

Введение в GEANT4

Иванов Артем Викторович

E-mail: arivanov@jinr.ru

Базовые класс

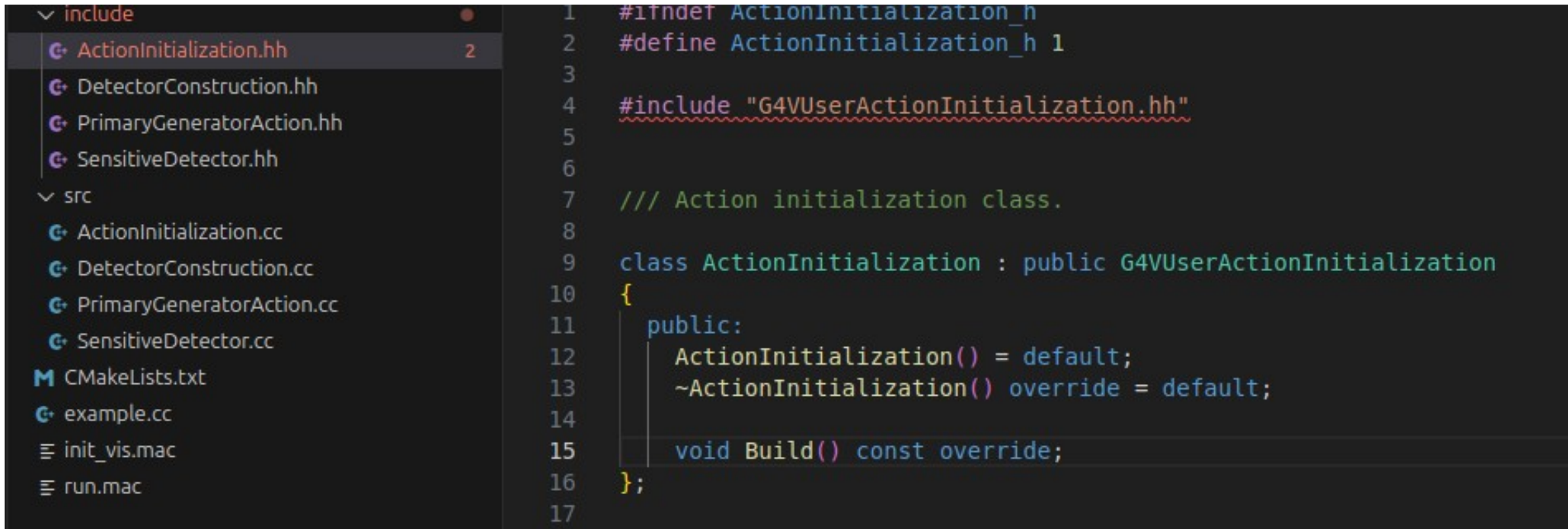
Обязательными для наследования являются **3 класса**:

- *G4VUserDetectorConstruction*
задается геометрия детектора и используемые материалы
- *G4VUserPhysicsList*
задается используемые частицы, и взаимодействия в которых они участвуют;
- *G4VUserActionInitialization*
используется для инициализации всех классов действий, одним из которых является класс, отвечающий за запуск первичных частиц
 - *G4VUserPrimaryGeneratorAction*
создаются первичные частицы – задается их тип, направление движения, энергия и т.д.

Дополнительный классы

- *G4UserRunAction*
позволяет задать действия в начале/конце run-a
- *G4UserEventAction*
позволяет задать действия в начале/конце event-a
- *G4UserTrackingAction*
позволяет задать действия при начале/завершении движения частицы
- *G4UserSteppingAction*
позволяет задать действия выполняемые на каждом шаге движения частиц.

Дополнительный классы



The image shows a code editor interface. On the left is a file explorer with a tree view. The 'include' directory is expanded, showing files: ActionInitialization.hh (selected), DetectorConstruction.hh, PrimaryGeneratorAction.hh, and SensitiveDetector.hh. The 'src' directory is also expanded, showing: ActionInitialization.cc, DetectorConstruction.cc, PrimaryGeneratorAction.cc, and SensitiveDetector.cc. Other files listed are CMakeLists.txt, example.cc, init_vis.mac, and run.mac. On the right, the content of ActionInitialization.hh is displayed, with line numbers 1 through 17 on the left margin. The code is as follows:

```
1  #ifndef ActionInitialization_h
2  #define ActionInitialization_h 1
3
4  #include "G4VUserActionInitialization.hh"
5
6
7  /// Action initialization class.
8
9  class ActionInitialization : public G4VUserActionInitialization
10 {
11     public:
12         ActionInitialization() = default;
13         ~ActionInitialization() override = default;
14
15         void Build() const override;
16 };
17
```

Дополнительный классы



The image shows a code editor interface. On the left is a file explorer with a dark background. It has two main sections: 'include' and 'src'. Under 'include', there are four files: 'ActionInitialization.hh', 'DetectorConstruction.hh', 'PrimaryGeneratorAction.hh', and 'SensitiveDetector.hh'. Under 'src', there are four files: 'ActionInitialization.cc', 'DetectorConstruction.cc', 'PrimaryGeneratorAction.cc', and 'SensitiveDetector.cc'. Below these, there are three other files: 'CMakeLists.txt', 'example.cc', 'init_vis.mac', and 'run.mac'. The 'ActionInitialization.cc' file is selected and highlighted. On the right side of the editor, the content of 'ActionInitialization.cc' is displayed. The code is as follows:

```
1
2
3  #include "ActionInitialization.hh"
4
5  #include "PrimaryGeneratorAction.hh"
6
7
8  //....ooo0000ooo.....ooo0000ooo.....ooo0000ooo.....ooo0000ooo.....
9
10
11 //....ooo0000ooo.....ooo0000ooo.....ooo0000ooo.....ooo0000ooo.....
12 |
13 void ActionInitialization::Build() const
14 {
15     SetUserAction(new PrimaryGeneratorAction);
16 }
17
18 //....ooo0000ooo.....ooo0000ooo.....ooo0000ooo.....ooo0000ooo.....
19
20
```

Дополнительный классы

```
void ActionInitialization::Build() const
{
    SetUserAction(new PrimaryGeneratorAction);
    SetUserAction(new RunAction);

    EventAction *ev = new EventAction();
    SetUserAction(ev);

    TrackingAction *tr = new TrackingAction(ev);
    SetUserAction(tr);
    SetUserAction(new SteppingAction);
}
```

Дополнительный классы

| | |
|-----------------------------|---|
| include | |
| • ActionInitialization.hh | M |
| • DetectorConstruction.hh | M |
| • EventAction.hh | M |
| • PrimaryGeneratorAction.hh | M |
| • Py8Decayer.hh | U |
| • Py8DecayerPhysics.hh | U |
| • RunAction.hh | M |
| • SteppingAction.hh | U |
| • TrackerSD.hh | M |
| • TrackingAction.hh | U |
| src | |
| • ActionInitialization.cc | M |
| • DetectorConstruction.cc | M |
| • EventAction.cc | M |
| • PrimaryGeneratorAction.cc | M |
| • Py8Decayer.cc | U |
| • Py8DecayerPhysics.cc | U |
| • RunAction.cc | M |
| • SteppingAction.cc | U |
| • TrackerSD.cc | M |
| • TrackingAction.cc | U |

Дополнительный классы

```
#ifndef B2RunAction_h
#define B2RunAction_h 1

#include "G4UserRunAction.hh"
#include "globals.hh"

class G4Run;

/// Run action class

class RunAction : public G4UserRunAction
{
public:
    void BeginOfRunAction(const G4Run* run) override;
    void EndOfRunAction(const G4Run* run) override;
};

#endif
```


Дополнительный классы

| | |
|-----------------------------|---|
| include | |
| • ActionInitialization.hh | M |
| • DetectorConstruction.hh | M |
| • EventAction.hh | M |
| • PrimaryGeneratorAction.hh | M |
| • Py8Decayer.hh | U |
| • Py8DecayerPhysics.hh | U |
| • RunAction.hh | M |
| • SteppingAction.hh | U |
| • TrackerSD.hh | M |
| • TrackingAction.hh | U |
| src | |
| • ActionInitialization.cc | M |
| • DetectorConstruction.cc | M |
| • EventAction.cc | M |
| • PrimaryGeneratorAction.cc | M |
| • Py8Decayer.cc | U |
| • Py8DecayerPhysics.cc | U |
| • RunAction.cc | M |
| • SteppingAction.cc | U |
| • TrackerSD.cc | M |
| • TrackingAction.cc | U |

G4UserRunAction

Содержит два виртуальных метода:

void BeginOfRunAction(const G4Run* run)

Вызывается перед началом обработки событий, в начале Run

- Открываются файлы для записи данных.
- Инициализируются гистограммы (через G4AnalysisManager).
- Выводятся стартовые сообщения.

void EndOfRunAction(const G4Run* run)

Вызывается после обработки всех событий, в конце Run

- Сохраняются или анализируются накопленные данные.
- Закрываются файлы.
- Выводятся итоговые результаты

Дополнительный классы

```
#ifndef B2RunAction_h
#define B2RunAction_h 1

#include "G4UserRunAction.hh"
#include "globals.hh"

class G4Run;

/// Run action class

class RunAction : public G4UserRunAction
{
public:
    void BeginOfRunAction(const G4Run* run) override;
    void EndOfRunAction(const G4Run* run) override;
};

#endif
```

G4UserRunAction

```
void BeginOfRunAction(const G4Run* run) {  
    G4cout << "Запуск " << run->GetRunID() << " " << G4endl;  
}
```

run.mac

Установили размеры детектора 10x10x10 и electron 10 GeV

/run/initialize

/run/beamOn 10

Установили размеры детектора 15x15x11 и pion 100 GeV

/run/initialize

/run/beamOn 100

G4UserEventAction

Содержит два виртуальных метода

void BeginOfEventAction(const G4Event*)

Вызывается в начале обработки каждого события.

- Инициализируются счетчики и переменные для события
- Подготавливаются структуры данных для сбора информации
- Выводится информация о прогрессе симуляции

void EndOfEventAction(const G4Event*)

Вызывается после обработки всех треков в событии.

- Анализируются накопленные данные
- Записываются результаты в файлы или гistogramмы
- Выполняется фильтрация событий по заданным критериям

G4UserEventAction

Объявляем в class-е *double fEnergyDeposited*

```
void BeginOfEventAction(const G4Event* event) {  
  
    // Вывод прогресса каждые N событий  
    G4int eventID = event->GetEventID();  
    if (eventID % 1000 == 0) {  
        G4cout << "Обрабатывается событие " << eventID << G4endl;  
    }  
  
    fEnergyDeposited = 0.0;  
}
```

G4UserEventAction

Считаем для каждого трека или степа $fEnergyDeposited += Energy$

```
void EndOfEventAction(const G4Event* event) {  
    // Анализ данных события  
  
    if (fEnergyDeposited > 10.0*MeV) {  
        // Запись в гистограмму  
        G4AnalysisManager* analysisManager = G4AnalysisManager::Instance();  
  
        AnalysisManager->FillH1(0, что-то );  
    }  
}
```

G4UserTrackingAction

Содержит два виртуальных метода:

void PreUserTrackingAction(const G4Track*)

Вызывается перед началом обработки нового трека:

- Получить информацию о рождающейся частице
- Принять решение о необходимости отслеживания

void PostUserTrackingAction(const G4Track*)

Вызывается после завершения обработки трека:

- Анализировать историю частицы
- Собирать статистику

G4UserTrackingAction

```
void PreUserTrackingAction(const G4Track* track) {  
  
    if (track->GetKineticEnergy ()>100*Mev) {  
  
        Убиваем текущий трек  
        track->SetTrackStatus(fStopAndKill);  
  
    }  
  
}
```

G4UserSteppingAction

Это абстрактный базовый класс в Geant4, который позволяет пользователю вмешиваться в процесс моделирования на уровне отдельных шагов (steps) частиц.

void UserSteppingAction(const G4Step*)

- Собирать микроскопическую информацию о движении частиц
- Влиять на поведение частиц на каждом отрезке траектории
- Регистрировать физические процессы в момент их возникновения

G4UserSteppingAction

```
#ifndef B1SteppingAction_h
#define B1SteppingAction_h 1

#include "G4UserSteppingAction.hh"
#include "globals.hh"

class SteppingAction : public G4UserSteppingAction
{
public:
    void UserSteppingAction(const G4Step*) override;
}

//....ooo00000ooo.....ooo00000ooo.....ooo00000ooo.....

#endif
```

G4UserSteppingAction

```
void UserSteppingAction(const G4Step* step) {  
  
    // Получаем энерговыделение на шаге  
    G4double edep = step->GetTotalEnergyDeposit();  
  
    if (edep > 100.0) {  
        // Определяем объем, где произошло энерговыделение  
        G4VPhysicalVolume* volume = step->GetPreStepPoint()->GetPhysicalVolume();  
        G4String volumeName = volume->GetName();  
  
        if (volumeName == "myDetector"){  
  
            // Передаем данные в EventAction  
  
            eventAction->AddEnergyDeposit(edep);  
        }  
    }  
}
```

G4VUserPrimaryGeneratorAction

/// The primary generator action class with particle gun.

```
class PrimaryGeneratorAction : public G4VUserPrimaryGeneratorAction  
{  
  public:  
    PrimaryGeneratorAction();  
    ~PrimaryGeneratorAction() ;  
  
    // method from the base class  
    void GeneratePrimaries(G4Event*) ;  
  
  private:  
  
    G4ParticleGun *fParticleGun;  
  
};
```

G4VUserPrimaryGeneratorAction

PrimaryGeneratorAction::PrimaryGeneratorAction()

```
{  
    G4int n_particle = 1;  
    fParticleGun = new G4ParticleGun(n_particle);  
  
    // default particle kinematic  
    G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable();  
    G4String particleName;  
    G4ParticleDefinition* particle = particleTable->FindParticle(particleName = "proton");  
  
    fParticleGun->SetParticleDefinition(particle);  
    fParticleGun->SetParticleMomentumDirection(G4ThreeVector(0., 0., 1.));  
    fParticleGun->SetParticleEnergy(100 * MeV);  
}
```

PrimaryGeneratorAction::~~PrimaryGeneratorAction()

```
{  
    delete fParticleGun;  
}
```

G4VUserPrimaryGeneratorAction

```
void PrimaryGeneratorAction::GeneratePrimaries(G4Event* event){
```

```
    G4double x0 = 0;
```

```
    G4double y0 = 0;
```

```
    G4double z0 = -40;
```

Задаем начальные координаты

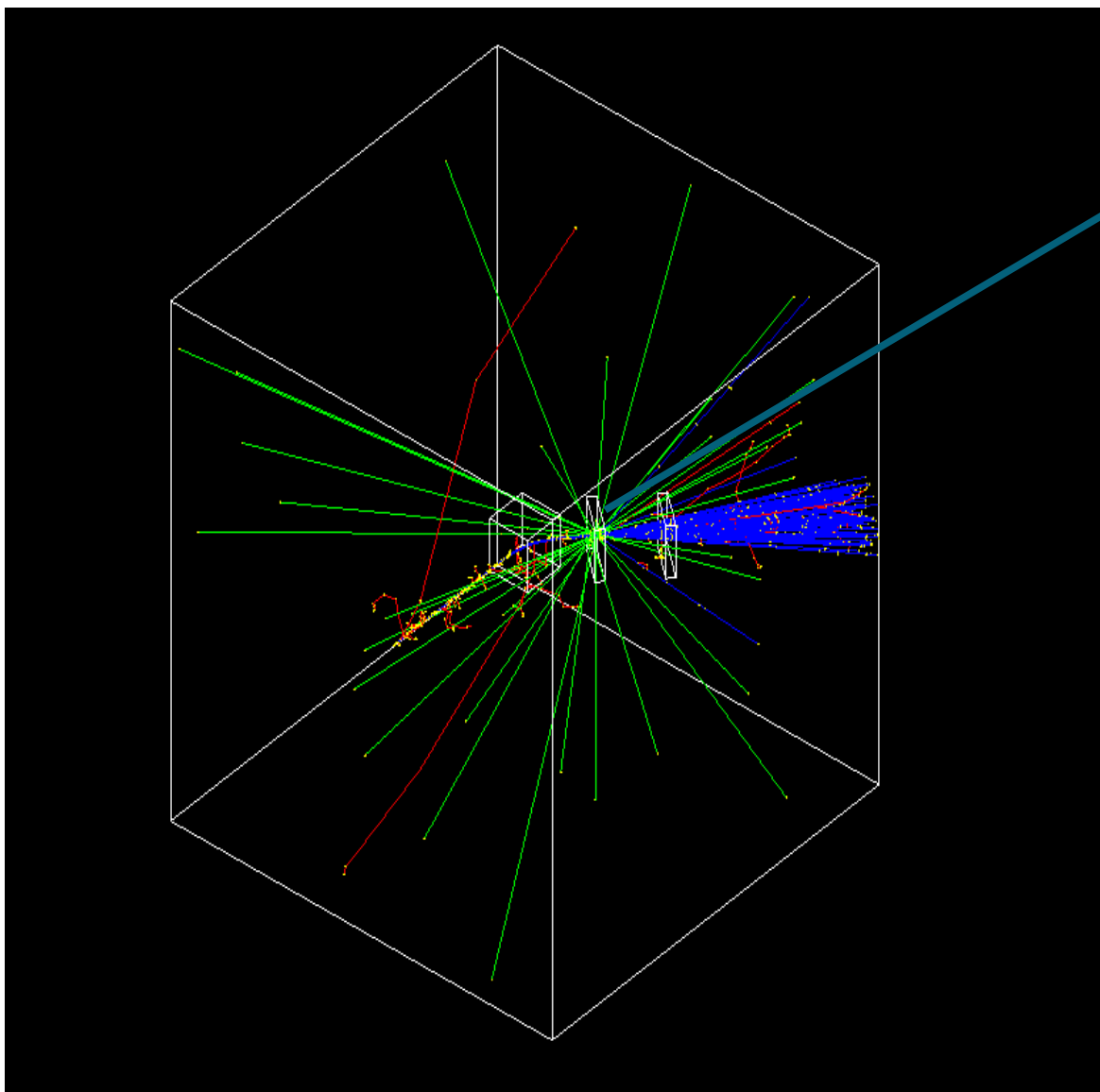
```
    fParticleGun->SetParticlePosition(G4ThreeVector(x0, y0, z0));
```

```
    fParticleGun->GeneratePrimaryVertex(event);
```

Генерируем событие

```
}
```

Задание



G4_W

1) создать RunAction

src/RunAction.cc

include/RunAction.hh

а) в нем задать **AnalisisManager**

2) Создать SteppingAction

src/SteppingAction.cc

include/SteppingAction.hh

а) заполнить h2 гистограмму

XY координатами. Сделать это только для
вашего детектора.

3) Задать π on 1 ГэВ, запустите под углом

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

```
RunAction::RunAction()
{
    auto analysisManager = G4AnalysisManager::Instance();

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

    analysisManager->CreateH1("EDep", "Energy deposit", 100, 0., 800*MeV);
}
```

```
void RunAction::BeginOfRunAction(const G4Run* run)
{
    auto analysisManager = G4AnalysisManager::Instance();
    analysisManager->OpenFile("MyFile");
}
```

```
void RunAction::EndOfRunAction(const G4Run* run)
{
    auto analysisManager = G4AnalysisManager::Instance();

    analysisManager->Write();
    analysisManager->CloseFile();
}
```

Заполнить Гистограммы

```
void UserSteppingAction(const G4Step* step) {  
    {  
        // Заполняем гистограмму h2 (X,Y) для вашего детектора  
        Проверка через имя.  
    }  
}
```

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
```

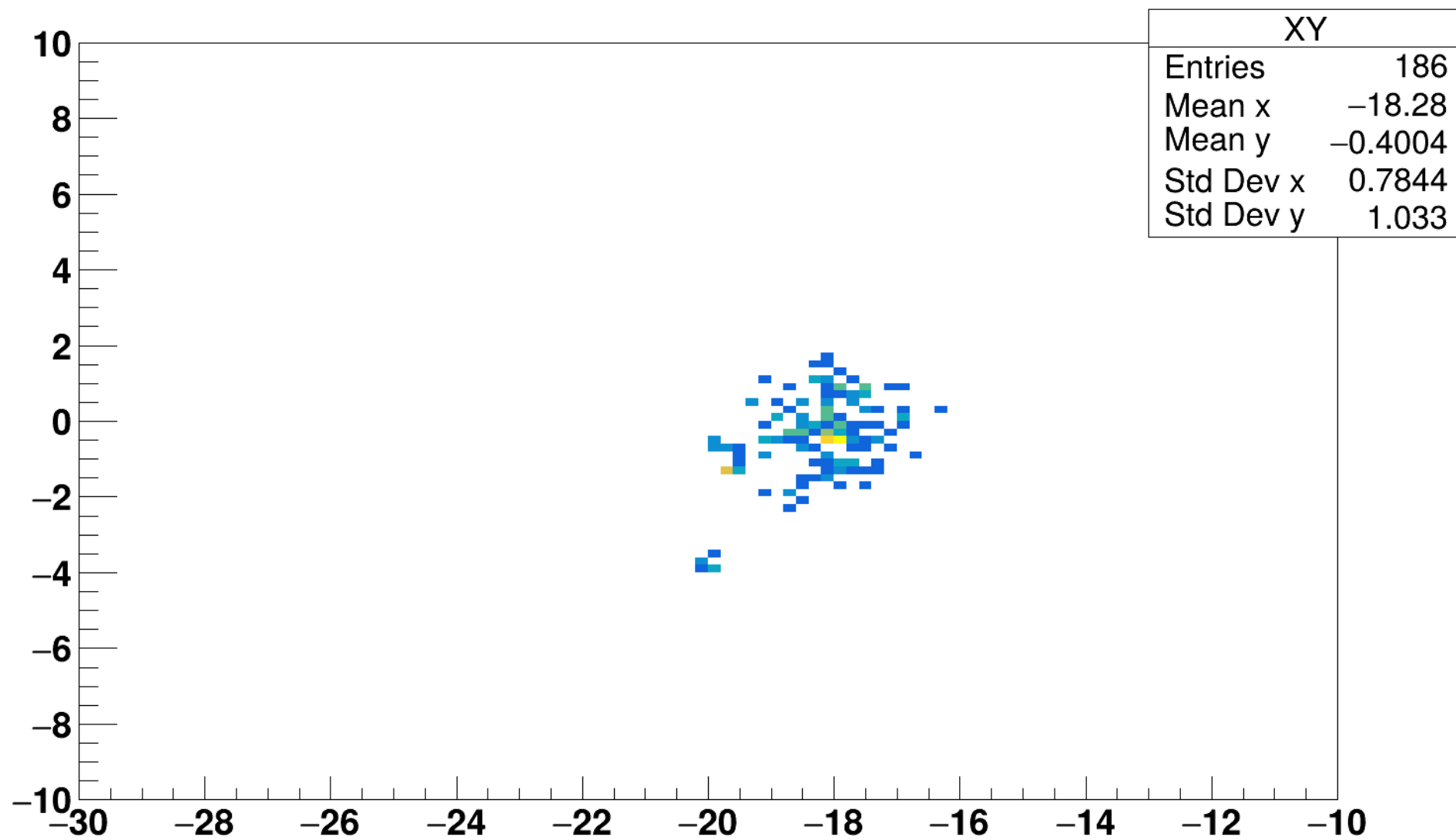
Проверяем что геометрия правильная

Второй способ через batch

```
./SimpleExample run.mac
```

*Запускаем на 1000 событий
/run/beamOn 1000*

Задание



Задание

- Получаем файл с именем файла.root

Чтобы открыть его

- root имя.root

далее пишем

- TBrowser a