A very simple Geant4 application

Geant4 Tutorial, Marshall Space Flight Center
April 2012

Daniel Brandt
Tutorial based on Geant4 v9.5-p01
Required components

• A physics list...
  ...so Geant4 knows what processes to simulate

• A detector construction...
  ...so Geant4 knows what geometry to simulate

• A particle generator...
  ...so Geant4 knows what particles to propagate
A complete G4 main method

```cpp
int main(int argc, char** argv){
    G4RunManager* runManager = new G4RunManager;

    MyGeometry* geom = new MyGeometry();
    runManager->SetUserInitialization(geom);

    MyPhysicsList* physics = new MyPhysicsList();
    runManager->SetUserInitialization(physics);

    MyPrimaryGeneratorAction* generator = new MyPrimaryGeneratorAction();
    runManager->SetUserAction(generator);

    runManager->Initialize();

    runManager->BeamOn(1);

    delete runManager;
    return 0
}
```
The geometry

The geometry inherits from \texttt{G4VUserDetectorConstruction}. It has to implement \texttt{G4VPhysicalVolume\* Construct()}.

```c++
Class MyGeometry : public G4VUserDetectorConstruction{
    G4VPhysicalVolume\* Construct();
}
```

The \textit{Construct()} method must return a pointer to a single volume containing all other volumes in the simulation. This is called the „World“. 
The physics list

The physics list inherits from `G4VUserPhysicsList`. The easiest way to implement this is to use a Geant4 standard physics list

```cpp
G4VUserPhysicsList* physics = new FTFP_BERT();
runManager->SetUserInitialization(physics);
```

Alternatively you could write your own. It must implement `ConstructParticle()`, `ConstructProcess()` and `SetCuts()`.

```cpp
class MyPhysics : public G4VUserPhysicsList{
    void ConstructParticle();
    void ConstructProcess();
    void SetCuts();
}
```
The primary generator action

Provides a method to create primary particles, the starting point of the simulation. It inherits from `G4VUserPrimaryGeneratorAction` and must implement `GeneratePrimaries(G4Event*)`.

```cpp
class myGenerator : public G4VUserPrimaryGeneratorAction{

    void GeneratePrimaries(G4Event*);
}
```

Particles can be generated by `G4ParticleGun`:

```cpp
G4ParticleGun* myGun = new G4ParticleGun(int n_particle = 1);
myGun->SetMomentumDirection(G4ThreeVector(1,0,0));
myGun->SetKineticEnergy(50.*MeV);
myGun->GeneratePrimaryVertex(G4Event* anEvent);
```
A complete Geant4 application

• To run a Geant4 application:

  ➢ Obtain pointer to the G4RunManager

  ➢ Register physics list G4VUserPhysicsList

  ➢ Register geometry G4VUserDetectorConstruction

  ➢ Register particle generator G4VPrimaryGeneratorAction

  ➢ G4RunManager.Initialize();

  ➢ G4RunManager.BeamOn(int number_of_particles);