An introduction to GEANT4

July 2021

Overview:

- About GEANT4
- Where to get help
- Classes
- Detector construction
- Visualization/debugging
- Analysis

What is Geant4?

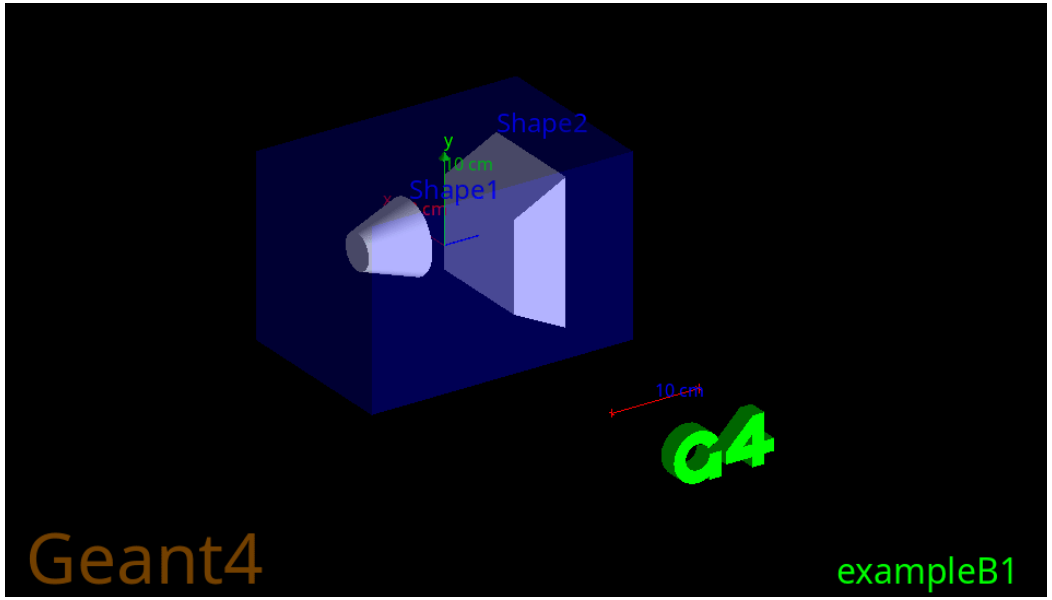
- Geant4 is a detector simulation toolkit for high-energy physics
- C++ Objective Oriented Programming
- We tell it what our experimental setup looks like, what particles we want to simulate and what physics we want to include
- Simulates millions of events – up to us to work out what we want to measure/extract from this Monte Carlo simulation

Where to get help

- GEANT4 website tutorial: http://geant4.web.cern.ch/collaboration/workshops/users2002/tutorial
- List of examples (basic/extended/advanced)
- Google-fu
- I am happy to help with any issues (time/length of problem permitting):
  - jackedwardbishop@gmail.com

Working through an example

- Firing gamma rays/protons at bone/tissue inside water
- Special Geant4 variable – prefixed with G4 (i.e. G4double)
- OOP C++ relies on a number of different classes which interact together to form our simulation – looking at example B1 (basic 1) – 6 MeV gamma
    - DetectorConstruction (defining our setup)
    - PrimaryGeneratorAction (particle source)
    - SteppingAction (finds the energy deposition in a selected volume for each particle step)
    - EventAction (event by event energy deposition)
    - RunAction (ties the simulation together and gets the event energy deposition)



Before we start.... Units!

G4SystemofUnits.hh

When inputting a variable, we must always multiply the variable by the units.

I.e. G4double density = 5*mg/cm3

When outputting a value, divide by the unit we want it in:

G4cout<< density/(g/cm3)<<G4endl;

Geant4 Internal Units

millimeter	(mm)
nanosecond	(ns)
Mega electron Volt	(MeV)
positron charge	(eplus)
degree Kelvin	(kelvin)
the amount of substance	(mole)
luminous intensity	(candela)
radian	(radian)
steradian	(steradian)

Detector Construction

- Defining our setup – done in terms of different volumes
  - Solid Volumes
    - Physical shape: G4Box, G4Sphere, G4Cons (segment of a cone), G4Tubs etc….
    - Can be combined, intersected, subtracted to make more complex shape from underlying polygons
    - Contains the geometric information about the shape
  - Logical Volumes
    - Tells us about the material, any EM fields it has and what solid volume (shape) it is made from
  - Can then place the logical volume and give location and orientation

```
G4LogicalVolume( G4VSolid*              pSolid,
                 G4Material*            pMaterial,
                 const G4String&        Name,
                 G4FieldManager*        pFieldMgr=0,
                 G4VSensitiveDetector*  pSDetector=0,
                 G4UserLimits*          pULimits=0,
                 G4bool                 Optimise=true )
```

Detector Construction

- First volume we need to define is the World volume
  - Simulations cannot be in absolute vacuum – need to be in a world of at least very high vacuum. World volume can also just be air.
  - Make sure your world is big enough to fit everything in!!!

```
G4Material* vacuum =
    new G4Material("Vacuum",       //Name as String
       1,               //Atomic Number,  in this case we use 1 for hydrogen
       1.008*g/mole,    //Mass per Mole "Atomic Weight"  1.008*g/mole for Hydoren
       1.e-25*g/cm3,    //Density of Vaccuum  *Cant be Zero, Must be small insted
       kStateGas,       //kStateGas for Gas
       2.73*kelvin,     //Temperature for gas
       1.e-25*g/cm3);   //Pressure for Vaccum
```

Define the size of our world:

env_sizeXY=20*cm

G4 types have units built in

G4Material – use built-in library to get properties of air

```
//
// World
//
G4double world_sizeXY = 1.2*env_sizeXY;
G4double world_sizeZ  = 1.2*env_sizeZ;
G4Material* world_mat = nist->FindOrBuildMaterial("G4_AIR");

G4Box* solidWorld =
  new G4Box("World",                       //its name
     0.5*world_sizeXY, 0.5*world_sizeXY, 0.5*world_sizeZ);     //its size

G4LogicalVolume* logicWorld =
  new G4LogicalVolume(solidWorld,          //its solid
                      world_mat,           //its material
                      "World");            //its name
```

Detector Construction

- First volume we need to define is the World volume
  - Simulations cannot be in absolute vacuum – need to be in a world of at least very high vacuum. World volume can also just be air.
  - Make sure your world is big enough to fit everything in!!!

Define the size of our world:

env_sizeXY=20*cm

G4 types have units built in

G4Material – use built-in library to get properties of air

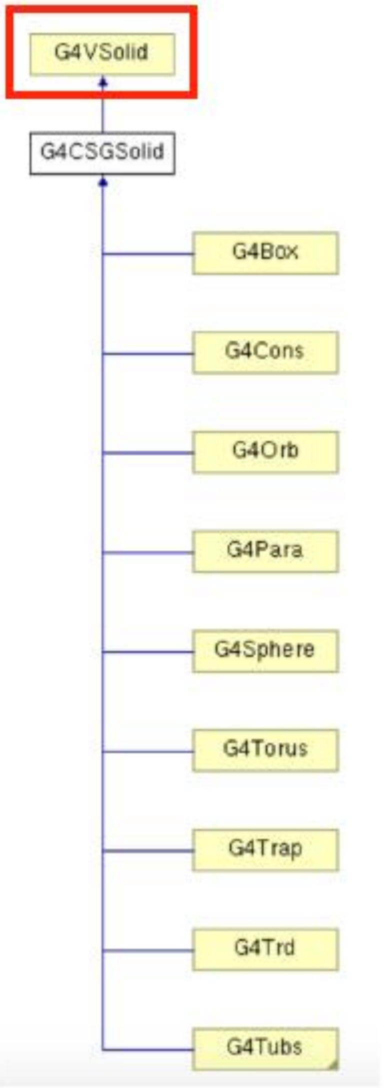
Define a physical volume – a box of ***half-widths*** x,y,z

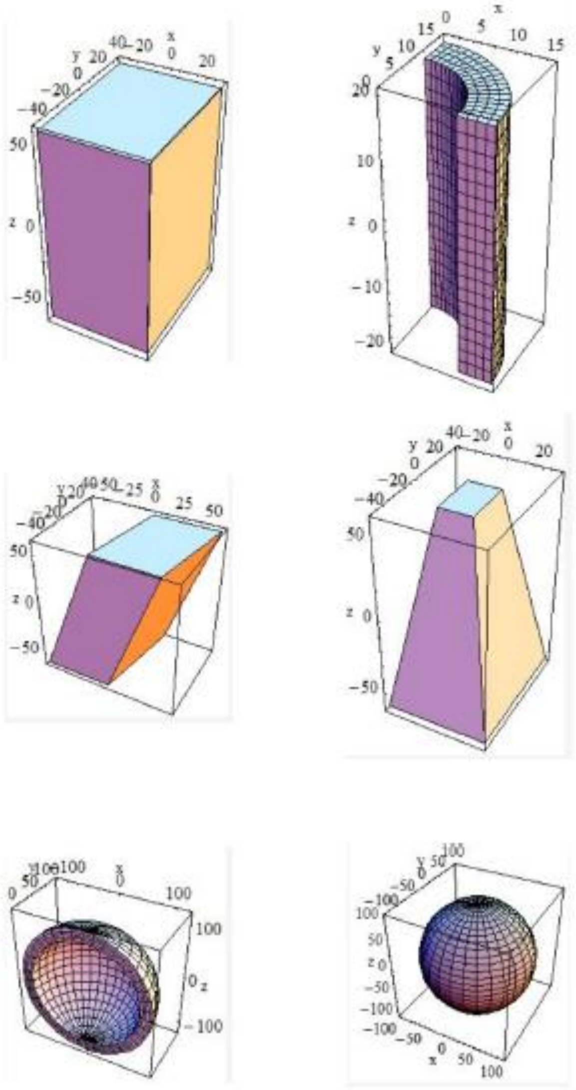
Define a logical volume – our physical box defined above, made of our world material (air)

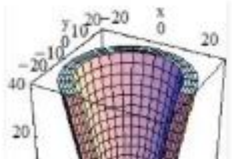
Place our logical volume, unrotated, at the origin, mother volume is 0 as we are defining the world (we are not placing the object into anything else)

G4cout<<env_sizeXY/cm<<" cm"<<endl;

```
//
// World
//
G4double world_sizeXY = 1.2*env_sizeXY;
G4double world_sizeZ  = 1.2*env_sizeZ;
G4Material* world_mat = nist->FindOrBuildMaterial("G4_AIR");

G4Box* solidWorld =
  new G4Box("World",                       //its name
     0.5*world_sizeXY, 0.5*world_sizeXY, 0.5*world_sizeZ);     //its size

G4LogicalVolume* logicWorld =
  new G4LogicalVolume(solidWorld,          //its solid
                      world_mat,           //its material
                      "World");            //its name

G4VPhysicalVolume* physWorld =
  new G4PVPlacement(0,                     //no rotation
                    G4ThreeVector(),       //at (0,0,0)
                    logicWorld,            //its logical volume
                    "World",               //its name
                    0,                     //its mother  volume
                    false,                 //no boolean operation
                    0,                     //copy number
                    checkOverlaps);        //overlaps checking
```

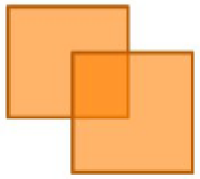




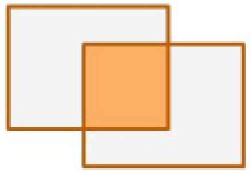


Boolean Solids for complicated structures

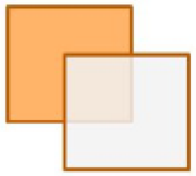
1. Union



2. Intersection



3. Subtraction



```
G4Box*  box =
  new G4Box("Box",20*mm,30*mm,40*mm);
G4Tubs* cyl =
  new G4Tubs("Cylinder",0,50*mm,50*mm,0,twopi);  // r:     0 mm -> 50 mm
                                                 // z:   -50 mm -> 50 mm
                                                 // phi:   0 ->  2 pi
G4UnionSolid* union =
  new G4UnionSolid("Box+Cylinder", box, cyl);
G4IntersectionSolid* intersection =
  new G4IntersectionSolid("Box*Cylinder", box, cyl);
G4SubtractionSolid* subtraction =
  new G4SubtractionSolid("Box-Cylinder", box, cyl);
```

G4Materials

- A few ways to get materials
    - Can find material in database (CsI)
    - Can define via G4Element
    - Then can mix to make your own materials (e.g. Havar)

[http://geant4-userdoc.web.cern.ch/geant4-userdoc/UsersGuides/ForApplicationDeveloper/html/Appendix/materialNames.html]

```
G4Material* si = G4Material::GetMaterial("G4_Si");
G4Material* al = new G4Material("al",2.7*g/cm3,1);
G4Material* be = new G4Material("be",1.848*g/cm3,1);
G4Element* Al = new G4Element("Aluminum","Al",13, 26.982*g/mole);
al->AddElement(Al,1);
G4Element* Be = new G4Element("Beryllium","Be",4, 9.012*g/mole);
be->AddElement(Be,1);
G4Material* CsI = G4Material::GetMaterial("G4_CESIUM_IODIDE");
G4Element* Cr = new G4Element("Chrome", "Cr", 25,   51.996*g/mole);
G4Element* Fe = new G4Element("Iron"  , "Fe", 26,   55.845*g/mole);
G4Element* Co = new G4Element("Cobalt", "Co", 27,   58.933*g/mole);
G4Element* Ni = new G4Element("Nickel", "Ni", 28,   58.693*g/mole);
G4Element* W  = new G4Element("Tungsten","W", 74,  183.850*g/mole);
G4Element* H  = new G4Element("Hydrogen","H", 1,  1.008*g/mole);
G4Element* C  = new G4Element("Carbon","C", 6,  12.011*g/mole);
G4Element* D = new G4Element("Deuterium","D",1,2.014102*g/mole);
G4Element* Au = new G4Element("Gold","Au",79,196.97*g/mole);
G4Material* Havar =
  new G4Material("Havar", 8.3*g/cm3, 5);
Havar->AddElement(Cr, 0.1785);
Havar->AddElement(Fe, 0.1822);
Havar->AddElement(Co, 0.4452);
Havar->AddElement(Ni, 0.1310);
Havar->AddElement(W , 0.0631);
```

Number of elements in G4Material

Fractional mass

Detectors

Embedded into logicWorld

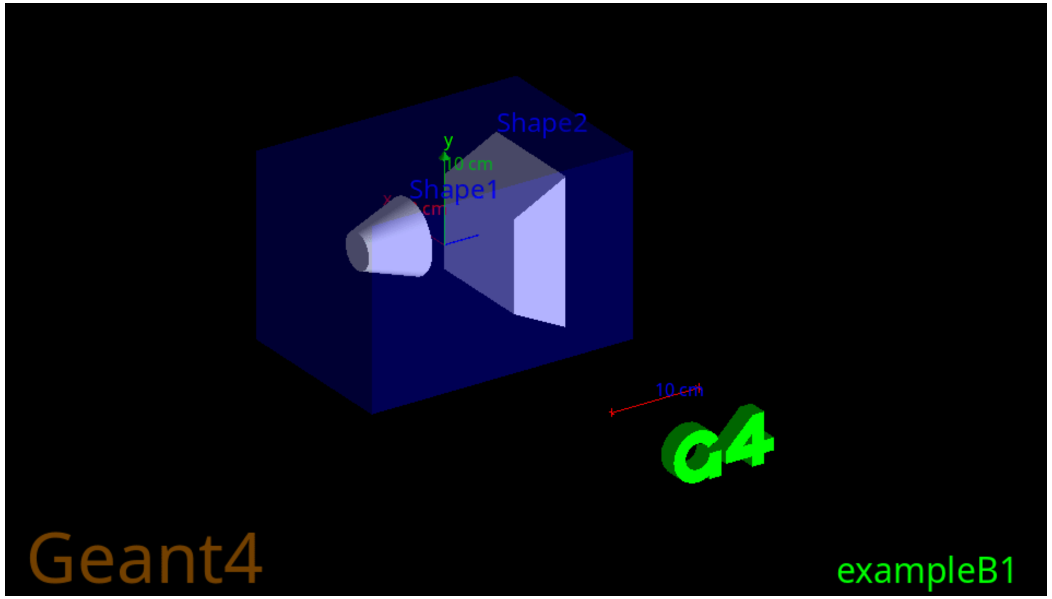
```
// Envelope parameters
//
G4double env_sizeXY = 20*cm, env_sizeZ = 30*cm;
G4Material* env_mat = nist->FindOrBuildMaterial("G4_WATER");

//
// Envelope
//
G4Box* solidEnv =
  new G4Box("Envelope",                       //its name
      0.5*env_sizeXY, 0.5*env_sizeXY, 0.5*env_sizeZ); //its size

G4LogicalVolume* logicEnv =
  new G4LogicalVolume(solidEnv,            //its solid
                      env_mat,             //its material
                      "Envelope");         //its name

new G4PVPlacement(0,                       //no rotation
                  G4ThreeVector(),         //at (0,0,0)
                  logicEnv,                //its logical volume
                  "Envelope",              //its name
                  logicWorld,              //its mother  volume
                  false,                   //no boolean operation
                  0,                       //copy number
                  checkOverlaps);          //overlaps checking
```

Embedded into Envelope

```
//
// Shape 1
//
G4Material* shape1_mat = nist->FindOrBuildMaterial("G4_A-150_TISSUE");
G4ThreeVector pos1 = G4ThreeVector(0, 2*cm, -7*cm);

// Conical section shape
G4double shape1_rmina =  0.*cm, shape1_rmaxa = 2.*cm;
G4double shape1_rminb =  0.*cm, shape1_rmaxb = 4.*cm;
G4double shape1_hz = 3.*cm;
G4double shape1_phimin = 0.*deg, shape1_phimax = 360.*deg;
G4Cons* solidShape1 =
  new G4Cons("Shape1",
  shape1_rmina, shape1_rmaxa, shape1_rminb, shape1_rmaxb, shape1_hz,
  shape1_phimin, shape1_phimax);

G4LogicalVolume* logicShape1 =
  new G4LogicalVolume(solidShape1,         //its solid
                      shape1_mat,          //its material
                      "Shape1");           //its name

new G4PVPlacement(0,                       //no rotation
                  pos1,                    //at position
                  logicShape1,             //its logical volume
                  "Shape1",                //its name
                  logicEnv,                //its mother  volume
                  false,                   //no boolean operation
                  0,                       //copy number
                  checkOverlaps);          //overlaps checking
```

Embedded into Envelope

```
//
// Shape 2
//
G4Material* shape2_mat = nist->FindOrBuildMaterial("G4_BONE_COMPACT_ICRU");
G4ThreeVector pos2 = G4ThreeVector(0, -1*cm, 7*cm);

// Trapezoid shape
G4double shape2_dxa = 12*cm, shape2_dxb = 12*cm;
G4double shape2_dya = 10*cm, shape2_dyb = 16*cm;
G4double shape2_dz  = 6*cm;
G4Trd* solidShape2 =
  new G4Trd("Shape2",                      //its name
            0.5*shape2_dxa, 0.5*shape2_dxb,
            0.5*shape2_dya, 0.5*shape2_dyb, 0.5*shape2_dz); //its size

G4LogicalVolume* logicShape2 =
  new G4LogicalVolume(solidShape2,         //its solid
                      shape2_mat,          //its material
                      "Shape2");           //its name

new G4PVPlacement(0,                       //no rotation
                  pos2,                    //at position
                  logicShape2,             //its logical volume
                  "Shape2",                //its name
                  logicEnv,                //its mother  volume
                  false,                   //no boolean operation
                  0,                       //copy number
                  checkOverlaps);          //overlaps checking

// Set Shape2 as scoring volume
//
fScoringVolume = logicShape2;
```



Primary Generator Action

```
G4int n_particle = 1;
fParticleGun  = new G4ParticleGun(n_particle);

// default particle kinematic
G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable();
G4String particleName;
G4ParticleDefinition* particle
  = particleTable->FindParticle(particleName="gamma");
fParticleGun->SetParticleDefinition(particle);
fParticleGun->SetParticleMomentumDirection(G4ThreeVector(0.,0.,1.));
fParticleGun->SetParticleEnergy(6.*MeV);
```

- Get a G4ParticleTable (list of lots of different particles that Geant4 already knows about), then find one called “gamma”
- Has proton, neutron, electron, alpha and heavier ions (mostly what we will be using)
- Our G4ParticleDefinition particle then has all the information we want about a gamma ray
- Our G4ParticleGun can then take this definition so we are loading ‘gamma ray ammo’ into the gun
- We define the direction we want to fire the gamma ray: along the z-axis
- We can set the energy to be 6 MeV

- Particle gun hasn’t been fired yet

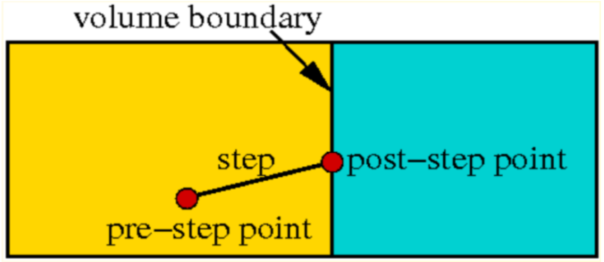
Generate Primaries

- Each time we fire, the generate primaries is called and what it does is find the envelope volume and randomly produce the gamma ray within 80% of the envelope

- We could also randomly change the direction/energy here if we desired – or make every n^{th} random particle a neutron/geantino. Whatever we want.

```
G4double envSizeXY = 0;
G4double envSizeZ = 0;

if (!fEnvelopeBox)
{
  G4LogicalVolume* envLV
    = G4LogicalVolumeStore::GetInstance()->GetVolume("Envelope");
  if ( envLV ) fEnvelopeBox = dynamic_cast<G4Box*>(envLV->GetSolid());
}

if ( fEnvelopeBox ) {
  envSizeXY = fEnvelopeBox->GetXHalfLength()*2.;
  envSizeZ = fEnvelopeBox->GetZHalfLength()*2.;
}
else  {
  G4ExceptionDescription msg;
  msg << "Envelope volume of box shape not found.\n";
  msg << "Perhaps you have changed geometry.\n";
  msg << "The gun will be place at the center.";
  G4Exception("B1PrimaryGeneratorAction::GeneratePrimaries()",
   "MyCode0002",JustWarning,msg);
}

G4double size = 0.8;
G4double x0 = size * envSizeXY * (G4UniformRand()-0.5);
G4double y0 = size * envSizeXY * (G4UniformRand()-0.5);
G4double z0 = -0.5 * envSizeZ;

fParticleGun->SetParticlePosition(G4ThreeVector(x0,y0,z0));

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

Stepping Action



- Each time we have an interaction, we get a step
- When we have a boundary between volumes we will have a step
- Stepping Action therefore runs at each step:
  - Checks to see what volume our particle is currently in (current step)
  - Accumulates the energy deposited in this step for the scoring volume – keeps track with an Event Action object

```
{
  if (!fScoringVolume) {
    const B1DetectorConstruction* detectorConstruction
      = static_cast<const B1DetectorConstruction*>
        (G4RunManager::GetRunManager()->GetUserDetectorConstruction());
    fScoringVolume = detectorConstruction->GetScoringVolume();
  }

  // get volume of the current step
  G4LogicalVolume* volume
    = step->GetPreStepPoint()->GetTouchableHandle()
      ->GetVolume()->GetLogicalVolume();

  // check if we are in scoring volume
  if (volume != fScoringVolume) return;

  // collect energy deposited in this step
  G4double edepStep = step->GetTotalEnergyDeposit();
  fEventAction->AddEdep(edepStep);
}
```

Event Action

- This is performed at the end of an event (i.e. primary generator has been created and all secondary particles have stopped or have exited our world)
- Accumulates all the energy deposited during the steps in the scoring volume
- Feeds into a Run Action object

```
{
  // accumulate statistics in run action
  fRunAction->AddEdep(fEdep);
}
```

Run Action

- Takes all of the Event Action energy deposits (event-by-event) and calculates a total dose that the phantom patient receives
- Prints dose after all of our gamma rays have been fired

```
{
  G4int nofEvents = run->GetNumberOfEvent();
  if (nofEvents == 0) return;

  // Merge accumulables
  G4AccumulableManager* accumulableManager = G4AccumulableManager::Instance()
  accumulableManager->Merge();

  // Compute dose = total energy deposit in a run and its variance
  //
  G4double edep  = fEdep.GetValue();
  G4double edep2 = fEdep2.GetValue();

  G4double rms = edep2 - edep*edep/nofEvents;
  if (rms > 0.) rms = std::sqrt(rms); else rms = 0.;

  const B1DetectorConstruction* detectorConstruction
   = static_cast<const B1DetectorConstruction*>
     (G4RunManager::GetRunManager()->GetUserDetectorConstruction());
  G4double mass = detectorConstruction->GetScoringVolume()->GetMass();
  G4double dose = edep/mass;
  G4double rmsDose = rms/mass;

  // Run conditions
  //  note: There is no primary generator action object for "master"
  //        run manager for multi-threaded mode.
  const B1PrimaryGeneratorAction* generatorAction
   = static_cast<const B1PrimaryGeneratorAction*>
     (G4RunManager::GetRunManager()->GetUserPrimaryGeneratorAction());
  G4String runCondition;
  if (generatorAction)
  {
    const G4ParticleGun* particleGun = generatorAction->GetParticleGun();
    runCondition += particleGun->GetParticleDefinition()->GetParticleName();
    runCondition += " of ";
    G4double particleEnergy = particleGun->GetParticleEnergy();
    runCondition += G4BestUnit(particleEnergy,"Energy");
  }

  // Print
  //
  if (IsMaster()) {
    G4cout
     << G4endl
     << "--------------------End of Global Run-----------------------";
  }
  else {
    G4cout
     << G4endl
     << "--------------------End of Local Run------------------------";
  }

  G4cout
     << G4endl
     << " The run consists of " << nofEvents << " "<< runCondition
     << G4endl
     << " Cumulated dose per run, in scoring volume : "
     << G4BestUnit(dose,"Dose") << " rms = " << G4BestUnit(rmsDose,"Dose")
     << G4endl
     << "------------------------------------------------------------"
     << G4endl
     << G4endl;
}
```

Running it all via ‘main’

- We can run either in interactive mode or not (visualization options/can give macro commands in command line), multithreaded mode or not
  - Define randomness (important for MC)
  - Define runManager which takes everything together
  - Feed in our B1DetectorConstruction
  - Tell it what physics to include (many different physics lists that work well for different energy regimes and particles)
  - Feed in our ActionInitialization which just defines our EventAction, SteppingAction and RunAction
  - Can define visualization of our detectors and the particle tracks in 3D (not advisable over VPN)
  - Give the program a macro file

```
int main(int argc,char** argv)
{
  // Detect interactive mode (if no arguments) and define UI session
  //
  G4UIExecutive* ui = 0;
  if ( argc == 1 ) {
    ui = new G4UIExecutive(argc, argv);
  }

  // Choose the Random engine
  G4Random::setTheEngine(new CLHEP::RanecuEngine);

  // Construct the default run manager
  //
#ifdef G4MULTITHREADED
  G4MTRunManager* runManager = new G4MTRunManager;
#else
  G4RunManager* runManager = new G4RunManager;
#endif

  // Set mandatory initialization classes
  //
  // Detector construction
  runManager->SetUserInitialization(new B1DetectorConstruction());

  // Physics list
  G4VModularPhysicsList* physicsList = new QBBC;
  physicsList->SetVerboseLevel(1);
  runManager->SetUserInitialization(physicsList);

  // User action initialization
  runManager->SetUserInitialization(new B1ActionInitialization());

  // Initialize visualization
  //
  G4VisManager* visManager = new G4VisExecutive;
  // G4VisExecutive can take a verbosity argument - see /vis/verbose guidance.
  // G4VisManager* visManager = new G4VisExecutive("Quiet");
  visManager->Initialize();

  // Get the pointer to the User Interface manager
  G4UImanager* UImanager = G4UImanager::GetUIpointer();

  // Process macro or start UI session
  //
  if ( ! ui ) {
    // batch mode
    G4String command = "/control/execute ";
    G4String fileName = argv[1];
    UImanager->ApplyCommand(command+fileName);
  }
  else {
    // interactive mode
    UImanager->ApplyCommand("/control/execute init_vis.mac");
    ui->SessionStart();
    delete ui;
  }

  // Job termination
  // Free the store: user actions, physics_list and detector_description are
  // owned and deleted by the run manager, so they should not be deleted
  // in the main() program !

  delete visManager;
  delete runManager;
```

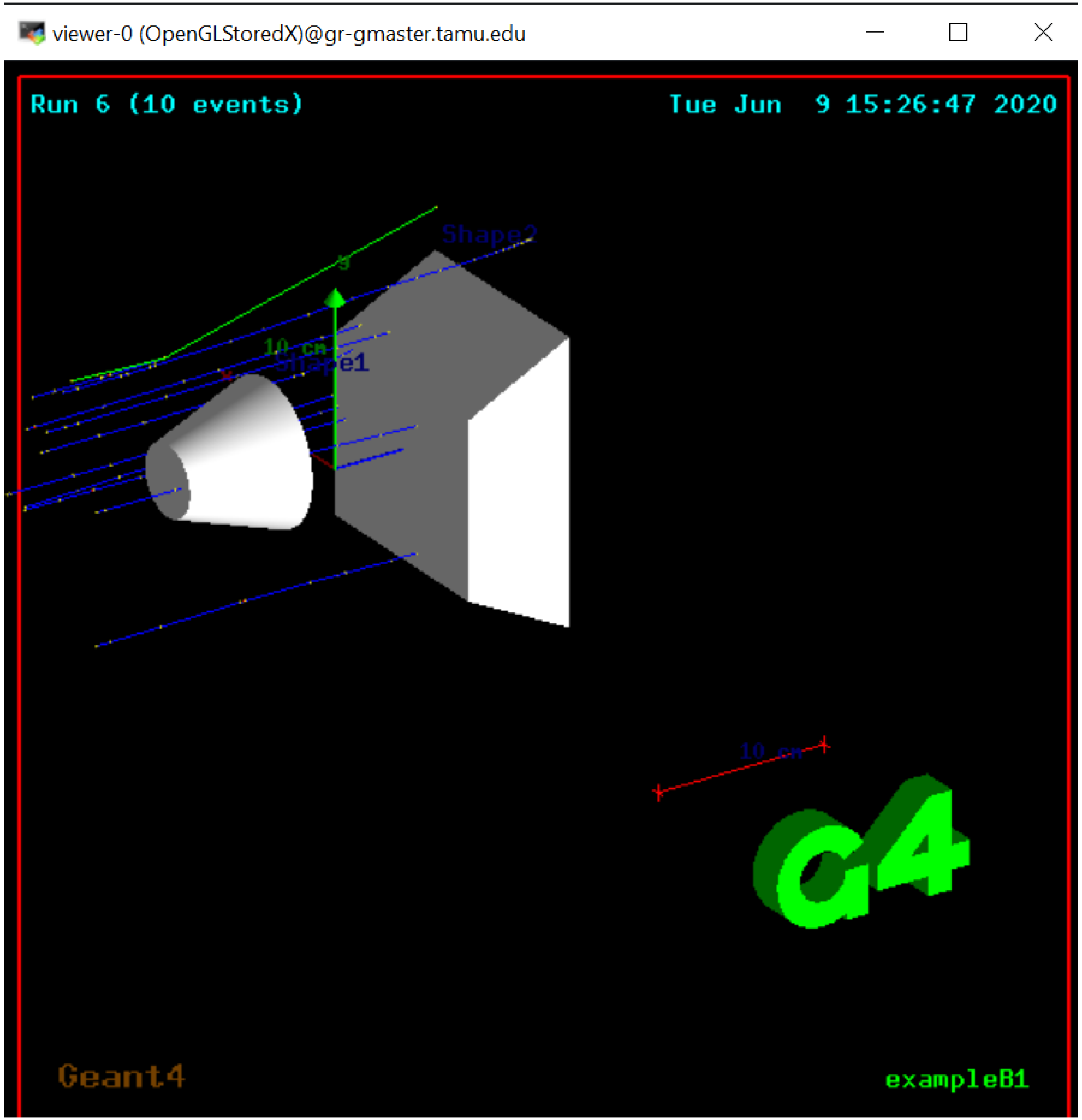
Macro files

- Macro files can:
- Launch the visualization window and set view angles, color options for different particles, hide different particles, wireframes, rotate in 3D and save to file to make gifs etc.
- Be used to set the particle gun parameters and fire
- Change verbosity for different parts of the program

```
# Macro file for example B1
#
# Can be run in batch, without graphic
# or interactively: Idle> /control/execute run1.mac
#
# Change the default number of workers (in multi-threading mode)
#/run/numberOfWorkers 4
#
# Initialize kernel
/run/initialize
#
/control/verbose 2
/run/verbose 2
/event/verbose 0
/tracking/verbose 1
#
# gamma 6 MeV to the direction (0.,0.,1.)
#
/gun/particle gamma
/gun/energy 6 MeV
#
/run/beamOn 5
#
# proton 210 MeV to the direction (0.,0.,1.)
#
/gun/particle proton
/gun/energy 210 MeV
/tracking/verbose 2
#
/run/beamOn 1
```

Making the file

- mkdir build
- cd build
- cmake .. (I prefer ccmake .. then use “c”, “c”, “g” to compile and generate)
- make
- We then get the exampleB1 file
- Run with ./exampleB1 and the visualization will slowly pop up
- In the command prompt, we can type /control/execute run1.mac and our macro will run
- Alternatively, we can avoid the visualization by ./exampleB1 run1.mac




```
****************************************************************************************
* G4Track Information:   Particle = gamma,   Track ID = 1,   Parent ID = 0
****************************************************************************************

Step#    X(mm)    Y(mm)    Z(mm) KinE(MeV)  dE(MeV) StepLeng TrackLeng  NextVolume ProcName
    0    -68.7    -22.7     -150         6        0        0         0    Envelope initStep
    1    -68.7    -22.7      150         6        0      300       300       World Transportation
    2    -68.7    -22.7      180         6        0       30       330  OutOfWorld Transportation

****************************************************************************************
* G4Track Information:   Particle = gamma,   Track ID = 1,   Parent ID = 0
****************************************************************************************

Step#    X(mm)    Y(mm)    Z(mm) KinE(MeV)  dE(MeV) StepLeng TrackLeng  NextVolume ProcName
    0     70.6    -54.7     -150         6        0        0         0    Envelope initStep
    1     70.6    -54.7      150         6        0      300       300       World Transportation
    2     70.6    -54.7      180         6        0       30       330  OutOfWorld Transportation

****************************************************************************************
* G4Track Information:   Particle = gamma,   Track ID = 1,   Parent ID = 0
****************************************************************************************

Step#    X(mm)    Y(mm)    Z(mm) KinE(MeV)  dE(MeV) StepLeng TrackLeng  NextVolume ProcName
    0    -19.3     77.4     -150         6        0        0         0    Envelope initStep
    1    -19.3     77.4      150         6        0      300       300       World Transportation
    2    -19.3     77.4      180         6        0       30       330  OutOfWorld Transportation

****************************************************************************************
* G4Track Information:   Particle = gamma,   Track ID = 1,   Parent ID = 0
****************************************************************************************

Step#    X(mm)    Y(mm)    Z(mm) KinE(MeV)  dE(MeV) StepLeng TrackLeng  NextVolume ProcName
    0    -47.4     52.3     -150         6        0        0         0    Envelope initStep
    1    -47.4     52.3     64.7         6        0      215       215      Shape2 Transportation
    2    -47.4     52.3      100         6        0     35.3       250    Envelope Transportation
    3    -47.4     52.3      136     0.795        0     36.3       286    Envelope compt
    4    -57.4       58      142      0.74        0     12.8       299    Envelope compt
    5    -79.5     62.6      150      0.74        0       24       323       World Transportation
    6     -120     70.9      165      0.74        0     43.9       367  OutOfWorld Transportation

****************************************************************************************
* G4Track Information:   Particle = e-,   Track ID = 3,   Parent ID = 1
****************************************************************************************

Step#    X(mm)    Y(mm)    Z(mm) KinE(MeV)  dE(MeV) StepLeng TrackLeng  NextVolume ProcName
    0    -57.4       58      142    0.0541        0        0         0    Envelope initStep
    1    -57.4       58      142         0   0.0541     0.05      0.05    Envelope eIoni
```

```
=====================================================================

### Run 1 starts.

****************************************************************************************
* G4Track Information:   Particle = proton,   Track ID = 1,   Parent ID = 0
****************************************************************************************

Step#    X(mm)    Y(mm)    Z(mm) KinE(MeV)  dE(MeV) StepLeng TrackLeng  NextVolume ProcName
    0     30.4     25.8     -150       210        0        0         0    Envelope initStep
    1     30.9     25.7    -92.6       185     24.8     57.4      57.4    Envelope hIoni
    2     31.3     25.7    -64.5       172     13.5     28.1      85.5      Shape1 Transportation
    3     32.1     26.2      -40       158     13.9     24.5       110    Envelope Transportation
    4       33     26.8    -8.49       142     16.2     31.5       142    Envelope hIoni
    5     33.4     27.4     20.4       125       17     28.9       170    Envelope hIoni
    6     33.6     27.9       40       111     13.5     19.6       190      Shape2 Transportation
    7     33.8     28.1       51        97     14.1       11       201      Shape2 hIoni
    8     34.1     28.2     59.6      84.5     12.5     8.67       210      Shape2 hIoni
    9     34.2     28.4     66.5      74.2     10.3     6.82       217      Shape2 hIoni
   10     34.3     28.6     71.9      65.4     8.73     5.44       222      Shape2 hIoni
   11     34.1     28.8     76.3      57.5     7.96     4.38       226      Shape2 hIoni
   12     33.9       29     79.8      50.2      7.3      3.5       230      Shape2 hIoni
   13     33.7     29.2     82.5      43.9     6.25     2.78       233      Shape2 hIoni
   14     33.7     29.3     84.7      38.2     5.72     2.22       235      Shape2 hIoni
   15     33.6     29.4     86.5      33.1     5.11     1.76       237      Shape2 hIoni
   16     33.6     29.4     87.9      28.9     4.23      1.4       238      Shape2 hIoni
   17     33.6     29.5       89      24.8     4.01     1.13       239      Shape2 hIoni
   18     33.6     29.6     89.9      21.3      3.5    0.896       240      Shape2 hIoni
   19     33.6     29.6     90.6      17.9     3.43    0.719       241      Shape2 hIoni
   20     33.6     29.7     91.2      15.2     2.73    0.566       241      Shape2 hIoni
   21     33.6     29.7     91.6      12.5     2.65     0.46       242      Shape2 hIoni
   22     33.6     29.7       92      9.72     2.81    0.369       242      Shape2 hIoni
   23     33.6     29.7     92.3       7.2     2.53    0.287       242      Shape2 hIoni
   24     33.5     29.7     92.5      4.58     2.62    0.222       243      Shape2 hIoni
   25     33.5     29.7     92.7      1.52     3.05    0.154       243      Shape2 hIoni
   26     33.5     29.8     92.7         0     1.52   0.0289       243      Shape2 hIoni
 Run terminated.
Run Summary
  Number of events processed : 1
  User=0.01s Real=0.54s Sys=0s

--------------------End of Global Run-----------------------
 The run consists of 1 proton of 210 MeV
 Dose in scoring volume : 10.27620774222224 picoGy  +- 0 picoGy
------------------------------------------------------------
```

Can you try changing:

- The particles fired (neutron/alpha/pion)?
- The energy

macro

- The size of the 'body' components
- Make one of them out of something else
- Change the scoring volume

source code