Primary generator

Geant4 School 2010

Koichi Murakami
Geant4 Collaboration
KEK/CRC

Contents

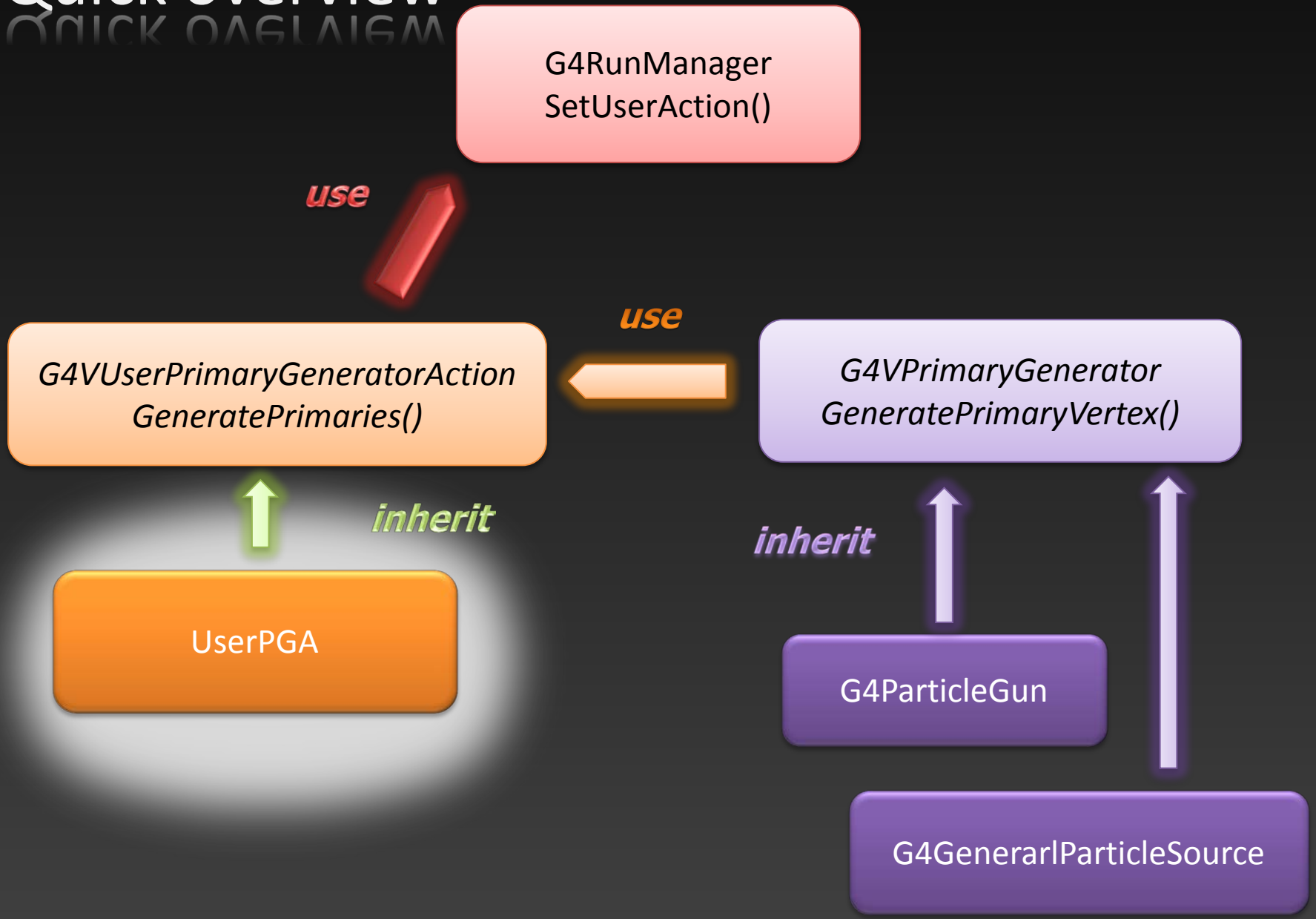
G4VUserPrimaryGeneratorAction

- *mandatory user action*

Built-in primary particle generators

- Particle Gun (G4ParticleGun)
- General Particle Source (GPS) (G4GeneralParticleSource)

Quick overview



Primary Generator Action

A mandatory user action class to control the generation of primaries.

- derived from *G4VUserPrimaryGeneratorAction*
- set with G4RunManager::SetUserAction()
- invoked during an event loop

Code example in main() :

```
// mandatory user action classes
PrimaryGeneratorAction* gen_action =
   new PrimaryGeneratorAction;
runManager-> SetUserAction(gen_action);
```

Implementation of PGA

Your PGA class should be inherited from *G4VPrimaryGgeneratorAction*, and implemented.

PGA typically calls GeneratePrimaryVertex() of primary generator (G4VPrimaryGenerator) in GeneratePrimaries() method.

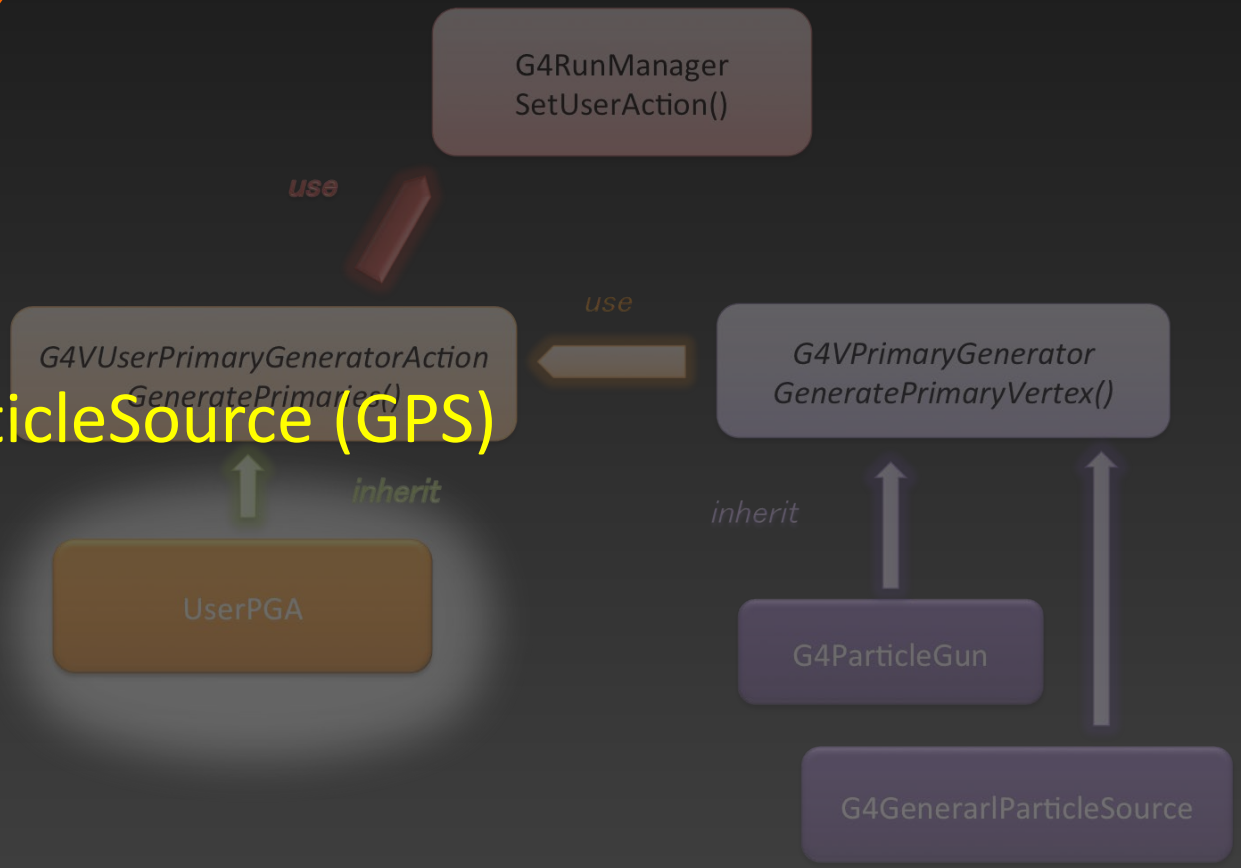
How to implement :

- *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)
  - ✓ Invoke GeneratePrimaryVertex() method of primary generator(s)

Built-in primary particle generators

Geant4 provides some concrete implementations of *G4VPrimaryGenerator*.

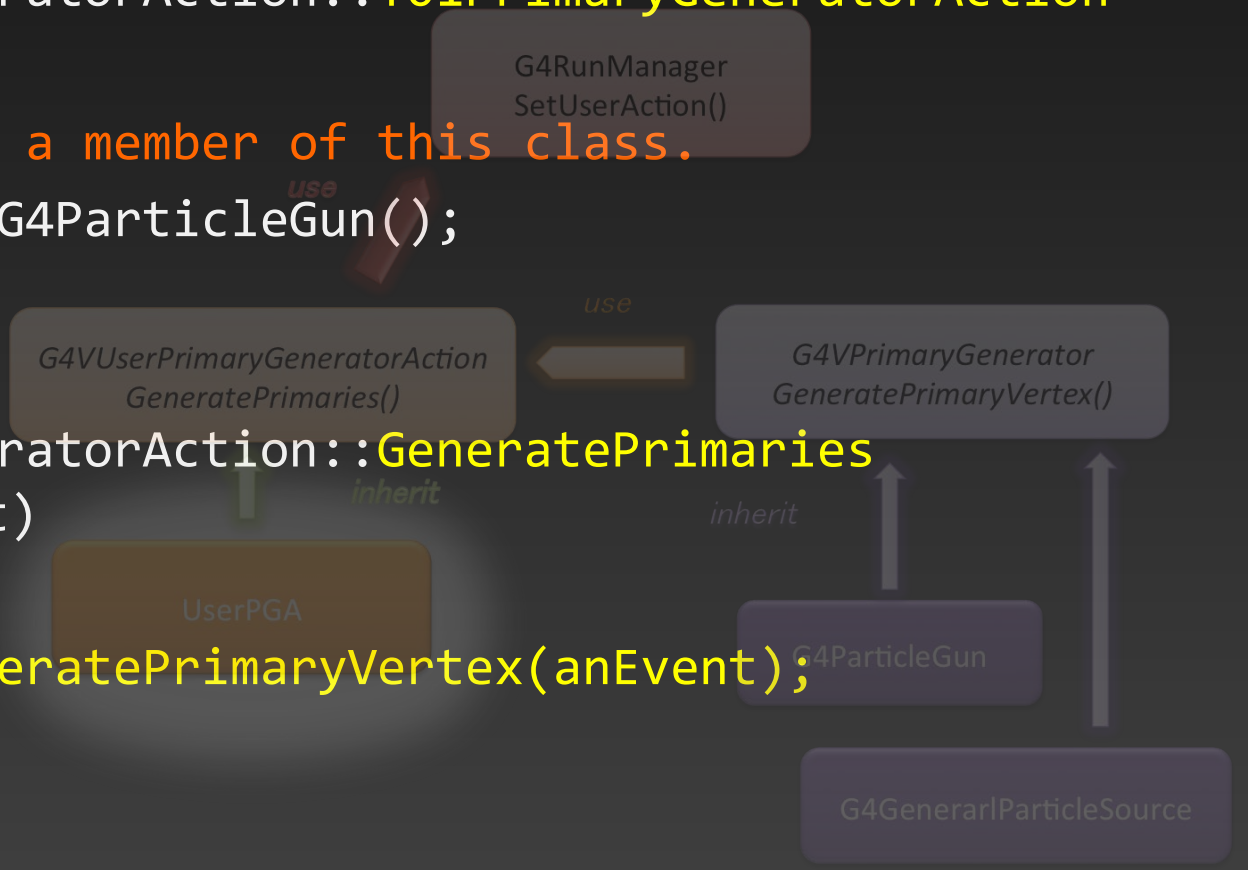
- G4ParticleGun
- G4GeneralParticleSource (GPS)



Example of user PGA using G4ParticleGun (simple case)

```
// constructor
void T01PrimaryGeneratorAction::T01PrimaryGeneratorAction
{
  // particleGun is a member of this class.
  particeGun = new G4ParticleGun();
}

void T01PrimaryGeneratorAction::GeneratePrimaries
   (G4Event* anEvent)
{
  particleGun-> GeneratePrimaryVertex(anEvent);
}
```



G4ParticleGun

A concrete implementations of *G4VPrimaryGenerator*

It shoots primary particles of a certain *energy* from a certain *point* at a certain *time* to a certain *direction.*

- a complete set of functions is available.

UI commands are also available for setting initial values.

/gun/list	List available particles
/gun/particle	Set particle type 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 your PGA to model complex source types or distributions.

Usecase:

- Generate desired distributions by shooting random numbers
- Use (C++) set methods of G4ParticleGun

Example of user PGA using G4ParticleGun (complex case)

```
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 momentum = 100.*MeV;
  G4double pp = momentum + (G4UniformRand()-0.5) * sigmaMomentum;
  G4double mass = particle-> GetPDGMass();
  G4double Ekin = sqrt(pp*pp+mass*mass) - mass;
  particleGun-> SetParticleEnergy(Ekin);

  G4double sigmaAngle = 10.*deg;
  G4double angle = (G4UniformRand()-0.5) * sigmaAngle;
  particleGun->
    SetParticleMomentumDirection(G4ThreeVector(sin(angle),0.,cos(angle)));

  particleGun-> GeneratePrimaryVertex(anEvent);
}
```

particle selection

energy distribution

angular distribution

General particle source (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

Provides 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++ direct implementation or UI commands.

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

Example of user PGA using GPS

```
// constructor
MyPrimaryGeneratorAction::MyPrimaryGeneratorAction()
{
  // gps is a member of this class
  gps= new G4GeneralParticleSource();
}

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

All user instructions given via macro UI commands.

UI commands for GPS

Many examples are available here.

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

```
/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 ...
```

protons

mono energetic beam of 500 MeV

planar emission from a zx plane along -y axis

Summary

In order to shoot primary particles,
you must derive a user action class from
G4VUserPrimaryGeneratorAction.

In this class,

- instantiate *primary generator*(s) in the constructor.
- set shooting values to it in the *GeneratePrimaries()* method.

You can use concrete implementation of primary generators.

- G4ParticleGun
- G4GeneralParticleSource
- UI commands to control parameters are available.