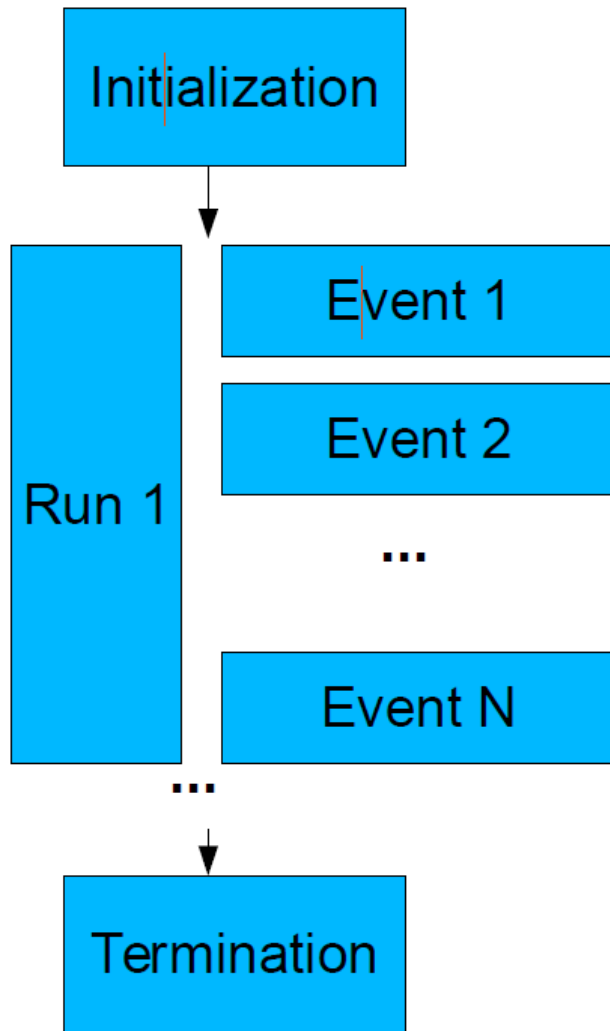


# Цикл моделирования



## Необходимые параметры для запуска

- Распределение вещества в детекторе и поля
- Генератор первичной вершины
- Список физических процессов, учитываемых в моделировании

## Дополнительные настройки

- Чувствительные элементы и способы моделирования отклика
- Способы визуализации
- Графический интерфейс
- Пользовательские расширения

# Самый главный класс Geant4

Под названием Geant4 скрываются сотни классов, решающих ту или иную задачу. Однако связь между собой имеет лишь ограниченное число, базовых, а часто абстрактных базовых классов, вместе образующих ядро Geant4. Для инициализации ядра в Geant4 существует специальный класс G4RunManager. Этот класс является управляющим. Он контролирует последовательность выполнения программы, а также управляет циклом событий во время запуска. Кроме того, пользователь может использовать иную версию данного класса — G4MTRunManager, если работает в многопоточном режиме.

Выделение места в динамической памяти для объекта класса G4RunManager и возврат указателя на созданный объект:

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

abстрактный класс c++ | C++ | Чистые виртуальн... | Абстрактные классы (C+... | Leonov A.A. - Outlook W... | GEANT4 G4RunManag

www.google.ru/search

В Госдуме сообщил... | A. Egon Cholakian ~... | Электричество и м...

Google

GEANT4 G4RunManager

Все | Картинки | Видео | Новости | Книги | Ещё | Инструменты

Результатов: примерно 3 590 (0,25 сек.)

Совет. Оставить только результаты на русском языке. Подробнее о фильтрации по языку...

laboratoire APC  
https://apc.u-paris.fr/html/ · Перевести эту страницу

Geant4: G4RunManager Class Reference

Definition at line 445 of file **G4RunManager.cc**. References  
G4RunManagerKernel::DefineWorldVolume(), and kernel. Referenced by...

laboratoire APC  
https://apc.u-paris.fr/html/ · Перевести эту страницу

Geant4.10: G4RunManager Class Reference

Referenced by **G4RunManager()**, **GetRandomNumberStoreDir()**,  
**RestoreRandomNumberStatus()**, **rndmSaveThisEvent()**, **rndmSaveThisRun()**,...

Кафедра 7 НИЯУ МИФИ  
http://www.kaf07.mephi.ru/ · Geant4 + lecture\_04 PDF

Интерфейс пользователя

**G4RunManager\*** runManager = new **G4RunManager**... // set mandatory initialization ... **Geant4**  
version Name: **geant4-08-00-patch-01** (10-February-2006). Copyright ...

https://www.google.ru/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwiH6aWxz7eEAxUyExAlHekgDC8QFnoFCAsQAQ&url=ht

Main Page | Namespaces | Data Structures | Files |  
Alphabetical List | Data Structures | Class Hierarchy | Data Fields

## G4RunManager Class Reference

#include <G4RunManager.hh>

### Public Member Functions

```
G4RunManager ()  
virtual ~G4RunManager ()  
virtual void BeamOn (G4int n_event, const char *macroFile=0, G4int n_select=-1)  
virtual void Initialize ()  
virtual void DefineWorldVolume (G4VPhysicalVolume *worldVol, G4bool topologyIsChanged=true)  
virtual void AbortRun (G4bool softAbort=false)  
virtual void AbortEvent ()  
virtual void InitializeGeometry ()  
virtual void InitializePhysics ()  
virtual G4bool ConfirmBeamOnCondition ()  
virtual void RunInitialization ()  
virtual void DoEventLoop (G4int n_event, const char *macroFile=0, G4int n_select=-1)  
virtual void RunTermination ()  
virtual void InitializeEventLoop (G4int n_event, const char *macroFile=0, G4int n_select=-1)  
virtual void ProcessOneEvent (G4int i_event)  
virtual void TerminateOneEvent ()  
virtual void TerminateEventLoop ()  
virtual G4Event * GenerateEvent (G4int i_event)  
virtual void AnalyzeEvent (G4Event *anEvent)  
void DumpRegion (const G4String &name) const  
void DumpRegion (G4Region *region=0) const  
virtual void rndmSaveThisRun ()  
virtual void rndmSaveThisEvent ()  
virtual void RestoreRandomNumberStatus (const G4String &fileN)  
void SetUserInitialization (G4VUserDetectorConstruction *userInit)  
void SetUserInitialization (G4VUserPhysicsList *userInit)  
void SetUserAction (G4UserRunAction *userAction)  
void SetUserAction (G4VUserPrimaryGeneratorAction *userAction)  
void SetUserAction (G4UserEventAction *userAction)  
void SetUserAction (G4UserStackingAction *userAction)  
void SetUserAction (G4UserTrackingAction *userAction)
```

одной из функций `runManager` является связь пользовательских классов с ядром Geant4  
для корректной работы любого приложения необходимо передать информацию:

## ГДЕ ПРОИСХОДИТ МОДЕЛИРОВАНИЕ?

## ПО КАКИМ ЗАКОНАМ ПРОИСХОДИТ МОДЕЛИРОВАНИЕ?

## ЧТО ПОЛЬЗОВАТЕЛЬ ХОЧЕТ УЗНАТЬ О РЕЗУЛЬТАТАХ?

На первый вопрос отвечает класс геометрии

На второй — класс, содержащий список всех физических процессов, которые могут произойти  
в рамках данного моделирования.

на третий вопрос отвечает класс пользовательских действий, включающий информацию о том,  
какие первичные частицы рождаются

Класс `G4RunManager` содержит метод `G4RunManger::SetUserInitialization()`, который позволяет инициализировать и связывать с ядром разные части процесса моделирования (*геометрия, описание физических процессов или классы действий*) .

```
runManager->SetUserInitialization(new Geometry);
```

```
runManager->SetUserInitialization(physicsList);
```

```
runManager->SetUserInitialization(new Action());
```

После того как вся необходимая информация передана, вызывается метод `G4RunManager::Initialize()`

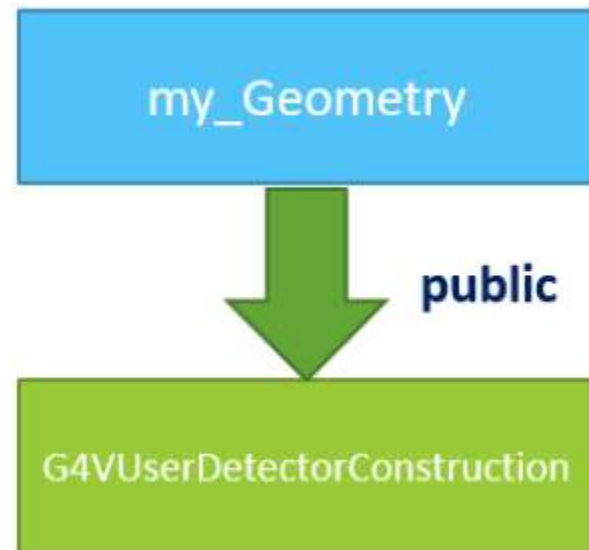
```
runManager->Initialize();
```



```
runManager->SetUserInitialization(new Geometry);
```

G4VuserDetectorConstruction базовый абстрактный класс для любой пользовательской геометрии. Реализация потомка данного класса является обязательным этапом в каждом проекте моделирования.

Класс Geometry является потомком класса G4UserDetectorConstruction



G4UserDetectorConstruction обладает единственным чисто виртуальным методом

**Construct()**

Данный метод вызывается объектом класса **G4RunManager** во время инициализации

то есть, *во время вызова метода **G4RunManager::Initialize()***

Результатов: примерно 2 310 (0,20 сек.)



laboratoire APC

<https://apc.u-paris.fr/html> · Перевести эту страницу

## Geant4: G4VUserDetectorConstruction Class Reference

**G4VUserDetectorConstruction** Class Reference · Public Member Functions · Detailed

Description · Constructor & Destructor Documentation · Member Function ...



laboratoire APC

<https://apc.u-paris.fr/html> · Перевести эту страницу

## Geant4.10: G4VUserDetectorConstruction Class Reference

Definition at line 107 of file **G4VUserDetectorConstruction.cc**. References assert,

G4FieldManager::Clone(), FatalException, G4Exception(), G4LogicalVolumeStore:: ...

Main Page

Namespaces

Data Structures

Files

Alphabetical List

Data Structures

Class Hierarchy

Data Fields

## G4VUserDetectorConstruction Class Reference

```
#include <G4VUserDetectorConstruction.hh>
```

### Public Member Functions

	<a href="#">G4VUserDetectorConstruction ()</a>
virtual	<a href="#">~G4VUserDetectorConstruction ()</a>
virtual <b>G4VPhysicalVolume *</b>	<b><a href="#">Construct ()=0</a></b>
void	<a href="#">RegisterParallelWorld (G4VUserParallelWorld *)</a>
<b>G4int</b>	<a href="#">ConstructParallelGeometries ()</a>
<b>G4int</b>	<a href="#">GetNumberOfParallelWorld () const</a>
<b>G4VUserParallelWorld *</b>	<a href="#">GetParallelWorld (G4int i) const</a>

# Построение модели детектора

Модель детектора строится из простых элементов - объемов

- Описание материалов
- Описание объемов
- *Описание электромагнитных полей*
- *Свойства визуализации*
- *Детектирующие свойства объемов*



# Классы, описывающие материалы

Материал для Geant4 = количество атомов данного вида на единицу объема (длины пробега)

- **G4Isotope**

описывает свойства атома: атомное число, количество нуклонов, молярную массу и т.д.

- **G4Element**

описывает свойства элемента: эффективное атомное число, эффективную молярную массу, число изотопов

- **G4Material**

описывает макроскопические свойства вещества:

плотность, состояние, температуру, давление, радиационную длину, длину свободного пробега и т.д

Для создания объёма детектора нужно определить материал  
Структура материалов в Geant4 отражает то, что реально существует в  
природе: материалы состоят из отдельных элементов или их смесей, элементы в  
свою очередь состоят из отдельных изотопов или их смесей.

## Создание элемента из изотопов

обязательными полями являются: имя, атомный номер и количество нуклонов. G4int iz,n,ncomponents;  
если не указана, то молярная масса рассчитается автоматически G4double a;  
G4String name,symbol;

```
G4Isotope* U5 = new G4Isotope(name="U235", iz=92, n=235, a=235.01*g/mole);  
G4Isotope* U8 = new G4Isotope(name="U238", iz=92, n=238, a=238.03*g/mole);  
G4Element* elU = new G4Element(name="enriched Uranium", symbol="U", ncomponents=2);
```

```
elU->AddIsotope(U5, abundance= 90.*perCent);  
elU->AddIsotope(U8, abundance= 10.*perCent);
```

Для каждого создаваемого элемента данный  
метод должен вызываться ровно столько раз, сколько  
изотопов было указано в конструкторе.

Создание материала:

```
density=19*g/cm2;  
G4Material* matU = new G4Material(name="matU", density, ncomponents=1);  
matU->AddElement(elU, natoms=1);
```

# Создание нового элемента

имя может быть любым, а символ должен соответствовать символу элемента в таблице Менделеева

```
a = 1.01*g/mole;
```

```
G4Element* elH = new G4Element(name="Hydrogen",symbol="H" , z= 1., a);
```

```
a = 14.01*g/mole;
```

```
G4Element* elN = new G4Element(name="Nitrogen",symbol="N" , z= 7., a);
```

При использовании этого конструктора не нужно создавать изотопы напрямую.

Они будут сконструированы автоматически, исходя из природного соотношения, хранимого в базе.

## Описание простых материалов

```
G4double density = 2.700*g/cm3;
```

```
G4double a = 26.98*g/mole;
```

```
G4Material* Al = new G4Material(name="Aluminum", z=13., a, density);
```

```
G4double density = 1.390*g/cm3;
```

```
G4double a = 39.95*g/mole;
```

```
G4Material* lAr = new G4Material(name="liquidArgon", z=18., a, density);
```

# Описание материалов по химической формуле

`a = 1.01*g/mole;`

`G4Element* elH = new G4Element(name="Hydrogen",symbol="H" , z= 1., a);`

`a = 16.00*g/mole;`

`G4Element* elO = new G4Element(name="Oxygen" ,symbol="O" , z= 8., a);`

`G4double density = 1.000*g/cm3;`

`G4Material* H2O = new G4Material(name="Water", density, ncomponents=2);`

`H2O->AddElement(elH, natoms=2);`

`H2O->AddElement(elO, natoms=1);`



## Описание материала через массовые доли компонент

```
a = 14.01*g/mole;
```

```
G4Element* elN = new G4Element(name="Nitrogen",symbol="N" , z= 7., a);
```

```
a = 16.00*g/mole;
```

```
G4Element* elO = new G4Element(name="Oxygen" ,symbol="O" , z= 8., a);
```

```
G4double density = 1.290*mg/cm3;
```

```
G4Material* Air = new G4Material(name="Air " , density, ncomponents=2);
```

```
Air->AddElement(elN, fractionmass=0.7);
```

```
Air->AddElement(elO, fractionmass=0.3);
```



Конструктор, позволяющий создавать материалы из созданных ранее элементов или материалов

```
G4Material(const G4String& name,           //имя
           G4double density,               //плотность
           G4int nComponents,              //количество компонентов
           G4State state = kStateUndefined, //состояние
           G4double temp = NTP_Temperature, //температура
           G4double pressure = CLHEP::STP_Pressure); //давление
```

Поле *state* отвечает за "состояние" материала. Оно может быть: твердым, жидким, газообразным или "неопределенным".

0	<i>kStateUndefined</i>
1	<i>kStateSolid</i>
2	<i>kStateLiquid</i>
3	<i>kStateGas</i>

По умолчанию значения температуры(*temp*) и давления(*pressure*) соответствуют нормальным условиям.

# Описание газов

## Из элементов

```
G4double density    = 27.*mg/cm3;
```

```
G4double pressure   = 50.*atmosphere;
```

```
G4double temperature = 325.*kelvin;
```

```
G4Material* CO2 = new G4Material(name="Carbonic gas", density,  
                                   ncomponents=2, kStateGas, temperature, pressure);
```

```
CO2->AddElement(elC, natoms=1);
```

```
CO2->AddElement(elO, natoms=2);
```

## Из материалов

```
density    = 0.3*mg/cm3;
```

```
pressure   = 2.*atmosphere;
```

```
temperature = 500.*kelvin;
```

```
G4Material* steam = new G4Material(name="Water steam ",  
                                   density, ncomponents=1, kStateGas, temperature, pressure);
```

```
steam->AddMaterial(H2O, fractionmass=1.);
```

при создании материала из других материалов не учитываются промежуточные свойства его компонентов. материал состоит из элементов, а его плотность, температура и т.п. определяются только финальным объектом.

# Задание материала с новыми свойствами из уже существующих материалов

```
G4Material(const G4String& name,           //имя
           G4double density,               //плотность
           const G4Material* baseMaterial,  //базовый материал
           G4State state = kStateUndefined, //состояние
           G4double temp  = NTP_Temperature, //температура
           G4double pressure = CLHEP::STP_Pressure); //давление
```

```
G4double density = 1.000*g/cm3;
```

```
G4Material* H2O = new G4Material(name="Water", density, ncomponents=2);
```

```
H2O->AddElement(elH, natoms=2);
```

```
H2O->AddElement(elO, natoms=1);
```

```
G4Material* new_Water = new G4Material("new_Water", 5*g/cm3, H2O, kStateLiquid, 1000*kelvin);
```

# Вакуум

Описывается как разреженный газ:

```
density = universe_mean_density; //from PhysicalConstants.h  
pressure = 1.e-19*pascal;  
temperature = 0.1*kelvin;  
new G4Material(name="Galactic", z=1., a=1.01*g/mole, density,  
               kStateGas, temperature, pressure);
```

# База материалов National Institute of Standards and Technology (NIST)

В Geant4 представлена база материалов, составленная NIST Physical Measurement Laboratory. В данной базе доступно более 3000 изотопов, а все представленные элементы составлены исходя из природного баланса изотопов. Элементы доступны от Водорода до Калифорния (98). Кроме того, в базе доступны различные готовые материалы, такие как:

- составные вещества и смеси (ткань, эквивалентная пластику; морской воздух и т.д.)
- биохимические материалы (жировая ткань, цитозин, тимин и т.д.)
- композитные материалы (например, кевлар)

## Как посмотреть таблицу изотопов, элементов и материалов

```
G4cout << *(G4Isotope::GetIsotopeTable()) << G4endl;
```

```
G4cout << *(G4Element::GetElementTable()) << G4endl;
```

```
G4cout << *(G4Material::GetMaterialTable()) << G4endl;
```

Перенаправление стандартного вывода и стандартного вывода ошибок в файл: `./TrackStack > file.dat 2>&1`



# Библиотека материалов Geant4

```
#include "G4NistManager.hh"
```

```
G4NistManager* man = G4NistManager::Instance();
```

```
// define elements
```

```
• G4Element* elAl = man->FindOrBuildElement("Al");
```

```
// define pure NIST materials
```

```
G4Material* Al = man->FindOrBuildMaterial("G4_Al");
```

```
G4Material* Cu = man->FindOrBuildMaterial("G4_Cu");
```

```
// define NIST materials
```

```
G4Material* H2O = man->FindOrBuildMaterial("G4_WATER");
```

```
G4Material* Sci = man-> FindOrBuildMaterial("G4_PLASTIC_SC_VINYLTOLUENE");
```

```
G4Material* Vacuum = man-> FindOrBuildMaterial("G4_Galactic");
```

Полный список доступных материалов в Geant4 можно найти в разделе:

[Geant4 Book For Application Developers > Appendix > Geant4 Material Database](#)

# Geant4

типы переменных

ВВОД-ВЫВОД

библиотека CLHEP

система единиц

# *Основные типы переменных Geant4*

Для достижения переносимости кода в Geant4 переопределены основные типы переменных

*G4int, G4long, G4float, G4double,  
G4bool, G4complex, G4String*

## *Ввод и вывод в коде Geant4*

Можно использовать обычные printf() и cout

Однако более правильно использовать переопределенные потоки Geant4:

```
G4cout << "test" << G4endl;
```

```
G4cerr << "error" << G4endl;
```

*Библиотека CLHEP* <http://cern.ch/clhep>  
*Class Library for High Energy Physics*

используется в коде Geant4

Содержит описание стандартных математических объектов, часто используемых в ФВЭ

- 3-векторы и 4-векторы
- действия с матрицами
- геометрические объекты и преобразования
- генераторы случайных чисел
- система единиц и основные физические константы

**G4ThreeVector**

– трехкомпонентный (x,y,z) вектор и действия с ним

• **G4LorentzVector**

– четырехкомпонентный (x,y,z,t) вектор

• **G4RotationMatrix**

– матрица 3x3, определяющая вращение 3-вектора

• **G4LorentzRotation**

– матрица 4x4, определяющая вращение 4-вектора

• Геометрические объекты и преобразования

– **G4Plane3D, G4Transform3D, G4Normal3D, G4Point3D, G4Vector3D**

# *Система единиц Geant4*

Для некоторых единиц измерения в Geant4 установлено значение «1».

- millimeter (mm) - миллиметр
- nanosecond (ns) - наносекунда
- MeV - Мега электронвольт
- eplus - заряд позитрона
- kelvin - кельвин
- mole - моль
- candela -сила света
- radian - радиан
- steradian - стерадиан

Все остальные единицы задаются через указанные в этом списке величины  
единицы измерения описаны в файле G4SystemOfUnits.hh.



Любое число, имеющее размерность, должно быть умножено на соответствующую единицу для перевода во внутреннюю систему единиц Geant4

*$length = 10.0 * cm;$*   
 *$kinetic\_energy = 5.0 * GeV;$*

```
double length = 5. * cm;  
G4cout << length << '\n';
```

 Выводимым в консоль результатом будет 50.

Для вывода величины в желаемых единицах следует делить число на единицу измерения

*$G4cout << eDep / MeV << "[MeV]" << G4endl;$*

```
double mass = 5. * g;  
G4cout << mass / g << '\n';
```

 Выводимый результат будет 5

если существует необходимость в добавлении единиц измерения, то можно добавить в файл с описанием геометрии строки

```
#include <G4SystemOfUnits.hh>
```

```
static const G4double inch = 2.54*cm;
```

чтобы увидеть весь список доступных единиц измерения, можно воспользоваться командой:

*/units/list* (например, воспользоваться списком команд в меню визуализатора)

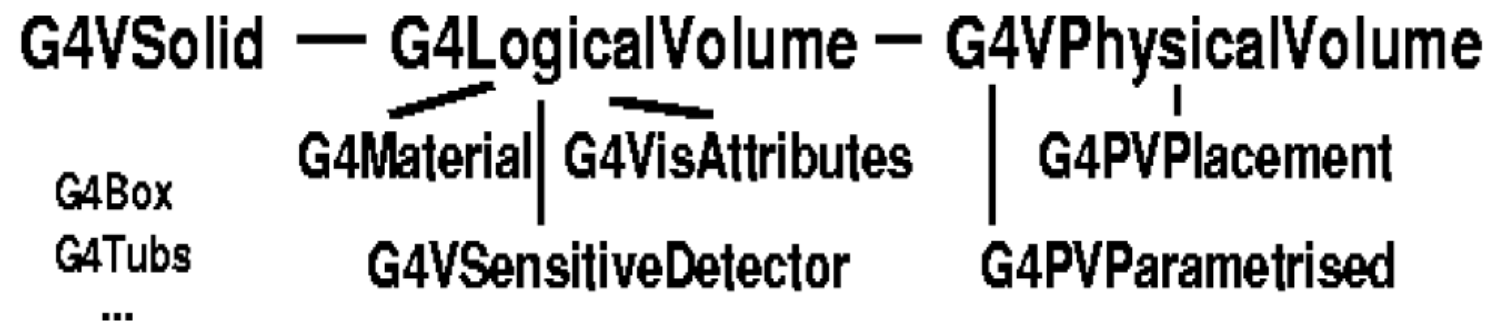
## Геометрия.

В основе геометрии Geant4 лежит концепция логических и физических объемов. Логический объем представляет собой элемент детектора определенной формы заданного материала, чувствительности и т. п. Кроме того, каждый логический объем может содержать внутри другие объемы. Физический объем представляет собой пространственное расположение логического объема относительно логического материнского объема

### Объем описывается в три этапа

- форма (*G4VSolid*)
- логический объем (*G4LogicalVolume*)
- физический объем (*G4VPhysicalVolume*)

- На первом этапе создается основа будущего логического объема - его форма, включающая геометрические характеристики.
- На втором - описываются свойства формы, такие как: материал, сцинтилляционные свойства и т. п. Кроме того, на втором этапе форме можно установить различные визуальные атрибуты, упрощающие дальнейшую визуализацию моделирования.
- На третьем этапе осуществляется расположение модифицированной формы в пространстве, её поворот, смещение и т. п.



в Geant4 представлен целый набор примитивных форм, являющихся потомками абстрактного класса **G4SCGSolid** : коробки, сферы, конусы, трубки и др.

Простейшая форма: прямоугольный параллелепипед (коробка):

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

в качестве значений по X, Y, Z принимаются ПОЛОВИНЫ от реальной длины, ширины и высоты.

Данное свойство характерно для большинства форм, используемых в G4.

реализация объекта простейшей формы будет выглядеть следующим образом:

```
G4Box *box = new G4Box("box", 5*cm, 5*cm, 5*cm);
```

Все геометрические формы в Geant4 реализованы в соответствии с концепцией

**Конструктивной блочной геометрии** Constructive Solid Geometry (CSG). Большинство комплексных форм описывается за счет их граничной поверхности, которая может быть первого, второго порядка, или B-spline поверхностью. Все это сделано, опираясь на ISO STEP стандарт CAD систем.

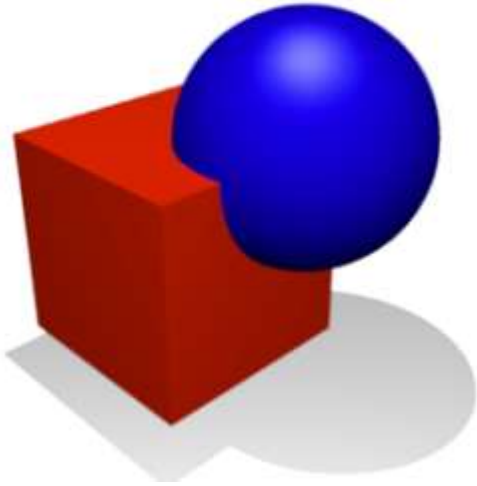
Простейшие тела, используемые в конструктивной блочной геометрии — **примитивы** (англ. *primitives*), тела с простой формой:

куб, цилиндр, призма, пирамида, сфера, конус.

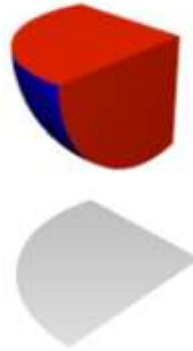
Построение более сложного объекта происходит путём применения к описаниям объектов **булевых** (двоичных) **операций** на **множествах** — **объединение, пересечение и разность**.

Примитивы могут быть скомпонованы в составные объекты с помощью таких операций:

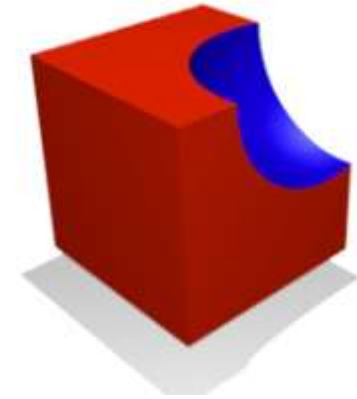
булево объединение



булево пересечение



булева разность



В Geant4 для всех базовых форм предусмотрены методы для расчета занимаемого геометрического объема

```
G4cout << "Volume = " << box->GetCubicVolume() << '\n';
```

а также площади поверхности:

```
G4cout << "Surface Area = " << box->GetSurfaceArea() << '\n';
```



# Формы

Geant4 Book For Application Developers > Geometry

## Простые формы

*G4Box, G4Tubs, G4Cons, G4Trd, ...*

## Специальные формы

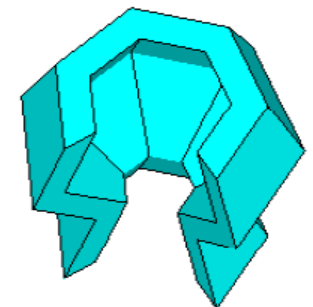
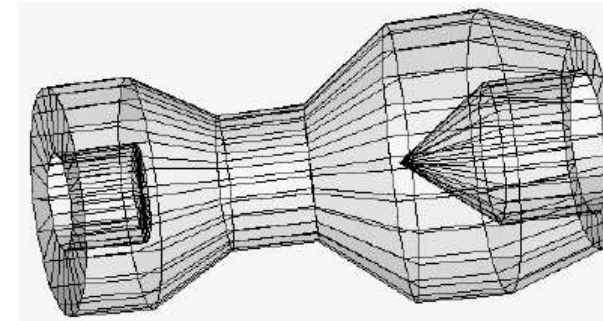
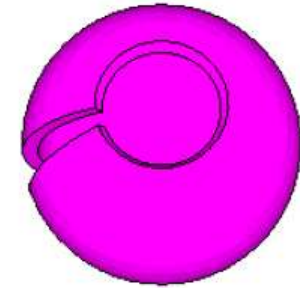
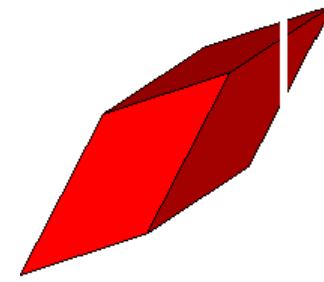
*G4polycone, G4Polyhedra, G4Hype, ...*

## Определяемые поверхностью

*G4BREPSSolidPolycone, G4BSplineSurface, ...*

## Булевы формы

*G4UnionSolid, G4SubtractionSolid, ...*

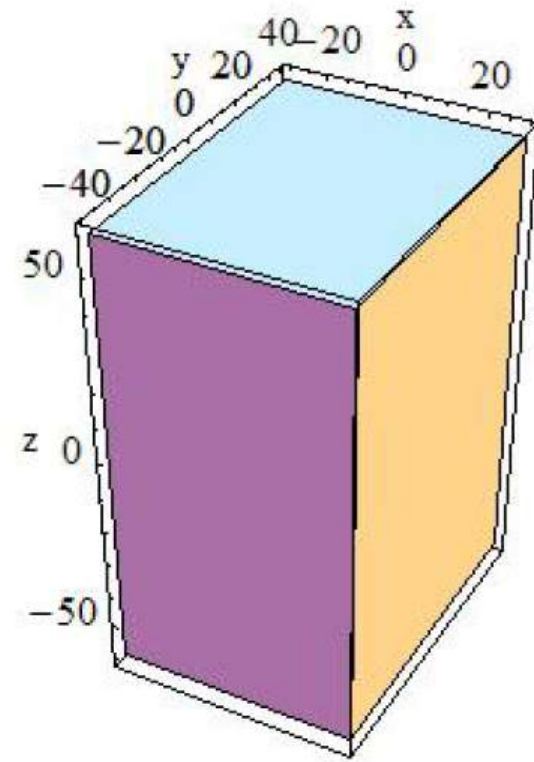


## Параллелепипед

```
G4VSolid* scint_solid = new  
    G4Box(const G4String& pName,  
          G4double pX,  
          G4double pY,  
          G4double pZ);
```

Пример:

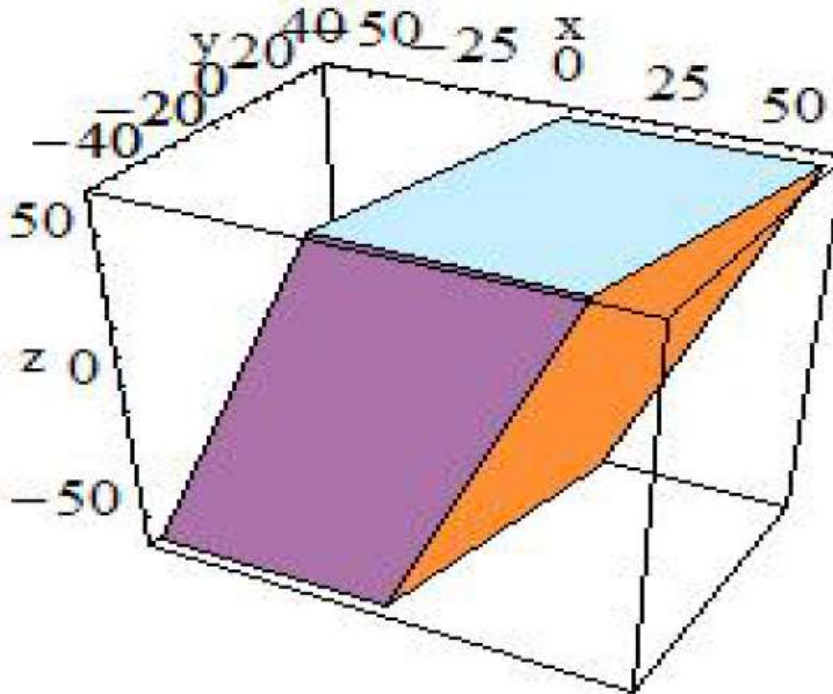
```
G4Box* aBox = new  
G4Box("BoxA", 20.0*cm, 40.0*cm, 60.0*cm);
```



# Параллелепипед в общем случае

```
G4VSolid* aSolid = new  
  G4Para(const G4String& pName,  
    G4double dx,  
    G4double dy,  
    G4double dz,  
    G4double alpha,  
    G4double theta,  
    G4double phi)
```

alpha – угол между осью Y и плоскостью, соединяющей центры боковых поверхностей в основании, параллельном плоскости XZ, и проходящей через  $-dy$  и  $dy$

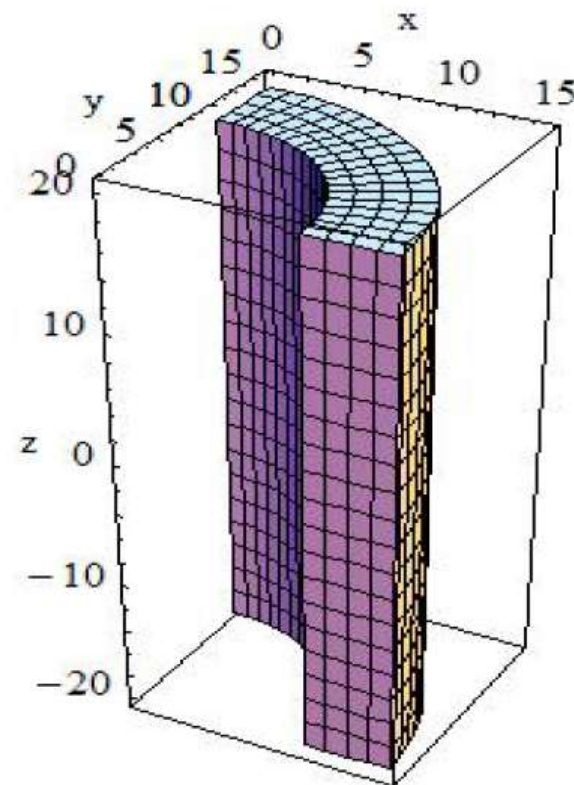


theta – полярный угол для линии, соединяющей центры боковых поверхностей в основании и проходящей через плоскости  $-dz$  и  $dz$

phi – азимутальный угол для линии, соединяющей центры боковых поверхностей в основании и проходящей через плоскости  $-dz$  и  $dz$

## Цилиндр

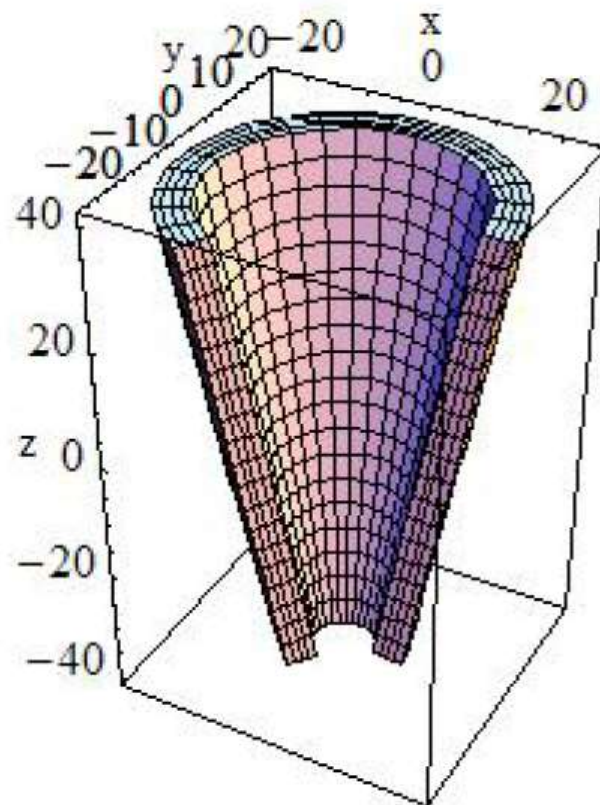
```
G4VSolid* calor_solid = new  
  G4Tubs(const G4String& pName,  
          G4double pRMin,  
          G4double pRMax,  
          G4double pDz, - полувысота  
          G4double pSPhi,  
          G4double pDPhi)
```





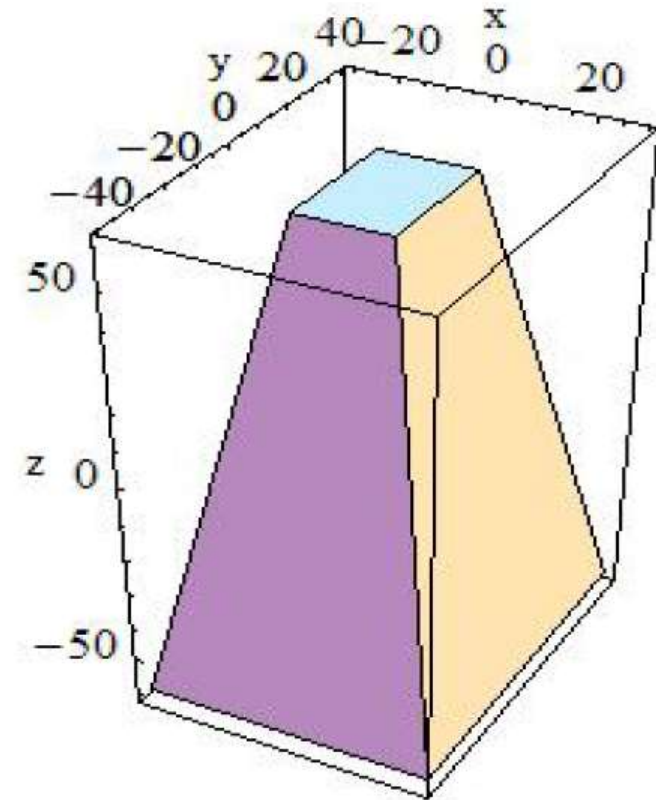
## Конус

```
G4VSolid* scint_solid = new  
G4Cons(const G4String& pName,  
        G4double pRmin1,  
        G4double pRmax1,  
        G4double pRmin2,  
        G4double pRmax2,  
        G4double pDz,  
        G4double pSPhi,  
        G4double pDPhi)
```



## Трапезоид

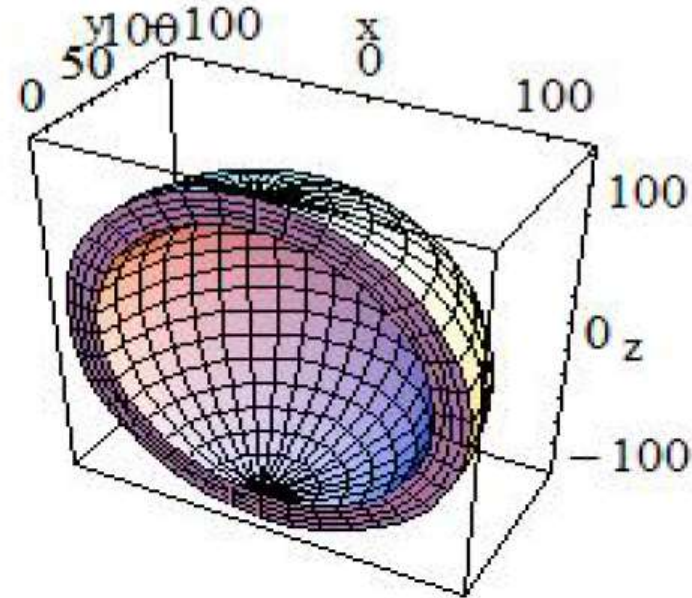
```
G4VSolid* aSolid = new  
G4Trd(const G4String& pName,  
        G4double dx1,  
        G4double dx2,  
        G4double dy1,  
        G4double dy2,  
        G4double dz)
```





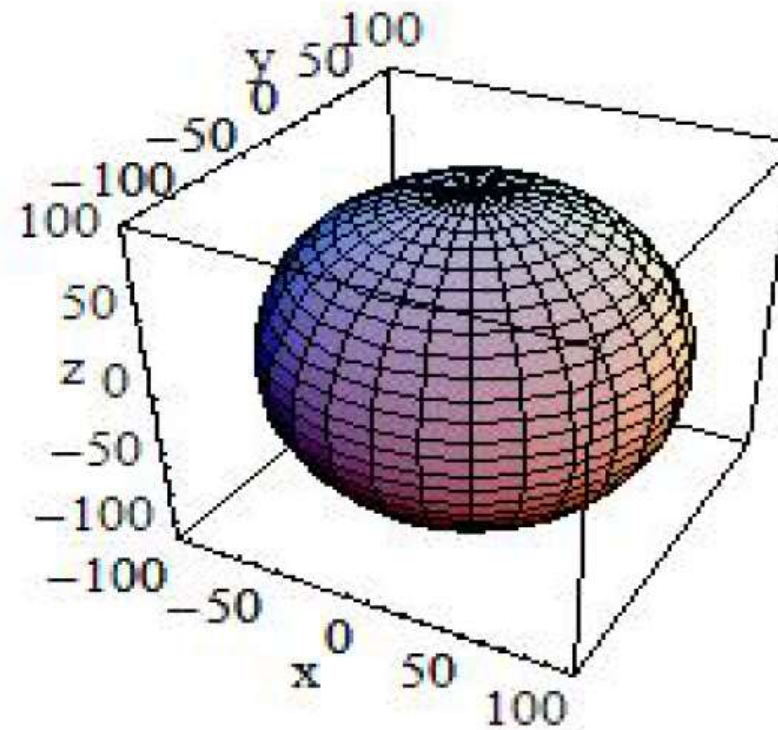
## Сфера

```
G4VSolid* aSolid = new  
  G4Sphere(const G4String& pName,  
            G4double pRmin,  
            G4double pRmax,  
            G4double pSPhi,  
            G4double pDPhi,  
            G4double pSTheta,  
            G4double pDTheta )
```



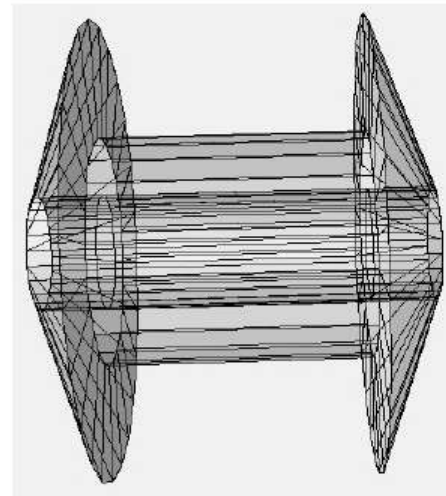
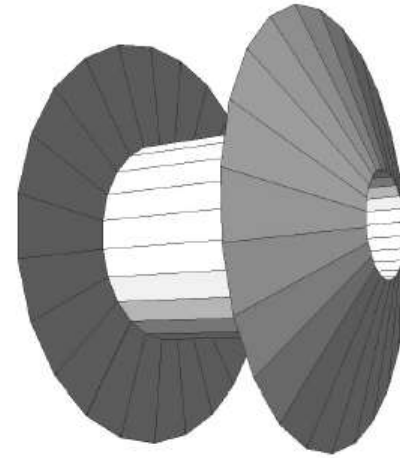
# Шар

```
G4VSolid* aSolid = new  
  G4Orb(const G4String& pName,  
        G4double pRmax)
```



## Формы, определяемые поверхностью

- Для задания такой формы необходимо описать все ограничивающие ее поверхности
- Поверхности могут быть
  - элементарные (плоскости, поверхности 2 порядка и т.д.)
  - сплайны, B-сплайны, NURBS (описываются в Geant4 используя интерфейс к системам САПР)



## Булевы формы

- Объединение двух форм при помощи логической операции
- *G4UnionSolid*, *G4SubtractionSolid*, *G4IntersectionSolid*
- При описании положение второй формы описывается в координатной системе первой
- Не следует злоупотреблять булевыми формами, т.к. усложняется трекинг и увеличивается время моделирования события. По возможности лучше использовать обычные вложенные объемы

```
G4Box* box = new G4Box("Box", 20*mm, 30*mm, 40*mm);  
G4Tubes* cyl = new G4Tubes("Cylinder", 0, 50*mm, 50*mm,0,twopi);
```

```
G4UnionSolid* union = new G4UnionSolid("Box+Cylinder", box, cyl);  
G4IntersectionSolid* intersection = new G4IntersectionSolid("Box*Cylinder", box, cyl);  
G4SubtractionSolid* subtraction = new G4SubtractionSolid("Box-Cylinder", box, cyl);
```





# Логический объем

Сама по себе форма не содержит никакой информации об объекте, кроме его геометрических размеров. Однако для моделирования взаимодействия излучения с веществом необходимо описать свойства этого самого вещества, в котором осуществляется моделирование. С этой целью в Geant4 предоставлен класс объектов, называемый логический объем. Одной из основных целей логического объема является связь формы с физическими свойствами объекта, как пример, материал, из которого она(форма) сделана.

- Кроме геометрических параметров, содержит описание материала, заполняющего объем, свойства визуализации объема и описание детектирующей способности

Объекты логического объема относятся к классу G4LogicalVolume.

Конструктор данного класса выглядит следующим образом:

```
G4LogicalVolume* aLogical =  
    new G4LogicalVolume( G4VSolid* pSolid,  
                          G4Material*      pMaterial,  
                          const G4String&   Name,  
                          G4FieldManager*    pFieldMgr=0,  
                          G4VSensitiveDetector* pSDetector=0,  
                          G4UserLimits*      pULimits=0,  
                          G4bool             optimise=true );
```

поля «форма и материал» (pSolid and pMaterial) не должны быть равными **nullptr**.

Поля: pFieldMgr, pSDetector, pULimits являются опциональными

Рассматривая формы, следует обратить внимание на визуализацию геометрических объектов, например цвет или прозрачность, так как визуальные атрибуты являются частью именно логической формы



Для того, чтобы изменить цвет того или иного геометрического объекта, следует вызвать метод:

```
void SetVisAttributes (const G4VisAttributes& VA);
```

который в качестве аргумента принимает интересующий нас параметр визуализации.

```
G4double box_size = 5 * cm;
```

```
G4Box *box = new G4Box("box", box_size, box_size, box_size);
```

```
G4LogicalVolume* box_log = new G4LogicalVolume(box, nist->FindOrBuildMaterial("G4_SODIUM_IODIDE"), "box_LOG");
```

```
box_log->SetVisAttributes(G4Colour::Blue());
```

Объект box\_log представляет собой логический объем, в качестве аргументов - принимающий форму и моделируемый материал. Кроме того, осуществляется передача логическому объему визуального атрибута «синий цвет». Стоит отметить, что теперь все физические объёмы которые будут использовать в качестве параметра данный логический объем, окрасятся синим цветом.

Отдельно стоит отметить имена форм и логических объемов. В рамках объектов своего класса эти имена должны быть уникальны. Для формы, логического объема и физического объема одного геометрического объекта можно использовать одно имя.

Расчёт массы:

```
G4cout << "Mass = " << box_log->GetMass() / g << " g\n";
```

# Физический объем

Последним этапом построения геометрического объекта средствами Geant4 является реализация его физического объема или, иначе говоря, расположение в пространстве.

- Строится на основе логического объема
- Описывает положение объема в пространстве
- Позволяет одновременно описать серию одинаковых объемов или параметризовать свойства объема в зависимости от номера копии

## Вложенность объемов

- Все объемы должны быть вложены один в другой  
Перекрывание объемов не допускается!
- В любой модели существует только один “самый верхний объем” (экспериментальный зал), в который “вкладываются” все остальные
- Дочерний объем позиционируется в локальной системе координат, связанной с родительским объемом. Положение любого объекта (объема, частицы и т.д.) одновременно вычисляется как в глобальной координатной системе, связанной с экспериментальным залом, так и в локальной, связанной с объемом, в котором объект в данный момент находится

# Размещение объёма

**G4PVPlacement** = один объем

Единственная копия данного объема размещается в материнском объеме

```
G4PVPlacement(G4RotationMatrix *pRot,           //матрица поворота
               const G4ThreeVector &tlate,        //вектор смещения
               G4LogicalVolume *pCurrentLogical,  //логический объем
               const G4String& pName,             //имя
               G4LogicalVolume *pMotherLogical,   //материнский объем
               G4bool pMany,                      //не используется
               G4int pCopyNo,                    //номер копии
               G4bool pSurfChk=false);            //проверка на
                                                    //пересечение с другими объемами
```

где pRot — матрица поворота, tlate — вектор смещения, pCurrentLogical — логический объем, pName — имя (идентификатор), pMotherLogical — материнский объем, pMany — данный параметр не реализован в текущей версии Geant4 (можно использовать false), pCopyNo - номер копии логического объема, для первого физического объема следует использовать значение 0, затем 1 и т.д.



должен существовать один единственный материнский логический объем, который становится изначальным или главным для всех последующих объемов.

```
world_pvpl = new G4PVPlacement(0, G4ThreeVector(0,0,0), world_log, "world_pvpl", 0, false, 0);
```

такой объем называют миром.

В качестве указателя на материнский объем для этого физического объема следует использовать нулевой указатель.

единственный необходимый для реализации геометрии метод: `virtual G4VPhysicalVolume* Construct() = 0;`

возвращает указатель на физический объем. Объем, на который возвращается указатель, является главным материнским объемом или представленным выше миром, с привязанными к нему всеми дочерними объемами, используемыми в геометрии.



## Как задать матрицу поворота?

```
G4RotationMatrix* rotm = new G4RotationMatrix();  
rotm->rotateX(90.*deg);  
rotm->rotateZ(45.*deg);
```

## Как задать трансляцию?

```
G4double pos_x = -1.0*cm;  
G4double pos_y = 0.0*cm;  
G4double pos_z = 0.0*cm;  
G4ThreeVector(pos_x, pos_y, pos_z);
```

Матрица поворота: ZERO\_RM, трансляция (смещение): DE1\_vect

```
DE1_pvpl = new G4PVPlacement(ZERO_RM, DE1_vect, DE1_log, "DetEl_pvpl", world_log, 0, false, 0);
```

## Контейнеры-хранилища

Все создаваемые в процессе моделирования формы, логические объемы и физические объемы помещаются в специальные контейнеры-хранилища для удобного доступа к элементам геометрии из других частей проекта.

- Для форм — G4SolidStore
- Для логических объемов — G4LogicalStore
- Для физических объемов — G4PhysicalVolumeStore.

Для получения доступа к контейнеру необходимо вызвать статический метод `GetInstance()`, возвращающий указатель на выбранное хранилище. хранилище является контейнером, с ним можно работать с помощью итераторов.

```
for (auto item : *G4SolidStore::GetInstance())  
    G4cout << item->GetName() << '\n';
```

Аналогичную информацию можно получить для хранилищ логических и физических объемов.

```
for (auto item : *G4PhysicalVolumeStore::GetInstance())  
    G4cout << item->GetName() << '\n';
```

Возможность динамического изменения геометрии

```
G4GeometryManager::GetInstance()->OpenGeometry();
```

Очистка хранилища физических объёмов

```
G4PhysicalVolumeStore::GetInstance()->Clean();
```

Очистка хранилища логических объёмов

```
G4LogicalVolumeStore::GetInstance()->Clean();
```

Очистка хранилища форм объёмов

```
G4SolidStore::GetInstance()->Clean();
```