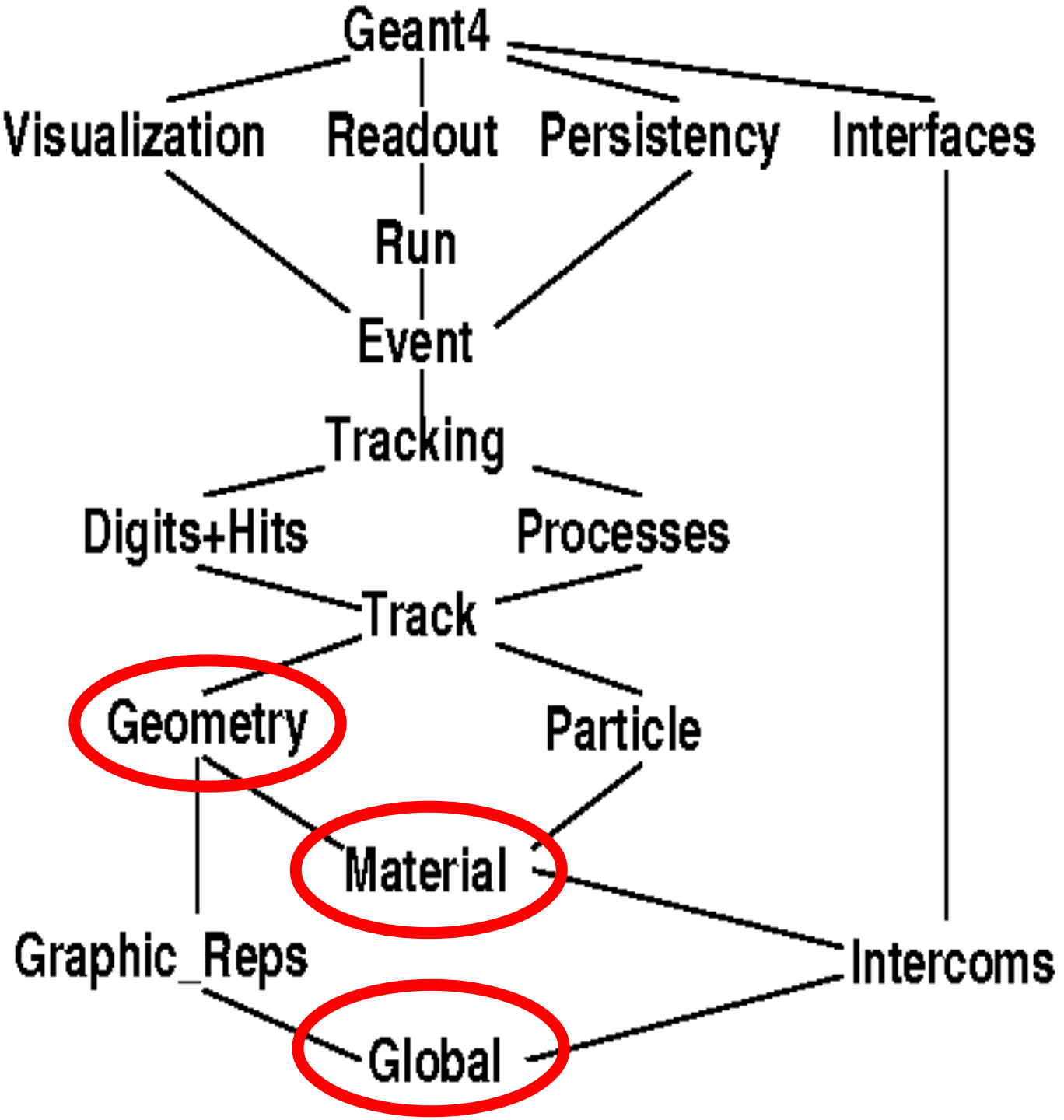
Построение модели детектора


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

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

Geant4 ничего не знает про молекулы и связанные с ними физические явления!

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;
```

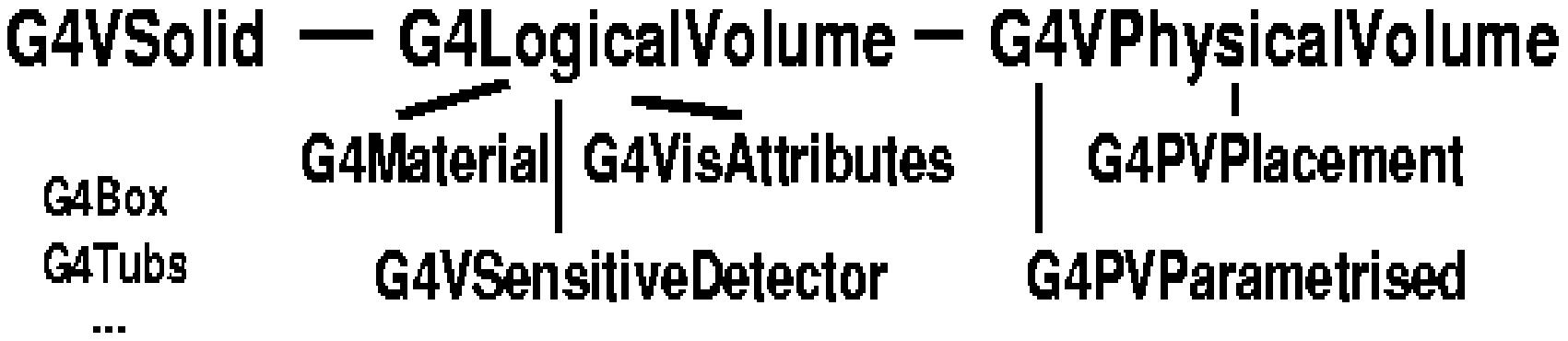
Библиотека материалов Geant4

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

Описание объема

Объем описывается в три этапа

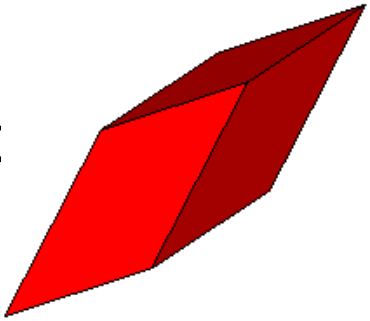
- форма (G4VSolid)
- логический объем (G4LogicalVolume)
- физический объем (G4VPhysicalVolume)



Формы

- Простые формы (CGS – Constructed Solid Geomet

G4Box, G4Tubs, G4Cons, G4Trd, ...

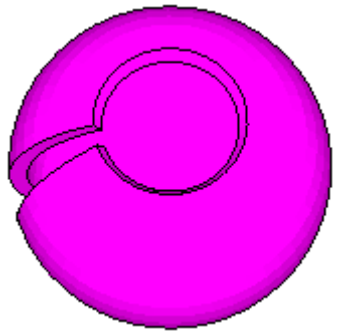


- Специальные формы

G4polycone, G4Polyhedra, G4Hype, ...

- Определяемые поверхностью

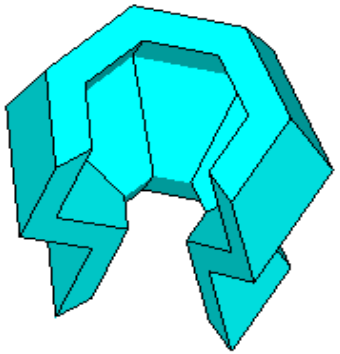
(BREP-Boundary REPresented)

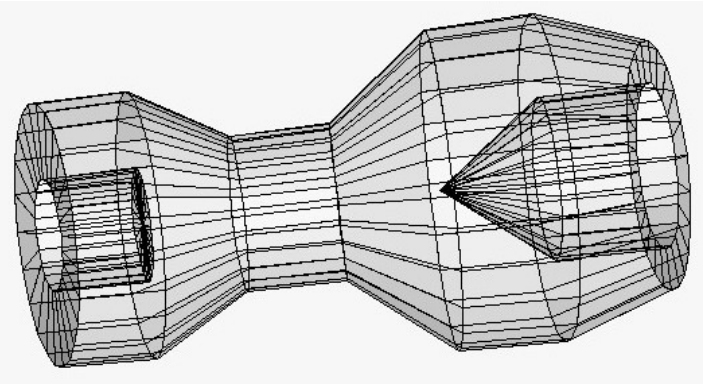


G4BREPSolidPolycone, G4BSplineSurface, ...

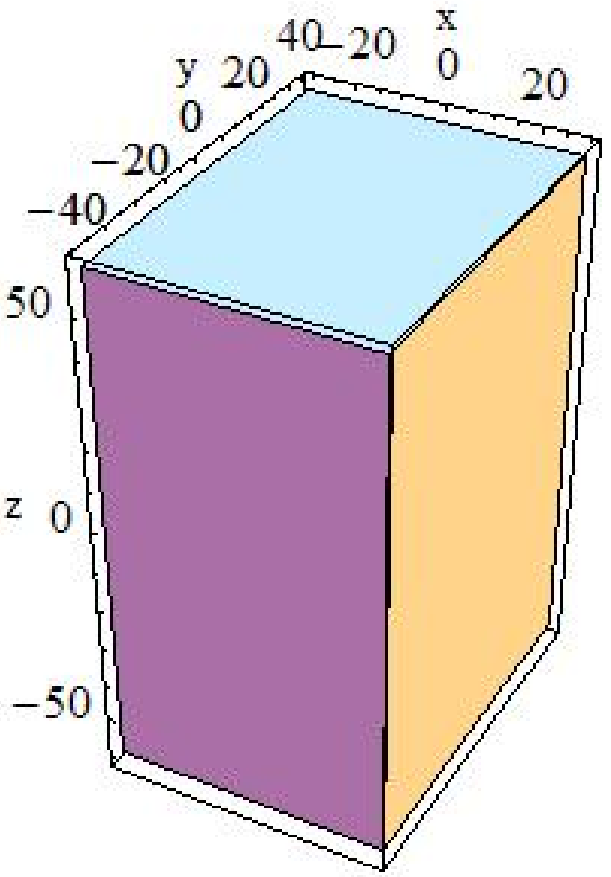
- Булевы формы

G4UnionSolid, G4SubtractionSolid, ...





Параллелепипед



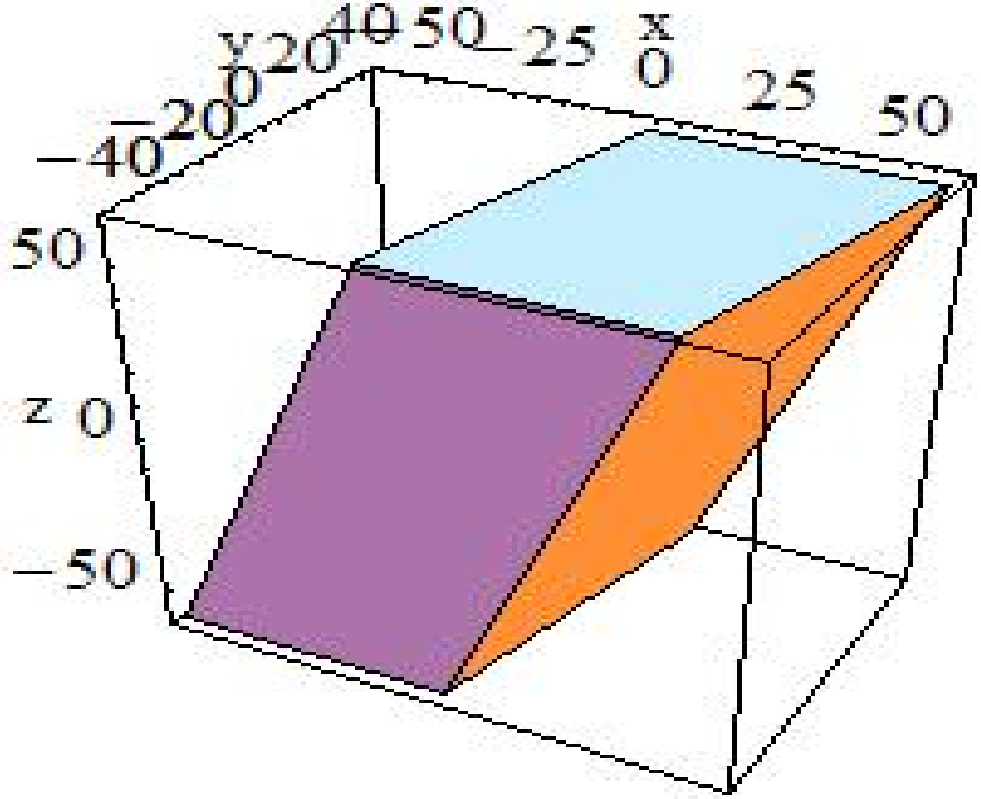

```
G4VSolid* scint_solid = new
    G4Box(const G4String& pName,
                G4double  pX,
                G4double  pY,
                G4double  pZ);
```

Пример:

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

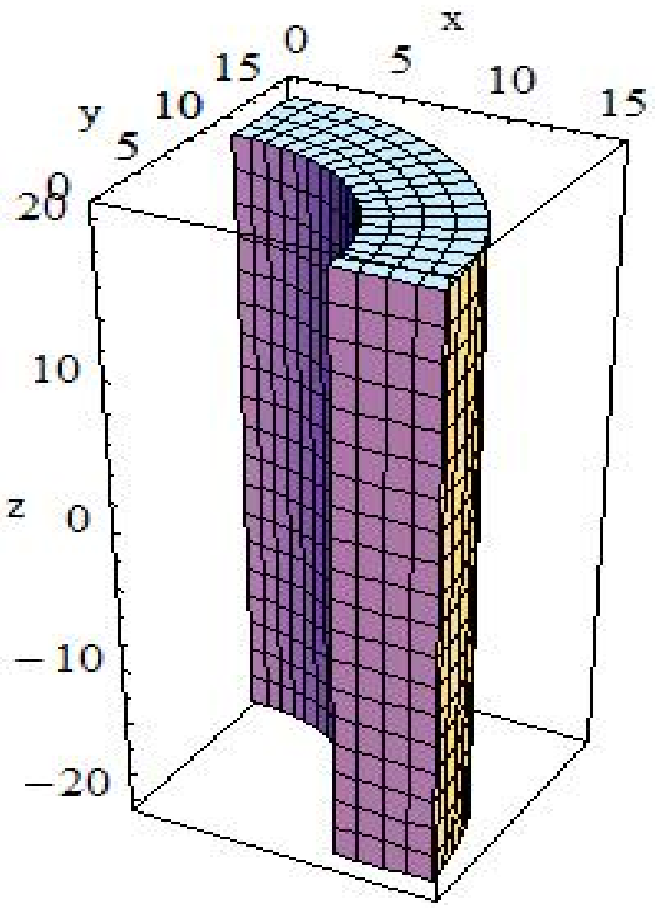
Параллелепипед в общем случае

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



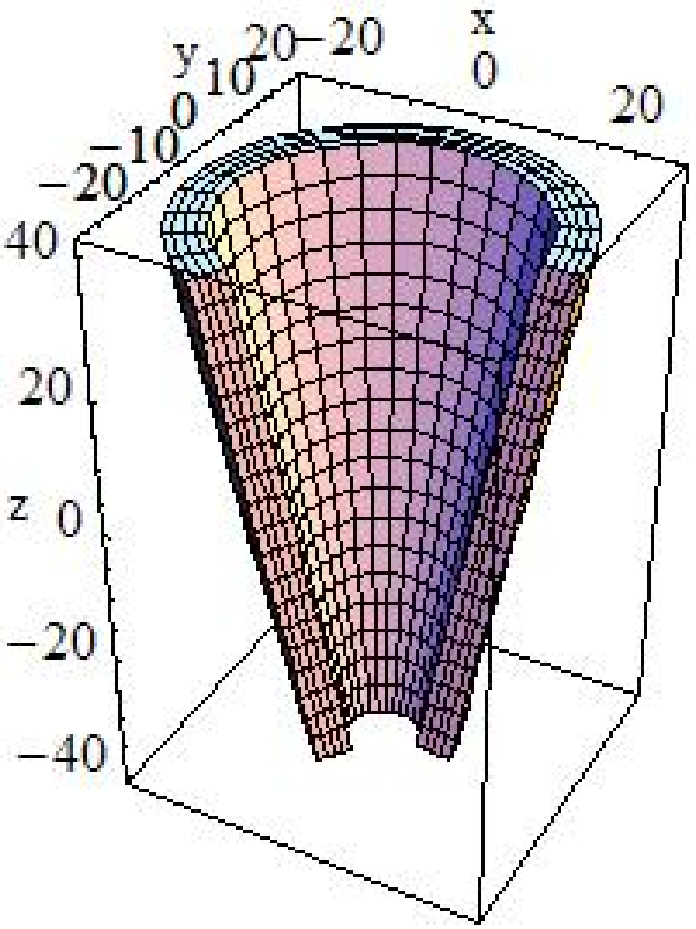
Цилиндр

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



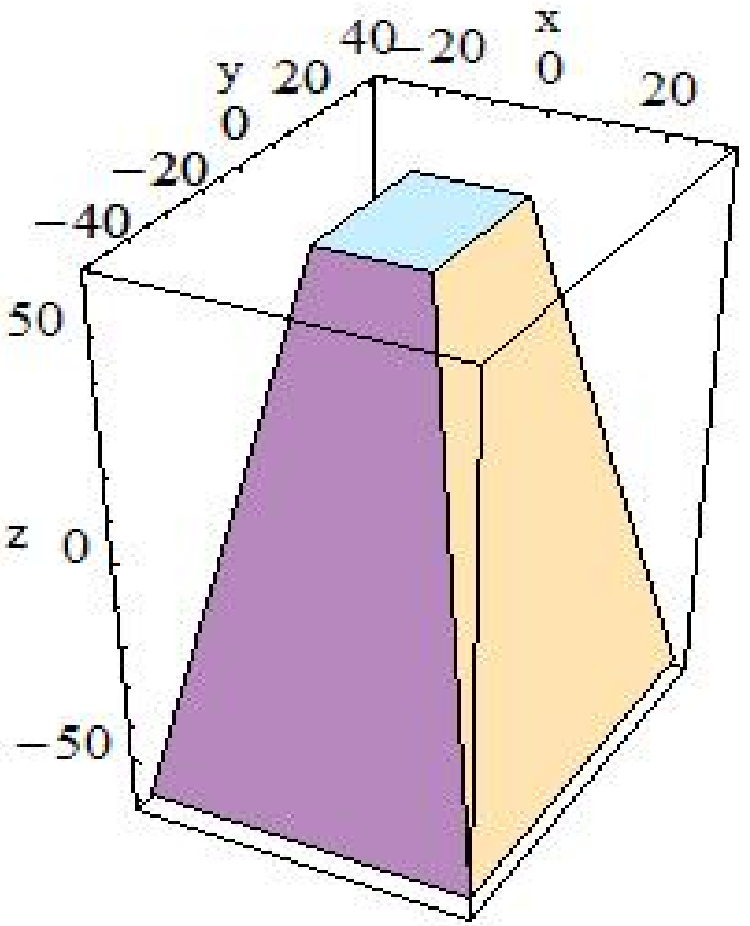
Конус

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



Трапезоид

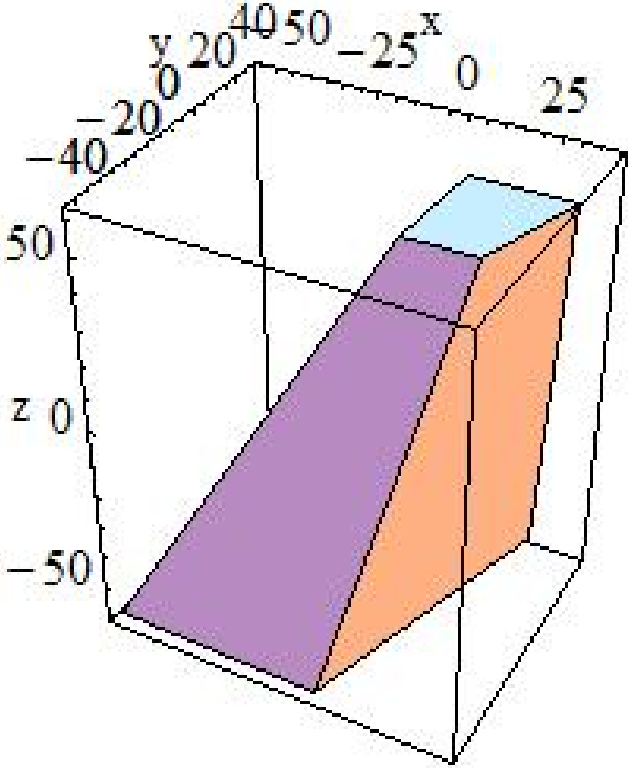
```
G4VSolid* aSolid = new
 G4Trd(const G4String& pName,
             G4double  dx1,
             G4double  dx2,
             G4double  dy1,
             G4double  dy2,
             G4double  dz)
```



Трапезоид в общем случае

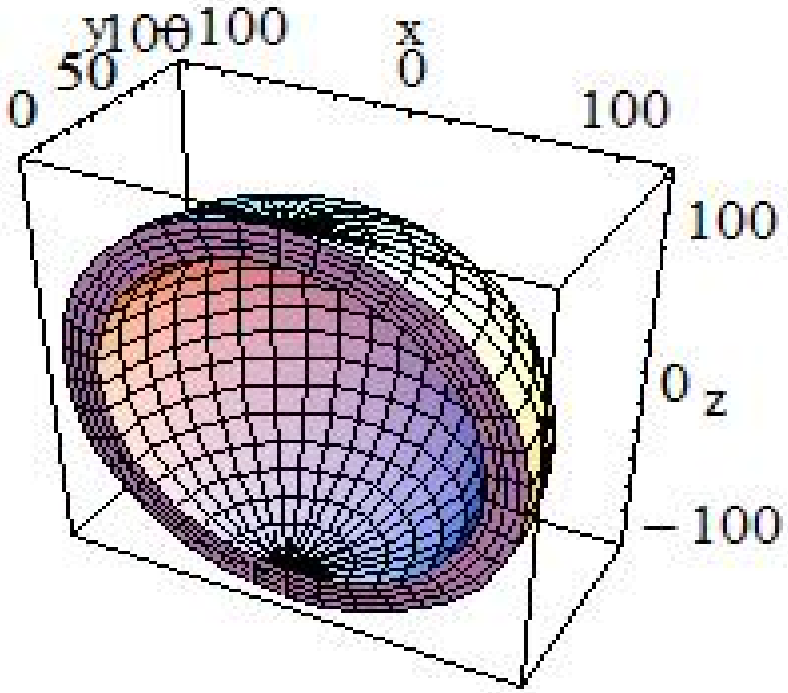
G4Trap(const G4String& pName,
G4double pZ,
G4double pY,
G4double pX,
G4double pLTX)

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)



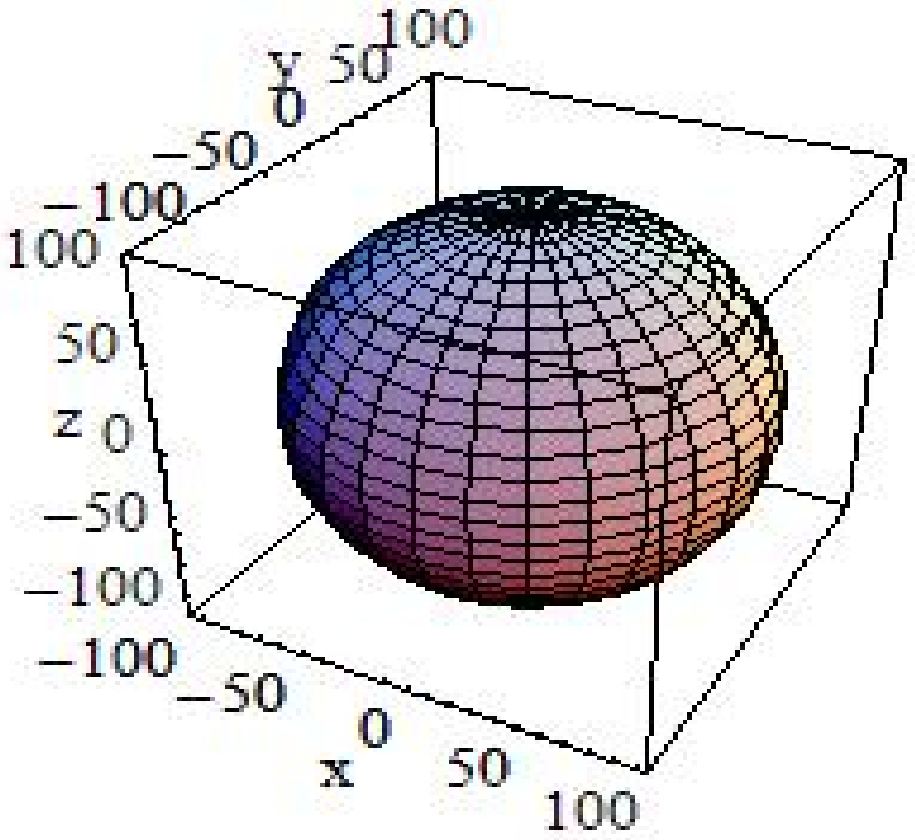
Сфера

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



Шар

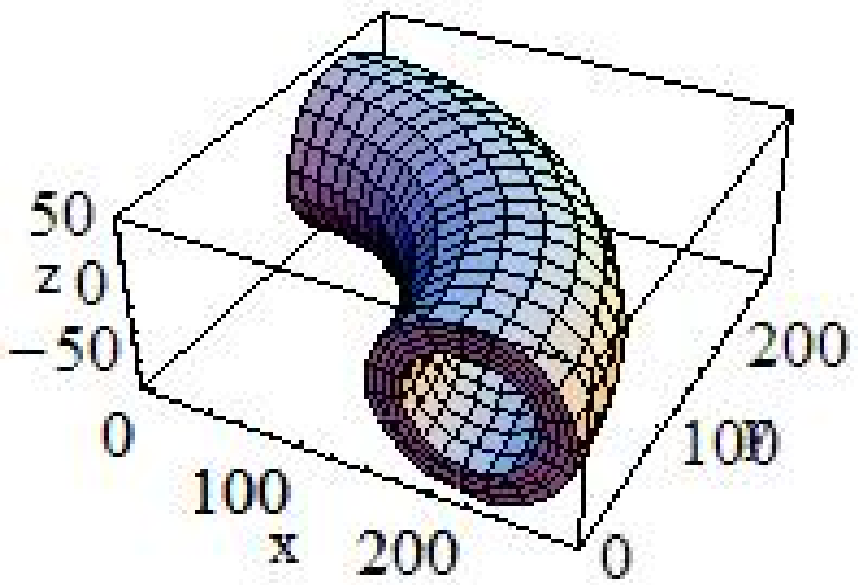
G4VSolid* aSolid = new
G4Orb(const G4String& pName,
G4double pRmax)



Top

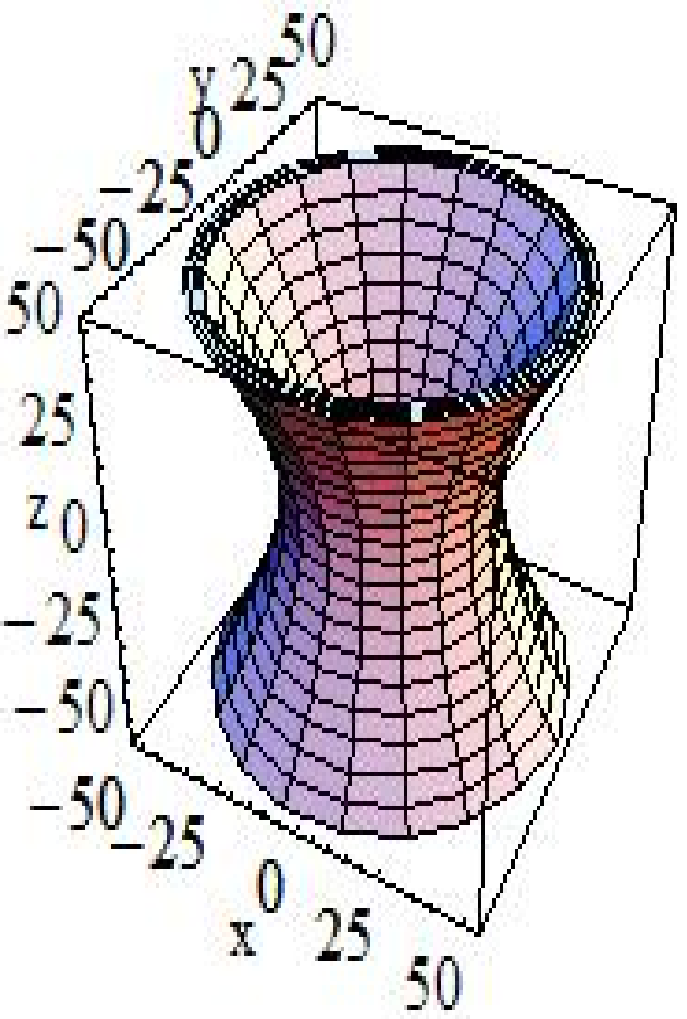
G4VSolid* aSolid = new
G4Torus(const G4String& pName,

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



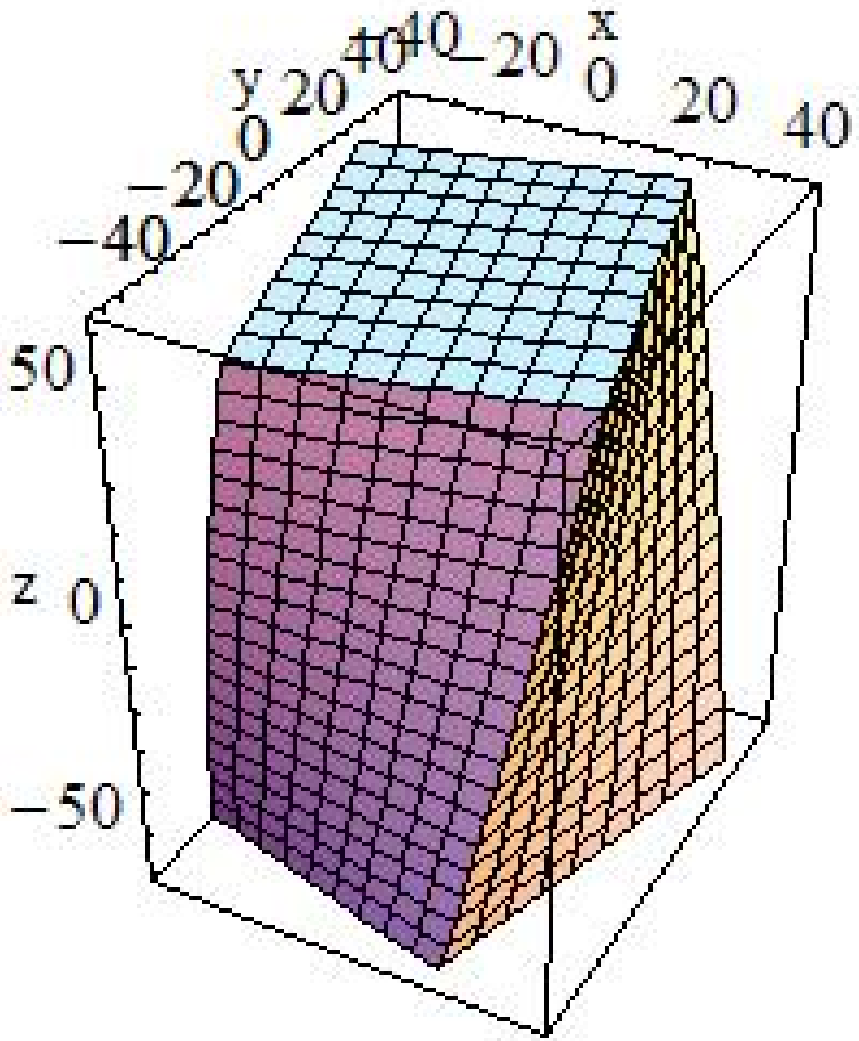
Гиперболическая поверхность

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



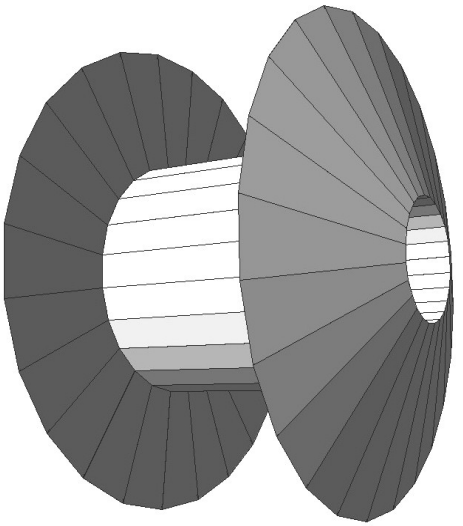
Перекрученный параллелепипед

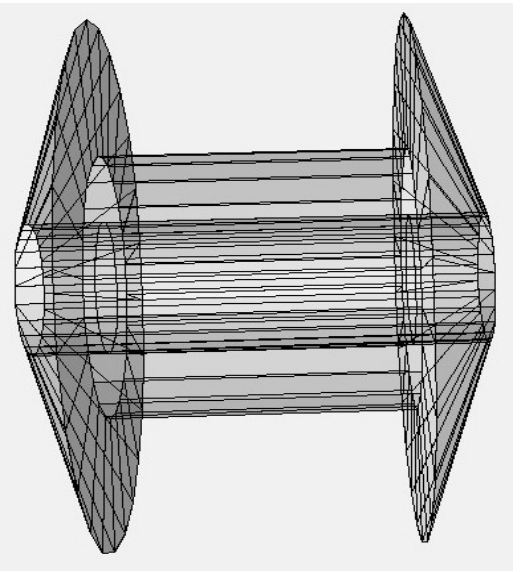
```
G4VSolid* aSolid = new
  G4TwistedBox(
          const G4String& pName,
          G4double  twistedangle,
          G4double  pDx,
          G4double  pDy,
          G4double  pDz)
```



Формы, определяемые поверхностью

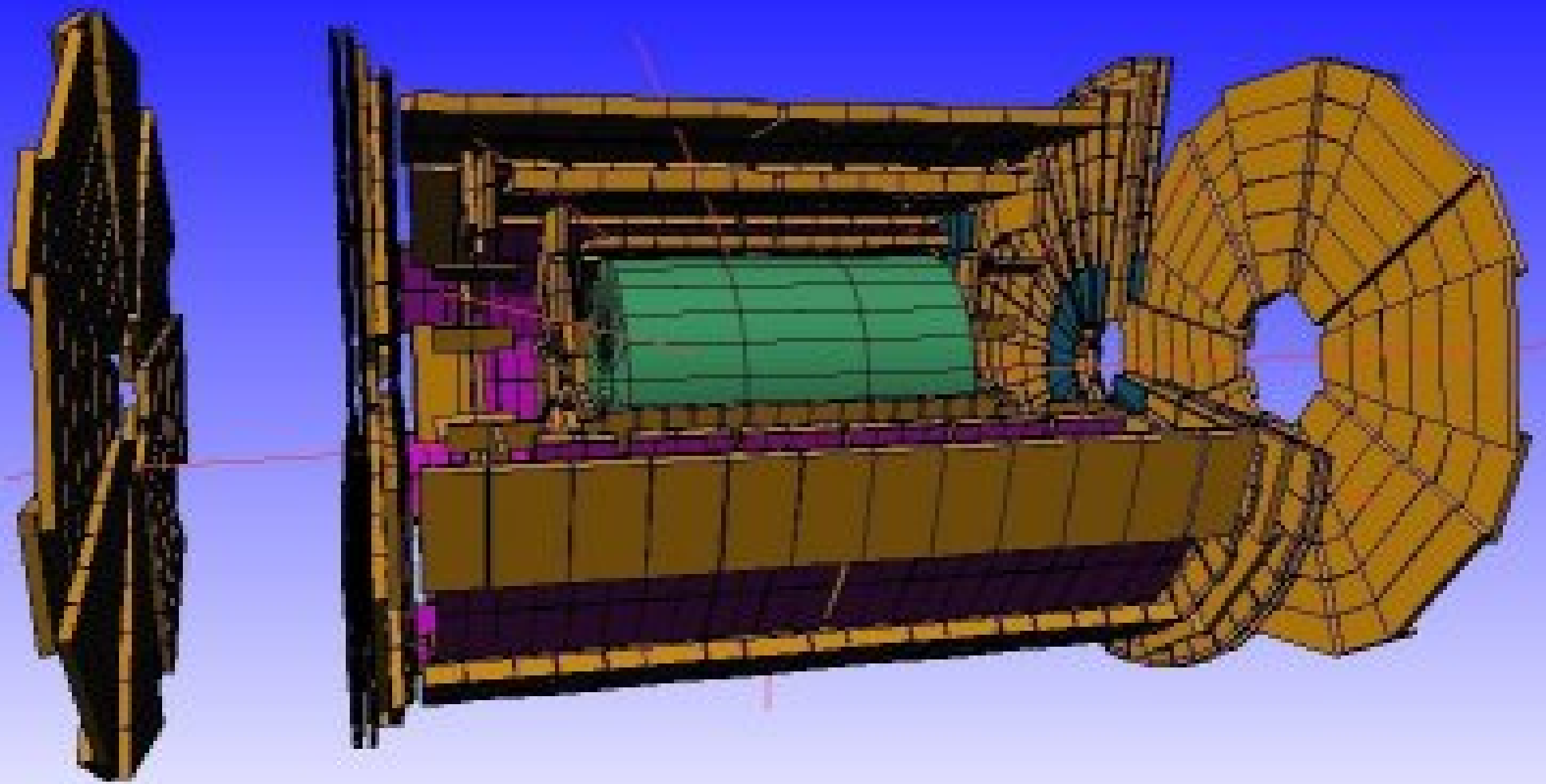
- Для задания такой формы необходимо описать все ограничивающие ее поверхности
- Поверхности могут быть
  - элементарные (плоскости, поверхности 2 порядка и т.д)
  - сплайны, B-сплайны, NURBS (описываются в Geant4 используя интерфейс к системам САПР)





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



ATLAS

Логический объем

- Строится на основе формы

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

G4LogicalVolume

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

G4Region

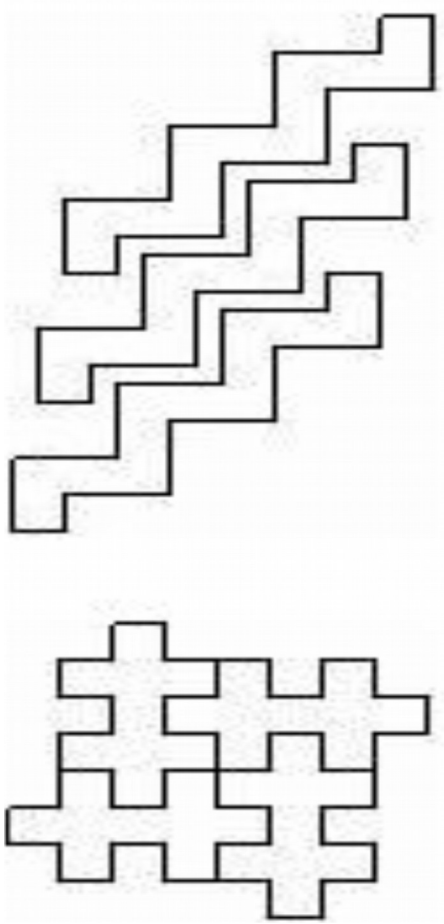
Объединение нескольких логических объемов, описывающих какую-либо подсистему детектора, для оптимизации скорости моделирования путем установки единых порогов трекинга для всей подсистемы

```
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 или параметризовать свойства объема в зависимости от номера копии

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

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.)

G4VPlacement

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

Как задать матрицу поворота?

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

и т. д.

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);
```

G4PVReplica

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

Параметризация физического объема

- Пользователь должен написать свой класс, отнаследованный от **G4VPVParameterisation** и определить в нем свойства объема как функцию номера копии:
- Можно параметризовать:
  - материал (*ComputeMaterial(copyNo, pPhysVol)*)
  - форму объема (*ComputeSolid(copyNo, pPhysVol)*)
  - положение (*ComputeTranslation(copyNo, pPhysVol)*)
  - размеры (*ComputeDimensions(solid, copyNo, pPhysVol)*)
- **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

Пример

```
void MyParameterisation::ComputeTransformation(const
G4int copyNo, G4VPhysicalVolume* physVol) const
{
  G4double fStartZ = 0*cm;
  G4double fSpacing = 1*cm;
  G4double Zposition = fStartZ + copyNo * fSpacing;
  G4ThreeVector origin(0,0,Zposition);
  physVol->SetTranslation(origin);
  physVol->SetRotation(0);
}
```

Проверка перекрытия объемов

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

ИЛИ

Idle> /geometry/test/run
Running geometry overlaps check...
Checking overlaps for volume Active ... OK!
Checking overlaps for volume Detector ... OK!
Checking overlaps for volume Coll1 ... OK!
Checking overlaps for volume line6_top1 ...
-------- WWWW ------- G4Exception-START -------- WWWW -------
*** G4Exception : GeomVol1002
issued by : G4PVPlacement::CheckOverlaps()
Overlap with volume already placed !
Overlap is detected for volume line6_top1
with line6_top2 volume's
local point (-76.8103,185,-6.38222), overlapping by at least: 1.11778 mm
NOTE: Reached maximum fixed number -1- of overlaps reports for this volume !
*** This is just a warning message. ***

```
#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 материалов

```
//////////////////////////////////////////////////////////
G4VPhysicalVolume* ExN01DetectorConstruction::Construct()
//////////////////////////////////////////////////////////
{
  //------------------------------------------------- materials
  G4double a;  // atomic mass
  G4double z;  // atomic number
  G4double density;

  G4Material* Ar =
    new G4Material("ArgonGas", z= 18., a= 39.95*g/mole, density= 1.782*mg/cm3);

  G4Material* Al =
    new G4Material("Aluminum", z= 13., a= 26.98*g/mole, density= 2.7*g/cm3);

  G4Material* Pb =
    new G4Material("Lead", z= 82., a= 207.19*g/mole, density= 11.35*g/cm3);
```

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);
```

Описание электрического и магнитного полей

- Описание электрического и магнитного полей задается при описании геометрии детектора
- Моделирование движения частицы в поле осуществляется численным решением уравнения движения с учетом величины поля в данной точке
- Geant4 позволяет описывать как простые (однородные) поля, так и сложные поля, в том числе задаваемые экспериментально измеренными картами напряженности поля
- Поля могут быть как стационарными, так и изменяющимися во времени

Описание однородного магнитного поля

- Создается объект класса G4UniformMagneticField

```
G4UniformMagField* magField  = new
G4UniformMagField(G4ThreeVector(0.,0.,fieldValue=1*tesla));
```

- Этот объект передается объекту G4FieldManager, которой управляет движением частицы в поле

```
G4FieldManager* fieldMgr
 = G4TransportationManager::GetTransportationManager()->
GetFieldManager();
fieldMgr->SetDetectorField(magField);
```

- Создается объект, рассчитывающий траекторию движения

```
fieldMgr->CreateChordFinder(magField);
```

Описание однородного электрического поля

- Создается объект класса G4UniformElectricField
- Создается объект класса G4EqMagElectricField, задающий уравнение движения в поле
- Создается объект, численно решающий уравнение движения (например G4ClassicalRK4, для решения методом Рунге-Кутта 4-го порядка)
- Аналогично случаю магнитного поля, объект G4UniformElectricField передается объекту G4FieldManager
- Создается объект класса G4MagInt_Driver, применяющий определенный выше метод решения уравнения движения к заданному полю.
- Создается объект, рассчитывающий траекторию движения, и передается в G4FieldManager

```
G4ElectricField*             fEMfield;
G4EqMagElectricField*        fEquation;
G4MagIntegratorStepper*      fStepper;
G4FieldManager*              fFieldMgr;
G4double                     fMinStep ;
G4ChordFinder*               fChordFinder ;

fEMfield = new G4UniformElectricField(G4ThreeVector(0.0,100000.0*kilovolt/cm,0.0));

// уравнение движения в поле
fEquation = new G4EqMagElectricField(fEMfield);
G4int nvar = 8; // число переменных

fStepper = new G4ClassicalRK4( fEquation, nvar );

fFieldManager= G4TransportationManager::GetTransportationManager()->
    GetFieldManager();

fFieldManager->SetDetectorField(fEMfield );
fMinStep     = 0.010*mm ;  // минимальный шаг 10 микрон
fIntgrDriver = new G4MagInt_Driver(fMinStep,
                        fStepper,  fStepper->GetNumberOfVariables() );

fChordFinder = new G4ChordFinder(fIntgrDriver);
fFieldManager->SetChordFinder( fChordFinder );
```

Описание полей в определенном логическом объеме

- Способом, описанным выше, поле задается во всей установке одновременно.
- Однако поле может быть также задано только в определенном логическом объеме, включая или исключая все его дочерние объемы. Для этого создается собственный FieldManager, и передается логическому объему:

```
G4bool includeDaugthers = true;
G4FieldManager* UserFM = MyFieldSetup->GetLocalFieldManager();
logicVolumeWithField -> SetFieldManager(UserFM, includeDaugthers);
```

- Подробнее в примере *examples/extended/field/field03*

MyFieldSetup

```
class MyFieldSetup {
  MyFieldSetup()
  {
   fLocalFieldManager = new G4FieldManager();
   // Initialize FieldManager as described above
  ...
  }

  G4FieldManager*  GetLocalFieldManager() { return fLocalFieldManager;}
...
protected:

  G4FieldManager*        GetGlobalFieldManager();
    // Returns the global Field Manager

  G4FieldManager*        fLocalFieldManager;
...
};
```

Более простой вариант

```
G4UniformMagField* myVolumeField = new

    G4UniformMagField(G4ThreeVector((0,0,0.5*tesla)
    ));

G4FieldManager*   myFieldMgrVolume = new
    G4FieldManager(myVolumeField);.

G4bool includeDaugthers = false; // или true

myLogicVolume ->
    SetFieldManager(myFieldMgrVolume,
    includeDaugthers);
```

Указание точности вычислений

Точность вычислений определяется тремя параметрами:

- максимальной длиной шага, на котором заново вычисляется траектория движения
- минимально допустимой относительной ошибкой решения уравнения движения
- максимально допустимой относительной ошибкой решения уравнения движения

```
G4FieldManager * globalFieldManager;

G4TransportationManager *transportMgr=
    G4TransportationManager::GetTransportationManager();

globalFieldManager = transportMgr->GetFieldManager();
                    // Relative accuracy values:
G4double minEps= 1.0e-5;  //   Minimum error value for smallest steps
G4double maxEps= 1.0e-4;  //   Maximum error value for largest steps

globalFieldManager->SetMinimumEpsilonStep( minEps );
globalFieldManager->SetMaximumEpsilonStep( maxEps );
globalFieldManager->SetDeltaOneStep( 0.5e-3 * mm );  // 0.5 um
```

Описание сложных полей

- Необходимо создать свой класс – наследник класса G4Field (в частных случаях только магнитного или электрического полей – соответственно G4MagneticField или G4ElectricField)
- Описать в нем методы

void GetFieldValue(const double Point[4], double *pField);

аргументы: Point[3] – координаты точки в пространстве и время

pField - возвращаемый массив параметров поля в данной точке

G4bool DoesFieldChangeEnergy();

- В случае, если поле отличается от простого электрического или магнитного, надо дополнительно описать класс - наследник класса G4Mag_EqRh, в котором задать уравнение движения частицы в поле