Geant4

Lecture 2: Geometry

Overview

3-stage process:

- Solid - the shape and its dimensions
- Logical volume - include materials
- Physical volume - position it in space

…

Making Solids

Simplest(?) example: G4Box
G4Box* G1=new G4Box('boxname',x,y,z);// x,y,z are the half-widths

Also common:
G4Tubs(name,rmin,rmax,length,phimin,phimax);// tube section
G4Cons(name,rmin1,rmax1,rmin2,rmax2,length,phimin,phimax);// cone section
G4Trd(name,x1,x2,y1,y2,length); //trapezoid
G4Orb(name, r); // solid sphere
G4Sphere(name,rmin,rmax,phimin,phimax,thetamin,thetamax); // spherical shell

Maybe more obscure
G4CutTubs for sliced tube, G4Para for parallellepiped, G4Trap for general trapezoid, G4Torus, G4EllipticalTube,, G4Ellipsoid, G4EllipticalCone, G4Paraboloid, G4Hype for hyperboloid, G4Tet for tetrahedron, G4TwistedBox, G4TwistedTrap and G4TwistedTrd for twisted trapezoid, G4GenericTrap , G4TwistedTub

Generic solids: G4Polycone, G4Polyhedra, G4ExtrudedSolid, G4TesselatedSolid, G4TriangularFacet, G4QuadranguarFacet

If you need a shape it is probably there, as somebody in the past will have needed it

A short lesson on C++ pointers (apologies to those who know it already)

Various types: int, float, char, G4int… and defined classes like G4Box

```
int j; float x,y,z; G4float r,theta, phi; G4Box myBox;
```

There are also pointers to these types:

```
int* pj; float *px,*py,*pz; G4float *r, *theta, *phi;G4Box* BX;
```

Conversions: syntax allows

```
       j= * pj; // j is contents-of pj
       pj = & j ; // pj is address-of j
```

often easier to pass around pointers rather than the actual contents - specially in function arguments

Note * is also multiplication so brackets often necessary/helpful

```
x=y+(*pz);
```

Handy feature of notation

```
G4Box* pbox;
```

`(*pbox).GetCubicVolume()` same as `pbox -> GetCubicVolume()`

Pointers in G4
`new` creates object and returns pointer

Why `G4Box* G1=new G4Box(‘boxname’,x,y,z);`
not `G4Box G1(‘boxname’,x,y,z);`?

Because that would get deleted on exit from the function.
These objects are put on a stack behind the scenes.

That’s why you need ‘boxname’, to refer to it later

Solids: Boolean combinations

Union ('or') Intersection ('and') subtraction ('and not')

```
G4UnionSolid* union = new G4UnionSolid("Box+Cylinder", box, cyl);
G4IntersectionSolid* intersection =new G4IntersectionSolid("Box*Cylinder", box, cyl);
G4SubtractionSolid* subtraction =new G4SubtractionSolid("Box-Cylinder", box, cyl);
```

Warning - avoid combining solids that share surfaces

More advanced example: shifts the 2nd solid

```
G4RotationMatrix* yRot = new G4RotationMatrix;
yRot->rotateY(-M_PI/4.*rad);
G4ThreeVector zTrans(0, 0, 50);
G4Transform3D transform(yRot, zTrans);
G4UnionSolid* unionMoved =
new G4UnionSolid("Box+CylinderMoved", box, cyl, transform);
```

Note new classes: G4ThreeVector, G4RotationMatrix, G4Transform3D

Worry about sense of rotation and shift and their order
You can also specify the rotation and translation separately -

Logical Volume

```
G4LogicalVolume(G4Solid*,G4Material*,name)
```

Again: declare using **`new`**

Defining a material

Materials are made of elements

Elements are made of isotopes. So define isotopes, then define elements, then define materials.

But there are shortcuts!

Isotopes (easy)

`G4Isotope* deuterium=new G4Isotope(“deut”,1,2);`
Aside: documentation says `G4Isotope(“deut”,z=1,n=2);`
This works, but is just trying to be helpful; `G4Isotope(“deut”,n=2,z=1);` compiles but gives z=2,n=1

To see what’s REALLY happening, don't just look at the documentation, look at the code. It’s in
`/home/software/geant/geant4.10.03/source/materials/include/G4Isotope.hh`

Find class `G4Isotope` and then the constructor(s)
The constructor is a function whose name is the name of the class, and has noreturn value (not even void!

```
class G4Isotope
{
 public:  // with description
    // Make an isotope
    //
    G4Isotope(const G4String& name,                 //its name
                    G4int     z,                    //atomic number
                    G4int     n,                    //number of nucleons
                    G4double  a = 0.,               //mass of mole
                    G4int     m = 0);               //isomer level
```

Elements

Two ways to do this

1) For more than one isotope

```
G4Element* H= new G4Element(‘hydrogen’,’H’,2);
H -> AddIsotope(deuterium,.0001);
H -> AddIsotope(normalH,.9999);
```

2) Shortcut for just one isotope: give Z and A

```
G4Element* H= new G4Element(‘hydrogen’,’H’,1,1);
```

Materials

3 ways of defining a material

1) From a single element (shortcut)

```
G4Material* Hmat=new G4Material(‘hydro’,1,1,1.234);
```

arguments are Z,A,density

2) As a mixture of elements and materials

```
G4Material* Hmat=new G4Material(‘hydro’,1.234,2);
```

followed by calls to

```
    Hmat->AddElement(element,n)
or  Hmat->AddElement(element,fraction)
or  Hmat->AddMaterial(mat,fraction)
```

Note: n, number of atoms, must be integer, f, mass fraction, must not
Water could be (with O,H defined) from (O,1), and (H,2) or (O,0.89) and (H,0.11)

3) From an existing material but changing the density

```
G4Material* Hnew=new G4Material(‘hydro-X’,1.324,Hmat);
```

Another aside about linux

- Finding stuff: two powerful tools
- `grep` *`string file`*
- file may involve wildcards: `grep Detector *.cc`
- often used recursively: `grep -r Detector *.cc` will look in subdirectories and sub-subdirectories
- Search for a file `find` *`path filename`*
- Generates lots of useless messages: filter with `find` *`path filename`* `| grep` *`filename`*

Another aside about C++

You are liable to get messages saying ‘can't find the function…’

Maybe you got the function name wrong: more likely you did not give it the right arguments. C++ arguments must match exactly

BUT

There may be different versions of the function with different arguments which do different things

Arguments are optional if default values given in the .hh file

If no match found, the compiler will try applying rules for ‘casting’ one type to another.

Shortcut for common materials

Water and Air etc are known about and can be summoned. (NIST isthe National Institute of Standards and Technology)

```
G4NistManager* man = G4NistManager::Instance();

G4Material* water=man-> FindOrBuildMaterial(“G4_WATER”);

G4Material* air = man ->FindOrBuildMaterial(“G4_AIR”);
```

Check results by printout

```
 Can do this from inside the program:
G4cout << *(G4Material::GetMaterialTable());
Or by commands (at the console or in input file)
/material/nist/listmaterials all
```

Units

What units does G4 use? You don't need to know!

Useful units so you don't have to remember

g, mg, kg…, cm3,… meter, mm… perCent

So say den=1.234*g/cm3; or den=1234.0 mg/cm3;

For a complete list: command /units/list or look in source files: global/management/include/G4SystemOfUnits.hh

Physical Volume (1)

Basic idea: Create a physical volume and add logical volumes

1) Create logical volume - call it LV

2) Create transform. Can be done in various ways

```
G4RotationMatrix* rot = new G4RotationMatrix(phi,theta,psi);
G4ThreeVector offset(x, y, z);
```

3) Do it

```
G4VPhysicalVolume* PhysVol=new
G4PVPlacement(rot,offset,"volname",LV,parent);
```

`parent` is another physical volume, or 0 for the 'world'. Idea is that you position (say) the drift chamber and the calorimeter and the muon detector within the overall experiment, then individal cells within the drift chamber, then wires within the cells…

Physical Volume (2)

Adding a second logical volume to your physicalvolume

1) Create logical volume - call it LV2

2) Add it. Note that the value is thrown away

```
new G4PVPlacement(rot,offset,LV2,’name’,parent,0,0);
```

3) To add a third - say, another LV2

```
new G4PVPlacement(rot2,offset2,LV2,’othername’,parent,0,1);
```

BTW if there is no rotation, just use 0 for rot

Different Styles

- In placement, parent can be a physical volume or a logical volume

- Top down: create world, then create and place daughter volumes (e.g. beamline, target, detector), then create and place grand-daughters (e.g. collimator, window, magnet…) positioning in physical volume

- Bottom-up: define collimator, window, magnet…. Then define logical beamline and position components, likewise target, detector. Finally define world and position daughters

Assessment

- Define a nuclear reactor with cylindrical fuel rods. Each has a radius of 1 cm and is surrounded by a 1 mm steel casing. It is composed of uranium dioxide, uranium trioxide, or diuranium pentoxide (to be assigned in class). The rods are 1.5 m long and there are 100 of them arranged in a square grid of dimension 5 cm. They are in a tank of heavy water, 2m by 2m by 2m.The uranum is 5% ^{235}U and 95% ^{238}U.

- Construct the geometry. Fire some neutrons at it and see what happens.