

Środowisko programistyczne GEANT4

Leszek Adamczyk

Wydział Fizyki i Informatyki Stosowanej
Akademia Górniczo-Hutnicza

Wykłady w semestrze zimowym 2015/2016

Materiały

- GEANT4 umożliwia używanie następujących materiałów:
 - pierwiastków, związków chemicznych, mieszanin pierwiastków i/lub związków chemicznych;
 - gazów, cieczy i ciał stałych;
 - o różnych gęstościach, temperaturach i ciśnieniach;
- Ogólny schemat definicji materiałów w GEANT4:
 - **pierwiastek** jest instancją klasy **G4Element**;
 - **związki** definiujemy przypisując kilka **pierwiastków** do instancji klasy **G4Material**;
 - **mieszanki** definiujemy przypisując kilka **związków** i/lub **pierwiastków** do instancji klasy **G4Material**;
 - instancja klasy **G4Material** może być zdefiniowana za pomocą **jednego pierwiastka**
 - można tworzyć własne **pierwiastki** z izotopów (instancji klasy **G4Isotope**)

Materiały, pierwiastki, izotopy

- klasy **G4Isotope** i **G4Element** określają własności jądra atomowego i atomu:
 - należy określić ich: liczbę i masę atomową;
 - można dodatkowo określić: liczbę masową, energie elektronów na orbitach, przekroje czynne na atom,...
- klasa **G4Material** opisuje własności makroskopowe materii
 - należy określić gęstość;
 - można określić: temperaturę, ciśnienie, stan skupienia, długość radiacyjną
- materiały muszą być zdefiniowane w klasie wyprowadzonej z klasy **G4VUserDetectorConstruction**

Materiały: Pojedynczy pierwiastek

```
G4double a, z, density;  
new G4Material(  "Titanium",      // nazwa  
                z=22.,           // liczba masowa  
                a=47.90*g/mole,   // masa atomowa  
                density=4.540*g/cm3); // gęstość
```

Materiały: Związki chemiczne

```
G4int ncomp,natoms;
G4string symbol;

G4Material * SiO2 =
new G4Material( "Quartz",           // nazwa
               density=2.65*g/cm3, // gęstość
               ncomp=2);           // ilość komponentów

G4Element * elSi =
new G4Element( "Silicon",          // nazwa
              symbol="Si",         // symbol chem.
              z=14.,               // liczba masowa
              a=28.09*g/mole);     // masa atomowa
G4Element * elO = .....

SiO2->AddElement(elSi, natoms=1);
SiO2->AddElement(elO, natoms=2);
```

Materiały: Mieszanki pierwiastków i/lub materiałów

```
G4Material * Aero =  
new G4Material( "Aerogel",           // nazwa  
               density=0.2*g/cm3,   // gęstość  
               ncomp=3);           // ilość komp.  
  
G4Element * elC = .....           // węgiel  
G4Material * SiO2 = .....         // kwarc  
G4Material * H2O = .....          // woda  
  
G4double fracMass;  
Aero->AddElement(elC,  fracMass= 0.1*perCent);  
Aero->AddMaterial(H2O,  fracMass=37.4*perCent);  
Aero->AddMaterial(SiO2,fracMass=62.5*perCent);
```

Materiały: Pierwiastek o nienaturalnym składzie izotopowym

```
G4Element * elDH =
new G4Element( "DH",          // nazwa
              symbol="H",    // symbol chem.
              ncomp=2);      // ilość komp.

G4Isotope * isoH1 =
new G4Isotope( "H1",        // nazwa
              iz = 1,       // liczba atomowa
              ia = 1,       // liczba masowa
              a=1.008*g/mole); // masa atomowa

G4Isotope * isoH2 =
new G4Isotope( "H2",        // nazwa
              iz = 1,       // liczba atomowa
              ia = 2,       // liczba masowa
              a=2.014*g/mole); // masa atomowa

G4double abund;
elDH->AddIsotope(isoH1, abund= 50.*perCent);
elDH->AddIsotope(isoH2, abund= 50.*perCent);
```

Materiały: Stan skupienia

- Materiał o gęstości $< 10\text{mg}/\text{cm}^3$ domyślnie jest gazem;
- często zachodzi konieczność określenia temperatury i ciśnienia gazu.

```
G4Material * CO2 =  
new G4Material("CarbonicGas",           // nazwa  
              density=27*mg/cm3,        // gęstość  
              ncomp=2,                  // ilość komponentów  
              kStateGas,                // stan gazowy  
              temp=325*kelvin,          // temperatura  
              pressure=50.0*atmosphere); // ciśnienie
```

- `kStateSolid` - ciało stałe
- `kStateLiquid` - płyn

Materiały: Baza materiałów NIST

- GEANT4 ma wbudowaną bazę materiałów NIST;
- baza zawiera definicje wszystkich pierwiastków o naturalnym składzie izotopowym;
- oraz szeregu materiałów z nich skonstruowanych; Aby korzystać z bazy danych należy uzyskać wskaźnik do menedżera bazy:

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

użyć którejs z metody:

- `FindOrBuildElement("O");`
- `FindOrBuildElement(G4int z = 8);`
- `FindOrBuildMaterial("Nist_name");`
- `ConstructNewMaterial(...)`
tworzy związki chem. i mieszanki materiałów dostępnych w bazie
- `ConstructNewGasMaterial(...)`
zmienia temperaturę i ciśnienie gazu zdefiniowanego w bazie

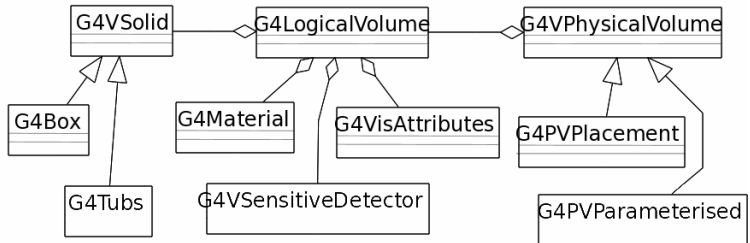
Materiały: Przydatne funkcje

- `/material/nist/printElement`
- `/material/nist/printMaterial`
- `G4cout << wskaźnik_do_materiału;`
- `G4cout << *(G4Material::GetMaterialTable()) << endl;`

Geometria detektora

Trzy poziomy definicji geomterii detektora

- **G4VSolid** - kształt, rozmiar
- **G4LogicalVolume** - materiał, wizualizacja
- **G4VPhysicalVolume** - umiejscowienie



Geometria: Typowy schemat

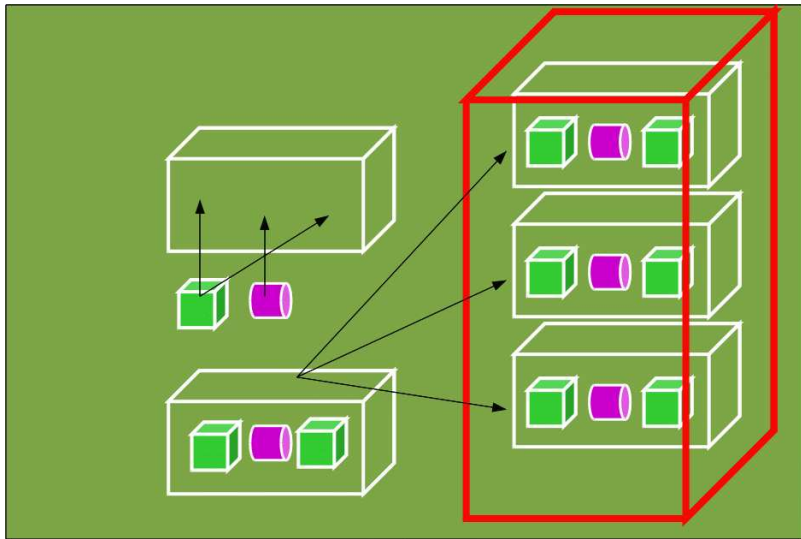
```
// World volume
G4Box* worldSolid =
    new G4Box("World_Solid",          // Name
              2.0*m, 2.0*m, 2.0*m);   // Half lengths

fpWorldLogical =
    new G4LogicalVolume(worldSolid,    // Solid
                        air,           // Material
                        "World_Logical"); // Name

fpWorldPhysical =
    new G4PVPlacement(0,                // Rotation matrix pointer
                      G4ThreeVector(), // Translation vector
                      fpWorldLogical,   // Logical volume
                      "World_Physical", // Name
                      0,                 // Mother volume
                      false,             // Unused boolean parameter
                      0);                // Copy number
```

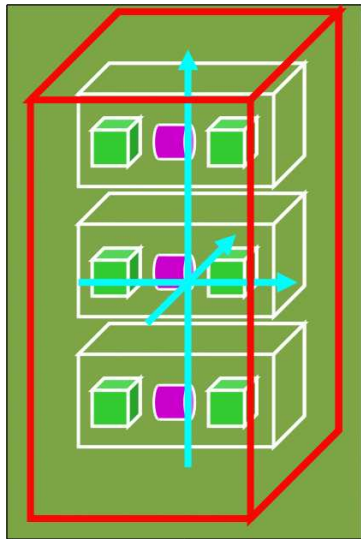
- każdy obszar jest umieszczany wewnątrz swojego obszaru matki;
- pozycja i orientacja obszaru określana jest względem układu współrzędnych matki
- obszar musi być całkowicie zawarty wewnątrz obszaru matki.

Geometria detektora



Geometria detektora

- Obszar logiczny może zawierać wiele obszarów fizycznych;
- Obszar logiczny można umieścić w wielu różnych miejscach(wraz z całą zawartością);
- Obszar **world** może być tylko jeden i musi zawierać całkowicie pozostałe obszary;
- Obszar **world** definiuje **globalny układ współrzędnych**;
- Pozycje czastek, depozytów energii, ..., podawane są w globalnym układzie współrzędnych.



Geometria: G4VUserDetectorConstruction

- Definicje materiałów, geometrii, wizualizacji powinny znajdować się w klasie wyprowadzonej z klasy **G4VUserDetectorConstruction**;
- Użytkownik musi zaimplementować czysto wirtualną metodę **Construct()**
- Opis klasy **G4VUserDetectorConstruction**

This is the abstract base class of the user's mandatory initialization class for detector setup. It has only one pure virtual method `Construct()` which is invoked by `G4RunManager` when its `Initialize()` method is invoked. The `Construct()` method must return the `G4VPhysicalVolume` pointer which represents the world volume.

- Program główny `My.CC`

```
runManager->SetUserInitialization(new MyDetectorConstruction);  
runManager->Initialize();
```

- Kod możemy umieścić w konstruktorze klasy **MyDetectorConstruction** lub w metodzie **MyDetectorConstruction::Construct()**

Geometria: MyDetectorConstruction.hh

```
#ifndef MYDETECTORCONSTRUCTION_HH
#define MYDETECTORCONSTRUCTION_HH

#include "G4UserDetectorConstruction.hh"
#include "globals.hh"

class G4LogicalVolume;
class G4VPhysicalVolume;

class MyDetectorConstruction : public G4UserDetectorConstruction {
public:

    // Constructor
    MyDetectorConstruction();

    // Destructor
    virtual ~MyDetectorConstruction();

    // Method
    virtual G4VPhysicalVolume* Construct();

private:

    // Helper methods
    void DefineMaterials();
    void SetupGeometry();

    // World logical and physical volumes
    G4LogicalVolume* fpWorldLogical;
    G4VPhysicalVolume* fpWorldPhysical;
};
#endif
```


Geometria: MyDetectorConstruction.cc

```
MyDetectorConstruction::MyDetectorConstruction()  
    :fpWorldLogical(0),fpWorldPhysical(0)  
{.....}  
  
G4VPhysicalVolume* MyDetectorConstruction::Construct()  
{  
    // Material Definition  
    DefineMaterials();  
  
    // Geometry Definition  
    SetupGeometry();  
  
    // Return world volume  
    return fpWorldPhysical;  
}  
  
void MyDetectorConstruction::DefineMaterials()  
{.....}  
  
void MyDetectorConstruction::SetupGeometry()  
{.....}
```

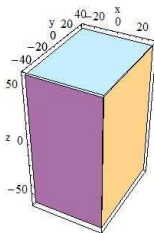
Geometria: Kształt i rozmiar (Bryły)

- Wszystkie bryły dostępne w GEANT4 są wyprowadzone z klasy **G4VSolid**.
- Klasa bazowa wymaga implementacji metod potrzebnych do:
 - Obliczenia odległości brzegu bryły do dowolnego punktu
DistanceToOut, DistanceToIn
 - Obliczenia wektora normalnego do powierzchni bryły w dowolnym punkcie
SurfaceNormal
 - Sprawdzenia czy dowolny punkt znajduje się wewnątrz bryły
Inside
 - Obliczania objętości, pola powierzchni, ...
GetSurfaceArea, GetCubicVolume, ...
- Użytkownik może definiować własne bryły.

Geometria: Bryły elementarne

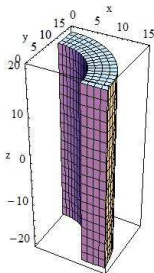
Prostopadłościan

```
G4Box(const G4String& Name, // name
      G4double X, // half length in X
      G4double Y, // half length in Y
      G4double Z) // half length in Z
```



Wycinek powłoki cylindrycznej

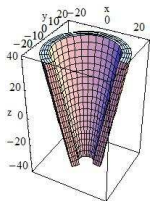
```
G4Tubs(const G4String& Name, // name
      G4double RMin, // inner radius
      G4double RMax, // outer radius
      G4double Dz, // half length in Z
      G4double SPhi, // the starting phi angle
      G4double DPhi) // the angle of the segment
```



Geometria: Pozostałe bryły elementarne

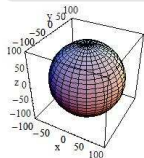
G4Cons

Powłoka stożkowa



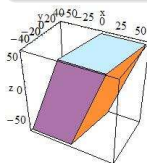
G4Orb

Pełna kula



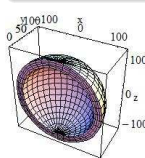
G4Para

Graniastosłup



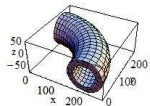
G4Sphere

Powłoka sferyczna



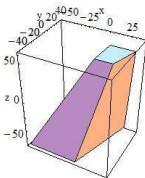
G4Torus

Torus



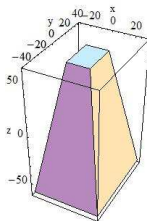
G4Trap

Ostrosłup



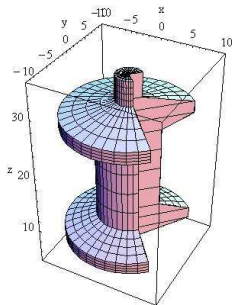
G4Trd

Ostrosłup foremny



Opis wszystkich brył elementarnych w rozdziale 4.1.2 Przewodnika dla twórców aplikacji.

Geometria: Bryły specjalne

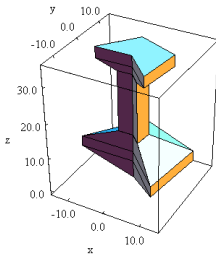


Wycinek wielostożka

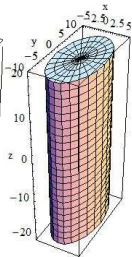
```
G4Polycone(const G4String& Name,           // name
            G4double   hiStart,           // starting phi angle
            G4double   hiTotal,          // total phi angle
            G4int      numRZ,             // number of corners in R-Z space
            const G4double r[],           // r coordinate of corners
            const G4double z[])           // z coordinate of corners
```

Geometria: Pozostałe bryły specjalne

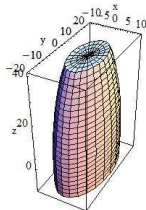
G4Polyhedra



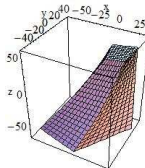
G4EllipticalTube



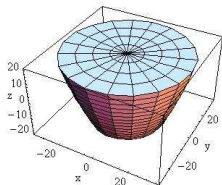
G4Ellipsoid



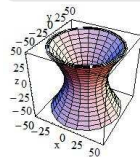
G4TwistedTrap



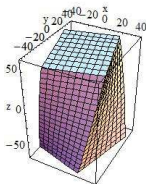
G4Paraboloid



G4Hype



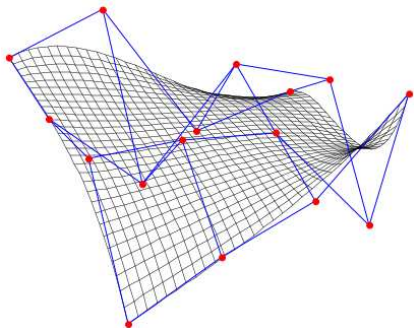
G4TwistedBox



Opis wszystkich brył specjalnych w rozdziale 4.1.2 Przewodnika dla twórców aplikacji.

Geometria: Bryły BREP (Boundary REPresentented solid)

- można definiować bryłę zadając wszystkie płaszczyzny które ją ograniczają;
- płaszczyzny mogą być: płaskie, zakrzywione, płyty Beziera.



Geometria: Operacje logiczne na bryłach

- można definiować bryły poprzez operacje logiczne na istniejących już bryłach;
- **G4UnionSolid** - suma brył;
- **G4SubtractionSolid** - różnica brył;
- **G4IntersectionSolid** - część wspólna brył;




```
G4LogicalVolume( G4VSolid*           Solid,  
                 G4Material*         Material,  
                 const G4String&     Name,  
                 G4FieldManager*     FieldMgr=0,  
                 G4VSensitiveDetector* SDetector=0,  
                 G4UserLimits*       ULimits=0);
```

- obszar logiczny posiada pełną informację o obszarze oprócz jego położenia i orientacji w przestrzeni;
- określa kształt i rozmiar - [G4VSolid](#);
- materiał - [G4Material](#);
- pole magnetyczne - [G4FieldManager](#);
- przypisuje atrybut obszaru czułego detektora - [G4VSensitiveDetector](#);
- warunki produkcji cząstek wtórnych - [G4UserLimits](#);
- atrybutów wizualizacji [SetVisAttributes](#)
- pozycje i orientacje obszarów córek.

Obszary fizyczne dzielimy na:

- jednokrotne: obszar umieszczony jednokrotnie
jeden fizyczny obszar = jeden obszar rzeczywisty
- wielokrotne: obszar umieszczany wielokrotnie
jeden fizyczny obszar = wiele obszarów rzeczywistych
Obszary wielokrotne ze względu na sposób powielania dzielimy na:
 - parametryzowane numerem kopii;
 - powielane wzdłuż jednej osi

Obszar "matka" może zawierać:

- albo wiele obszarów jednokrotnych;
- albo jeden obszar wielokrotny;

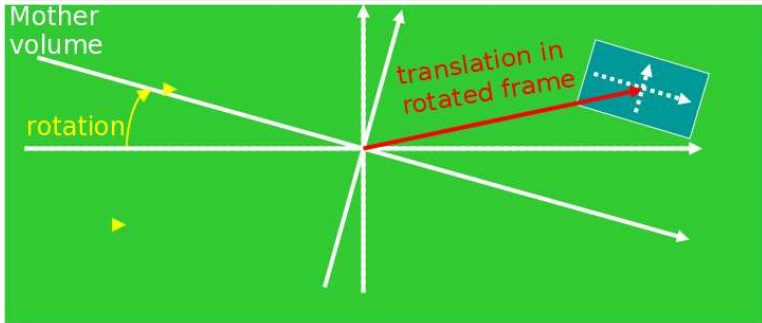
Geometria: Obszary fizyczne : Przegląd

- **G4Placement**
reprezentuje obszar jednokrotny;
- **G4Parameterised**
reprezentuje obszar wielokrotny parametryzowany numerem kopii
 - kształt, rozmiar, materiał,... mogą być parametryzowane numerem kopii;
 - powielane obszary mogą zawierać obszary "wnuczki" o ile "wnuczki" mają identyczne rozmiary i kształty;
 - wymagana jest implementacja klasy **G4PVParameterisation**
 - implementacja klasy **G4PVNestedParameterisation** umożliwia parametryzację materiału,... także numerem kopii swojego "przodka"

- **G4PVReplica** (obszar wielokrotny)
obszary "córk" mające ten sam kształt ustawiane są wzdłuż jednej osi i wypełniają cały obszar "matki" bez żadnych przerw;
- **G4PVDivision** (obszar wielokrotny)
tak jak G4PVReplica ale dopuszcza istnienie przerwy między brzegiem obszaru "matki" a najbardziej zewnętrznymi "córkami", brak przerw między "córkami"
- **G4ReflectionFactory** (para obszarów)
umożliwia umieszczenie obszaru i jego odbicia;
- **G4AssemblyVolume** (wiele obszarów)
umożliwia umieszczenie jednokrotnych obszarów połączonych w grupę

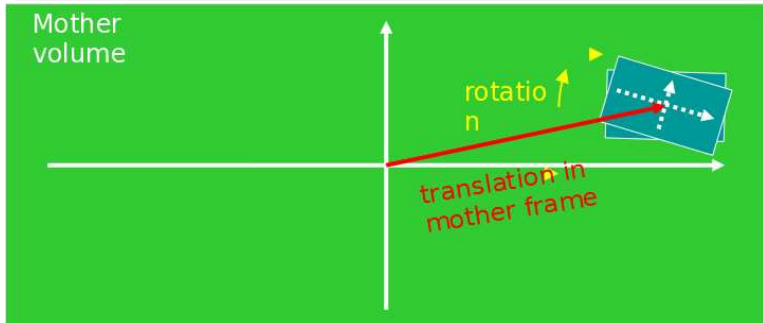
Geometria: G4PVPlacement I

```
G4PVPlacement( G4RotationMatrix* Rot, // obrót układu matki
               const G4ThreeVector& trans, // pozycja w obróconym układzie
               G4LogicalVolume* CurrentLogical,
               const G4String& Name,
               G4LogicalVolume* MotherLogical,
               G4bool Many, // false
               G4int CopyNo, // dowolna unikalna liczba całk.
               G4bool SurfChk=false )
```



Geometria: G4PVPlacement II

```
Tra = G4Transform3D (  
    G4RotationMatrix &pRot,    // obrót układu "córki"  
    const G4ThreeVector &trans); // pozycja w układzie "matki"  
  
G4PVPlacement( G4Transform3D Tra , // transformacja córki  
    .....
```

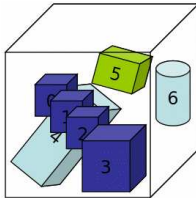


G4PVParameterised

```
G4PVParameterised(  
    const G4String&           Name,  
    G4LogicalVolume*         DLogical,  
    G4LogicalVolume*         MLogical,  
    const EAxis               Axis,  
    const G4int               n,  
    G4VPVParameterisation*   Param)
```

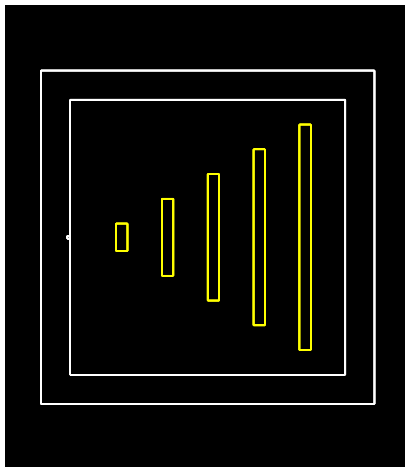
- Umieszcza obszar logiczny **DLogical** wewnątrz obszaru "matki" **MLogical** **n** razy używając parametryzacji **Param**;
- **Axis** sugeruje wzdłuż której osi będą powielane obszary: **kXAxis**, **kYAxis**, **kZAxis**, **kUndefined**

G4PVParameterised



- Użytkownik musi zaimplementować klasę wyprowadzoną z klasy **G4VVPParameterisation** definiując następujące wielkości w funkcji **numeru kopii**:
 - położenie i orientację **ComputeTransformation**
 - (opcjonalnie) rozmiar **ComputeDimensions**
 - (opcjonalnie) kształt **ComputeSolid**
 - (opcjonalnie) materiał **ComputeMaterial**
- wszystkie kopie muszą zawierać się w obszarze "matce" i nie nachodzić na siebie
- nie wszystkie bryły można w ten sposób powielić

Geometria: G4PVPParameterised przykład exampleN02



Układ pięciu komór śladowych o rosnących rozmiarach

G4PVParameterised: ExN02DetectorConstruction.cc

```
solidChamber = new G4Box(  
    "chamber", 100*cm, 100*cm, 10*cm);
```

```
logicChamber = new G4LogicalVolume(  
    solidChamber, ChamberMater, "Chamber", 0, 0, 0);
```

```
chamberParam = new ExN02ChamberParameterisation(  
    NbOfChambers,           // NoChambers  
    firstPosition,         // Z of center of first  
    ChamberSpacing,       // Z spacing of centers  
    ChamberWidth,         // Width Chamber  
    firstLength,          // lengthInitial  
    lastLength);          // lengthFinal
```

```
physiChamber = new G4PVParameterised(  
    "Chamber",             // their name  
    logicChamber,         // their logical volume  
    LogicTracker,         // Mother logical volume  
    kZAxis,               // Are placed along this axis  
    NbOfChambers,        // Number of chambers  
    chamberParam);       // The parametrisation
```

G4PVParameterised: ExN02ChamberParameterisation.hh

```
.....  
class ExN02ChamberParameterisation : public  
    G4VPVParameterisation{
```

```
public:  
    ExN02ChamberParameterisation(  
        G4int    NoChambers,  
        G4double startZ,  
        G4double spacing,  
        G4double widthChamber,  
        G4double lengthInitial,  
        G4double lengthFinal );
```

```
void ComputeTransformation (  
    const G4int copyNo,  
    G4VPhysicalVolume* physVol) const;
```

```
void ComputedDimensions (  
    G4Box & trackerLayer,  
    const G4int copyNo,  
    const G4VPhysicalVolume* physVol) const;
```

```
.....
```

G4PVParameterised: ExN02ChamberParameterisation.cc

```
ExN02ChamberParameterisation::  
  ExN02ChamberParameterisation(  
    G4int      NoChambers,  
    G4double  startZ,  
    G4double  spacingZ,  
    G4double  widthChamber,  
    G4double  lengthInitial,  
    G4double  lengthFinal )  
{  
  fNoChambers = NoChambers;  
  fStartZ     = startZ;  
  fHalfWidth  = widthChamber*0.5;  
  fSpacing    = spacingZ;  
  fHalfLengthFirst = 0.5 * lengthInitial;  
  .....  
}
```

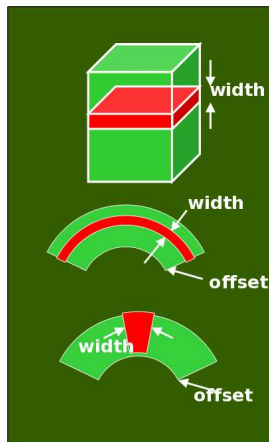
G4PVParameterised: ExN02ChamberParameterisation.cc

```
void ExN02ChamberParameterisation::ComputeTransformation(  
    const G4int copyNo,  
    G4VPhysicalVolume* physVol) const  
{  
    G4double      Zposition= fStartZ + (copyNo+1) * fSpacing;  
    G4ThreeVector origin(0,0,Zposition);  
    physVol->SetTranslation(origin);  
    physVol->SetRotation(0);  
}
```

```
void ExN02ChamberParameterisation::ComputeDimensions(  
    G4Box& trackerChamber,  
    const G4int copyNo,  
    const G4VPhysicalVolume*) const  
{  
    G4double halfLength= fHalfLengthFirst+copyNo*fHalfLengthIncr;  
    trackerChamber.SetXHalfLength(halfLength);  
    trackerChamber.SetYHalfLength(halfLength);  
    trackerChamber.SetZHalfLength(fHalfWidth);  
}
```

G4PVReplica

- Obszar "matki" musi być **całkowicie** wypełniony kopiami o **takich samych** grubościach i kształtach.
- Powielanie może odbywać się wzdłuż:
 - osi (X,Y,Z), plastry są prostopadłe do osi a lokalny układ wsp. związany jest ze środkiem plastra;
 - osi radialnej (R), powielane wycinki stożka/cylindra są współosiowe i nieobrócone a lokalne układy wsp. są takie same jak obszaru "matki"
 - osi kąta azymutalnego (Phi), powielane wycinki stożka/cylindra są współosiowe i obracane o stały kąt a lokalny układ wsp. jest obrócony względem układu "matki" w taki sposób, że oś X dzieli na pół każdy klin;



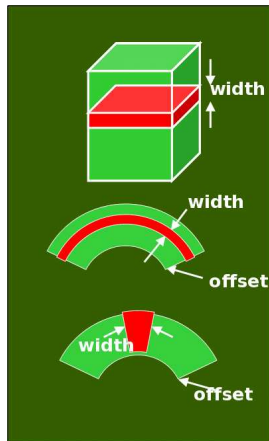
G4PVReplica

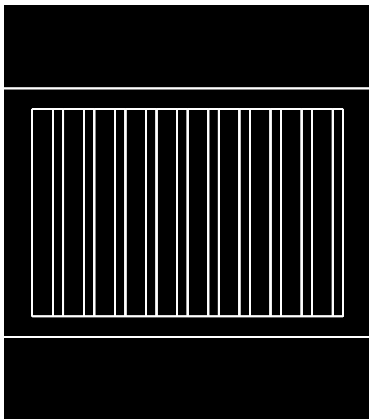
```
G4PVReplica( const G4String&      Name,  
             G4LogicalVolume*    CurrentLogical,  
             G4LogicalVolume*    MotherLogical,  
             const EAxis         Axis,  
             const G4int         nReplicas,  
             const G4double      width,  
             const G4double      offset=0 )  
}
```

- Obszar "matka" może sam być klasy **G4PVReplica**;
- Obszar klasy **G4Placement** może znajdować się w powielanym obszarze o ile powielanie nie następuje wzdłuż osi radialnej;
- Obszar klasy **G4Parameterised** nie może znajdować się w powielanym obszarze

G4PVReplica

- Osie kartezjańskie $pAxis=kXaxis$ (YZ) $offset$ musi mieć wartość 0
- Oś radialna $pAxis=kRaxis$ $offset$ musi być równy wewnętrznemu promieniowi obszaru "matki"
- Oś kąta azymutalnego $pAxis=kPhi$ $offset$ musi być równy początkowemu kątowi obszaru "matki"





Kalorymetr zbudowany z 10 kopii warstwy absorbera oraz warstwy czynnej

G4PVReplica: ExN02DetectorConstruction.cc

```
solidLayer = new G4Box(  
    "Layer"                                     //its name  
    LayerThickness/2,CalorSizeYZ/2,CalorSizeYZ/2); //size
```

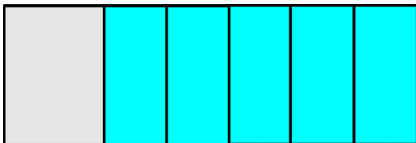
```
logicLayer = new G4LogicalVolume(  
    solidLayer,                                // its solid  
    defaultMaterial,                          // its material  
    "Layer");                                  // its name
```

```
physiLayer = new G4PVReplica(  
    "Layer",                                   // its name  
    logicLayer,                               // its logical volume  
    logicCalor,                               // its mother  
    kXAxis,                                   // axis of replication  
    NbOfLayers,                              // number of replica  
    LayerThickness);                         // width of replica
```

```
physiAbsorber = new G4PVPlacement(0,         // no rotation  
    G4ThreeVector(-GapThickness/2,0.,0.), // its position  
    logicAbsorber,                          // its logical volume  
    AbsorberMaterial->GetName(),            // its name  
    logicLayer,                              // its mother  
    false,                                   // no boolean operat  
    0);                                      // copy number
```

G4PVDivision

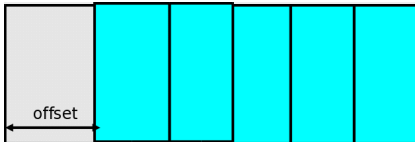
- $G4PVDivision = G4PVR Replica + G4PVParameterised$;
- Umożliwia podział obszaru matki na szereg identycznych kopii wzdłuż jakiejś z osi;
- **G4PVParameterisation** jest automatycznie generowana na podstawie parametrów konstruktora klasy **G4PVDivision**.
- Dopuszcza istnienie przerw między skrajnymi córkami a granicami obszaru matki.



G4PVDivision

```
G4PVDivision( const G4String& Name,  
             G4LogicalVolume* CurrentLogical,  
             G4LogicalVolume* MotherLogical,  
             const EAxis Axis,  
  
             const G4int nDivisions,  
             const G4double offset )
```

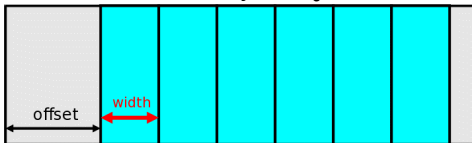
Szerokość kopii obliczona automatycznie tak aby wypełnić obszar "matki" do końca.



G4PVDivision

```
G4PVDivision( const G4String& Name,  
              G4LogicalVolume* CurrentLogical,  
              G4LogicalVolume* MotherLogical,  
              const EAxis Axis,  
  
              const G4double width,  
              const G4double offset )
```

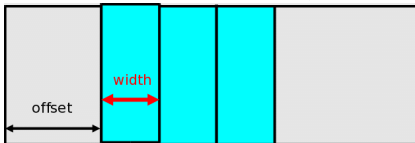
Ilość kopii obliczona automatycznie tak aby wypełnić obszar "matki" tak bardzo jak się da.



G4PVDivision

```
G4PVDivision( const G4String& Name,  
              G4LogicalVolume* CurrentLogical,  
              G4LogicalVolume* MotherLogical,  
              const EAxis Axis,  
  
              const G4int nDivisions,  
              const G4double width,  
              const G4double offset )
```

Ustalona zarówno ilość jak i szerokość kopii



G4PVDivision

Obecnie można podzielić następujące bryły:

G4Box Axis = kXAxis (Y,Z)

G4Tubs Axiz = kRho (Phi,Z)

G4Cons Axiz = kRho (Phi,Z)

G4Trd Axis = kXAxis (Y,Z)

G4Para Axis = kXAxis (Y,Z)

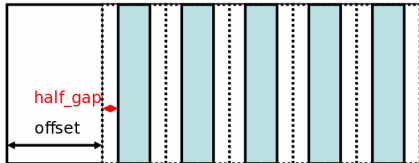
Polycone Axiz = kRho (Phi,Z)

Polyhedra Axiz = kRho (Phi,Z)

Ostatnie dwie bryły można dzielić wzdłuż osi Z **tylko** na taką ilość warstw z jakich same się składają.

G4PVReplicatedSlices

- Rozszerzenie klasy G4PVDivision;
- Umożliwia pozostawienie przerw między kolejnymi córkami;



G4PVNestedParameterisation

- **NestedParameterisation** umożliwia uzależnienie materiału córki nie tylko od swojego numeru kopii ale również od numeru kopii przodka bądź przodków w przypadku gdy obszary przodków są obszarami powielanymi: (Parameterised, Replica, Division, ReplicatedSlices).
- Wymaga implementacji trzech dodatkowych funkcji:
 - **ComputeMaterials()** - zwraca wskaźnik do materiału w zależności od numeru kopii własnej oraz przodków
 - **GetNumberOfMaterials()** - zwraca całkowitą liczbę materiałów jakie mogą być zwrócone przez ComputeMaterials()
 - **GetMaterial(G4int i)** – zwraca wskaźnik do i-tego materiału