Введение в (GEometry ANd Tracking) GEANT4

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

https://disk.jinr.ru/index.php/s/8ps2J3J43yijg46

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.

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 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)

DetectorConstruction

Надо создать два файла

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() {


    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.

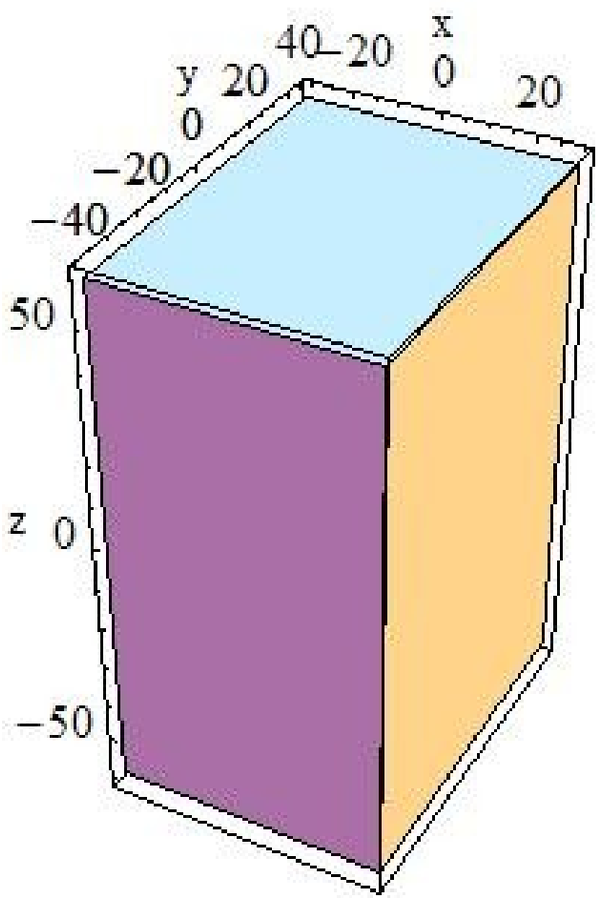
Физический объем (Physical Volume) — обеспечивает позиционирование логического объема в пространстве путем указания координат и ориентации относительно системы координат материнского логического объема.

Простейшие CSG формы

Box

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

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



pX = 30, pY = 40, pZ = 60

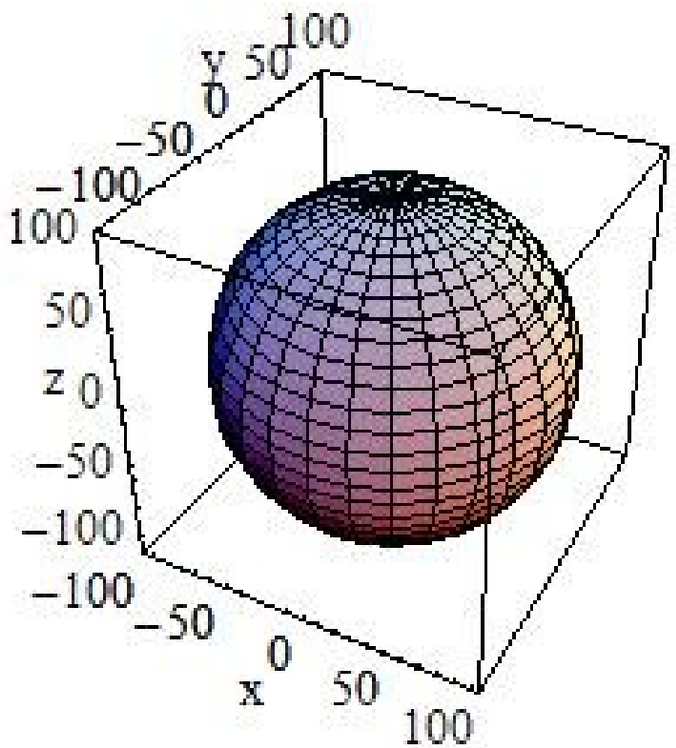
auto solidWorld =new G4Box("World", 30, 40, 60);

Простейшие CSG формы

Full Solid Sphere

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

- *pName-name*
- *pRmax - Outer radius*



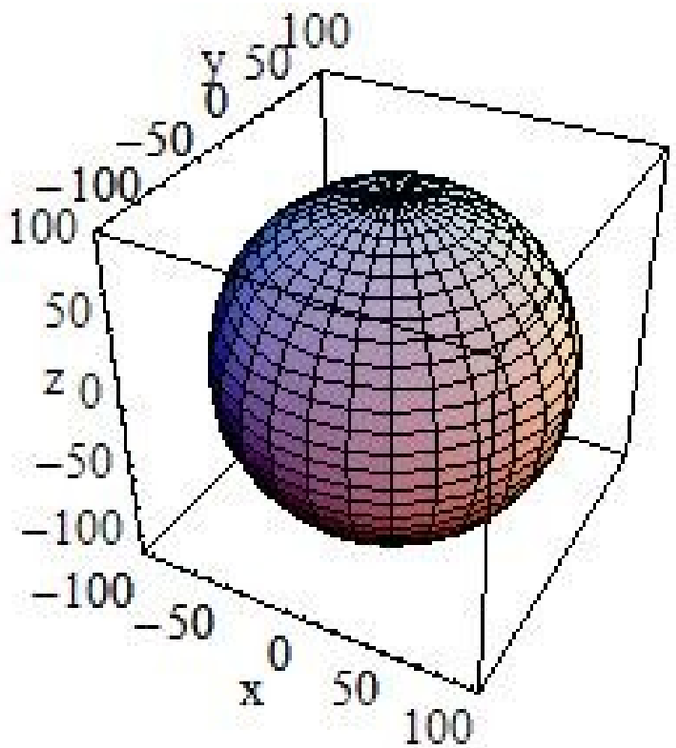
pRmax = 100

Простейшие CSG формы

Full Solid Sphere

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

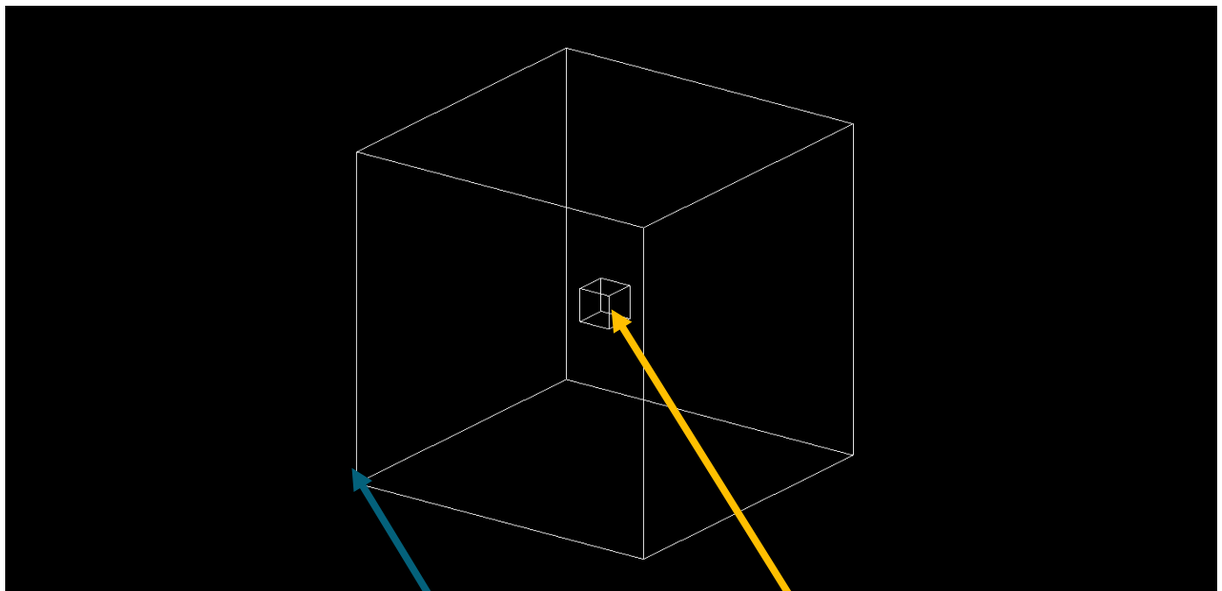
- *pName-name*
- *pRmax - Outer radius*



pRmax = 100

auto solidOrb = new G4Orb("Orb", 100);

Geometry



```
G4NistManager* nist = G4NistManager::Instance();
G4Material* env_mat = nist->FindOrBuildMaterial("G4_Al");
G4Material* world_mat = nist->FindOrBuildMaterial("G4_AIR");
```

```
G4bool checkOverlaps = true;
G4double world_sizeXY = 100;
G4double world_sizeZ = 100;
auto solidWorld =new G4Box("World",  0.5 * world_sizeXY, 0.5 * world_sizeXY, 0.5 * world_sizeZ);
auto logicWorld = new G4LogicalVolume(solidWorld,  world_mat, "World");
auto physWorld = new G4PVPlacement(nullptr,   G4ThreeVector(),  logicWorld,  "World",  nullptr,  false,  0,  checkOverlaps);
```

```
G4double env_sizeXY = 10;
G4double env_sizeZ = 10;
auto solidEnv = new G4Box("Box",   0.5 * env_sizeXY, 0.5 * env_sizeXY, 0.5 * env_sizeZ);
auto logicEnv = new G4LogicalVolume(solidEnv,  env_mat, "Box");
auto physEnv = new G4PVPlacement(nullptr, G4ThreeVector(), logicEnv,  "Box", logicWorld,  false,  0, checkOverlaps);
```

Geometry

```
G4NistManager* nist = G4NistManager::Instance();
G4Material* env_mat = nist->FindOrBuildMaterial("G4_Al");
G4Material* world_mat = nist->FindOrBuildMaterial("G4_AIR");
```

Создаем World, который у нас имеет форму куба

```
G4double world_sizeXY = 100;
G4double world_sizeZ = 100;
auto solidWorld =new G4Box("World",  0.5 * world_sizeXY, 0.5 * world_sizeXY, 0.5 * world_sizeZ);

auto logicWorld = new G4LogicalVolume(solidWorld,  world_mat, "World");

auto physWorld = new G4PVPlacement(0,   G4ThreeVector(0,0,0),  logicWorld,  "World",  0,  false,  0,  true);
```

Ноль, потому что это World

Создаем куб

```
G4double env_sizeXY = 10;
G4double env_sizeZ = 10;
auto solidEnv = new G4Box("Box",   0.5 * env_sizeXY, 0.5 * env_sizeXY, 0.5 * env_sizeZ);

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

auto physEnv = new G4PVPlacement(0,  G4ThreeVector(0,0,0),    logicEnv,  "Box", logicWorld,  false,  0, true);
```

Не ноль, потому что помещаем в World

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
  кто уже скачивал проект, делаем
- cd geant4
- git pull

- *mkdir practice1*
- *cd practice1*
- *cp -r ../geant4/p1 ./*
- *code p1 (открываем редактор в котором вы будете писать код)*

- mkdir build (создаем директорию в которой вы будете компилировать проект)

- *cd build*
- *cmake ../p1 (подготавливаем make файл)*

(когда код написан,делаем)
- *make (коплируется проект,получаем исполняемый файл,который можно потом запустить)*
- *./example*
  (видим окошко с геометрией, радуемся)

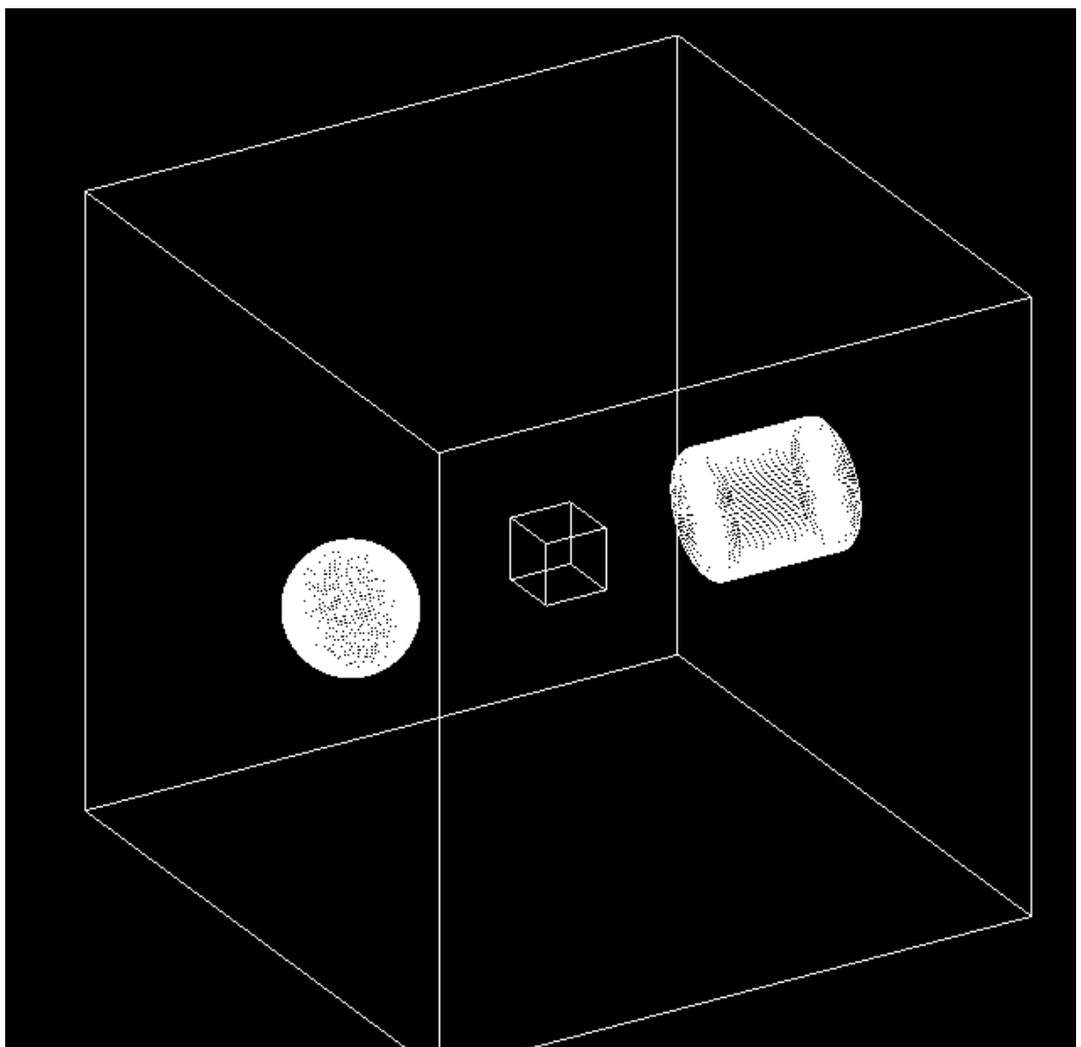
Визуализация

В файле **vis.mac**

/vis/open OGL
/vis/drawVolume

/vis/viewer/set/style wireframe
/vis/viewer/set/auxiliaryEdge true
/vis/viewer/set/lineSegmentsPerCircle 100

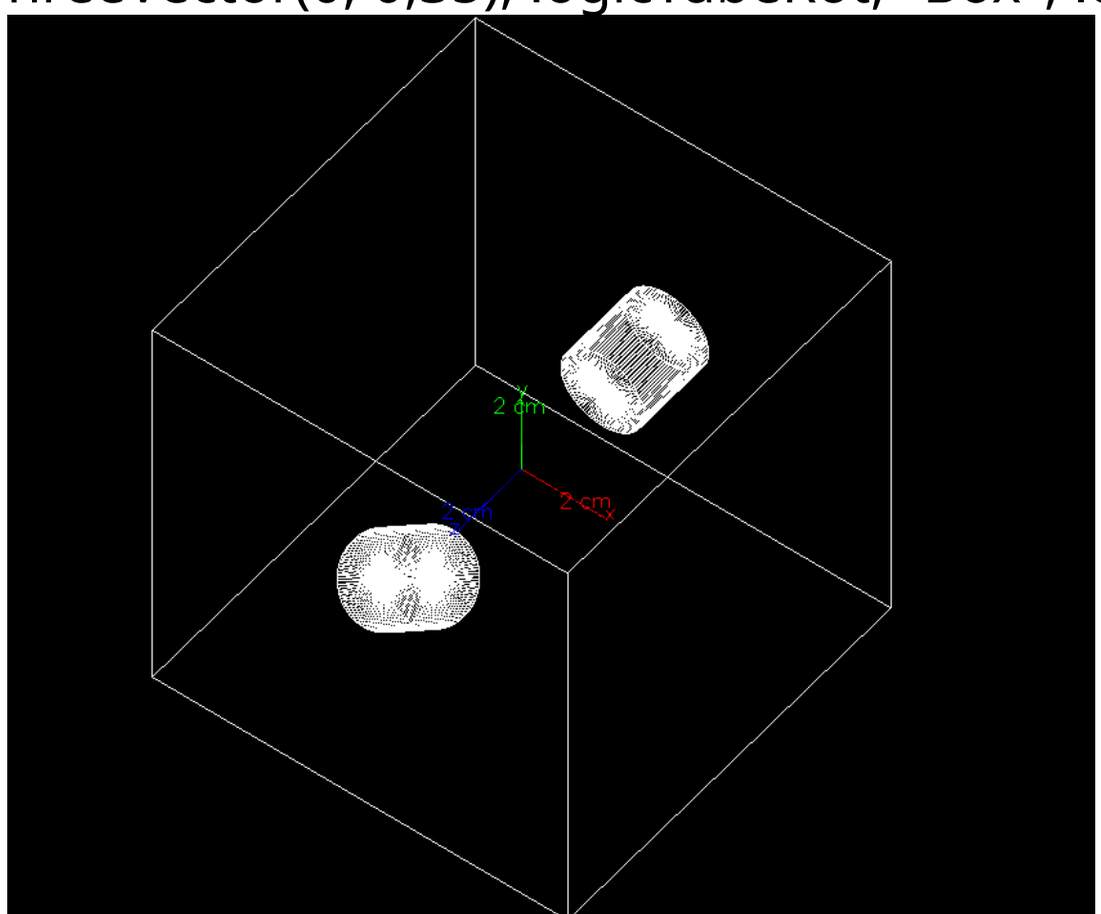
Практика 1



Вращение

```
G4RotationMatrix *rot =  new G4RotationMatrix();
rot->rotateX(40.0*deg);
```

```
new G4PVPlacement(rot, G4ThreeVector(0, 0,35), logicTubeRot, "Box", logicWorld,
false, 0, checkOverlaps);
```



Replica

G4PVReplica

- Множество повторяющихся объемов
- Дочерние элементы одинаковой формы выровнены вдоль одной оси
- Дочерние элементы полностью заполняют материнский элемент без зазоров между ними

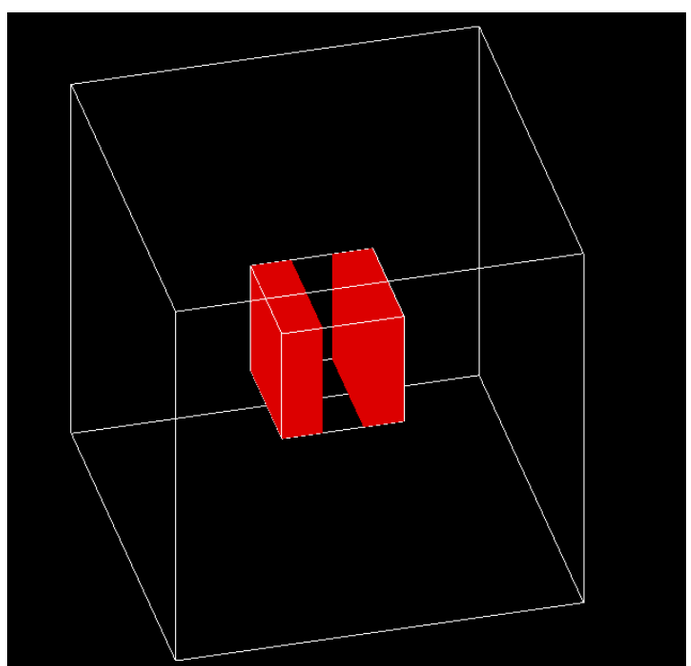
```
new G4PVReplica(
    "Layer",         // Имя физического объема
    logicLayer,      // Логический объем для репликации
    logicWorld,      // Материнский объем (мир)
    kZAxis,          // Ось репликации (Z)
    numberOfLayers, // Количество реплик
    layerThickness, // Толщина одного слоя (шаг)
    shift            // Смещение: центрируем всю структуру
);
```

Replica

```
G4double env_sizeXY = 30;
G4double env_sizeZ = 30;
auto solidEnv = new G4Box("Box",   0.5 * env_sizeXY, 0.5 * env_sizeXY, 0.5 * env_sizeZ);
auto logicEnv = new G4LogicalVolume(solidEnv,  air_mat, "Box");
new G4PVPlacement(0, G4ThreeVector(0,0,0), logicEnv,  "Box", logicWorld,  false,  0, checkOverlaps);
```

```
G4double mini_sizeXY = 30;
G4double mini_sizeZ = 10;
auto solidMini = new G4Box("Box",   0.5 * mini_sizeXY, 0.5 * mini_sizeXY, 0.5 * mini_sizeZ);
auto logicMini = new G4LogicalVolume(solidMini,  env_mat, "Box");
```

```
new G4PVReplica(   "Layer",  logicMini, logicEnv, kZAxis, 2, mini_sizeZ*2, 0 );
```

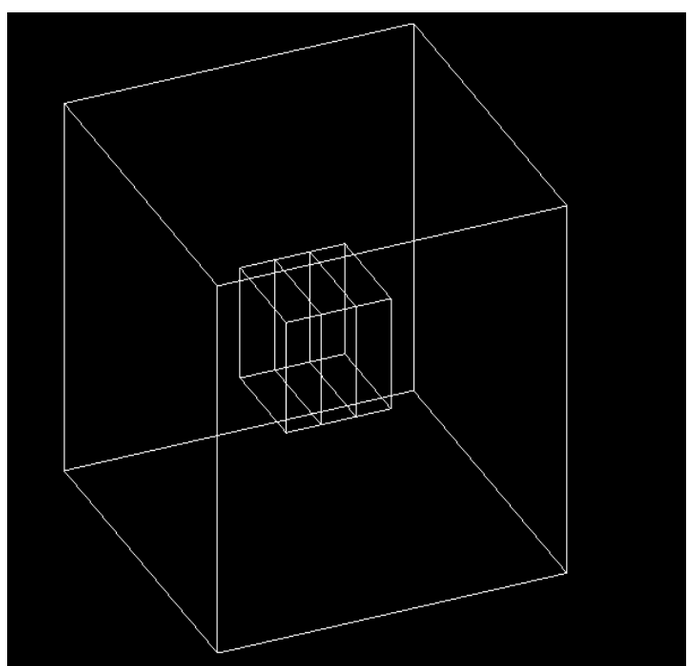


Replica

```
G4double env_sizeXY = 30;
G4double env_sizeZ = 30;
auto solidEnv = new G4Box("Box",   0.5 * env_sizeXY, 0.5 * env_sizeXY, 0.5 * env_sizeZ);
auto logicEnv = new G4LogicalVolume(solidEnv,  air_mat, "Box");
new G4PVPlacement(0, G4ThreeVector(0,0,0), logicEnv,  "Box", logicWorld,  false,  0, checkOverlaps);
```

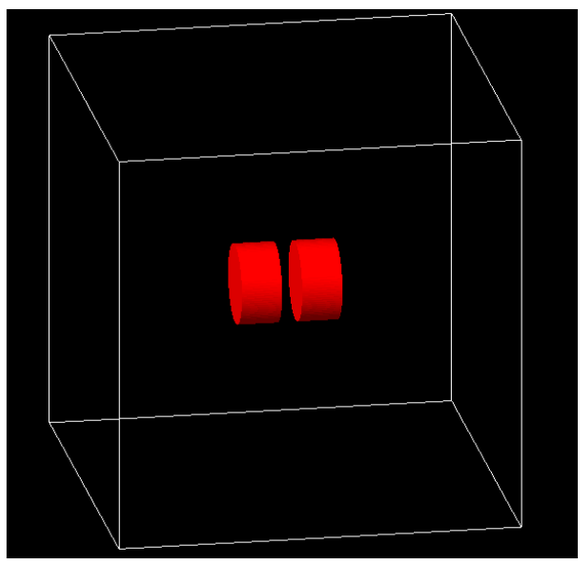
```
G4double mini_sizeXY = 30;
G4double mini_sizeZ = 10;
auto solidMini = new G4Box("Box",   0.5 * mini_sizeXY, 0.5 * mini_sizeXY, 0.5 * mini_sizeZ);
auto logicMini = new G4LogicalVolume(solidMini,  env_mat, "Box");
```

```
new G4PVReplica(   "Layer",  logicMini, logicEnv, kZAxis, 3, mini_sizeZ, 0 );
```



Практика 1 со звездочкой

1) По вращать параллелепипед.
2) Использовать Replica. Повторить.



Цвет для сигмента задается вот так

```
G4VisAttributes* visAtt = new G4VisAttributes(G4Colour(1.0, 0.0, 0.0)); // Красный
visAtt->SetVisibility(true); // Делаем объем видимым (по умолчанию true)
visAtt->SetForceSolid(true); // Или SetForceWireframe(false) для solid отображения
logic>SetVisAttributes(visAtt);
```