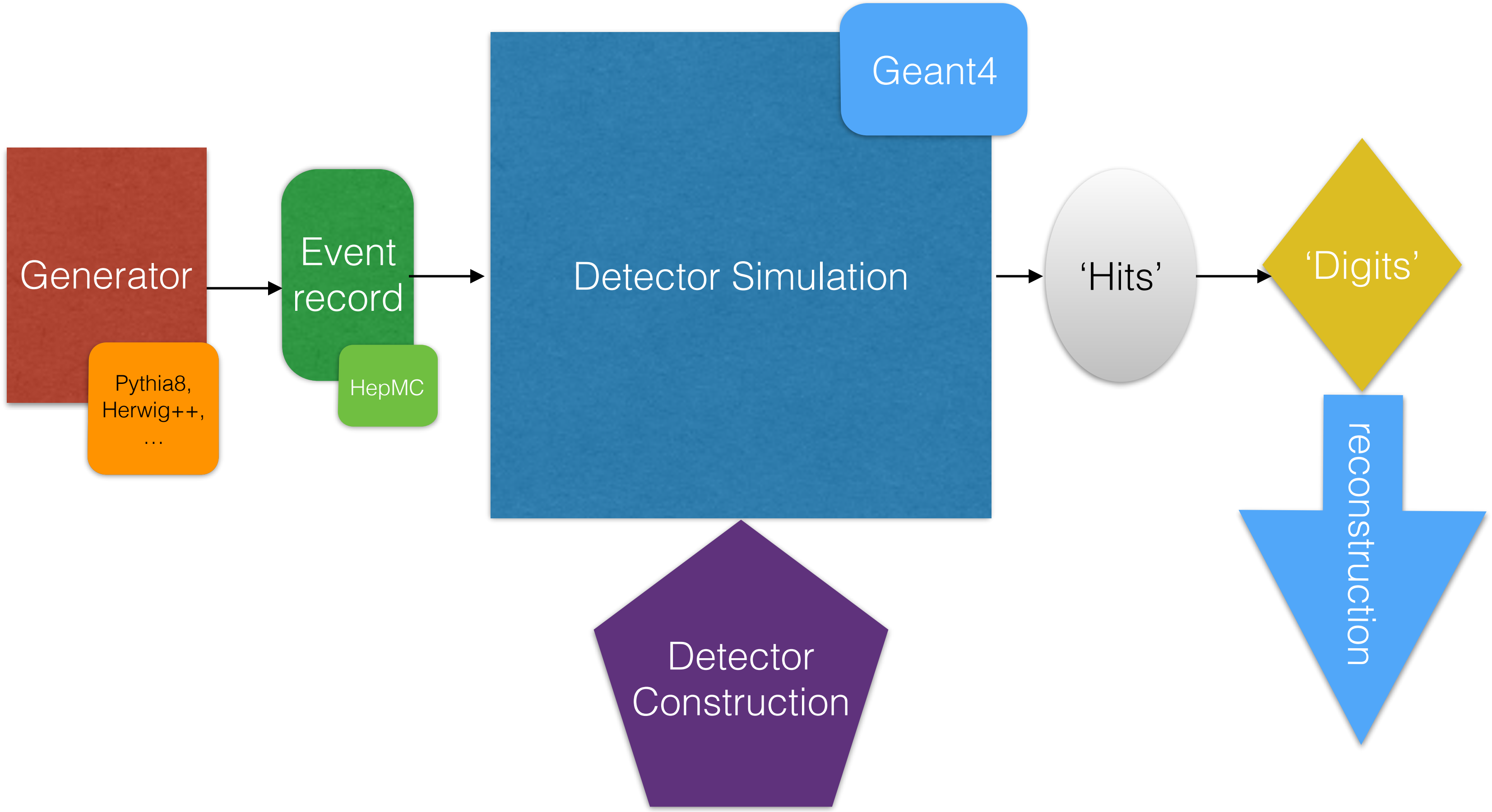




and to get something out of it…



Simulation chain for HEP experiment (slide from Introduction)



User Actions and Initializations

- **Initialization classes**
  - Use G4RunManager::SetUserInitialization() to define.
  - Invoked at the initialization
    - G4VUserDetectorConstruction
    - G4VUserPhysicsList

  $\leftarrow$ mandatory

- **Action classes**
  - Use G4RunManager::SetUserAction() to define.
  - Invoked during an event loop

    G4VUserPrimaryGeneratorAction $\leftarrow$ mandatory

    + G4UserRunAction / G4UserEventAction / G4UserStackingAction / G4UserTrackingAction/ G4UserSteppingAction / …

$\Rightarrow$ Main program (.cc file in your root development tree) :

```
// mandatory User Action classes
 G4VUserPrimaryGeneratorAction* gen_action = new PrimaryGeneratorAction;
 runManager->SetUserAction(gen_action);
```

G4VUserPrimaryGeneratorAction

- This class is one of the mandatory user classes and controls the generation of primaries ⇒ what kind of particle (how many) what energy, position, direction, polarisation, etc
- This class should NOT generate primaries itself but invoke **GeneratePrimaryVertex()** method of the selected primary generator(s) to make primaries
- **G4VPrimaryGenerator** class provides the primary particle generators

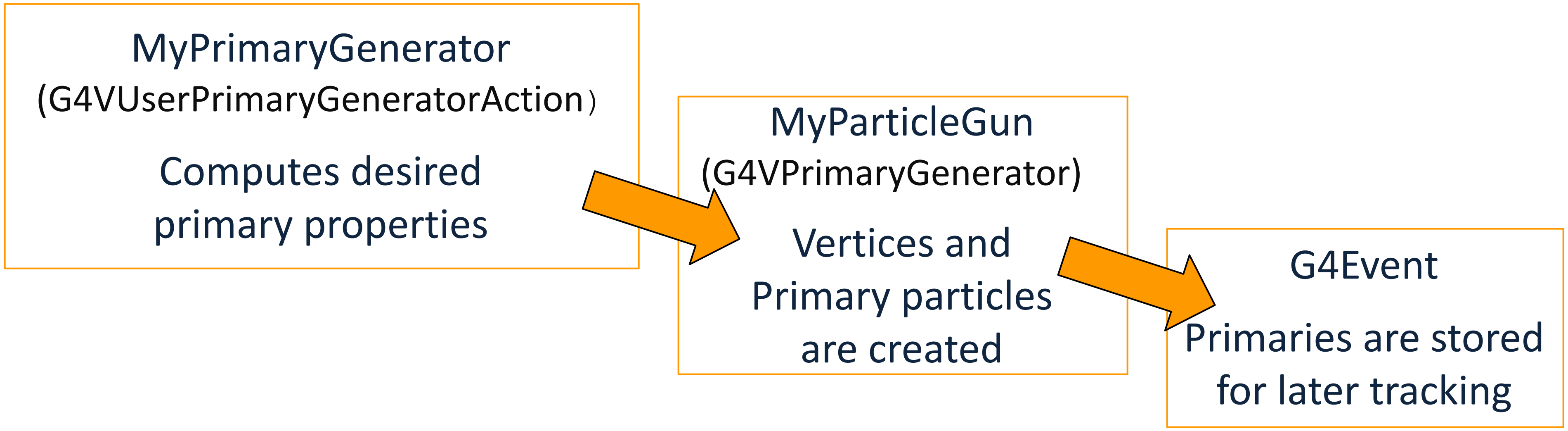
G4VUserPrimaryGeneratorAction class description :

- Constructor (& destructor)

  Instantiate primary generator and set default values
- **`GeneratePrimaries(G4Event *)`** method
  - Randomize particle-by-particle value(s)
  - Set these values to primary generator(s)
  - Invoke **`GeneratePrimaryVertex()`** method of primary generator

Primary vertices and primary particles

- Primary vertices and primary particles are stored in G4Event in advance to processing an event.
  - **G4PrimaryVertex** and **G4PrimaryParticle** classes
  - They will become “primary tracks” only at Begin-of-Event phase and put into a “stack”


MyPrimaryGenerator
(G4VUserPrimaryGeneratorAction)
Computes desired primary properties
MyParticleGun
(G4VPrimaryGenerator)
Vertices and Primary particles are created
G4Event
Primaries are stored for later tracking

Primary vertices and primary particles

- Capability of bookkeeping decay chains

⟹ primary particles may not necessarily be particles which can be tracked by Geant4

- Pre-assigned decay channels attached to particles
- Also, “exotic” particles can be imported from Particle Generators, followed by either decay or user defined physics processes

  (e.g. Higgs, W/Z boson, SUSY particle, …)

Built-in primary particle generators

- **Geant4 provides some concrete implementations of G4VPrimaryGenerator.**

  1. G4ParticleGun
  2. G4HEPEvtInterface, G4HEPMCInterface
  3. G4GeneralParticleSource

G4ParticleGun

- **Concrete implementations of G4VPrimaryGenerator**

It shoots one primary particle of a certain energy from a certain point at a certain time to a certain direction.
(a complete set of function is available)

- **UI 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

G4ParticleGun : complex sources

- **G4ParticleGun is basic, but it can be used from inside UserPrimaryGeneratorAction to model complex source types or 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

G4ParticleGun : complex sources

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);
}
```

← choose particle

← set particle

← set kinetic energy and momentum

← generate event

⟹ You can repeat this for generating more than one primary particles.

Interfaces to external event generators

Concrete implementations of G4VPrimaryGenerator

- Good examples for experiment-specific primary generator implementation
- Interface to external physics generators

⟹ **G4HEPEvtInterface**

- Event record structure based on **HEPEVT** common block
- Used by (FORTRAN) HEP physics generators
- Developed and agreed on within the framework of the 1989 LEP physics study
- ASCII file input

⟹ **G4HepMCInterface**

- **HepMC** Event record for MC generators. Object Oriented, C++
- Used by new (C++) HEP physics generators
- ASCII file input or direct linking to a generator through HepMC

User actions for external event generators

Adapted from examples/extended/eventgenerator/HepMC/HepMCEx01 and examples extended/runAndEvent/RE01

```
PrimaryGeneratorAction::PrimaryGeneratorAction() {
   // HepMC
   m_currentGenerator = new HepMCG4AsciiReader();
   // HEPEvt
   // G4String filename = "pythia_event.data";
   // m_currentGenerator = new G4HEPEvtInterface(filename);
}

PrimaryGeneratorAction::~PrimaryGeneratorAction() {
     delete m_currentGenerator;
}

void PrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent) {
     m_currentGenerator-> GeneratePrimaryVertex(anEvent);
}
```

+ UI macro commands

```
/generator/hepmcAscii/open filename
/run/beamOn 1
```

G4GeneralParticleSource (GPS)

- An advanced concrete implementation of G4VPrimaryGenerator
  - 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
- 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
- All features can be used via C++ or command line (or macro) UI

G4GeneralParticleSource (GPS)

Features available in GPS:

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

User Actions for GPS

Example of user PrimaryGeneratorAction using GPS

```
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

G4GeneralParticleSource (GPS)

Many examples are available here :

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

- Example 1

```
/gps/particle proton

/gps/ene/type Mono
/gps/ene/mono 500 MeV

/gps/pos/type Plane
/gps/pos/shape Rectangle
/gps/pos/rot1 0 0 1
/gps/pos/rot2 1 0 0
/gps/pos/halfx 46.2 cm
/gps/pos/halfy 57.2 cm
/gps/pos/centre 0. 57.2 0. cm

/gps/direction 0 -1 0

/run/beamOn ...
```

←

mono energetic beam
500 Mev

planar emission from a z×x plane
along -y axis

Conclusion

- User primary generator action is a mandatory class that user must implement
    - This class can re-use existing primary generators
- ‘particle guns’ used for test-beam or fixed target simulations
- interface to HepMC event record used for MC event generators