Введение в GEANT4

Иванов Артем Викторович
E-mail: arivanov@jinr.ru

Как создать свой Sensitive Detector?

Шаг 2 В файле include/SensitiveDetector.hh

```
#ifndef SensitiveDetector_h
#define SensitiveDetector_h 1

#include "G4VSensitiveDetector.hh"

class G4Step;

class SensitiveDetector : public G4VSensitiveDetector
{
  public:
    SensitiveDetector(const G4String& name);
   ~SensitiveDetector();

    G4bool ProcessHits(G4Step* step, G4TouchableHistory* history);

};

#endif
```

Использование Geant4 Analysis

Основные шаги

- Создать G4AnalysisManager.
- Объявить (создать) свои гистограммы и ntuples.
- Открыть файл.
- Заполнять значениями гистограммы и ntuples.
- Записать данные в файл и закрыть его.

Создание Менеджера Анализа

RunAction.cc

Менеджер анализа создается при первом вызове функции G4AnalysisManager::Instance()

```
#include "G4AnalysisManager.hh"

RunAction::RunAction()
{
    // Создаем менеджер анализа
    auto analysisManager = G4AnalysisManager::Instance();

    analysisManager->SetVerboseLevel(1);
    analysisManager->SetDefaultFileType("root");
}
```

Если установлен тип файла по умолчанию, имена файлов можно указывать без расширения.

Создать гистограммы

RunAction.cc

Пример создания одномерных гистограмм

```
#include "G4AnalysisManager.hh"

RunAction::RunAction()
{
    // Создаем или получаем менеджер анализа
    // ...
    // Создаем гистограммы
    analysisManager->CreateH1("EDep", "Energy deposit", 100, 0., 800*MeV);
    analysisManager->CreateH1("TLen", "Track length"  , 100, 0., 100*mm);



}
```

Открыть Файл

RunAction.cc

Пример открытия файла

```
#include "G4AnalysisManager.hh"

void RunAction::BeginOfRunAction(const G4Run* run)
{

    auto analysisManager = G4AnalysisManager::Instance();

    // Открываем выходной файл
    analysisManager->OpenFile("MyFile");
}
```

Расширение имени файла можно опустить, так как тип файла по умолчанию был установлен ранее. Иначе следует указать полное имя "MyFile.root".

Заполнить Гистограммы

EventAction.c

Пример заполнения одномерных гистограмм

```
#include "G4AnalysisManager.hh"

void EventAction::EndOfEventAction(const G4Event* event)
{
    // Получаем менеджер анализа
    auto analysisManager = G4AnalysisManager::Instance();

    // Заполняем гистограммы
    analysisManager->FillH1(0, fEdep);
    analysisManager->FillH1(1, fTrackLength);
}
```

Записать и Закрыть Файл

RunAction.cc

```
#include "G4AnalysisManager.hh"

void RunAction::EndOfRunAction(const G4Run* run)
{
    // Получаем менеджер анализа
    auto analysisManager = G4AnalysisManager::Instance();

    // Записываем и закрываем выходной файл
    analysisManager->Write();
    analysisManager->CloseFile();
}
```

Использование Geant4 Analysis. Основные шаги

- **Создать G4AnalysisManager**

  в конструкторе RunAction

- **Объявить (создать) гистограммы, n-кортежи**

  в конструкторе RunAction

- **Открыть файл**

  в BeginOfRunAction()

- **Заполнять значениями гистограммы, n-кортежи**

  в любом месте (во время обработки события)

- **Записать в файл и закрыть файл**

  в EndOfRunAction()

Выполнение шагов в предложенных классах и методах не является обязательным, но гарантирует корректную работу в многопоточном режиме.

Гистограммы

- Доступны 1D, 2D, 3D гистограммы
- Идентификатор гистограммы (целочисленное значение) автоматически генерируется при создании гистограммы функцией CreateH1
- Начальное значение по умолчанию 0
- Идентификаторы для 1D, 2D и 3D гистограмм определяются независимо.
- Можно получить прямой доступ к гистограмме с помощью функции GetH1(G4int id)

Ntuple

```
RunAction::RunAction()
{
    // Создаем или получаем менеджер анализа
    // ...
    // Создаем ntuple
    analysisManager->CreateNtuple("MyNtuple", "Edep and X");
    analysisManager->CreateNtupleDColumn("Eabs"); // Столбец типа double
    analysisManager->CreateNtupleIColumn("X"); // Столбец типа int
    analysisManager->FinishNtuple();
}
```

Заполнить ntuple

EventAction.cc

```
void EventAction::EndOfEventAction(const G4Event* event)
{
   // Получаем менеджер анализа
   auto analysisManager = G4AnalysisManager::Instance();

   // Заполняем ntuple
   analysisManager->FillNtupleDColumn(0, fEnergyAbs);
   analysisManager->FillNtupleIColumn(1,  fX);
   analysisManager->AddNtupleRow(); // Добавляем строку
}
```

Доступные типы столбцов:

integer (I), float (F), double (D), std::string (S)

std::vector этих типов.

Выходные файлы

- В зависимости от выбранного формата файла может создаваться несколько выходных файлов:

  ROOT, HDF5, XML, CSV

Задание

SensitiveDetector::SensitiveDetector(const G4String& name): G4VSensitiveDetector(name)

{

Создать G4AnalysisManager

Объявить свои гистограммы

Открыть файл

}

SensitiveDetector::~SensitiveDetector()

{

Записать данные в файл и закрыть его.

}

G4bool SensitiveDetector::ProcessHits(G4Step* aStep, G4TouchableHistory* history)

{

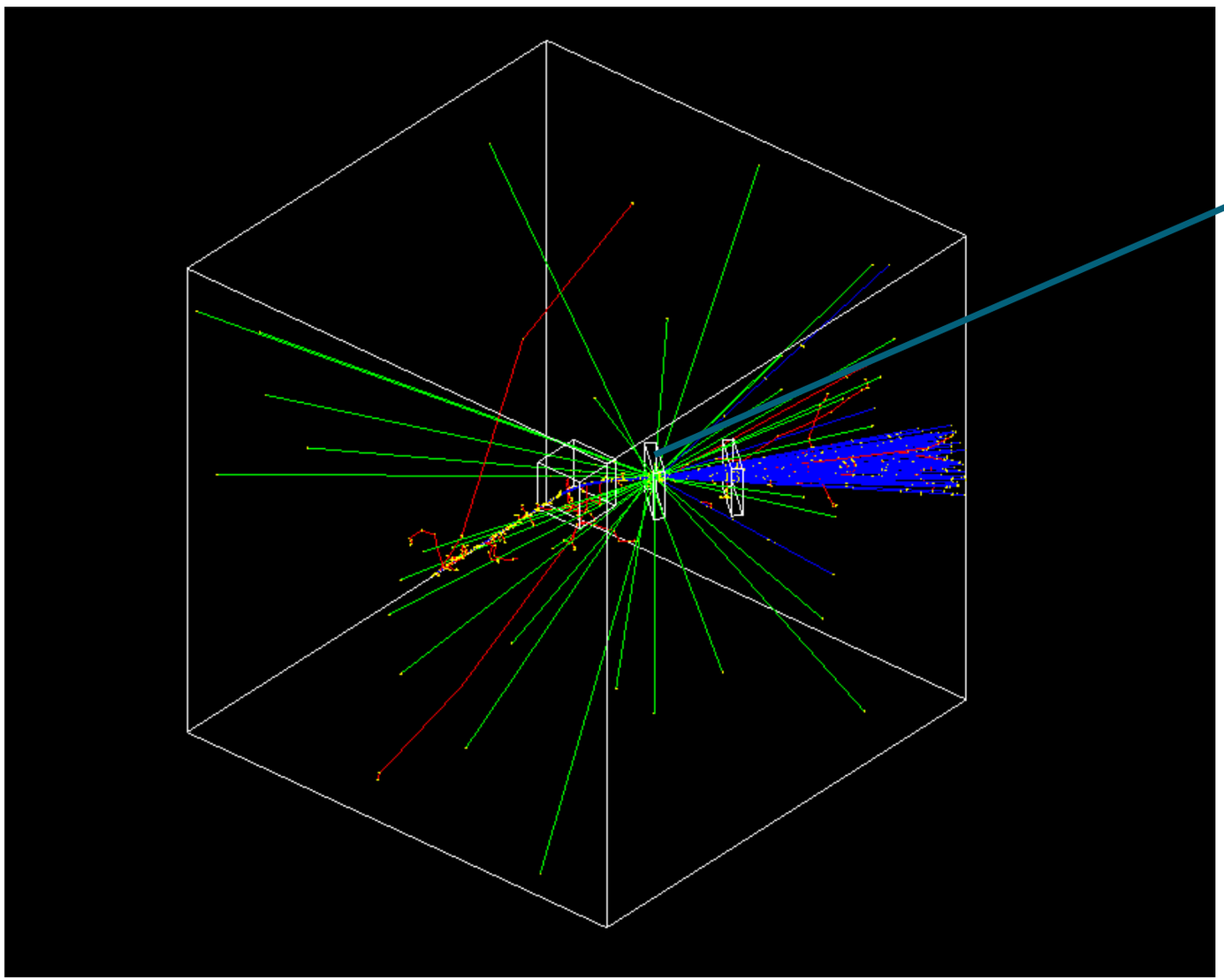
Заполнять значениями гистограммы

}

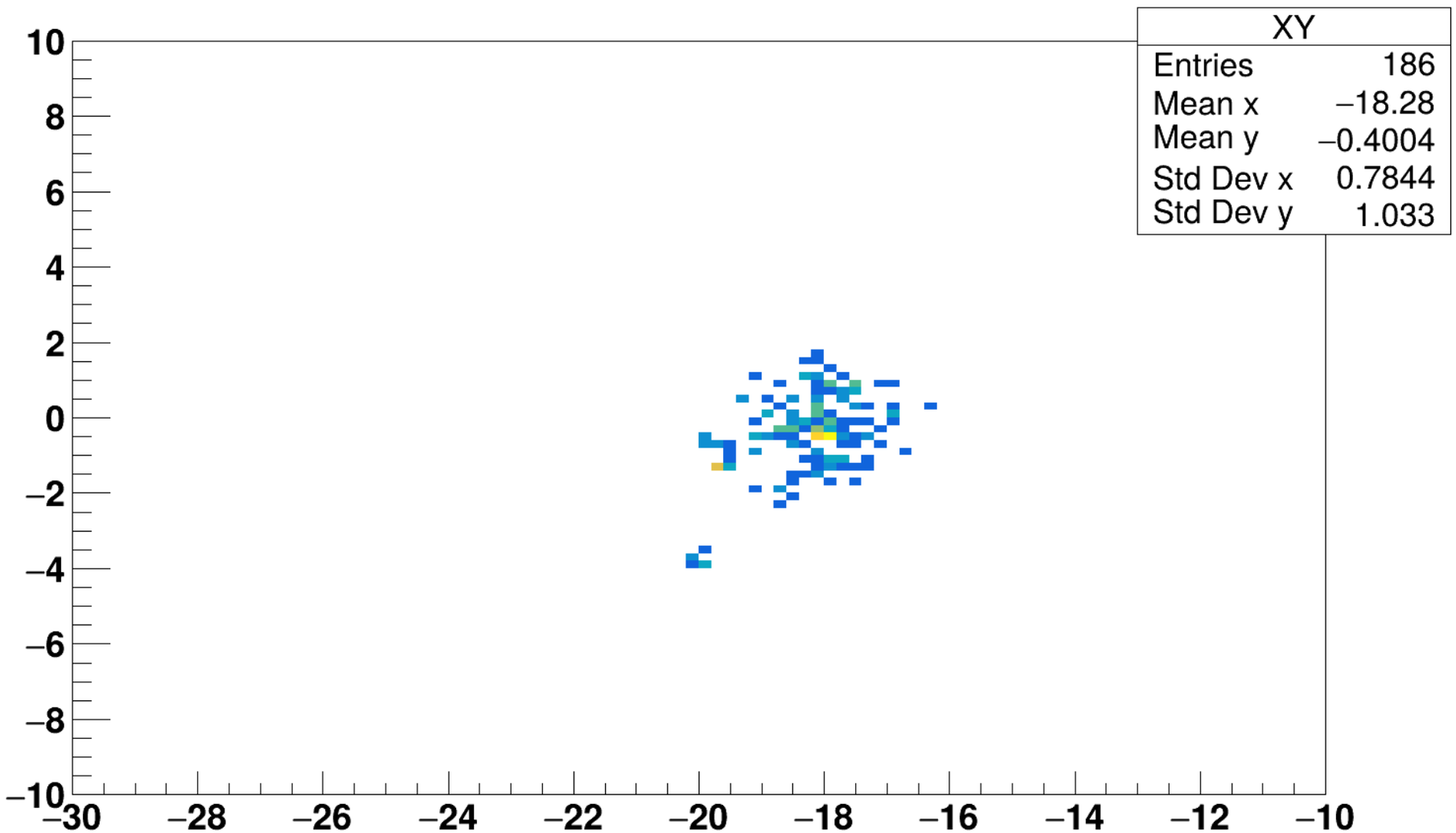
Задание

```
analysisManager->CreateH2("XY", "XY", binx, xmin, xmax, biny, ymin, ymax );

analysisManager->FillH2(0, x, y );

double x=pos_pre.x()
```

Задание



Задание


XY
Entries 186
Mean x −18.28
Mean y −0.4004
Std Dev x 0.7844
Std Dev y 1.033
10
8
6
4
2
0
−2
−4
−6
−8
−10
−30
−28
−26
−24
−22
−20
−18
−16
−14
−12
−10

Задание

source /opt/root/root_v6.36.04/install/bin/thisroot.sh
source /opt/geant4/geant4-v11.3.2/install/bin/geant4.sh

Первый способ через GUI

./SimpleExample

Проверяем что геометрия правильная

Второй способ через batch

./SimpleExample run.mac

Запускаем на 1000 событиий
/run/beamOn 1000

Задание

- Получаем файл с имяафайла.root

  Чтобы открыть его
- root имя.root

  дальше пишем
- TBrowser a