粒子物理与核物理实验中的数据分析

杨振伟

清华大学

第八讲:Geant4 的探测器模拟介绍(3)

上讲回顾

- **粒子定义**

  G4ParticleDefinition

  6大类粒子：G4LeptonConstructor
  G4BosonConstructor
  G4MesonConstructor
  G4BaryonConstructor
  G4IonConstructor
  G4ShortlivedConstructor

- **产生事例：** G4ParticleGun

- **物理过程：** **电磁、强作用、衰变、光轻子-强子作用、光学、参数化、输运(必要过程)**

本讲要点

- 灵敏探测器(Sensitive Detector)
- 读取灵敏探测器数据
  存入ROOT文件
- cmake方式编译Geant4应用程序
  （适用于Geant4 9.5之后的版本）

- 产生主事例（自学）

  G4HEPEvtInterface

灵敏探测器(Sensitive Detector)

灵敏探测器

- 灵敏探测器(SD)的首要任务是通过粒子“迹”(track)上的“步”(step)的信息，构造“击中”(hit)。

  这些击中经过数字化，被读出模块读出的信息是真正的模拟结果。(当然在模拟中我们也可以忽略数字化而直接读出hit的信息或者其它信息，这些信息实际上是所谓的"Monte Carlo Truth")

用户灵敏探测器继承自抽象基类G4VSensitiveDetector，用户需要完成3个主要函数：

ProcessHits(G4Step* aStep, G4TouchableHistory*)

构造“击中”，被G4SteppingManager调用

Initialize(G4HCofThisEvent* HCE)

初始化，事例开始时调用，指定构造的“击中”与当前事例关联起来

EndOfEvent(G4HCofThisEvent*)

事例结束时调用

参见例子N02/src/ExN02TrackerSD.cc

定义和添加灵敏探测器(1)

1.定义Hits，如ExN02TrakcerHit.cc
2.定义SD，如ExN02TrackerSD.cc
3.在DetectorConstruction()中添加SD

在探测器构造中添加敏感探测器，比如：

```
//SDManager
G4SDManager* SDman = G4SDManager::GetSDMpointer();
//创建敏感探测器
G4String trackerChamberSDname = "ExN02/TrackerChamberSD";
ExN02TrackerSD* aTrackerSD = new ExN02TrackerSD( trackerChamberSDname );
//添加到SDManager
SDman->AddNewDetector( aTrackerSD );
//为logical体积设定敏感探测器!!!
logicChamber->SetSensitiveDetector( aTrackerSD );
```

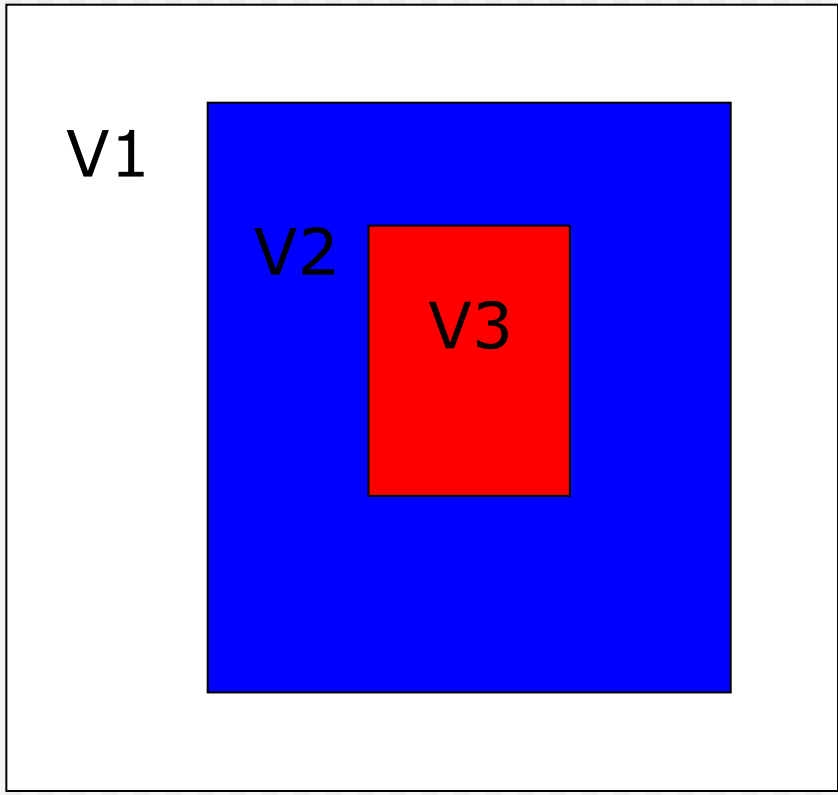
参见例子**N02/src/ExN02DetectorConstruction.cc**

定义和添加灵敏探测器(2)

- 将多个logical体积添加为灵敏探测器时：

  假设有3个体积V1，V2，V3

  如果定义这3个体积的时候，先定义V1，再定义V2，最后定义V3，则V1，V2，各自被覆盖掉一部分。



如果希望蓝色区域为SD，则需要
V2->SetSensitiveDetector(....)

读取灵敏探测器数据存入ROOT文件

读取敏感探测器的信息

在EventAction类的EndOfEventAction()函数中，可以读取该事例中存储的Hits。比如可以在ExN02EventAction.cc中加入下面代码，查看每个事例中的Hits数目：

```
//获得该事例的HitsCollection(可能不止一个)
G4HCofThisEvent* hc = evt->GetHCofThisEvent();
G4int NbOfColl = hc->GetNumberOfCollections();
//获得第0个HitsCollection，即ExN02TrackerHitsCollection
//也可以通过CollectionID获得
ExN02TrackerHitsCollection *hitsC = hc->GetHC(0);
//该Collection中Hits数目
G4int sizehits = hitsC->entries();
.....
```

当然，你也可以将hitsC中的Hits挨个读取出来，并获取这些Hits的详细信息。

将模拟结果写入root文件

1) GNUMakefile中添加调用root需要的头文件的目录和库，即在G4EXLIB := true一行后面加入：

```
ROOTCFLAGS    = $(shell root-config --cflags)
ROOTLIBS      = $(shell root-config --libs)
ROOTGLIBS     = $(shell root-config --glibs)
CPPFLAGS  += $(ROOTCFLAGS)
EXTRALIBS += $(ROOTLIBS) $(ROOTGLIBS)
```

2) 在main函数新建TFile，定义TTree (全局变量)
3) 在EventAction的EndOfEventAction()函数中收集需要的数据，填充到TTree。(也可以直接在SD中收集)
4) 在RunAction中将TFile写入硬盘。(也可以在主函数main()中写入)

参见
hep.tsinghua.edu.cn/~yangzw/CourseDataAna/examples/Lec8.tgz

提示：模拟过程信息的获取

- 模拟过程的各种信息几乎都可以从G4Step获得，参见 http://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/FAQ/html/ch01s04.html

```
G4StepPoint* point1 = step->GetPreStepPoint();
G4StepPoint* point2 = step->GetPostStepPoint();

G4ThreeVector pos1 = point1->GetPosition();
G4ThreeVector pos2 = point2->GetPosition();

G4TouchableHandle touch1 = point1->GetTouchableHandle();

G4VPhysicalVolume* volume = touch1->GetVolume();

G4double eDeposit      = step->GetTotalEnergyDeposit();
G4double sLength       = step->GetStepLength();
G4ThreeVector displace = step->GetDeltaPosition();
G4double tof           = step->GetDeltaTime();

G4Track* track          = step->GetTrack();
G4ThreeVector momentum = track->GetMomentum();
G4double kinEnergy     = track->GetKineticEnergy();
G4double globalTime    = track->GetGlobalTime();
...etc...
```

cmake方式
编译Geant4应用程序

什么是cmake

- cmake(www.cmake.org)是一个跨平台的开源编译系统，它通过与平台和编译器无关的纯文本配置文件(configuration file)控制软件的编译过程。它能根据配置文件自动生成Makefile和工作区(workspace)以进行编译和运行。
- 个人评价
  - ✓ 相当赞！

cmake编译c++程序(1)

- 假设要编译的HelloWorld.cxx放于HelloWorld目录，程序代码如下：

```
// Hello World
#include <iostream>
using namespace std;

int main (int argc, char *argv[])
{
  cout << "Hello World ! " << endl;
  return 0;
}
```

- 该目录下还需要存在cmake配置文件CMakeLists.txt，最基本的配置文件如下：

```
cmake_minimum_required (VERSION 2.6)
project (HelloWorld)
# add the executable
add_executable(HelloWorld HelloWorld.cxx)
```

cmake_minimum_required, project, add_executable等都是cmake的内置命令，用于指定版本要求、项目名称或利用指定源文件(HelloWorld.cxx)为项目添加一个可执行文件(HelloWorld)。参见手册(http://www.cmake.org/cmake/help/cmake-2-8-docs.html)

cmake编译c++程序(2)

- 编译HelloWorld.cxx (特点：源文件与编译文件分离)

```
YANG:HelloWorld-build yangzw$ cmake ../HelloWorld
-- The C compiler identification is GNU
-- The CXX compiler identification is GNU
-- Checking whether C compiler has -isysroot
-- Checking whether C compiler has -isysroot - yes
-- Checking whether C compiler supports OSX deployment target flag
-- Checking whether C compiler supports OSX deployment target flag - yes
-- Check for working C compiler: /usr/bin/gcc
-- Check for working C compiler: /usr/bin/gcc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Checking whether CXX compiler has -isysroot
-- Checking whether CXX compiler has -isysroot - yes
-- Checking whether CXX compiler supports OSX deployment target flag
-- Checking whether CXX compiler supports OSX deployment target flag - yes
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Configuring done
-- Generating done
-- Build files have been written to: /Users/yangzw/workdir/trycmake/Tutorial/HelloWorld-build
YANG:HelloWorld-build yangzw$ ls
CMakeCache.txt       CMakeFiles/          Makefile             cmake_install.cmake
YANG:HelloWorld-build yangzw$ make
Scanning dependencies of target HelloWorld
[100%] Building CXX object CMakeFiles/HelloWorld.dir/HelloWorld.cxx.o
Linking CXX executable HelloWorld
[100%] Built target HelloWorld
YANG:HelloWorld-build yangzw$ ./HelloWorld
Hello World !
```

假设当前目录HelloWorld中已经存放源文件HelloWorld.cxx和配置文件CMakeLists.txt和配置文件。

```
cd ../
mkdir HelloWorld-build
cd HelloWorld-build
cmake ../HelloWorld
ls
make
./HelloWorld
```

问题：如果需要include头文件，或者链接其他库文件怎么办？

cmake编译Geant4应用程序(1)

假设要编译Geant4.9.5例子(源文件所在)
$HOME/geant4/examples/basic/B1

假设Geant4.9.5安装于
/projects/soft/ext/geant4/geant4.9.5-install

则编译步骤如下：

```
cd $HOME/geant4
mkdir B1-build
cd B1-build/
cmake -DGeant4_DIR=/projects/soft/ext/geant4/geant4.9.5-install/lib/Geant4-9.5.0 $HOME/geant4/examples/basic/B1
make
source /projects/soft/ext/geant4/geant4.9.5-install/share/Geant4-9.5.0/geant4make/geant4make.sh
./exampleB1
```

cmake编译Geant4应用程序(2)

如果要在B1例子中使用ROOT，则需要在B1所在目录的CMakeLists.txt中加入几行配置命令，以便让编译器知道如何找ROOT的头文件，如何链接ROOT的库文件。

```
#added by yangzw, 2012.04.19, to find cmake config file of ROOT package
SET(CMAKE_MODULE_PATH /projects/soft/ext/geant4/geant4.9.5/cmake/Modules/)
#added by yangzw, 2012.04.19, to find ROOT package and give the include dir.
find_package(ROOT)
include_directories(${ROOT_INCLUDE_DIR})

#----------------------------------------------------------------------------
# Setup Geant4 include directories and compile definitions
```

```
#----------------------------------------------------------------------------
# Add the executable, and link it to the Geant4 libraries
#
add_executable(exampleB1 exampleB1.cc ${sources} ${headers})
target_link_libraries(exampleB1 ${Geant4_LIBRARIES})

#added by yangzw, 2012.04.19, to link ROOT libraries
target_link_libraries(exampleB1 ${ROOT_LIBRARIES})
```

用事例产生子接口

G4HEPEvtInterface产生主事例

事例产生子接口

- G4HEPEvtInterface

  很多时候，事例产生子已经存在，而且是Fortran语言。Geant4并不直接链接这些Fortran程序，而是提供了一个接口：

  G4HEPEvtInterface读取事例产生子生成的ASCII文件中的信息，重新生成G4PrimaryParticle对象，并关联到对应的G4PrimaryVertex

也就是说，G4HEPEvtInterface将/HEPEVT/公共块的信息转换为一个O-O数据结构。这个公共块在高能物理中被广泛使用。

用/HEPEVT/公共块生成ASCII文件

```
************************************************************
      SUBROUTINE HEP2G4
*
* Convert /HEPEVT/ event structure to an ASCII file
* to be fed by G4HEPEvtInterface
*
************************************************************
      PARAMETER (NMXHEP=2000)
      COMMON/HEPEVT/NEVHEP,NHEP,ISTHEP(NMXHEP),IDHEP(NMXHEP),
     >JMOHEP(2,NMXHEP),JDAHEP(2,NMXHEP),PHEP(5,NMXHEP),VHEP(4,NMXHEP)
      DOUBLE PRECISION PHEP,VHEP
*
      WRITE(6,*) NHEP
      DO IHEP=1,NHEP
       WRITE(6,10)
     >   ISTHEP(IHEP),IDHEP(IHEP),JDAHEP(1,IHEP),JDAHEP(2,IHEP),
     >   PHEP(1,IHEP),PHEP(2,IHEP),PHEP(3,IHEP),PHEP(5,IHEP)
10     FORMAT(4I10,4(1X,D15.8))
      ENDDO
*
      RETURN
      END
```

common block

将以下量写入文件中

第一行：NHEP，当前事例粒子数(包括中间态)
随后的NHEP行：每个粒子的ISTHEP,IDHEP,JDAHEP,PHEP信息
ISTHEP：粒子状态；IDHEP：粒子PDG号；JDAHEP：粒子衰变产物位置的指针；PHEP(1-3,5)：粒子x,y,z动量，能量，质量

以HEPEVT格式输出的ASCII文件

比如：下面这个事例表示该事例共102个粒子(包括中间态)，随后的102行分别为这102个粒子的具体信息：
第一列为粒子状态(3：对撞入射粒子或其它；2：衰变了；1：存在的粒子；0：空)，
第2列为粒子PDG号，
最后4列分别为粒子的x，y，z方向动量和质量。

```
102
     3      11       0       0  0.00000000E+00  0.00000000E+00  0.25000000E+03  0.51000000E-03
     3     -11       0       0  0.00000000E+00  0.00000000E+00 -0.25000000E+03  0.51000000E-03
     3      11       0       0  0.00000000E+00  0.00000000E+00  0.24999999E+03  0.00000000E+00
     3     -11       0       0  0.00000000E+00  0.00000000E+00 -0.25000000E+03  0.00000000E+00
     3      11       0       0  0.37396914E-02  0.15234913E-02  0.24138585E+03  0.00000000E+00
     3     -11       0       0 -0.93164320E-02  0.27396574E-01 -0.24687934E+03  0.00000000E+00
     3      23       0       0 -0.55767406E-02  0.28920065E-01 -0.54934906E+01  0.48823428E+03
     3       2       0       0  0.19070032E+02  0.24337596E+03 -0.48627266E+01  0.33000000E+00
     3      -2       0       0 -0.19075609E+02 -0.24334704E+03 -0.63076405E+00  0.33000000E+00
     2      23      16      26 -0.55767406E-02  0.28920065E-01 -0.54934906E+01  0.48823428E+03
     1      22       0       0  0.93164331E-02 -0.27396573E-01 -0.31205891E+01  0.00000000E+00
     1      22       0       0 -0.81046576E-03 -0.82301151E-04  0.14162632E+00  0.00000000E+00
     ......
175
     ......
```

使用HEPEvtInterface的例子

参见例子N04，在ExN04PrimaryGeneratorAction.cc中：

```
ExN04PrimaryGeneratorAction::ExN04PrimaryGeneratorAction()
{
  const char* filename = "pythia_event.data";
  //读取pythia_event.data
  HEPEvt = new G4HEPEvtInterface(filename);
}
void ExN04PrimaryGeneratorAction::GeneratePrimaries(G4Event*
anEvent)
{
   //设定主顶点位置，产生主顶点
   HEPEvt->SetParticlePosition(G4ThreeVector(0.*cm,0.*cm,0.*cm);
   HEPEvt->GeneratePrimaryVertex(anEvent);
}
```

其中HEPEvt在头文件中定义：

```
  G4VPrimaryGenerator* HEPEvt;
```

注：main函数或者mac文件中设定beamOn事例数不能超过ASCII中事例数。

小结

- 敏感探测器的添加和定义
  在DetectorConstruction中，不但要将SD添加给SDManager，还要指定相应的logical体积。
- 将结果存储到root文件中
  在EventAction中收集数据，或者在SD中直接收集。
- cmake方式编译Geant4应用程序
- G4HEPEvtInterface
  主产生子(PrimaryGenerator)的一种，直接读取ASCII文件中以HEPEVT格式存储的事例。

练习

- 在例子N02的基础上，将模拟的信息存储到root文件中。这些信息包括：粒子的PDG号、质量、能量沉积、径迹长度。生成root文件后画出这些信息的直方图，并进行分析
- 修改探测器物质和入射粒子，重新运行，得到新的root文件，并画出储存信息的直方图。
- 在N03的基础上，加入敏感探测器。

参考资料

1. Geant4应用开发手册3.6节
2. Geant4应用开发手册4.4节
3. Geant4例子novice/N02,N04