猪头 Geant4 讲座第一讲——预备知识

先来两句套话，本人知识有限，不当之处请大家多多指教:P~~~

预备知识，不爱聊天兄已经说过了，C++
这里我重点说一下，重点要学习 C++中的类的使用，包括类的调用、类的初始化、类的重构等

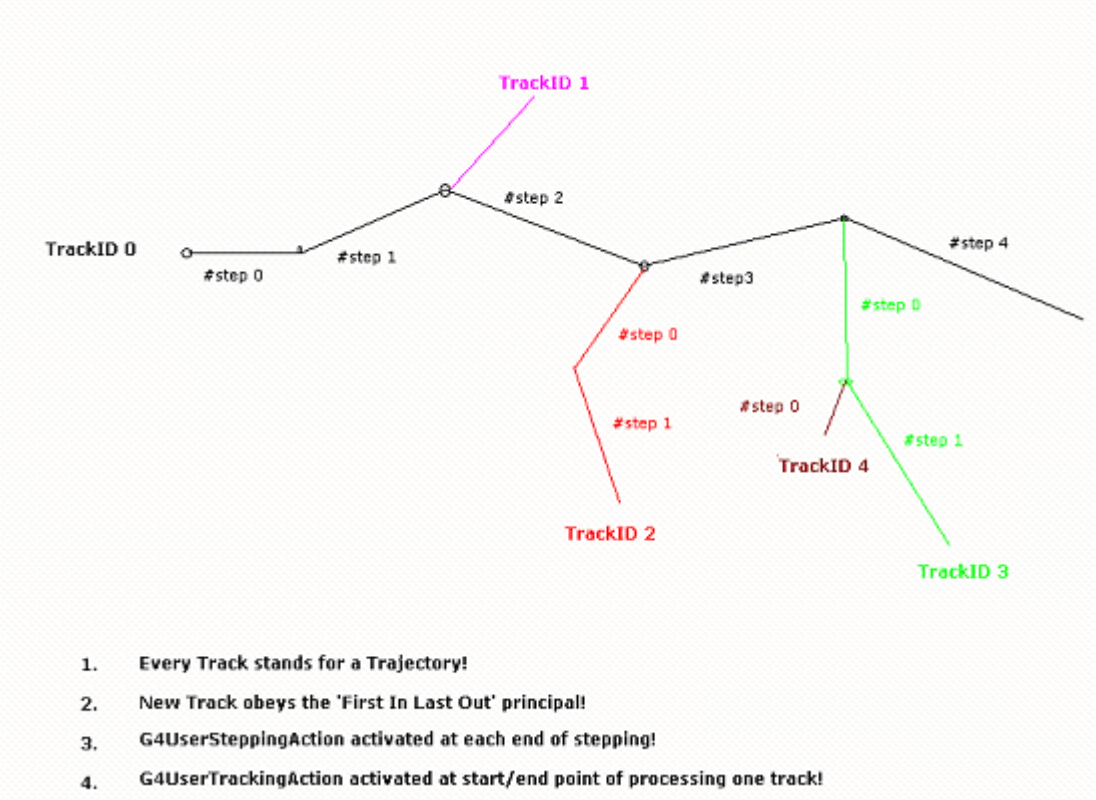
其次，基本物理概念和基本 Linux 基础也是必要的

当然，Geant4 是要安装好的，G4 手册是要有的。从哪儿找，怎么装，这些我都不再具体叙述了
大家自己翻前面的帖子吧。

猪头 Geant4 讲座第二讲——模拟算法

我们要学习 Geant4 首先应该学习 G4 是如何处理模拟过程的。

在 G4 中一个典型的模拟算法是这样的。
首先建立一次模拟，在 G4 中称为一次 Run
Run 建立后，需要对几何结构、物理过程进行初始化
初始化完成后就开始模拟过程了，
首先发射一个粒子，每一步都按照蒙卡方法进行模拟，具体模拟方法请参阅裴鹿成或许淑艳老师的书
这里不具体讲，因为不是重点
在 G4 中，发射一个（或一系列）粒子到所有次级粒子死亡的过程成为一次 Event。
而每次发射的初始粒子则有粒子发射器进行控制。
而在每一个 event 过程中，粒子与材料反应后会可能生成多个次级粒子，每个粒子都会有一条径迹，称之为 track
而每一个粒子（初始的或次级的）的径迹又是由很多步组成的，称之为 step
关于 track 和 step 的理解请参阅下图



最后总结一下，G4 模拟的基本算法是

A Run Start -> 初始化物理模型/几何模型
-> An Event Start -> 调用粒子发射器发射粒子
-> A Track Start
-> A Step Start
-> A Step End
-> Next Step Start
-> ……
-> All Step End
-> A Track End
-> Next Track Start
-> ……
-> All Track End
-> An Event End
-> Next Event Strat
-> ……
-> All Event End(All Primaries Shot)
-> A Run End
-> Next Run Start
-> ……

猪头 Geant4 讲座第三讲——程序结构及基本类

G4 是采用的 gcc 编译器，因此其程序结构是和 C++一样的。
首先包括有一个主程序 main，然后分别包含有子程序 src 和头文件 include
以及其他调用文件 others
G4 里面为了与 C++相区别，程序后缀都是.cc，头文件后缀都是.hh
其中头文件在.cc 里面写也没问题，但是那样看起来不方便，建议还是按照 c++的习惯一一对应比较好。
那么，关键问题是要进行一个模拟我们都需要写哪些 src 和 include 的？
下面我们首先看 G4 里面的几个基本类，这些基本类基本上是与 src 一一对应的。

G4RunManager——对应主程序
这个类在主程序中用以初始化模拟信息的，或者形象地说是用于连接子程序的，
而连接方式是通过 Set 函数来完成的
大家可以从$G4INSTALL/source/run/G4RunManager.hh 里面查看各种 Set 函数，如

```
  public: // with description
    inline void SetUserInitialization(G4VUserDetectorConstruction* userInit)
    { userDetector = userInit; }
    inline void SetUserInitialization(G4VUserPhysicsList* userInit)
    {
      physicsList = userInit;
      kernel->SetPhysics(userInit);
    }
```

可以说 G4RunManager 类是贯穿整个程序模拟过程的总线，因此一般说来只能有一个
而开始一次 Run 的信号则是通过 BeamOn 函数发出的，其格式是

```
    virtual void BeamOn(G4int n_event,const char* macroFile=0,G4int n_select=-1);
```

可以通过多次调用 BeamOn 来实现循环计算。

其余子类包括几何结构类、物理设定类、粒子发射器类（源描述类）、事件处理类、径迹处理类等。
这些类可以按照两种不同的分类方式分类，每种分类方式都可以分为两类。

子类按照重要性分为强制类和可选类。
其中几何结构类（DetectorConstruction）、物理设定类（PhysicsList）、
源描述类（PrimaryGenerator）都属于强制类，是必须有的，缺少任一个程序都无法运行。
而事件处理类（EventAction）、步数据处理类（SteppingAction）、径迹处理类（TrackingAction）、运行处理类（RunAction）
都属于可选类，用户可以通过设定这些类来获取感兴趣的信息，虽然没有这些类程序一样可以运行，但是如果一个都没有的话，
这样的模拟是没有意义的，除非是用来检验几何结构的完备性。

子类按照调用过程分为初始化类和用户干涉类
其中几何结构类（DetectorConstruction）、物理设定类（PhysicsList）属于初始化类，
这两个类在每次 Run 开始后就对模拟过程进行初始化，之后在粒子发射开始后（BeamOn 开始后）就不再继续调用了。
而其余的类则是在每发射一个粒子后都需要调用的，具体调用过程请复习第二讲中的模拟算法。

下面讲一个特殊的类。

G4UImanager——UI（user interface）管理器
这个类的英文全名是 user interface manager，顾名思义是用于人机交互的，用户可以通过这个类来向模拟过程发出“命令”，
这个类在一定程度上使用很简洁的命令语句代替了上述用户干涉类的功能。

需要说明的是，上面所说的几个类并没有包括所有可选类，根据每个用户的需求，用户可以有很多其他类。

下面以$G4INSTALL/example/novice/N01 的例子为例看一下这些类，
因为这个粒子仅为了说明 Geant4 的基本运用，所以这个例子里面只定义了 3 个强制类，没有使用可选类来输出感兴趣的信息。
但是通过 UI 管理器查看了粒子输运过程中的一些信息。

```
// Construct the default run manager
 //首先是要创建一个运行管理器的实例，根据 C++的知识我们知道每个类都必须通过他的实例来进行操作
 G4RunManager* runManager = new G4RunManager;

 // set mandatory initialization classes
 //通过 Set 函数设置几何结构和物理过程，这两个类属于强制类，也属于初始化类
 G4VUserDetectorConstruction* detector = new ExN01DetectorConstruction;
 runManager->SetUserInitialization(detector);
 //
 G4VUserPhysicsList* physics = new ExN01PhysicsList;
 runManager->SetUserInitialization(physics);

 // set mandatory user action class
 //通过 Set 函数进行源描述，这个类属于强制类，也属于用户干涉类
 G4VUserPrimaryGeneratorAction* gen_action = new ExN01PrimaryGeneratorAction;
 runManager->SetUserAction(gen_action);

 // Initialize G4 kernel
 //Set 完成后，通过 Initialize()函数初始化 G4 内核
 runManager->Initialize();
 // Get the pointer to the UI manager and set verbosities
 //创建 UI 管理器，通过 UI 管理器设置运行、事件、径迹的可视化精细程度。
 G4UImanager* UI = G4UImanager::GetUIpointer();
 UI->ApplyCommand("/run/verbose 1");
 UI->ApplyCommand("/event/verbose 1");
 UI->ApplyCommand("/tracking/verbose 1");
 // Start a run
 //通过 BeamOn()函数开始模拟，这里 macroFile 和 n_Select 都是使用了默认值
 G4int numberOfEvent = 3;
 runManager->BeamOn(numberOfEvent);
```

猪头 Geant4 讲座第四讲——基本单位

在进行后续讲座之前，我们首先了解一下 Geant4 中的单位，

在 Geant4 中的每个物理量虽然都有默认单位，但是每个物理量还可以带任意用户方便的单位，比如长度单位默认是 mm，但用户可以改为 m。

物理量的默认单位是采用的 Hep（High-Energy Physics ）中的单位：

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)

G4 中对每个单位都用常变量进行了定义，可以很方便地对物理量进行描述方法。

而且大部分常用单位都有全名和缩写两种。

比如定义一个 10cm 的长度可以用下述几种方法描述。

```
G4double length=100.;

G4double length=100. * mm;

G4double length=100. * millimeter;

G4double length=10. * cm;

G4double length=10. * centimeter;
```

这几种描述方法是等价的。

具体的单位设置可以看$G4INSTALL/source/global/management/include/G4SystemOfUnits.hh，这个文件是引用的 CLHEP 的一部分，其来源是$CLHEP/Units/SystemOfUnits.h。

猪头 Geant4 讲座第五讲——材料定义

几何结构类（DetectorConstruction）属于强制初始化类，其主要功能是构建模拟问题的几何结构，包括各部分的材料、形状、尺寸、位置等信息。

因此，在这个类里面我们就必须完成上述几个信息的设置工作。

首先，材料定义。

材料可以分为单质和化合物（混合物）两种。而不管是单质还是化合物都是由元素组成的，因此在定义材料前，必须首先定义元素。而元素的定义将决定在模拟过程中需要使用的截面库的选择（大部分是自动选择的，这里不重点讲）。

那么下面我们来看如何定义元素。

我们知道，每一种元素都可能有多个同位素，但是所有这些同位素的原子序数（核内质子数）都是相同的，其摩尔质量也可以根据各个同位素所占份额计算出来。因此，只需要有原子序数 Z 和摩尔质量 A 就可以定义出一个元素。这就是元素的直接定义法，参考$G4INSTLL/source/materials/include/G4Element.hh，如下：

```
  // Constructor to Build an element directly; no reference to isotopes
  //
  G4Element(const G4String& name,  //its name
            const G4String& symbol,  //its symbol
                G4double   Zeff,  //atomic number
                G4double   Aeff);  //mass of mole
```

其中元素名称和符号只是个标记，并不会影响元素的物理性质。

$G4INSTALL/example/novice/N02 中氮元素的定义就是采用的直接定义法。

```
  G4Element* N = new G4Element("Nitrogen", "N", z=7., a= 14.01*g/mole);
```

此外，既然每种元素都是由不同的同位素组成的，那如果事先定义了同位素，加上每个同位素所占份额不也可以确定一种元素，而不必麻烦地去计算摩尔质量吗？

确实如此，在 Geant4 中同样提供了另一种定义元素的方法，我将之称为间接定义法。

同样参考$G4INSTLL/source/materials/include/G4Element.hh，如下：

```
  // Constructor to Build an element from isotopes via AddIsotope
  //
  G4Element(const G4String& name,  //its name
            const G4String& symbol,  //its symbol
            G4int nbIsotopes);   //nb of isotopes
  void AddIsotope(G4Isotope* isotope,   //isotope
                  G4double    RelativeAbundance); //fraction of nb of
                                                  //atomes per volume
```

而同位素的定义则参考$G4INSTLL/source/materials/include/G4Isotope.hh

```
    G4Isotope(const G4String& name,  //its name
                    G4int      z,  //atomic number
                    G4int      n,  //number of nucleons
                    G4double   a = 0.);  //mass of mole
```

$G4INSTALL/example/novice/N03 中铀元素的定义就是采用的间接定义法。

```
// define an Element from isotopes, by relative abundance
//
G4Isotope* U5 = new G4Isotope("U235", iz=92, n=235, a=235.01*g/mole);
```

```
G4Isotope* U8 = new G4Isotope("U238", iz=92, n=238, a=238.03*g/mole);
G4Element* U  = new G4Element("enriched Uranium",symbol="U",ncomponents=2);
U->AddIsotope(U5, abundance= 90.*perCent);
U->AddIsotope(U8, abundance= 10.*perCent);
```

定义完元素之后我们就可以定义材料了。

前面我们说了，材料可以分为单质（只有一种元素）和化合物（或混合物，含有两种或两种以上的元素）。不管是单质还是化合物，如果我们知道其中每种元素的份额（单质可以认为其组成元素的份额是 100%），那我们就可以确定这种材料的组成了。再加上密度等信息就可以确定这种材料的具体状态了。这就是 Geant4 中的一种材料定义法。而 Geant4 中为了方便定义材料，将份额还分为了两种，分别是原子数份额和质量份额。

定义方法参考$G4INSTLL/source/materials/include/G4Material.hh 如下：

```
  // Constructor to create a material from a combination of elements
  // and/or materials subsequently added via AddElement and/or AddMaterial
  //
  G4Material(const G4String& name,    //its name
             G4double   density,      //density
             G4int      nComponents,  //nbOfComponents
             G4State    state     = kStateUndefined, //solid,gas
             G4double   temp      = STP_Temperature, //temperature
             G4double   pressure = STP_Pressure);  //pressure
  //
  // Add an element, giving number of atoms
  //
  void AddElement(G4Element* element,    //the element
                  G4int      nAtoms);    //nb of atoms in
          // a molecule
  //
  // Add an element or material, giving fraction of mass
  //
  void AddElement (G4Element* element ,    //the element
                   G4double    fraction);   //fractionOfMass
```

$G4INSTALL/example/novice/N03 中二氧化硅就是采用的原子数份额定义的，而空气则是采用的质量份额定义的。

```
G4Material* SiO2 =
new G4Material("quartz",density= 2.200*g/cm3, ncomponents=2);
SiO2->AddElement(Si, natoms=1);
SiO2->AddElement(O , natoms=2);

G4Material* Air =
new G4Material("Air"   , density= 1.290*mg/cm3, ncomponents=2);
Air->AddElement(N, fractionmass=0.7);
Air->AddElement(O, fractionmass=0.3);
```

值得一提的是，在 Geant4 中为了定义复杂的混合物，还提供了

```
AddMaterial(G4Material* material,   //the material
            G4double    fraction);  //fractionOfMass
```

函数，通过这个函数可以将已经定义好的材料作为新材料的一种组成直接添加。

如$G4INSTALL/example/novice/N03 中气凝胶就是由 62.5%的二氧化硅、37.4%的水和 0.1%的碳组成的，由于二氧化硅和水在之前都已经定义了，因此直接使用 AddMaterial(G4Material* material, G4double fraction) 添加即可，而碳则是一种元素需要用 AddElement(G4Element* element , G4double fraction)添加，如下：

```
G4Material* Aerog =
new G4Material("Aerogel", density= 0.200*g/cm3, ncomponents=3);
Aerog->AddMaterial(SiO2, fractionmass=62.5*perCent);
Aerog->AddMaterial(H2O , fractionmass=37.4*perCent);
Aerog->AddElement (C    , fractionmass= 0.1*perCent);
```

而对于单质，由于是由一种元素组成的，因此可以将元素定义和材料定义合并，

参考$G4INSTLL/source/materials/include/G4Material.hh 如下

```
  // Constructor to create a material from scratch.
  //
  G4Material(const G4String& name,    //its name
             G4double   z,     //atomic number
             G4double   a,     //mass of mole
             G4double   density,     //density
             G4State    state     = kStateUndefined, //solid,gas
             G4double   temp      = STP_Temperature, //temperature
             G4double   pressure = STP_Pressure);  //pressure
```

如$G4INSTALL/example/novice/N03 中铝材料就是直接定义的。

```
new G4Material("Aluminium", z=13., a=26.98*g/mole, density=2.700*g/cm3);
```

需要注意的是在采用直接定义法定义元素或者直接定义单质的时候，其中的原子序数的类型是 G4double，而不是 G4int，但是从$G4INSTLL/source/materials/include/G4Element.cc 中看，这个变量最终似乎是被转换为 G4int 型处理的。由于时间关系，本人没有进行深入地研究，有兴趣的同仁可以研究一下这里为什么不用 G4int 而用 G4double，是否还有别的含义。

猪头 Geant4 讲座第六讲——几何模型

上一讲我们讲过了，几何结构类（DetectorConstruction）属于强制初始化类，其主要功能是构建模拟问题的几何结构，包括各部分的材料、形状、尺寸、位置等信息。

前面我讲了材料如何定义，今天我主要讲几何模型的建立以及各部分材料的设置。

在讲如何建立几何模型前我想首先讲一下 Geant4 中所采用的几何建模思想。

学过 MCNP 或 Fluka 的都知道 MCNP 和 Fluka 采用的是 CG 模型，所谓 CG 模型就是 Combination

Geometry，我习惯翻译为组合几何模型，CG 模型顾名思义就是整个模型由一块块小模块组合而成，打个不恰当的比方就是搭积木。这种模型的要求就是“不交不空”，既不能有相交的部分，也不能有空白的地方。CG 模型是粒子输运蒙卡模拟中非常常用的一种。

Geant4 采用的模型则不同，目前我没看到有关此模型的确切名称，但我习惯将之称为嵌套模型或盒子模型，因为其建模的方式就如往大盒子里放小盒子。在 Geant4 中首先我们要建立一个最大的盒子，称为 World Volume，然后往这个大盒子里面放各种各样的小盒子（部件），然后每个小盒子（部件）里面还可以放更小的盒子（零件），放入的小盒子将自动代替大盒子原有部分。在 Geant4 中，将大盒子称为 Mother Volume（母体），小盒子称为 Daughter Volume（子体）。

除了 World Volume 之外，每个 Volume（体）都必须且只能有一个母体，但可以没有子体，也可以有多个子体。

Geant4 的这种盒子模型的要求是：“不交不超”。“不交”就是要求同一个大盒子里面的小盒子，即同一等级的子体不能有相交的部分；“不超”就是要求小盒子不能超过大盒子的范围，即子体不能超出母体。用集合的语言描述就是

Volume i(level n)∩Volume j(level n)=Φ; Daughter Volume∈Mother Volume.

下面就可以讲如何设置这些盒子（Volume）。

Geant4 中每个这样的盒子的安放又分为两步。第一步是构建盒子，称为 Logical Volume（逻辑体）；第二步是将盒子摆放到正确位置，变为 Physical Volume（物理体）。

构建盒子又分为两步，第一步是确定盒子形状，第二步是确定盒子的材料等属性。

形状在 Geant4 中被称为 Solid。在 Geant4 中提供了多种固有的形状，如球形、长方体、锥体等，可以在 $G4INSTALL/source/geometry/solids 里面查找。

用户也可以通过 G4VSolid 类构建自己的形状，请参见 $G4INSTALL/source/geometry/management/include/G4VSolid.hh。

此外，对于一些复杂的形状，用户也可以利用基本形状通过交并补等布尔运算方式完成，布尔运算的方式请参考$G4INSTALL/source/geometry/solids/Boolean。

确定了盒子形状后，就是设置盒子的材料、磁场等属性。

这些属性的设置通过 G4LogicalVolume 类来完成，设置方法如下：

```
G4LogicalVolume(G4VSolid* pSolid, //形状
G4Material* pMaterial, //材料
const G4String& name, //逻辑体名字
```

G4FieldManager* pFieldMgr=0, //场管理
G4VSensitiveDetector* pSDetector=0, //是否 SD 探测器

G4UserLimits* pULimits=0, //用户限制
G4bool optimise=true); //是否优化

盒子造完了就该摆放盒子了。

摆放盒子也有两种方法，一种是直接构建物理体，另一种是指定摆放方法。

直接构建物理体是通过 G4VPhysicalVolume 类，其定义方法如下：

G4VPhysicalVolume(G4RotationMatrix *pRot, //旋转方式

const G4ThreeVector &tlate, //摆放坐标
const G4String &pName, //物理体名字
G4LogicalVolume *pLogical, //对应的逻辑体
G4VPhysicalVolume *pMother); //母体

如果 pMother=0 就表明这个体是一个 World Volume，World Volume 必须且只能有一个。

在实际应用中，我们通常采用指定摆放方法的方式来完成物理体的构建。

指定摆放方法是通过 G4PVPlacement 类完成。G4PVPlacement 类是 G4VPhysicalVolume 的派生类，该类提供了多种方法描述 Logical Volume 的摆放方法。具体可以参考 $G4INSTALL/source/geometry/volumes/include/G4PVPlacement.hh。

用这种方法可以建立具有相同 Logical Volume 的物理体，同时给可以给每个物理体分配一个编号，以便区分具有相同 Logical Volume 的物理体。这些编号在 UserSteppingAction 等类中处理数据时有时会非常有用处。

需要注意的是，在 Geant4 中摆放坐标都是指的相对坐标，是子体中心相对母体中心的坐标。而 World Volume 建立后就等于建立了几何模型的绝对坐标系。

下面简单地讲一下第一个 novice 例子中几何模型的建立。

G4double expHall_x = 3.0*m;

```
G4double expHall_y = 1.0*m;
G4double expHall_z = 1.0*m;
G4Box* experimentalHall_box
= new G4Box("expHall_box",expHall_x,expHall_y,expHall_z);
```

这是建立了一个长方体，需要注意的是 Geant4 中 Solid 的原点通常是设置在这个形状的中心的，而 MCNP 和 Fluka 中则大多是设置在某个顶点或某个底面的。

此外，定义长方体等形状时大多是用半长度/半宽度作为参数，而不是整个长度和宽度。

```
experimentalHall_log = new G4LogicalVolume(experimentalHall_box, //对应的 Solid
Ar, //材料

"expHall_log", //名字

0, //无场管理

0, //不是 SD

0); //无用户限制
```

这里需要注意的是最后省略了 optimise，而是采用了默认值。

```
experimentalHall_phys = new G4PVPlacement(0, //无旋转

G4ThreeVector(), //放置在(0,0,0)
experimentalHall_log, //对应逻辑体

"expHall", //名字

0, //母体

false, //pMany

0); //Copy No.
```

1、这里母体为 0 表明这是个 World Volume；

2、pMany 目前没有用处，根据 Geant4 的描述，将来也许会用于重复结构；

3、这里同样省略了最后一个参数 pSurfChk。