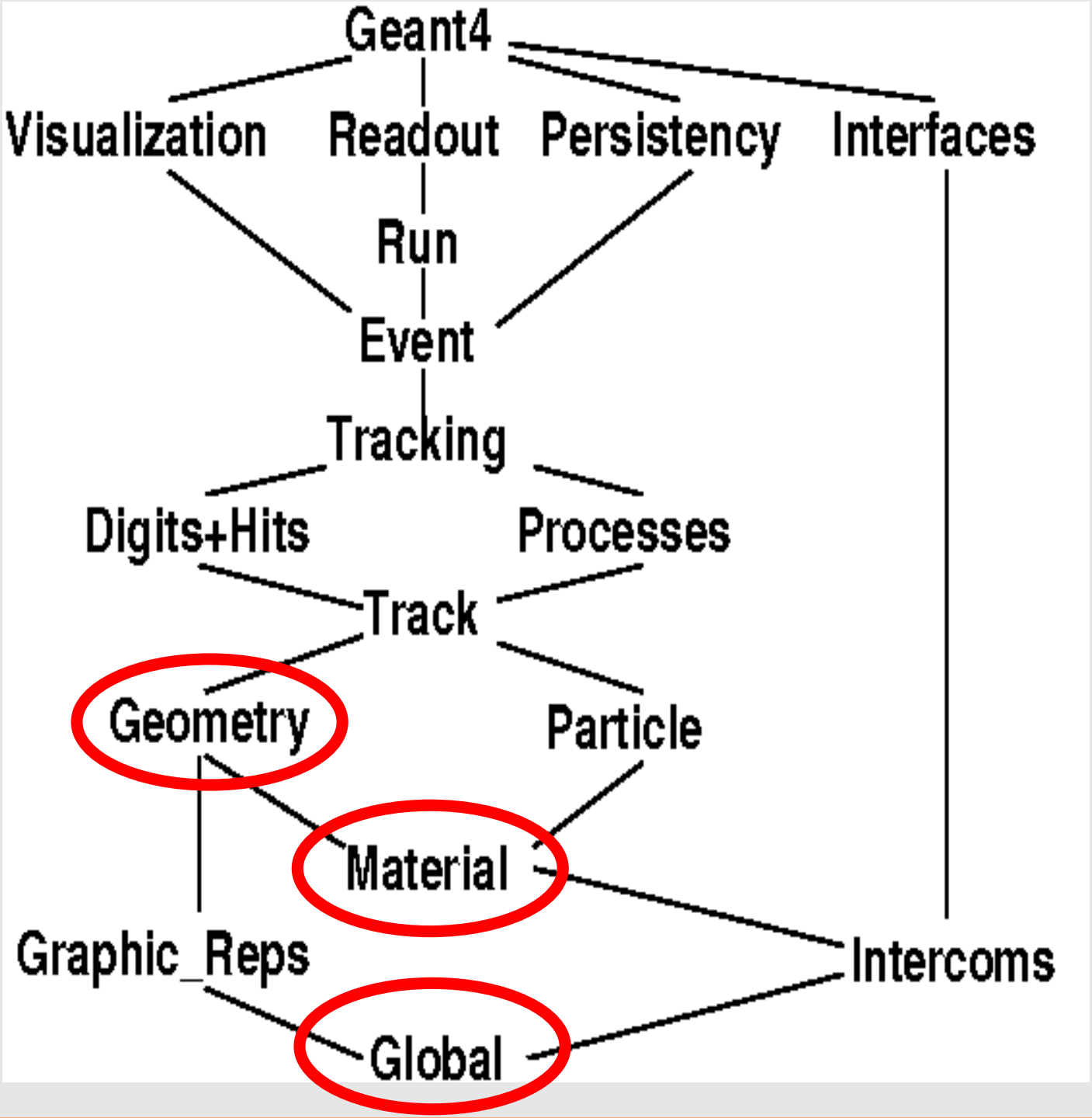
Построение модели детектора


Geant4
Visualization
Readout
Persistency
Interfaces
Run
Event
Tracking
Digits+Hits
Processes
Track
Geometry
Particle
Material
Graphic_Reps
Intercoms
Global

Основные типы переменных

Для достижения переносимости кода в Geant4 переопределены основные типы переменных

G4int, G4long, G4float, G4double,

G4bool, G4complex, G4String

Лучше использовать переопределенные типы

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*;

При этом будет корректно обрабатываться ввод-вывод в случае, если интерфейс пользователя отличен от командной строки

Библиотека CLHEP
Class Library for High Energy Physics

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

Широко используется в коде Geant4

Подробное описание

http://cern.ch/clhep

G4ThreeVector

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

G4LorenzVector

четырехкомпонентный (x,y,z,t) вектор

G4RotationMatrix

матрица 3x3, определяющая вращение 3-вектора

G4LorenzRotation

матрица 4x4, определяющая вращение 4-вектора

Геометрические объекты и преобразования

G4Plane3D, G4Transform3D, G4Normal3D, G4Point3D, G4Vector3D

Система единиц Geant4

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;

Для получения величины в желаемых единицах следует делить число на единицу

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

Все общеупотребительные единицы описаны в Geant4 (CLHEP). При необходимости можно определить свои единицы

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

- **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

Создание нового элемента

```
a = 1.01*g/mole;
G4Element* elH
        = new G4Element(name="Hydrogen",symbol="H" , z= 1., a);
a = 12.01*g/mole;
G4Element* elC
        = new G4Element(name="Carbon"  ,symbol="C" , z= 6., a);
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);
```

Создание элемента из изотопов

```
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);
```

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

```
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);

G4double density = 1.032*g/cm3;
G4Material* Sci = new G4Material(name="Scintillator", density,
   ncomponents=2);
Sci->AddElement(elC, natoms=9);
Sci->AddElement(elH, natoms=10);
```

Описание материала через массовые доли компонент

```
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);
```

Описание смеси нескольких материалов

```
G4double density = 0.200*g/cm3;
G4Material* Aerog
      = new G4Material(name="Aerogel", density, ncomponents=3);
Aerog->AddMaterial(SiO2,fractionmass=62.5*perCent);
Aerog->AddMaterial(H2O,fractionmass=37.4*perCent);
Aerog->AddElement (elC , fractionmass= 0.1*perCent);
```

Описание газов

```
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);

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.);
```

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);

density     = 1.e-5*g/cm3;
pressure    = 2.e-2*bar;
temperature = STP_Temperature;   //from PhysicalConstants.h
G4Material* beam = new G4Material(name="Beam ", density,
   ncomponents=1,  kStateGas, temperature, pressure);
beam->AddMaterial(Air, fractionmass=1.);
```

Как посмотреть таблицу изотопов, элементов и материалов

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

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

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

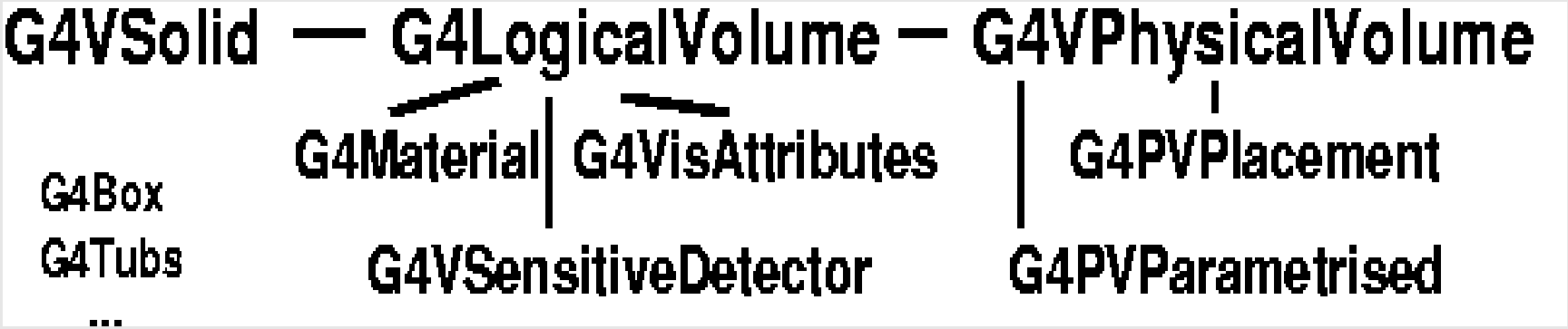
Библиотека материалов Geant4

```
#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");
```

Описание объема

Объем описывается в три этапа

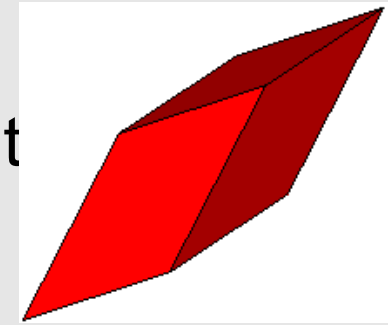
- форма (G4VSolid)
- логический объем (G4LogicalVolume)
- физический объем (G4VPhysicalVolume)



Формы

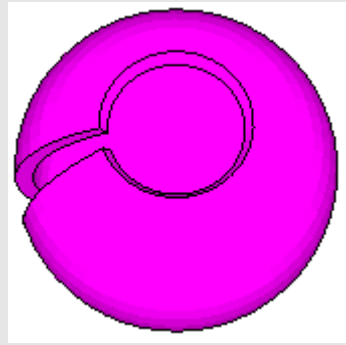
Простые формы (CGS – Constructed Solid Geomet

G4Box, G4Tubs, G4Cons, G4Trd, ...



Специальные формы

G4polycone, G4Polyhedra, G4Hype, ...

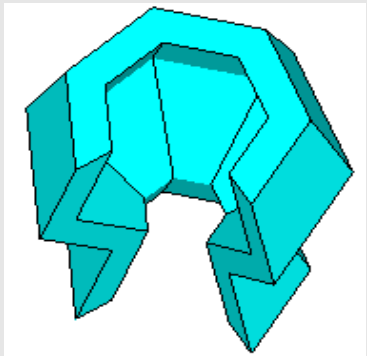


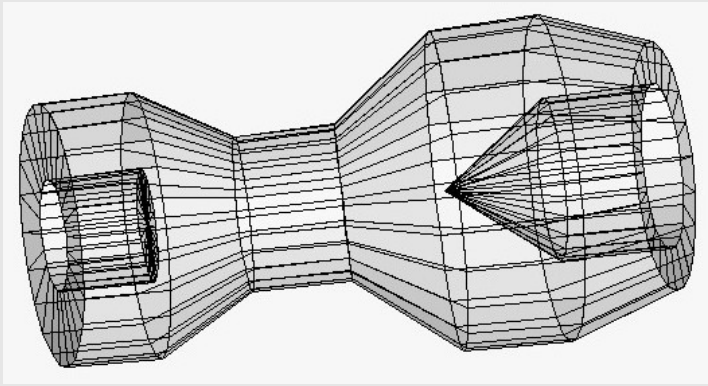
Определяемые поверхностью (BREP-Boundary REPresented)

G4BREPSolidPolycone, G4BSplineSurface, ...

Булевы формы

G4UnionSolid, G4SubtractionSolid, ...

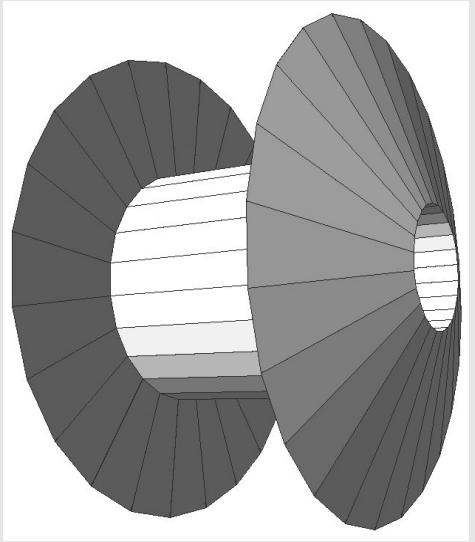


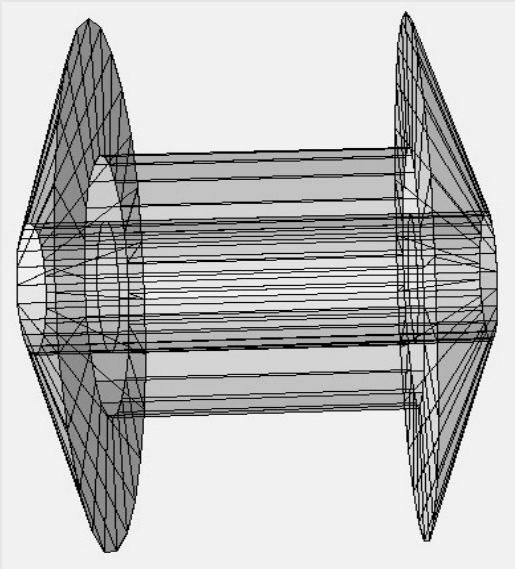


Формы, определяемые поверхностью

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)

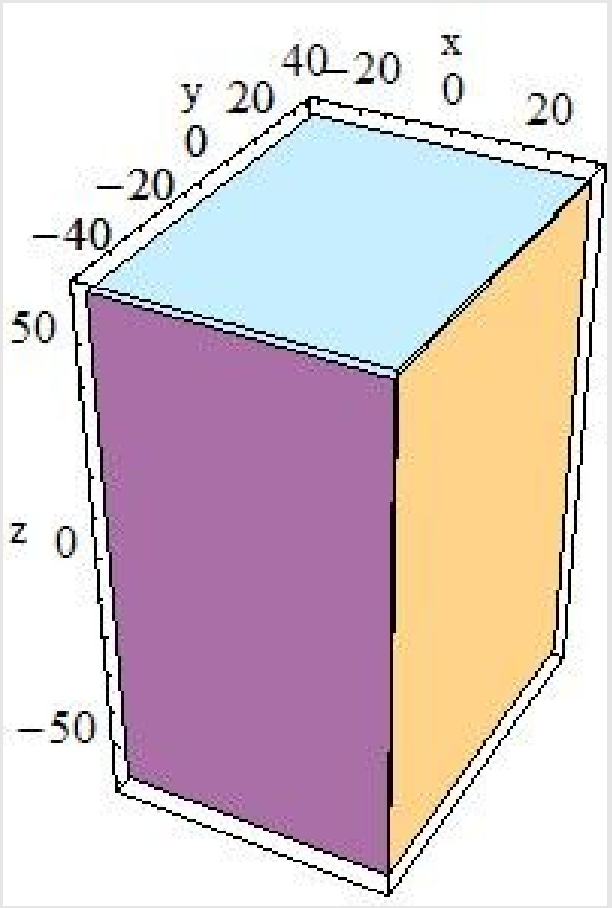




Булевы формы

- 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

Параллелепипед



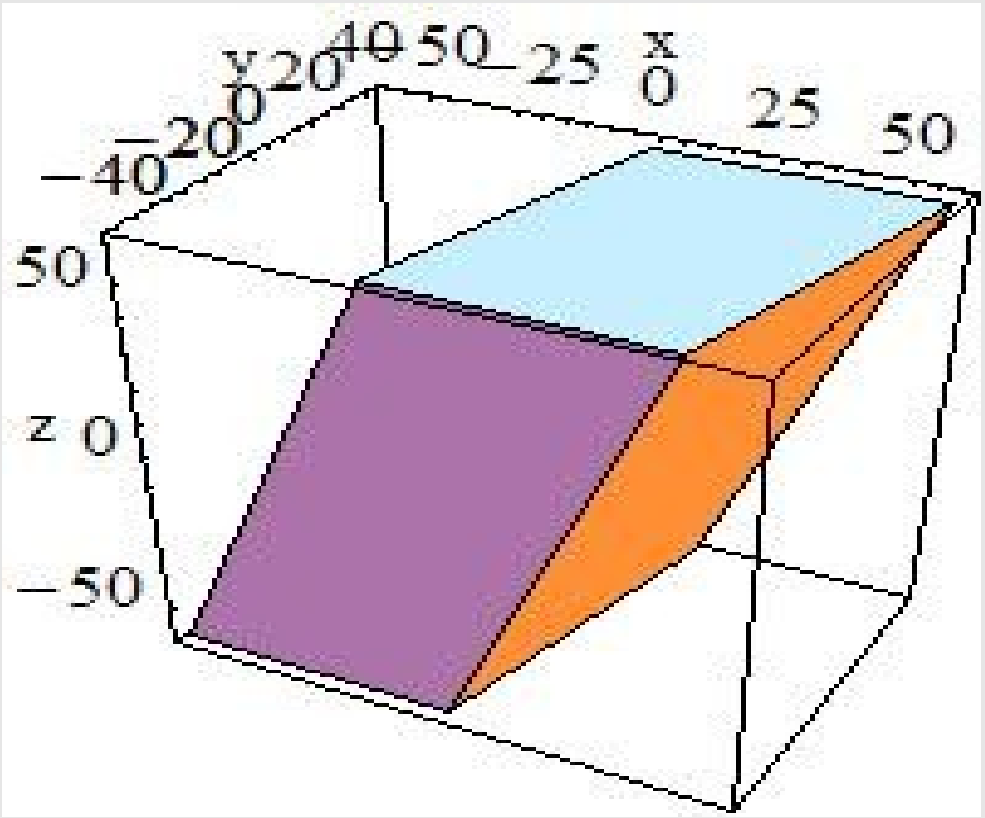

```
G4VSolid* scint_solid = new
    G4Box(const G4String& pName,
                G4double  pX,
                G4double  pY,
                G4double  pZ);
```

Пример:

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

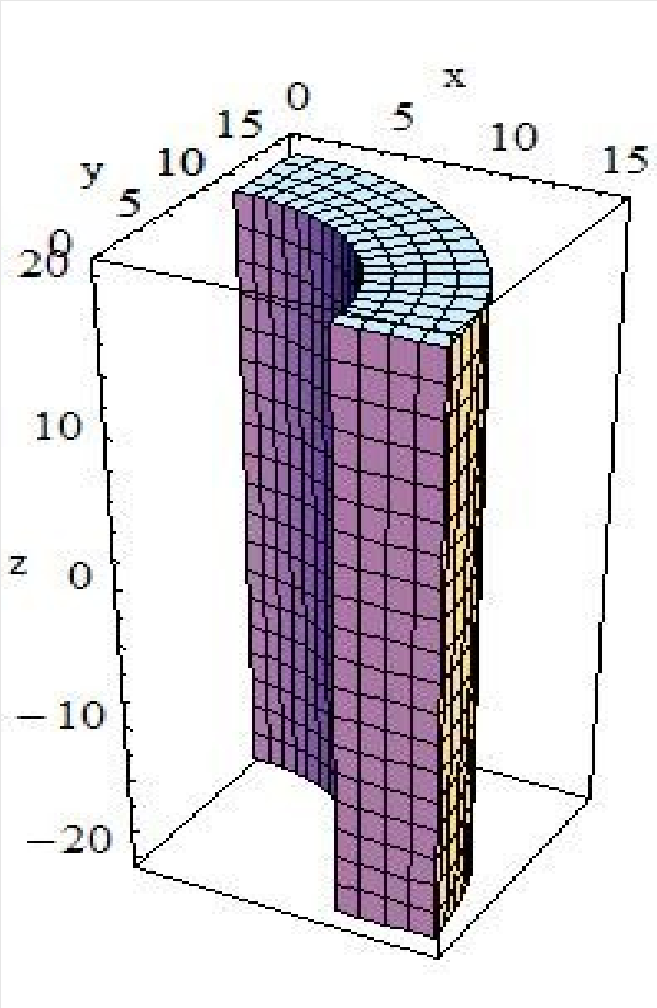
Параллелепипед в общем случае

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



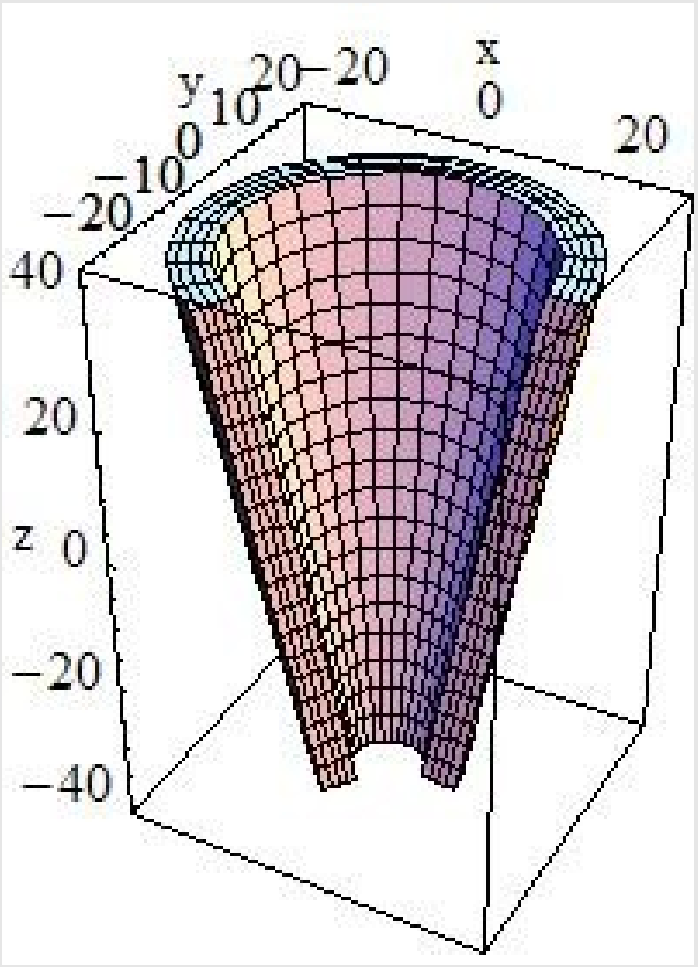
Цилиндр

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



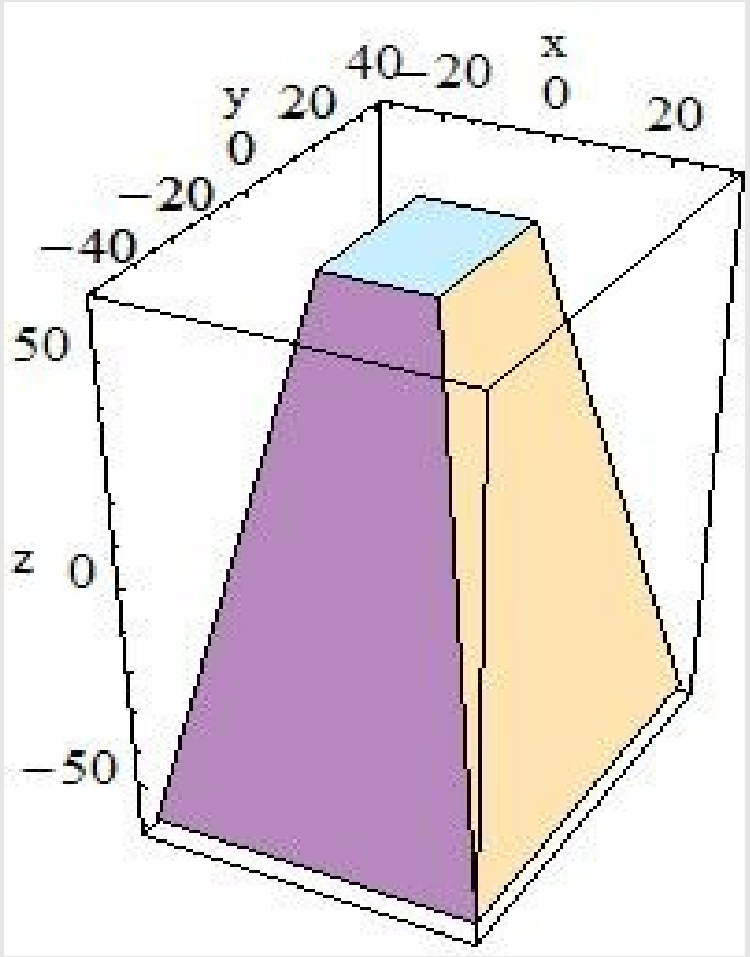
Конус

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



Трапезоид

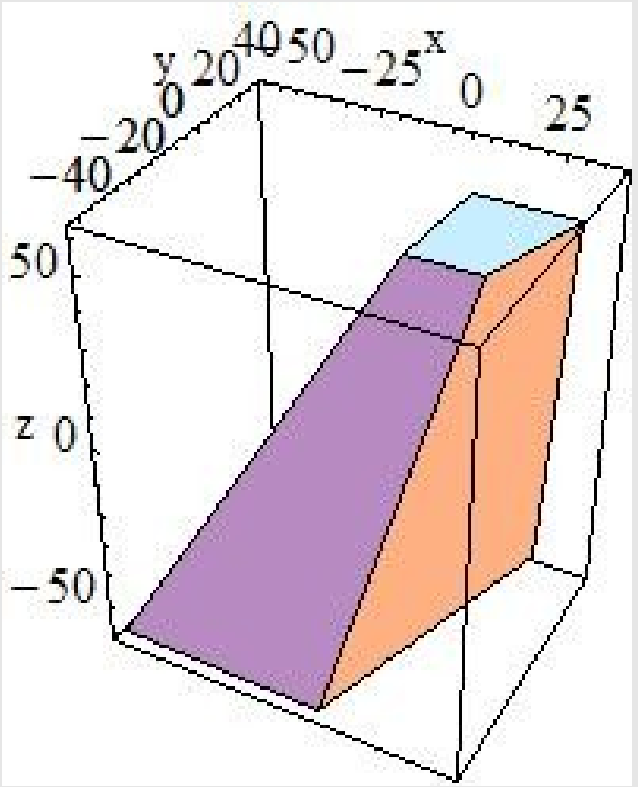
```
G4VSolid* aSolid = new
 G4Trd(const G4String& pName,
             G4double  dx1,
             G4double  dx2,
             G4double  dy1,
             G4double  dy2,
             G4double  dz)
```



Трапезоид в общем случае

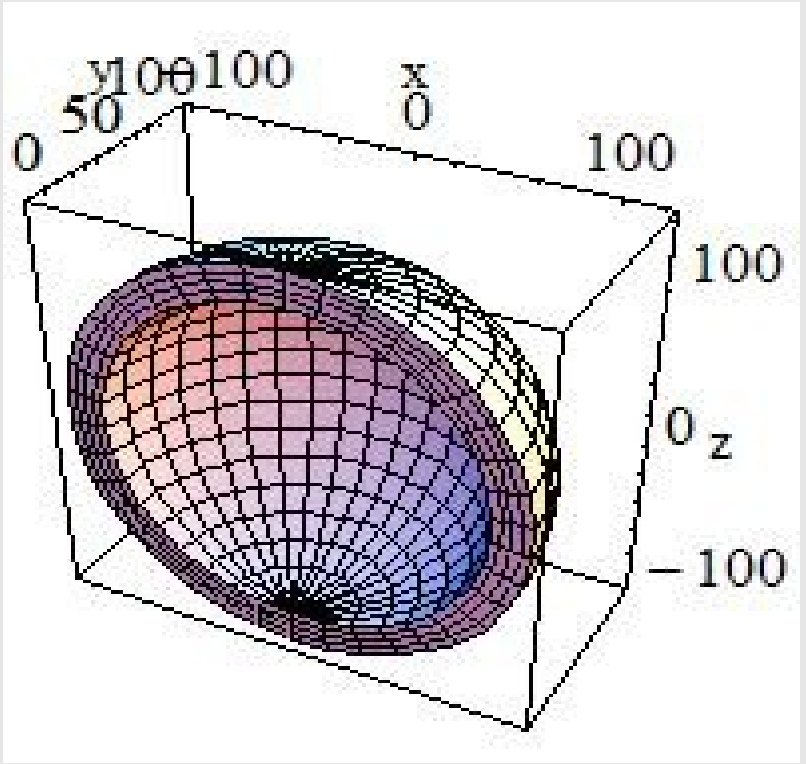
G4Trap(const G4String& pName,
G4double pZ,
G4double pY,
G4double pX,
G4double pLTX)

G4Trap(const G4String& pName,
G4double pDz, G4double pTheta,
G4double pPhi, G4double pDy1,
G4double pDx1, G4double pDx2,
G4double pAlp1, G4double pDy2,
G4double pDx3, G4double pDx4,
G4double pAlp2)



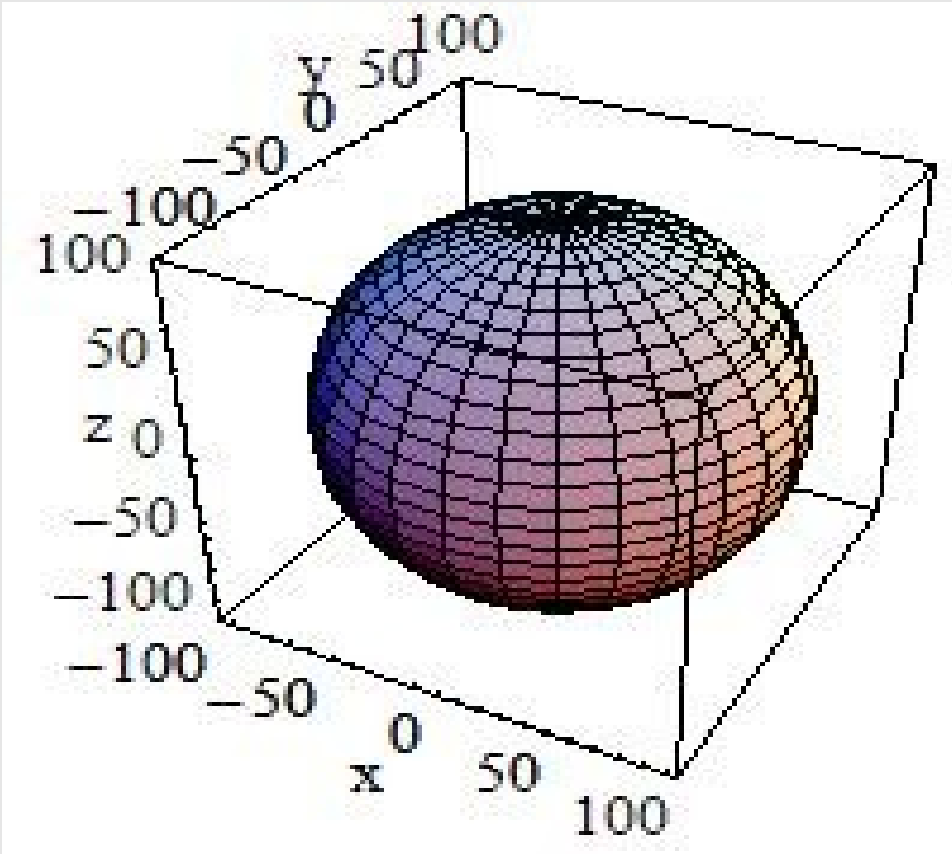
Сфера

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



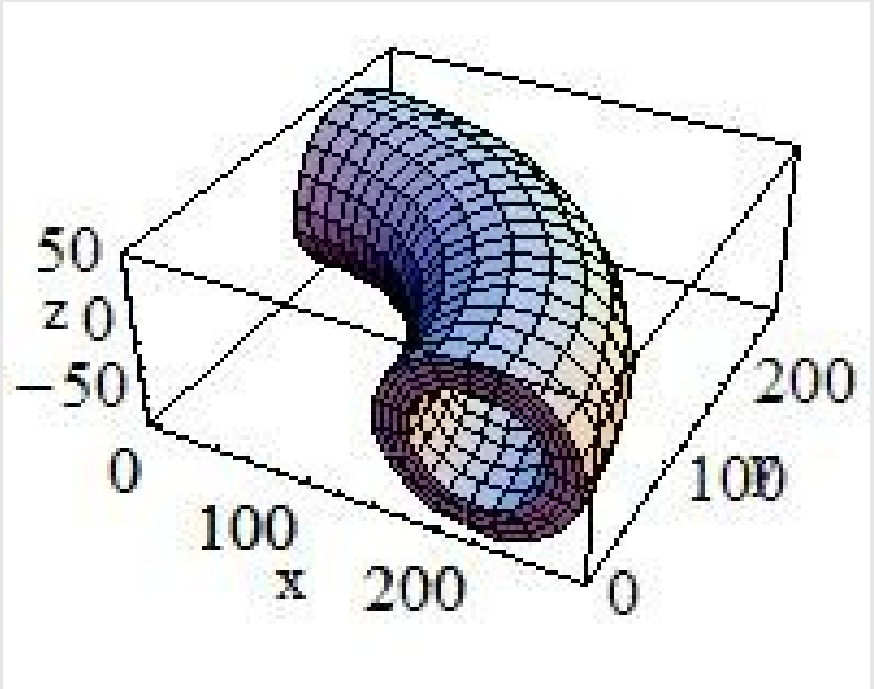
Шар

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



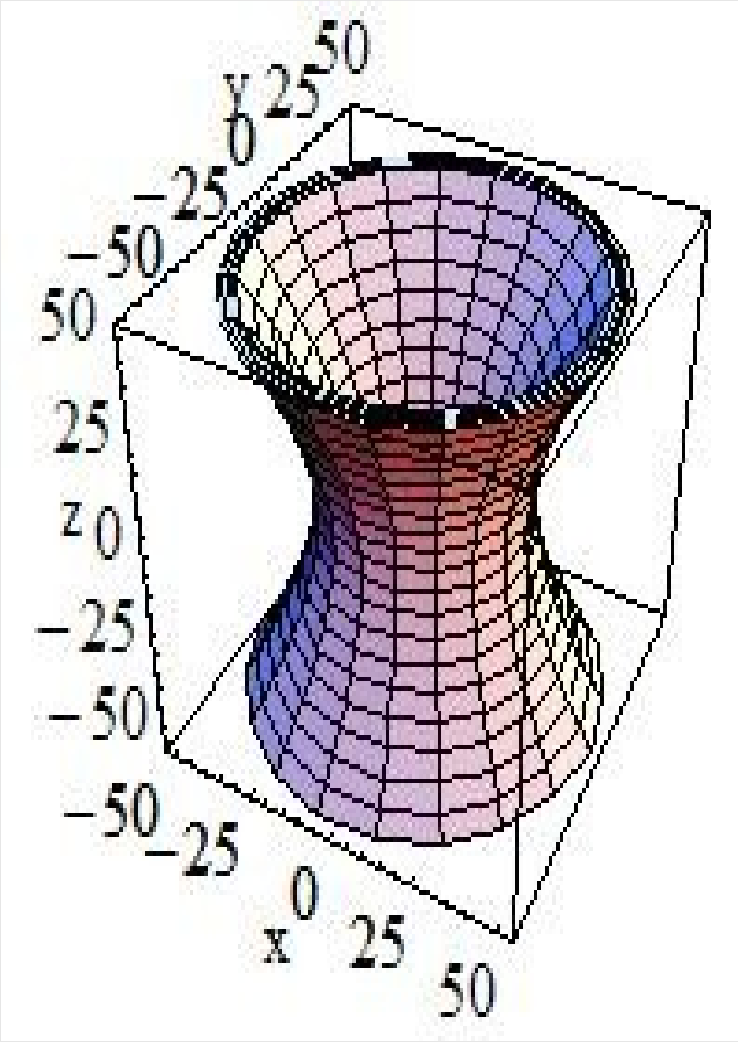
Top

```
G4VSolid* aSolid = new
  G4Torus(const G4String& pName,

                G4double   pRmin,
                G4double   pRmax,
                G4double   pRtor,
                G4double   pSPhi,
                G4double   pDPhi)
```



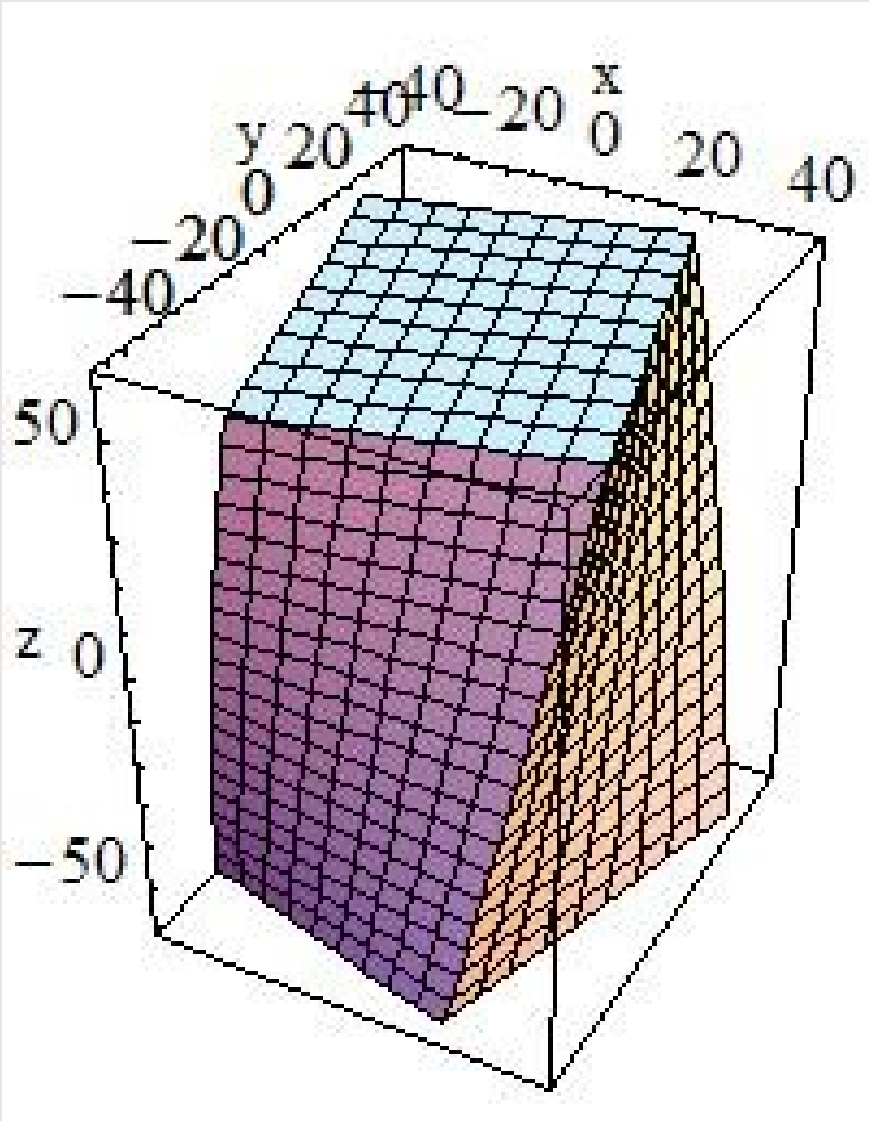
Гиперболическая поверхность

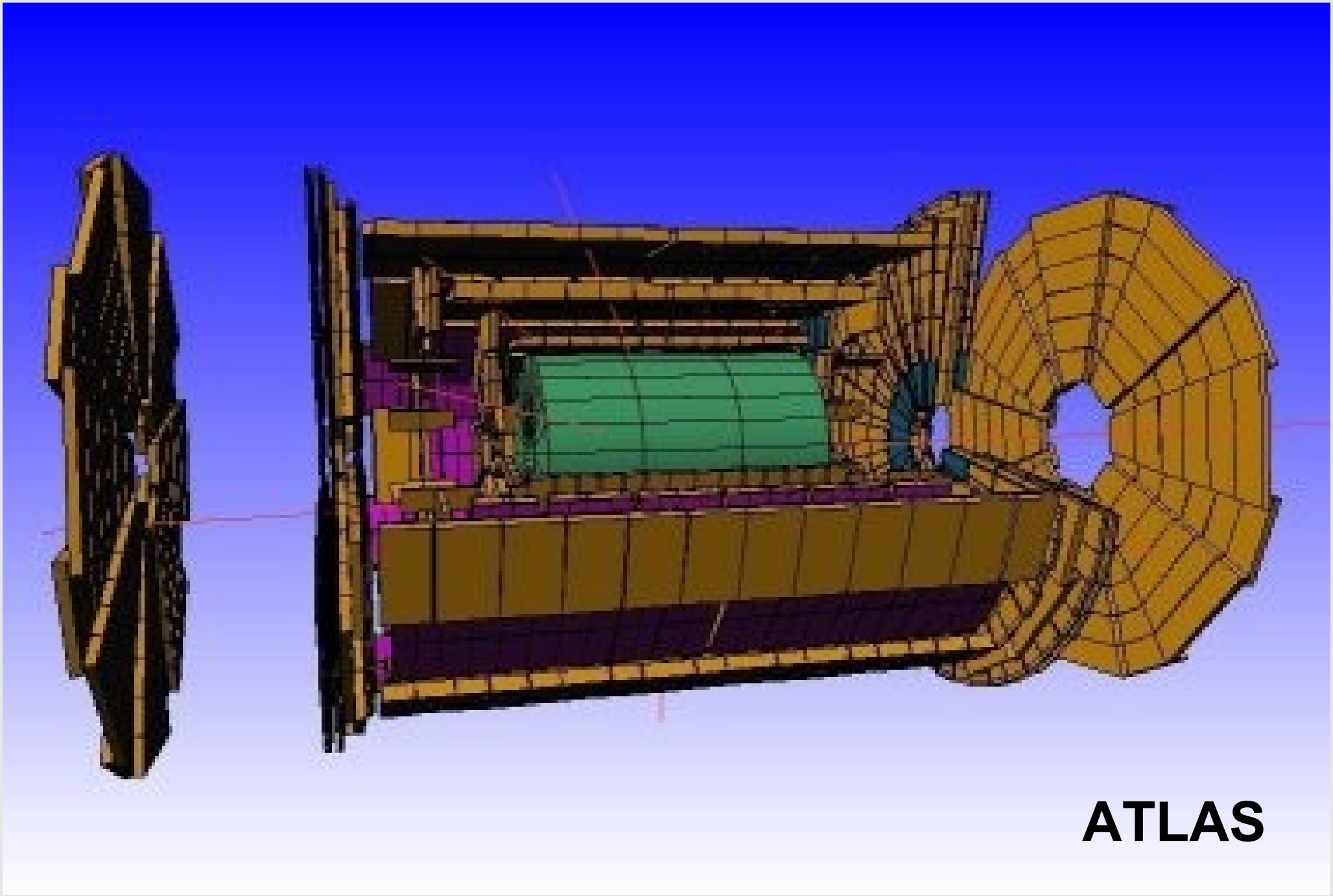
```
G4VSolid* aSolid = new
  G4Hype(const G4String& pName,
         G4double  innerRadius,
         G4double  outerRadius,
         G4double  innerStereo,
         G4double  outerStereo,
         G4double  halfLenZ)
```



Перекрученный параллелепипед

```
G4VSolid* aSolid = new
  G4TwistedBox(
          const G4String& pName,
          G4double  twistedangle,
          G4double  pDx,
          G4double  pDy,
          G4double  pDz)
```




ATLAS

Логический объем

Строится на основе формы

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

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

G4Region

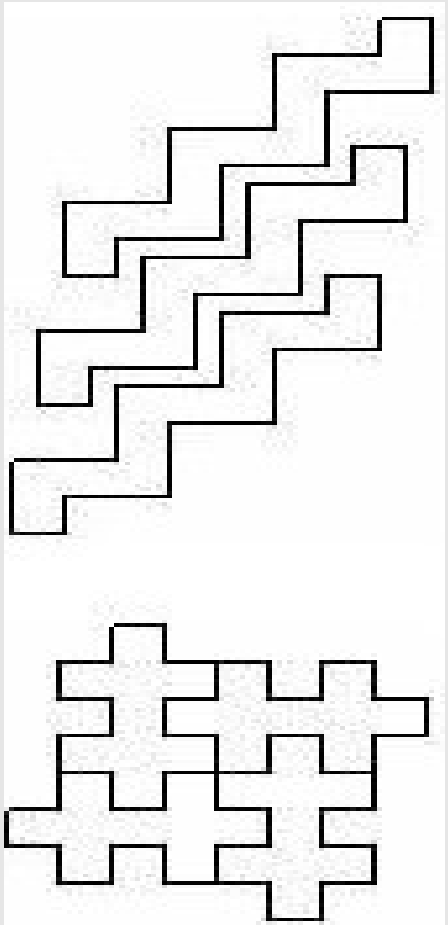
Объединение нескольких логических объемов, описывающих какую-либо подсистему детектора, для оптимизации скорости моделирования путем установки единых порогов трекинга для всей подсистемы

```
G4Region* emCalorimeter = new G4Region("EM-Calorimeter");

emCalorimeter->AddRootLogicalVolume(emCalorimeter_log);

emCalorimeter->SetProductionCuts(emCalCuts);
```

G4AssemblyVolume

Объединение нескольких логических объемов, позволяющее одновременно разместить несколько физических объемов

```
// Define one layer as one assembly volume
G4AssemblyVolume* assemblyDetector = new
G4AssemblyVolume();
// Fill assembly with logical volumes
assemblyDetector->AddPlacedVolume( LV,
G4Transform3D(Translation,RotM) );
// Place the assembly
 assemblyDetector->MakeImprint( worldLV,
G4Transform3D(Tm,Rm) );
```



Физический объем

Строится на основе логического объема

Описывает положение объема в пространстве

Позволяет одновременно описать серию одинаковых объемов (G4Replica) или параметризовать объем (G4VParametrised)

G4PVPlacement = один объем

Единственная копия данного объема размещается в материнском объеме

G4PVParameterised = одновременное описание нескольких объемов

Возможна параметризация формы, размеров, материала, положения и поворотов в пространстве в зависимости от номера копии

Ограничение: в настоящее время можно параметризовать только объемы, не имеющие дочерних, или одинаковые по форме и размерам

G4PVReplica = одновременное описание нескольких объемов

Материнский объем заполняется одинаковыми дочерними объемами

G4ReflectionFactory

одновременно размещается объем и его зеркальное отражение

G4AssemblyVolume

одновременно размещается несколько не вложенных друг в друга объемов, которые ведут себя как единое целое при геометрических преобразованиях (поворотах и т.д.)

Параметризация физического объема

Пользователь должен написать свой класс, отнаследованный от **G4VPVParameterisation** и определить в нем геометрические размеры и положение в пространстве как функцию номера копии

При желании можно также параметризовать:

- форму объема
- материал, детектирующие свойства и свойства визуализации

Ограничения:

Можно параметризовать только простые формы

Применение:

- при описании сложных детекторов с повторением одинаковых объемов (например ячейки калориметра)
- медицинские приложения – живая ткань описывается как параметризованные кубики с меняющимися свойствами

Вложенность объемов

- Все объемы должны быть вложены один в другой
  Перекрытие 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

Проверка перекрытия объемов

```
Idle> /geometry/test/grid_test  true
GeomTest: no daughter volume extending outside
mother detected.
GeomTest: no overlapping daughters detected.
```

G4VPlacement

```
G4PVPlacement(  G4RotationMatrix*  pRot,
                const G4ThreeVector&     translate,
                G4LogicalVolume*   pCurrentLogical,
                const G4String&          pName,
                G4LogicalVolume*   pMotherLogical,
                G4bool               pMany,
                G4int              pCopyNo,
                G4bool               pSurfChk=false )
```

G4PVReplica

```
G4PVReplica( const G4String&        pName,
             G4LogicalVolume*   pCurrentLogical,
             G4LogicalVolume*   pMotherLogical,
             const EAxis            pAxis,
             const G4int           nReplicas,
             const G4double            width,
             const G4double            offset=0 )
```

```
#include "G4RunManager.hh"
#include "G4UImanager.hh"
#include "ExN01DetectorConstruction.hh"
#include "ExN01PhysicsList.hh"
#include "ExN01PrimaryGeneratorAction.hh"
int main()
{
  // construct the default run manager
  G4RunManager* runManager = new G4RunManager;
  // set mandatory initialization classes
  runManager->SetUserInitialization(new ExN01DetectorConstruction);
  runManager->SetUserInitialization(new ExN01PhysicsList);
  // set mandatory user action class
  runManager->SetUserAction(new ExN01PrimaryGeneratorAction);
  // initialize G4 kernel
  runManager->initialize();
  // get the pointer to the UI manager and set verbosities
  G4UImanager* UI = G4UImanager::GetUIpointer();
  UI->ApplyCommand("/run/verbose 1");
  // start a run
  int numberOfEvent = 3;
  runManager->BeamOn(numberOfEvent);
  // job termination
  delete runManager;
  return 0;
}
```

Класс ExN01DetectorConstruction

```
class G4LogicalVolume;
class G4VPhysicalVolume;

#include "G4VUserDetectorConstruction.hh"

class ExN01DetectorConstruction : public G4VUserDetectorConstruction
{
  public:
    ExN01DetectorConstruction();
    ~ExN01DetectorConstruction();
    G4VPhysicalVolume* Construct();
  private:
    // Logical volumes
    //
    G4LogicalVolume* experimentalHall_log;
    G4LogicalVolume* tracker_log;

    // Physical volumes
    //
    G4VPhysicalVolume* experimentalHall_phys;
    G4VPhysicalVolume* tracker_phys;
};
```

Описание базового объема

```
//------------------------------ experimental hall (world volume)
 //------------------------------ beam line along x axis

 G4double expHall_x = 3.0*m;
 G4double expHall_y = 1.0*m;
 G4double expHall_z = 1.0*m;
 G4Box* experimentalHall_box
   = new G4Box("expHall_box",expHall_x,expHall_y,expHall_z);
 experimentalHall_log = new  G4LogicalVolume( experimentalHall_box,
                                              Ar,"expHall_log",0,0,0);
 experimentalHall_phys = new G4PVPlacement(0,G4ThreeVector(),
                                 experimentalHall_log,"expHall",0,false,0);
```

Описание остальных объемов

```
//---------------------------- a tracker tube

 G4double innerRadiusOfTheTube = 0.*cm;
 G4double outerRadiusOfTheTube = 60.*cm;
 G4double hightOfTheTube = 50.*cm;
 G4double startAngleOfTheTube = 0.*deg;
 G4double spanningAngleOfTheTube = 360.*deg;
 G4Tubs* tracker_tube = new
   G4Tubs("tracker_tube",innerRadiusOfTheTube,
                         outerRadiusOfTheTube,hightOfTheTube,
                         startAngleOfTheTube,spanningAngleOfTheTube);
 tracker_log = new G4LogicalVolume(tracker_tube,Al,"tracker_log",0,0,0);
 G4double trackerPos_x = -1.0*m;
 G4double trackerPos_y = 0.*m;
 G4double trackerPos_z = 0.*m;
 tracker_phys = new G4PVPlacement(0,
         G4ThreeVector(trackerPos_x,trackerPos_y,trackerPos_z),
         tracker_log,"tracker",experimentalHall_log,false,0);
```