Введение в (GEometry ANd Tracking) GEANT4

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

Основные понятие

Сеанс (Run)

- Как и в реальном эксперименте, сеанс начинается с “Beam On”

- Сеанс это период набора статистики, в котором не меняются условия проведения эксперимента.
  - параметры пучка
  - физические процессы
  - конфигурация детектора

  То есть, детектор не доступен во время сеанса.

- В начале прогона геометрия оптимизируется для навигации, и таблицы поперечных сечений рассчитываются в соответствии с материалами, представленными в геометрии, и заданными предельными значениями.

- Класс *G4Run*. Управляется *G4RunManager*.

Основные понятие

Событие (Event)

- Единичное независимое измерение

- Класс *G4Event* содержит
  - Первичные и вторичные частицы
  - Коллекцию хитов
  - Коллекцию траекторий
  - Оцифрованный сигнал

- *G4Event* создается объектом класса *G4RunManager* и передается объекту класса *G4EventManager*, который осуществляет управление событием

Основные понятие

Трек (Track)

- Треки представлены классом *G4Track*, и содержат информацию о первом и последнем шаге

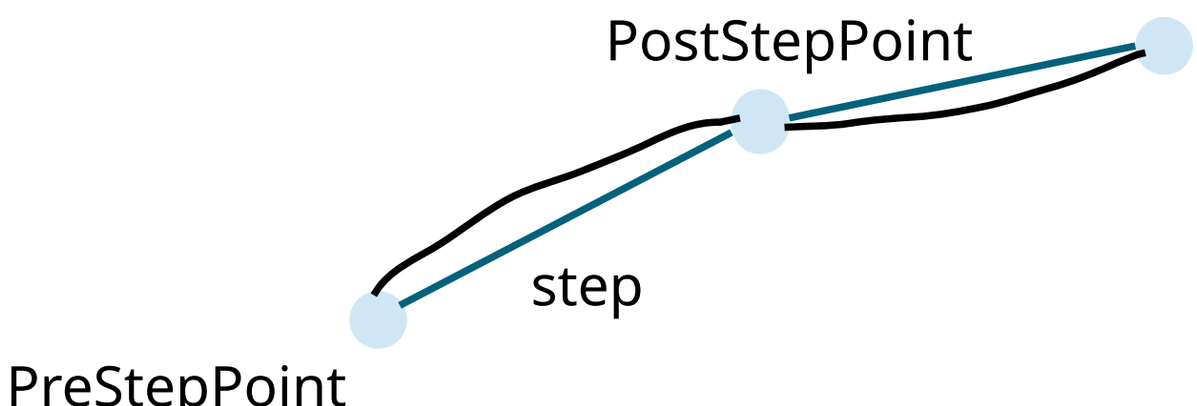
- Объект *G4Track*, таким образом, описывает полное продвижение частицы в веществе

- Трек удаляется, когда:
  - он выходит за пределы объёма мира;
  - он исчезает (например, распадается);
  - его кинетическая энергия падает до нуля и не требуется дополнительный процесс;
  - пользователь решает завершить его.

Основные понятие

Шаг (G4Step)

- Описывает минимальное продвижение частицы через вещество с учетом различных физических процессов.
- На каждом шаге частица может терять энергию, менять направление, порождать вторичные частицы
- Шаг имеет две точки, а также «дельта»-информацию о частице (потеря энергии, время пролёта, затраченное на шаг, и т. д.).



Основные понятие

Срабатывание (Hit)

- *Hit* - это запись о взаимодействии частицы с чувствительным объемом детектора
- Содержит информацию о координате и времени взаимодействия, энергии и импульсе частицы в этой точке, энерговыделении, геометрическую информацию
- Hits являются данными, которые затем преобразуются в оцифрованный сигнал (*Digi*)

Основные понятие

Оцифрованный сигнал (Digi)

- Оцифрованное представление *Hit*, имитирующее реальную электронику детектора с возможным добавлением эффектов шума, дискриминации, временных задержек и тд

- Позволяет максимально приблизить моделирование к реальному эксперименту

- Служит в основном для отладки алгоритмов реконструкции

Важно! Типы данных

- Псевдонимы для примитивных типов данных для обеспечения кроссплатформенной совместимости:
  G4double, G4float, G4int, G4bool, G4long

- Расширенная версия строки под названием **G4String**
  - наследует от **std::string** все методы и операторы
  - плюс несколько дополнительных методов

- Вместо **std::cout/std::endl** и **std::cerr**, в Geant4 используйте **G4cout, G4endl и G4cerr**

- **G4ThreeVector –** трехкомпонентный класс, соответствующий реальному физическому вектору
  G4ThreeVector dimensions {1.0, 2.0, 3.0 };

Пожалуйста, используйте эти типы для лучшей совместимости !

Важно! Физические единицы

По умолчанию

- миллиметр,
- наносекунда,
- мегаэлектрон-вольт,
- Кельвин,
- моль,
- кандела,
- радиан,
- стерадиан.

Не используйте единицы измерения по умолчанию!

При указании размеров всегда умножайте на соответствующую единицу измерения

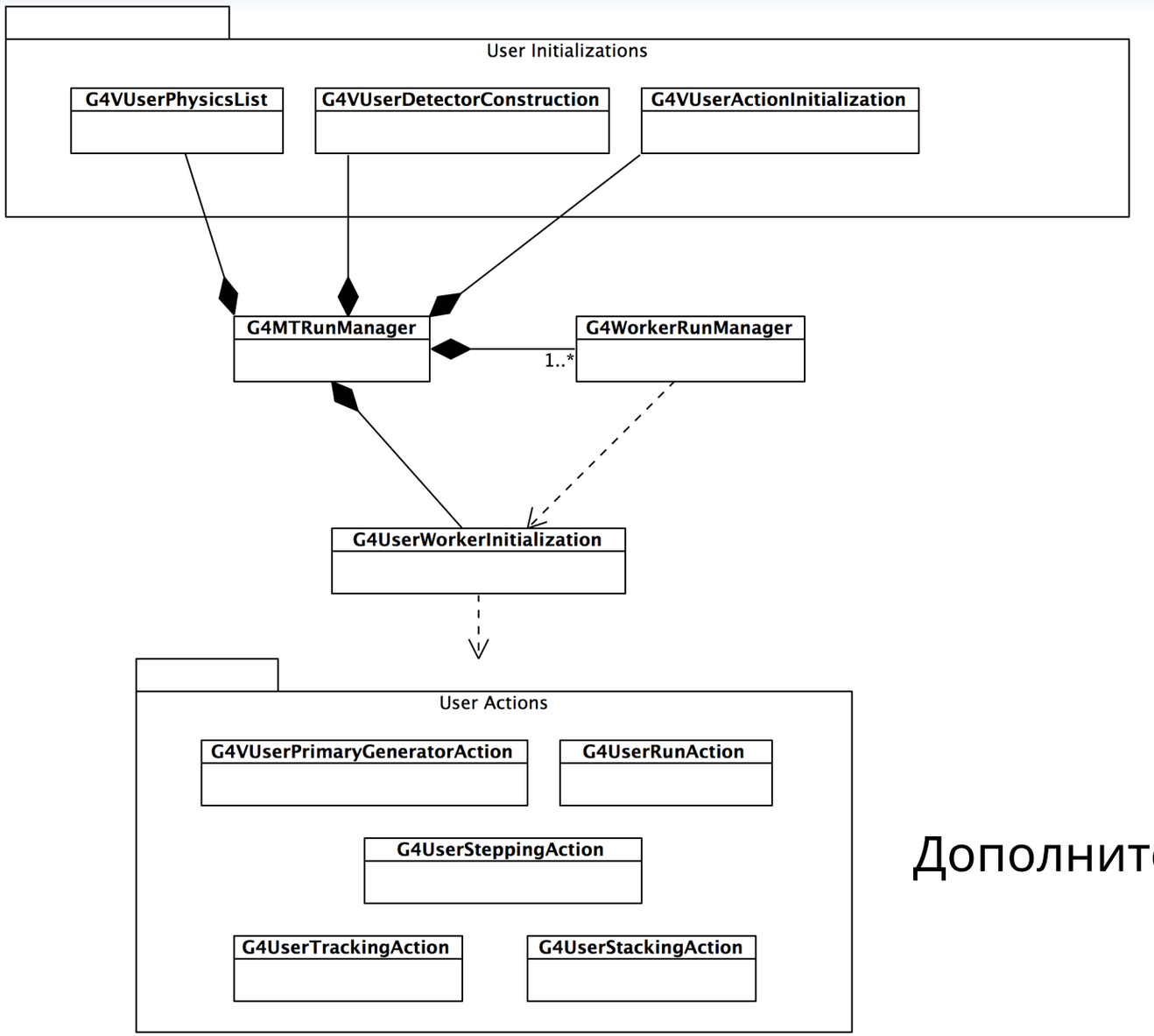
```
double kineticEnergy = 10 * MeV;
double size = 1*meter;
```

Наиболее распространённые единицы измерения определены в библиотеке CLHEP (входит в состав Geant4): *G4SystemOfUnits.hh, CLHEP/SystemOfUnits.hh*

Выходные данные в единицах измерения: **разделить** значение на единицу измерения:

```
G4cout << size/meter << "(meter)" << G4endl;
```

Схема Geant4



Базовые класс

Дополнительные классы

Базовые класс

Обязательными для работы являются **3 класса**:

- *G4VUserDetectorConstruction*
  задается геометрия детектора и используемые материалы

- *G4VUserPhysicsList*
  задается используемые частицы, и взаимодействия в которых они участвуют;

- *G4VUserPrimaryGeneratorAction*
  создаются первичные частицы – задается их тип, направление движения, энергия и т.д.

Базовые класс, Но

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-а

- *G4UserEventAction*
  позволяет задать действия в начале/конце event-а

- *G4UserStackingAction*
  позволяет задать действия в момент появления вторичных частиц

- *G4UserTrackingAction*
  позволяет задать действия при начале/завершении движения частицы

- *G4UserSteppingAction*
  позволяет задать действия выполняемые на каждом шаге движения частиц.

Run Manager

Связь с ядром Geant4 осуществляет объект класса

G4RunManager - однопоточный режим

```
G4RunManager* runManager = new G4RunManager;
```

или

G4MTRunManger – многопоточный режим

```
G4MTRunManager* runManager = new G4MTRunManager;
runManager->SetNumberOfThreads(nthreads);
```

```
auto* runManager = G4RunManagerFactory::CreateRunManager();
```

Фабрика анализирует переменные окружения (например, G4MULTITHREADED) и конфигурацию сборки Geant4, чтобы создать подходящий экземпляр

Run Manager

В Geant4 метод *SetUserInitialization* класса *G4RunManager* используется для установки пользовательских классов инициализации, которые определяют ключевые компоненты симуляции. Существует три основных перегруженных версии этого метода, каждая из которых отвечает за свой аспект инициализации:

```
virtual void SetUserInitialization (G4VUserActionInitialization *userInit)
virtual void SetUserInitialization (G4VUserDetectorConstruction *userInit)
virtual void SetUserInitialization (G4VUserPhysicsList *userInit)
```

Простейшая программа

```
#include "DetectorConstruction.hh"
#include "ActionInitialization.hh"
#include "G4RunManagerFactory.hh"
#include "FTFP_BERT.hh"
```

Подключаем заголовочные файлы

запуск программы
./example

```
int main(){
    G4RunManager* runManager = new G4RunManager;

    runManager->SetUserInitialization(new DetectorConstruction);

    auto physicsList = new FTFP_BERT;
    runManager->SetUserInitialization(physicsList);

    runManager->SetUserInitialization(new ActionInitialization);

    runManager->Initialize();

    runManager->BeamOn(3);

    delete runManager;
    return 0;
}
```

Создание класса G4RunManager, он контролирует выполнение программы. Управляет циклами по событиям и сеансом

Создание геометрии, материала детектора и мишени

Используется готовый физический лист из Geant4 (набора моделируемых частиц и физических процессов)

Объявление начальных частиц (параметры пучка) и подключение прочих классов

Инициализация ядра Geant4

Запускаем 3 частицы

Окончание работы, вызов деструктора (удаление) G4RunManager

Простейшая программа с macros

```
#include "DetectorConstruction.hh"
#include "ActionInitialization.hh"
#include "G4RunManagerFactory.hh"
#include "G4UImanager.hh"
#include "FTFP_BERT.hh"


int main(int argc, char** argv){

    G4RunManager* runManager = new G4RunManager;

    runManager->SetUserInitialization(new DetectorConstruction);

    auto physicsList = new FTFP_BERT;
    runManager->SetUserInitialization(physicsList);

    runManager->SetUserInitialization(new ActionInitialization);

    G4UImanager* UImanager = G4UImanager::GetUIpointer();
    G4String command = "/control/execute ";
    G4String fileName = argv[1];
    UImanager->ApplyCommand(command + fileName);

    delete runManager;
    return 0;
}
```

```
./example run.mac
          /run/initialize
          /run/beamOn 1
```

Получение указателя на менеджера взаимодействия с пользователем нужен, что бы можно было отправлять команды в проект

Выполнение команды, которая запускает выполнение макроса с указанным именем файла.

Простейшая программа с GUI

```
#include "DetectorConstruction.hh"
#include "ActionInitialization.hh"
#include "G4RunManagerFactory.hh"
#include "G4UImanager.hh"
#include "G4UIExecutive.hh"
#include "G4VisExecutive.hh"
#include "FTFP_BERT.hh"

int main(int argc, char** argv){

    G4RunManager* runManager = new G4RunManager;
    runManager->SetUserInitialization(new DetectorConstruction);
    auto physicsList = new FTFP_BERT;
    runManager->SetUserInitialization(physicsList);
    runManager->SetUserInitialization(new ActionInitialization);

    auto visManager = new G4VisExecutive(argc, argv);
    visManager→Initialize();

    G4UIExecutive* ui = new G4UIExecutive(argc, argv);

    G4UImanager* UImanager = G4UImanager::GetUIpointer();
    UImanager->ApplyCommand("/control/execute init_vis.mac");
    ui->SessionStart();
    delete ui;
    return 0;
}
```

./example

В коде мы вызываем макрос init_vis.mac

/run/initialize

/control/execute vis.mac

/vis/open OGL

/vis/drawVolume

Инициализация системы визуализации для графического отображения геометрии и треков частиц.

Создание интерактивного пользовательского интерфейса

Инициализация менеджера визуализации

Параметры отображения берем из заранее подготовленного файла

Запуск интерактивной сессии (программа ждет команд пользователя)

Visualization

В файле vis.mac

/vis/open OGL	Открывает визуализацию с выбором графической системы по умолчанию
/vis/drawVolume	Отрисовывает геометрию детектора, все объемы
/vis/scene/add/scale	Линейку масштаба
/vis/scene/add/axes	Координатные оси (X=red, Y=green, Z=blue)
`/vis/geometry/set/visibility World 0 false`	Сделать объем "World" невидимым.
`/vis/scene/add/trajectories smooth`	Добавляет траектории частиц с сглаживанием.

Геометрия в Geant4

```
#include "DetectorConstruction.hh"
#include "ActionInitialization.hh"
#include "G4RunManagerFactory.hh"
#include "G4UImanager.hh"
#include "G4UIExecutive.hh"
#include "G4VisExecutive.hh"
#include "FTFP_BERT.hh"
int main(int argc, char** argv){
    G4RunManager* runManager = new G4RunManager;

    runManager->SetUserInitialization(new DetectorConstruction);

    auto physicsList = new FTFP_BERT;
    runManager->SetUserInitialization(physicsList);
    runManager->SetUserInitialization(new ActionInitialization);

    auto visManager = new G4VisExecutive(argc, argv);
    visManager→Initialize();

    G4UIExecutive* ui = new G4UIExecutive(argc, argv);

    G4UImanager* UImanager = G4UImanager::GetUIpointer();
    UImanager->ApplyCommand("/control/execute init_vis.mac");
    ui->SessionStart();
    delete ui;
    return 0;
}
```

Создание геометрии, материала детектора и мишени

DetectorConstruction

DetectorConstruction.hh

```
#ifndef DETECTORCONSTRUCTION_HH
#define DETECTORCONSTRUCTION_HH
#include <G4VUserDetectorConstruction.hh>

class DetectorConstruction : public G4VUserDetectorConstruction{
public:
    G4VPhysicalVolume *Construct() override;
};
#endif
```

это директивы препроцессора,
используемые для защиты от множественного включения заголовочных файлов.

наследуемся от класса G4VUserDetectorConstruction

Объявление виртуального метода Construct(). Ключевое слово override указывает, что метод переопределяет базовый виртуальный метод.

DetectorConstruction.cxx

```
#include "DetectorConstruction.hh"
#include "G4SystemOfUnits.hh"


G4VPhysicalVolume *DetectorConstruction::Construct() {
   G4VPhysicalVolume* physWorld…..

    bla bla bla bla


   return physWorld;
}
```

Описываем детектор

Геометрия в Geant4

Тело (Solid) — определяет форму и размеры геометрического объема, задавая его пространственные границы и конфигурацию.

Логический объем (Logical Volume) — объединяет информацию о материале, электромагнитных полях и других свойствах, связывая их с определенным телом **(Solid)**. Также содержит указания о чувствительности объема для регистрации частиц.

Физический объем (Physical Volume) — обеспечивает позиционирование логического объема в пространстве путем указания координат и ориентации относительно системы координат материнского логического объема.

Геометрия в Geant4

- ***G4CSVSolid*** — базовый класс для твёрдых тел (формы: шар, бокс, цилиндр и т.д.).
- ***G4LogicalVolume*** — логический объём: связывает форму (solid) с материалом и чувствительными детекторами.
- ***G4Material*** — материал, из которого состоит объём.
- ***G4VPhysicalVolume*** — физический объём: размещает логический объём в пространстве (с позицией и поворотом).

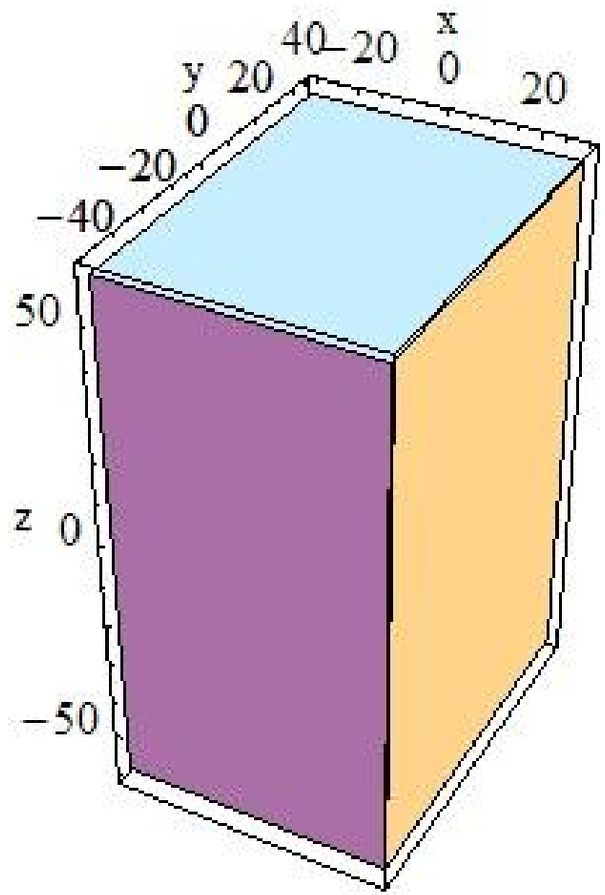
Геометрия в Geant4

- ***G4CSVSolid*** — базовый класс для твёрдых тел (формы: шар, бокс, цилиндр и т.д.).
- ***G4LogicalVolume*** — логический объём: связывает форму (solid) с материалом и чувствительными детекторами.
- ***G4Material*** — материал, из которого состоит объём.
- ***G4VPhysicalVolume*** — физический объём: размещает логический объём в пространстве (с позицией и поворотом).

Простейшие CSG формы

Box

```
G4Box(const G4String& pName,
                G4double  pX,
                G4double  pY,
                G4double  pZ)
```



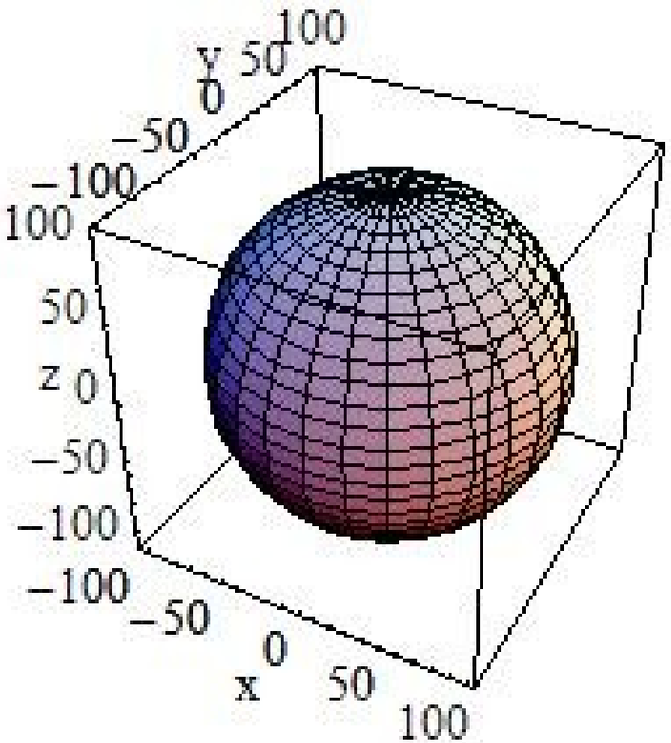
pX = 30, pY = 40, pZ = 60

- pX - half length in X
- pY - half length in Y
- pZ - half length in Z
- `pName-name`

Простейшие CSG формы

Full Solid Sphere

```
G4Orb(const G4String& pName,
                 G4double  pRmax)
```



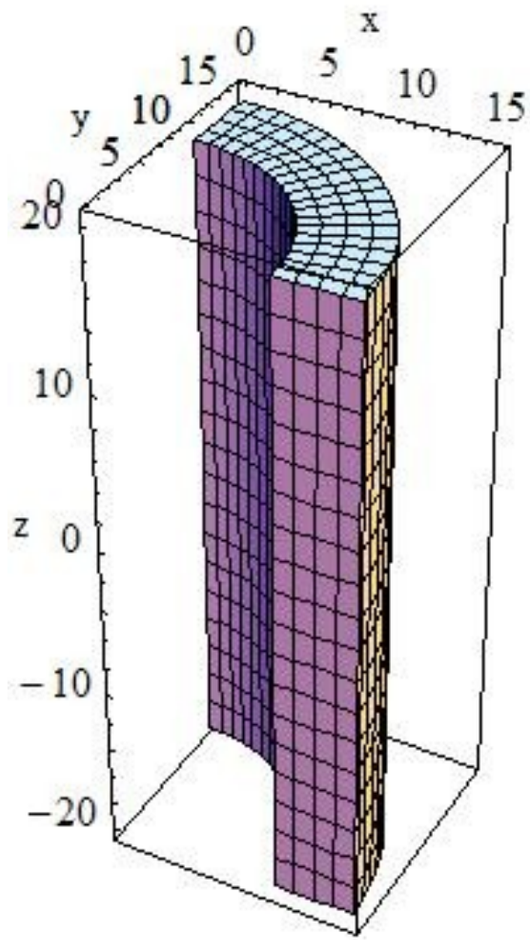
pRmax = 100

- pRmax - Outer radius
- pName-name

Простейшие CSG формы

Cylindrical Section or Tube

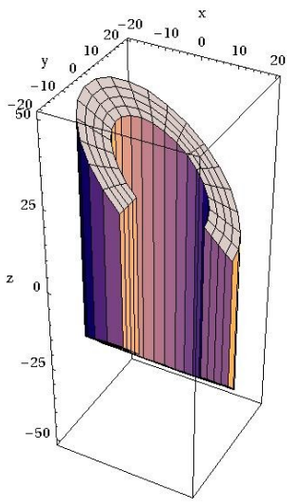
```
G4Tubs(const G4String& pName,
                 G4double  pRMin,
                 G4double  pRMax,
                 G4double  pDz,
                 G4double  pSPhi,
                 G4double  pDPhi)
```

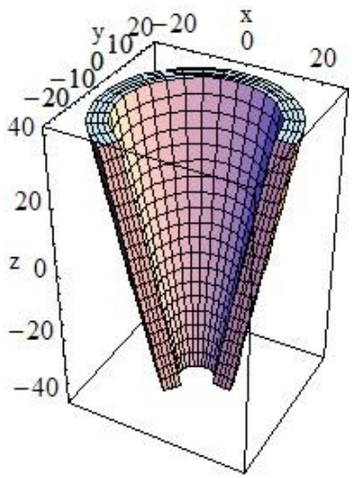


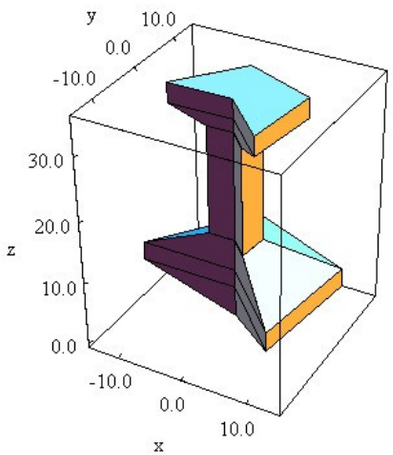
- pRMin - Inner radius
- pRMax - Outer radius
- pdz - Half length in Z
- pSPhi - Starting phi angle in radians
- pDPhi - Angle of the segment in radians
- `pName` - name

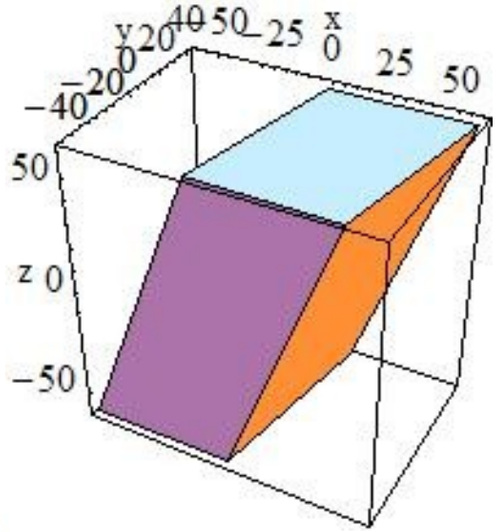
pRMin = 10, pRMax = 15, pDz = 20

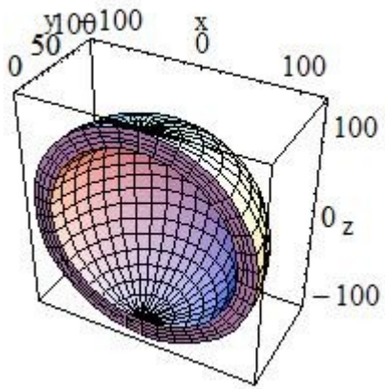
Простейшие CSG формы

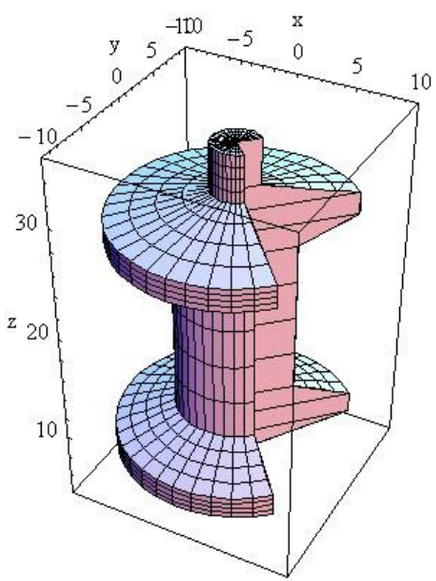


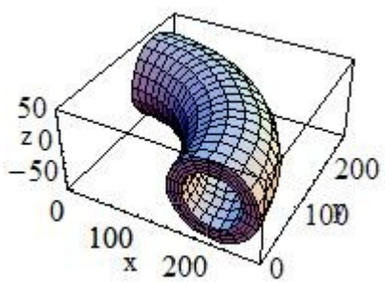


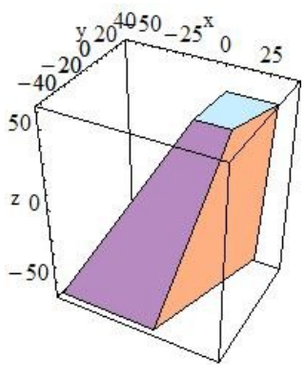


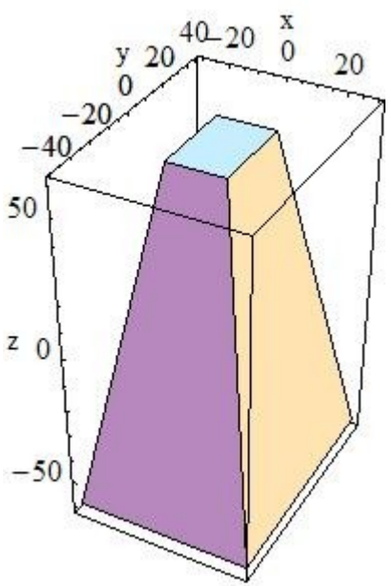


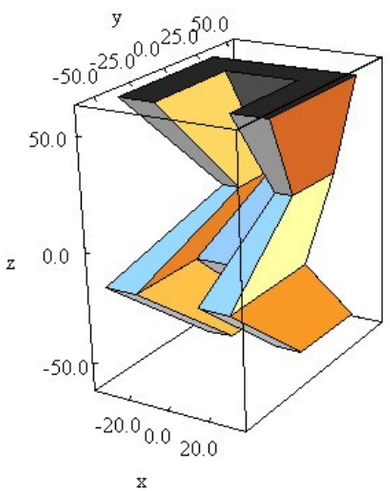


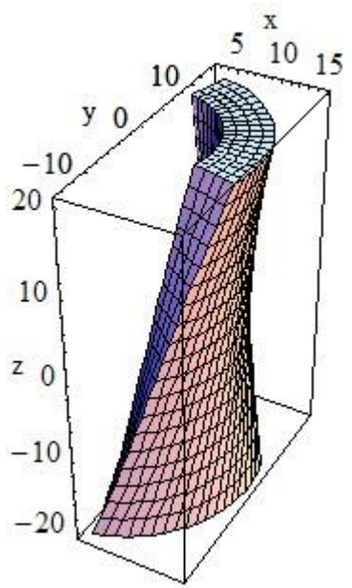












Геометрия в Geant4

Мир (World) — должен быть самым большим объёмом и содержать все остальные.

```
   G4double world_sizeXY = 100*meter;
.  G4double world_sizeZ  = 100*meter;

   G4Box* solidWorld =
           new G4Box( "World",
                      0.5*world_sizeXY,
                      0.5*world_sizeXY,
                      0.5*world_sizeZ);
```

Геометрия в Geant4

- ***G4CSVSolid*** — базовый класс для твёрдых тел (формы: шар, бокс, цилиндр и т.д.).
- ***G4LogicalVolume*** — логический объём: связывает форму (solid) с материалом и чувствительными детекторами.
- ***G4Material*** — материал, из которого состоит объём.
- ***G4VPhysicalVolume*** — физический объём: размещает логический объём в пространстве (с позицией и поворотом).

Логический объем

```
G4double world_sizeXY = 1*meter;
G4double world_sizeZ  = 1*meter;

G4Box* solidWorld = new G4Box( "World", 0.5*world_sizeXY, 0.5*world_sizeXY, 0.5*world_sizeZ);
```

Теперь, используя созданное тело, мы можем перейти к формированию логического объема. Однако перед этим необходимо определить материал, из которого будет состоять данный объем. Для этого можно либо самостоятельно создать новый материал, либо воспользоваться одним из предопределенных материалов через менеджер NistManager:

```
auto nist = G4NistManager::Instance();
G4Material* world_mat = nist->FindOrBuildMaterial("G4_AIR");


G4LogicalVolume* logicWorld =
        new G4LogicalVolume(solidWorld,           //its solid
                            world_mat,            //its material
                            "World");             //its name
```

Геометрия в Geant4

- ***G4CSVSolid*** — базовый класс для твёрдых тел (формы: шар, бокс, цилиндр и т.д.).
- ***G4LogicalVolume*** — логический объём: связывает форму (solid) с материалом и чувствительными детекторами.
- ***G4Material*** — материал, из которого состоит объём.
- ***G4VPhysicalVolume*** — физический объём: размещает логический объём в пространстве (с позицией и поворотом).

P4VPhysicalVolume

```
G4bool checkOverlaps = true; // проверка на пересечения объемов

G4VPhysicalVolume* physWorld =
        new G4PVPlacement(  0,                      //no rotation
                            G4ThreeVector(0, 0, 0),
                            logicWorld,           //its logical volume
                            "World",                 //its name
                            0,                       //its mother volume
                            false,                  //no boolean operation
                            0,                       //copy number
                            checkOverlaps);
```

Материнский объем, этот самый первый, поэтому

уникальный идентификатор экземпляра объема

P4VPhysicalVolume

```
G4double env_sizeXY = 10*meter;

G4double env_sizeZ = 10*meter;

auto solidEnv = new G4Box("Box",  0.5 * env_sizeXY, 0.5 * env_sizeXY, 0.5 * env_sizeZ);


G4Material* env_mat = nist->FindOrBuildMaterial("G4_Al");

auto logicEnv = new G4LogicalVolume(solidEnv,  env_mat,  "Box");


new G4PVPlacement(0,  G4ThreeVector(0,0,0),  logicEnv,  "Box",  logicWorld,  false,  0,
checkOverlaps);
```

Сборка проекта

- snap install code --classic (пароль dubna)
- source /opt/geant4/geant4-v11.3.2/install/bin/geant4.sh
- git clone https://github.com/Arivanoviktor/geant4.git

- Создаем директорию

```
mkdir practice1
cd practice1
cp -r ../geant4/p1 ./
code p1
```

- Создаем директорию

```
mkdir build
```

- Собираем проект

```
cd build
cmake ../p1
make
```