Моделирование физических процессов в пакете Geant4

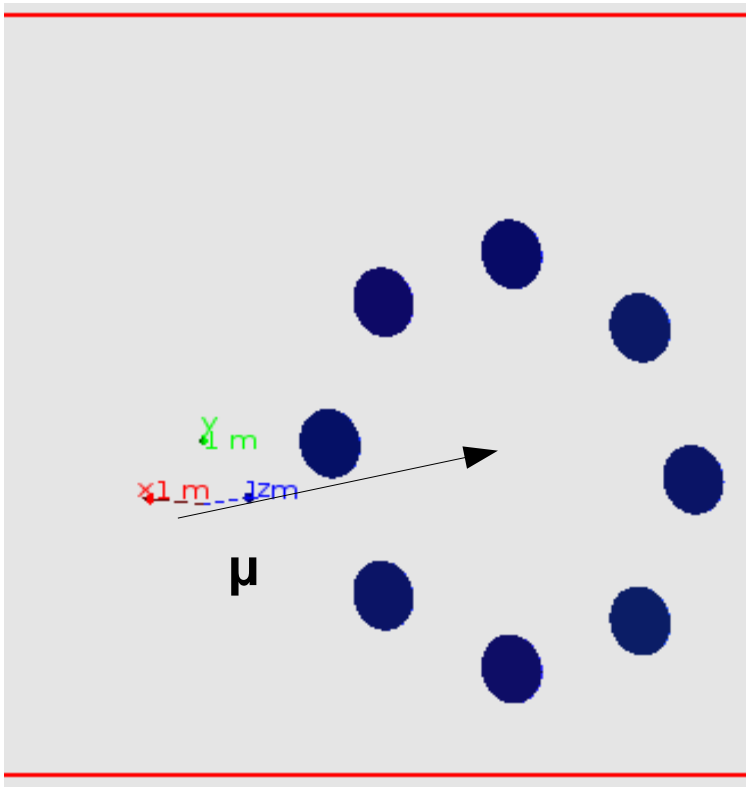
Алексей Жемчугов
ОИЯИ
E-mail: zhemchugov@jinr.ru

Иркутский государственный университет

28-29 апреля 2015

Более сложный пример

- SimpleHE.zip на http://geant4.jinr.ru
- Моделируются оптические фотоны (черенковское излучение, сцинтилляция, поглощение, рассеяние)
- Объем заполнен водой
- Создано 8 дисков - «чувствительных детекторов»
- Генератор мюонов (100 МэВ, направление перп. плоскости детекторов)
- При попадании оптического фотона в «детектор» в гистограммы ROOT заносится номер детектора, энергия, время.
- На каждом шаге мюона печатается полное энерговыделение.
- Графический и пакетный интерфейс, визуализация



Моделирование отклика детектора

- Любой логический объем в модели можно объявить детектирующим, или «чувствительным». При этом при прохождении частицы через данный объем моделируется срабатывание детектора.

- Детектирующих объемов одновременно может несколько. Обработка срабатываний при этом может происходить по разному.

- Дополнительно можно смоделировать оцифровку сигнала и электронный отклик детектора

Описание детектирующего объема (I)

- Создается класс-наследник класса G4VSensitiveDetector
- Описываются обязательные методы
  - **Initialize**() вызывается в начале каждого события
  - **ProcessHits**() вызывается на каждом шаге в детектирующем объеме. Позволяет получить информацию о характеристиках частицы в данной точке, о взаимодействии с веществом, и смоделировать срабатывание детектора
  - **EndOfEvent**() вызывается в конце события. Позволяет провести отбор срабатываний, и сохранить результаты

Описание детектирующего объема (II)

• Инициализируется объект класса G4SDManager

G4SDManager fSDM = G4SDManager::GetSDMpointer();*

• Объект, описывающий детектирующий объем, регистрируется в менеджере

fSDM->AddNewDetector(G4VSensitiveDetector)*

• Детектирующий объем ассоциируется с логическим объемом

logVolume->SetSensitiveDetector(G4VSensitiveDetector)*

Пример получения информации о частице

```
G4int pdgcode =  step->GetTrack()->GetDefinition()->GetPDGEncoding();

G4double energy =  step->GetTrack()->GetKineticEnergy();

G4double time =  step->GetPostStepPoint()->GetGlobalTime();

G4int id = step->GetPreStepPoint()->GetTouchableHandle()-
>GetCopyNumber();

G4String vol_name = step->GetPreStepPoint()->GetTouchableHandle()-
>GetVolume()->GetName();

G4double edep = step->GetTotalEnergyDeposit();

G4double trackid = step->GetTrack()->GetTrackID();
```

Действия, определяемые пользователем

- Классы-наследники G4RunAction, G4EventAction, G4UserTrackingAction, G4UserSteppingAction
- G4RunAction: для каждого сеанса (**BeginOfRun, EndOfRun**)
- G4EventAction: вызывются на каждом событии (**BeginOfEvent, EndOfEvent**)
- **G4UserTrackingAction**: для каждого трека (до и после трекинга частицы) вызываются методы соответственно:
    - ***virtual void PreUserTrackingAction(const G4Track*)***
    - ***virtual void PostUserTrackingAction(const G4Track*)***
- G4UserSteppingAction: на каждом шаге вызывается метод
    - ***virtual void UserSteppingAction(const G4Step*)***

Взаимодействие пользователя с программой моделирования

- Пакетный режим
- Пакетный режим, управляемый сценарием
- Интерактивный режим с командной строкой
- Интерактивный режим с графическим интерфейсом

Управление программой моделирования

- Geant4 имеет встроенный набор команд, управляющих моделированием
  - уровень диагностики
  - изменение геометрических параметров модели
  - изменение списка учитываемых процессов
  - ...
- Этот набор может быть расширен пользователем
- Команды могут использоваться в интерактивном режиме, считываться из файла сценария или быть запрограммированы в коде

Категории команд

/control	управление интерфейсом
/units	система единиц
/geometry	навигация и проверка перекрытий объемов
/tracking	управление объектами TrackingManager и SteppingManager
/event	управление объектом EventManager
/cuts	управление порогами рождения частиц
/run	управление сеансом
/random	управление генератором случайных чисел
/material	просмотр списка и свойств материалов
/particle	изменение списков и свойств частиц
/process	изменение списка активных процессов
/vis	визуализация
/gun	управление генератором первичной вершины
/hits	управление детектирующими объемами
/score	работа со счетчиками

/control/execute

Выполнение сценария:

/control/execute имя_файла_сценария

Например

/control/execute ~/run.mac

/tracking/verbose

/tracking/verbose уровень_детализации

Уровни детализации диагностики

0 : Сообщения отсутствуют

1 : Минимальная информация о каждом шаге

2 : Уровень=1 + информация о вторичных частицах

3 : Дополнительно информация о состоянии в начале и конце каждого шага

4 : Дополнительно информация о состоянии в начале и конце каждого шага с детализацией по отдельным процессам

5 : Дополнительно информация об оценках длины шага согласно каждому отдельному процессу

/run/beamOn

Основная команда при моделировании

Примеры

/run/beamOn 1000 - Начать моделирование 1000 событий

/run/beamOn 1000 event.mac 100

Начать моделирование 1000 событий, и после каждого из первых 100 событий выполнить сценарий event.mac

/random

Действия с начальным значением генератора случайных чисел

/random/setDirectoryName [fileName]

/random/setSavingFlag [flag]

/random/saveThisRun

/random/saveThisEvent

/random/resetEngineFrom [fileName]

Самые главные команды

```
Idle> help
Command directory path : /
 Sub-directories :
 1) /control/   UI control commands.
 2) /units/   Available units.
.....
Type the number ( 0:end, -n:n level back ) :
0
Exit from HELP.


Idle>exit
```

Сохранение результатов. Geant4 и ROOT.

•**Geant4 не имеет встроенных средств сохранения результатов на диск.**

•Пользователь может использовать любые способы сохранения результатов моделирования, используя методы Geant4 для доступа к этим результатам

- ASCII файл
- ROOT
- JAS
- HBOOK
- g4tools (AIDA)
- .....

Использование ROOT
Простой случай

В простых случаях – использование методов ROOT, например, при обработке срабатываний в детектирующем объеме

- Инициализация дерева и/или гистограмм в конструкторе
- Заполнение дерева и/или гистограмм в методах ProcessHits() и/или EndOfEvent()
- Сохранение в ROOT файл в деструкторе

Использование ROOT
Более сложный случай (I)

В сложных случаях – создание синглетного (объект которого существует в программе в единственном экземпляре) класса MyROOTManager:

```
class MyROOTManager
{
public:
static MyROOTManager* GetPointer()
{
    if (theInstance == 0) theInstance = new  MyROOTManager();
    return theInstance;
}
private:
MyROOTManager(){}; // создание объекта только методом GetPointer()
static MyROOTManager* theInstance;
};

MyROOTManager::theInstance = 0;
```

Использование ROOT
Более сложный случай (II)

- Класс MyROOTManager содержит все деревья и гистограммы, обеспечивает заполнение их и запись в файл

- Указатель на объект MyROOTManager в любом месте программы получется вызовом

```
MyROOTManager* fRM = MyROOTManager::GetPointer();
```

- Дальнейшая работа (заполнение гистограмм, запись на диск и т.д.) осуществляется по указателю fRM

Сборка Geant4 и ROOT: что исправить в CMakeLists.txt?

```
....
project(SimpleHE)
set(CMAKE_MODULE_PATH
      ${Geant4_DIR}/Modules/
      ${CMAKE_MODULE_PATH})
option(WITH_GEANT4_UIVIS "Build example with Geant4 UI and Vis drivers" ON)
if(WITH_GEANT4_UIVIS)
  find_package(Geant4 REQUIRED ui_all vis_all)
else()
  find_package(Geant4 REQUIRED)
endif()
#include (FindROOT.cmake)
find_package(ROOT REQUIRED)
include(${Geant4_USE_FILE})
include_directories(${PROJECT_SOURCE_DIR}/include)
#include(FindROOT.cmake)
set(INCLUDE_DIRECTORIES   ${ROOT_INCLUDE_DIR} )
include_directories( ${INCLUDE_DIRECTORIES})
set(LINK_DIRECTORIES      ${ROOT_LIBRARY_DIR}  )
file(GLOB sources ${PROJECT_SOURCE_DIR}/src/*.cc)
file(GLOB headers ${PROJECT_SOURCE_DIR}/include/*.hh)
add_executable(simpleHE.exe simpleHE.cc ${sources} ${headers})
target_link_libraries(simpleHE.exe ${Geant4_LIBRARIES} ${ROOT_LIBRARIES})
....
```

Оптические фотоны

Регистрация конструктора оптических процессов в дополнение к стандартному набору физ. процессов:

```
// Physics list
G4VModularPhysicsList* physicsList = new QGSP_BERT;
physicsList->SetVerboseLevel(1);
physicsList->RegisterPhysics(new G4OpticalPhysics);
runManager->SetUserInitialization(physicsList);
```

Оптические свойства материала

- В материалах, оптические свойства которых не заданы, оптические фотоны не моделируются
- Необходимо создать объект G4MaterialPropertiesTable, заполнить в нем таблицы оптических свойств и передать его в соотв. объект класса G4Material

```
G4MaterialPropertiesTable* MPT1 = new G4MaterialPropertiesTable();

Water->SetMaterialPropertiesTable(MPT1);
```

- Основные таблицы оптических свойств:
  - RINDEX - показатель преломления (в зависимости от энергии фотона)
  - ABSLENGTH - Длина поглощения (в зависимости от энергии фотона)
- Дополнительные свойства для сцинтилляции, Ми-рассеяния.