ジオメトリの組込み

演習パッケージ：　P02_Geometry

Geant4 10.3.P3準拠

Geant4　HEP/Space/Medicine 講習会資料

大学共同利用機関法人
高エネルギー加速器研究機構

本資料に関する注意

- 本資料の知的所有権は、高エネルギー加速器研究機構およびGeant4 collaborationが有します
- 以下のすべての条件を満たす場合に限り無料で利用することを許諾します
  - 学校、大学、公的研究機関等における教育および非軍事目的の研究開発のための利用であること
    - Geant4の開発者はいかなる軍事関連目的へのGeant4の利用を拒否します
  - このページを含むすべてのページをオリジナルのまま利用すること
    - 一部を抜き出して配布したり利用してはいけません
  - 誤字や間違いと疑われる点があれば報告する義務を負うこと
- 商業的な目的での利用、出版、電子ファイルの公開は許可なく行えません
- 本資料の最新版は以下からダウンロード可能です
  - http://geant4.kek.jp/lecture/
- 本資料に関する問い合わせ先は以下です
  - Email: lecture-feedback@geant4.kek.jp

演習の目標

1. BGO検出器ジオメトリの実装をすることで、最も基礎的なジオメトリ記述方法を学ぶ
2. pixel検出器ジオメトリの実装をすることで、Replicaの使用方法を学ぶ
3. これらの演習を通して、Geant4のジオメトリ記述のキーポイントである Logical Volumeの使い方を学ぶ

演習の準備

P02_Geometryプログラムのコピーとファイル構造の確認

課題:0　演習プログラムとして提供されているP02_Geometryの全体をユーザのワークディレクトリにコピーし、そのファイル構造を確認する

[注意]
1. コマンド入力には必ずtcsh補完機能を使う
2. スライドのコマンドを「コピペ」するのは危険

1) 演習プログラム全体を自分のワークデレクトリにコピー

```
$ cd ~/Geant4Tutorial20171129
$ cd UserWorkDir
$ cp -r ../TutorialMaterials/P02_Geometry .
```

ユーザ作業root dir

先ず、演習のルート・ディレクトリに行く

P02_Geometryの後ろに"/"をつけないこと

2) 演習プログラムのファイル構造の確認

```
$ ls P02_Geometry
source/    util/
$ ls P02_Geometry/source
Application_Main.cc    CMakeLists.txt    include/    src/
```

mainプログラム [注1]　cmakeビルドファイル　ヘッダファイル　ソースファイル

[注1] mainプログラムは前の演習 (P01_FirstStep_Vis) で使用したものと同じ
以後の演習では常に、このmainを使う

```
$ ls P02_Geometry/source/src
Geometry.cc          PrimaryGenerator.cc      UserActionInitialization.cc
```

ジオメトリ定義ファイル　入射粒子定義ファイル　ユーザ・アクション登録用ファイル
（.ccに対応する.hhはincludeデレクトリにある）[注2]

[注2] Geometry以外のプログラムは前の演習 (P01_FirstStep_vis) で使用したものと同じ
Geometry.ccはこの演習用のものを使う
Geometry.hhは前の演習用と同じ

```
$ ls P02_Geometry/util
G4Codes/          Help/          Macros/
```

演習で使うC++ファイル　救済用スクリプト　アプリ実行用Geant4マクロ

P02_Geometry: BGO_One

円筒型BGO検出器ジオメトリ

P02_Geometry: BGO_Oneプログラムの概要

■ 演習プログラムの目的

- P01_FirstStep_Visで使用した「水の直方体」ジオメトリが「円柱のBGO検出器」に変更されたプログラムをビルドして実行する

  ［注］この演習では、用意されているプログラムを編集することなく、そのまま使う

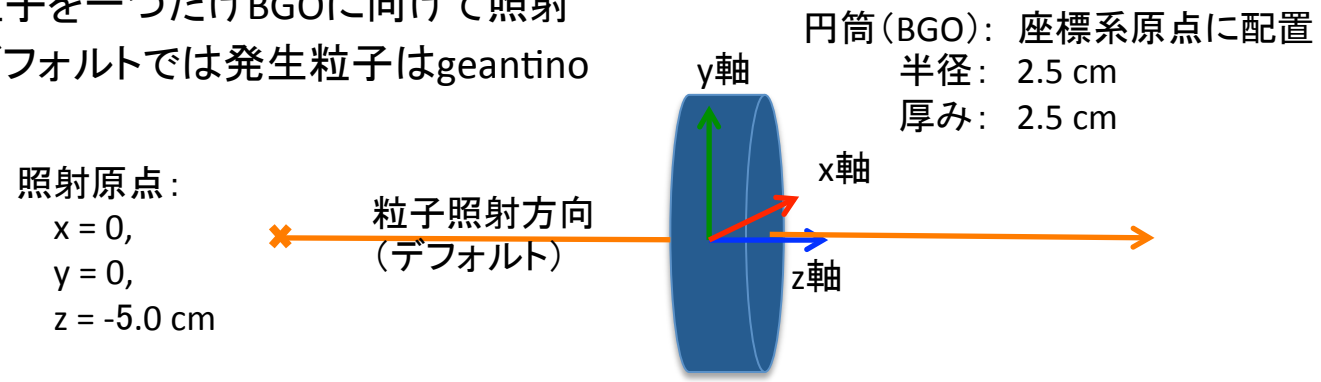
- ジオメトリ定義の基本的な手続きを学ぶ

■ プログラムの構成

- mainプログラム： P01_FirstStep_Visと同一
- ジオメトリ： 円柱BGO (下図参照) ： P01_FirstStep_Visでは水の直方体であった
- PhysicsList : P01_FirstStep_Visと同一
- PrimaryGenerator: P01_FirstStep_Visと同一

■ 組み込まれているジオメトリと粒子発生機能

- 粒子を一つだけBGOに向けて照射
- デフォルトでは発生粒子はgeantino



P02_Geometry: BGO_Oneのビルド

課題：1　P02_Geometry: BGO_Oneをビルドして実行ファイルを作成する

(ビルドに必要な全ファイルはコピーしたP02_Geometryにすでに用意されている)

1) P02_Geometryのディレクトリに移行する

```
$ cd ~/Geant4Tutorial20171129/UserWorkDir/P02_Geometry
```

CBDir (Current Base Directory)

2) ビルドするための作業デレクトリを作成

```
$ mkdir build
$ ls
build/  source/  util/
```

ビルド作業のデレクトリ名は習慣的にbuildとする

3) buildデレクトリでビルドを実行　[注1]

```
$ cd  build
$ cmake ../source
$ make
$ make install
```

cmakeでビルドを実行する時の慣用句

4) ビルド実行後のデレクトリ構造確認

```
$ cd  ..
$ ls
bin/        build/      source/    /util
$ ls bin
Application_Main*
```

[注1]
buildを失敗したら、CBdirに戻って、以下のスクリプトを実行すればbuildは自動完了：
./util/Help/Build_P02_BGO_One.sh

P02_Geometry: BGO_One(Application_Main)の実行

課題：2　Application_Main (BGO_One)を実行する

1) プログラム実行を行う作業デレクトリを新たに作成する

```
$ pwd
……/P02_Geometry
$ mkdir TestBench
$ ls
TestBench/ bin/        build/      source/
$ cd TestBench
$ cp ../util/Macros/* .
$ ../bin/Application_Main
```

現在デレクトリがCBDirであることを確認：
P02_Geometryでなければ、そこに移動する

作業デレクトリをつくる(名前は自由)

用意されているマクロファイルをTestBenchにコピー

TestBenchデレクトリでApplication_Main実行

2) 端末ウインドに以下のメッセージが出力され、続いてQtウインドが開く

```
*************************************************************
 Geant4 version Name: geant4-10-03-patch-03    (20-October-2017)
                      Copyright : Geant4 Collaboration
                      Reference : NIM A 506 (2003), 250-303
                            WWW : http://cern.ch/geant4
*************************************************************

<<< Geant4 Physics List simulation engine: FTFP_BERT 2.0
.....
### Adding tracking cuts for neutron  TimeCut(ns)= 10000  KinEnergyCut(MeV)= 0
Visualization Manager instantiating with verbosity "warnings (3)"...
Visualization Manager initialising...
Registering graphics systems...
.....
```

Qtウインドでアプリの動作を確認

課題：3　任意のUI コマンドを使って動作をチェックする

1) 以下のUIコマンドを先ず入力

```
/gun/particle proton
/gun/energy 60 MeV
/run/beamOn 1
```

右図はその出力
（マウスで視点を移動させ見た目を調整している）

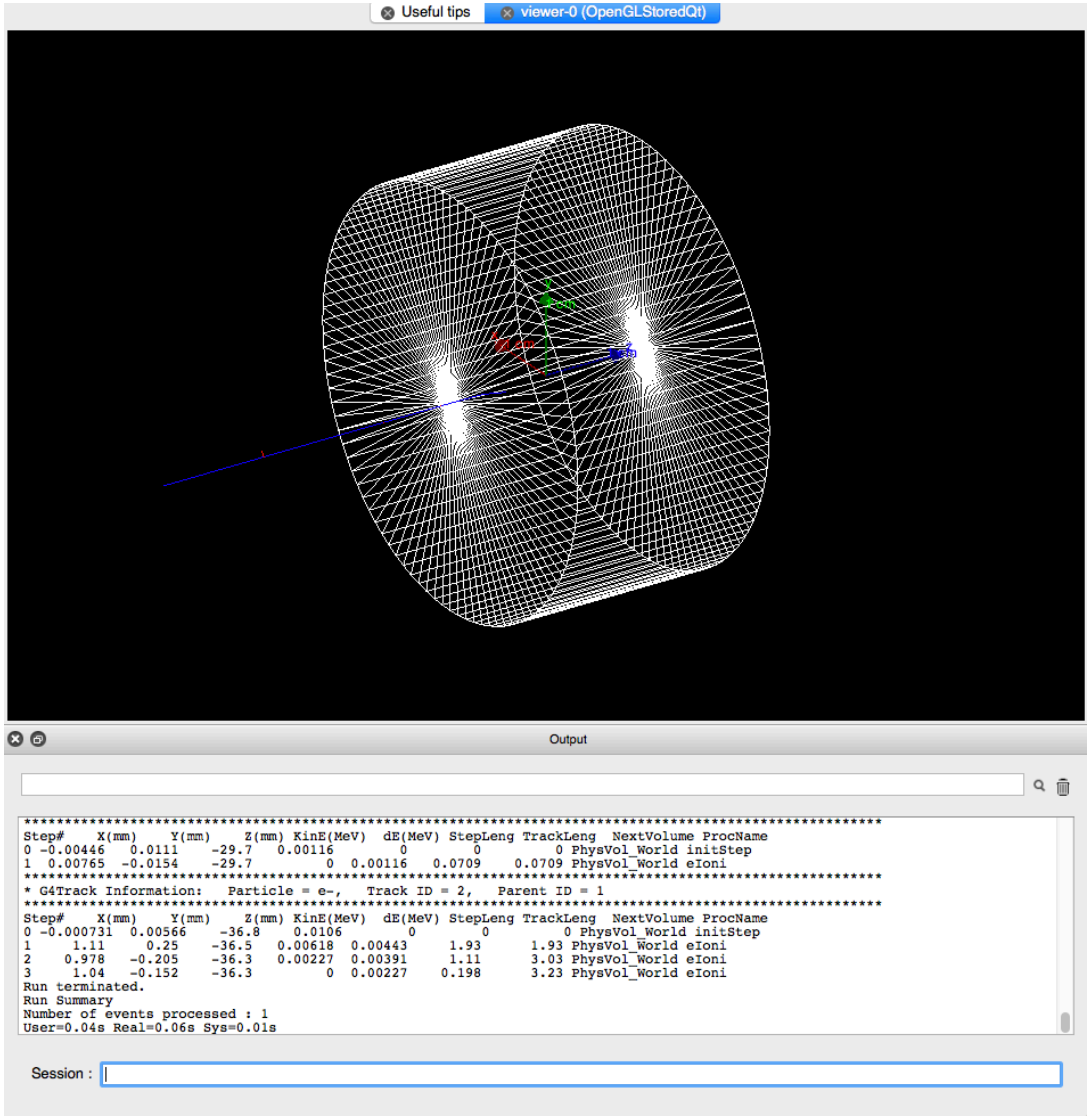
[注] UIコマンドでの設定値のチェック法

```
?/gun/energy
```

2) 上の設定で粒子の種類を変える、あるいはエネルギーを変える等してみる

3) 一旦、アプリを終了したのち、以下のマクロを編集して入射粒子の条件、照射回数を自由に変更し、アプリを再起動してみる

- GlobalSetup.mac
- primaryGeneratorSetup.mac



ジオメトリ定義の実際： Geomety.hhの内容

課題：4　P02_Geometryプログラムでのジオメトリ定義ヘッダーファイルを理解する

```
$ less ../source/include/Geometry.hh
```

```
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// Geometry.hh
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#ifndef Geometry_h
#define Geometry_h 1

#include "G4VUserDetectorConstruction.hh"
class G4VPhysicalVolume;

//-----------------------------------------------------------------------------
  class Geometry : public G4VUserDetectorConstruction
//-----------------------------------------------------------------------------
{
  public:
    Geometry();
   ~Geometry();

    G4VPhysicalVolume* Construct();
};
#endif
```

このヘッダーファイルはすべての演習プログラムで共通に使われている

➔多くの場合、ジオメトリ定義ヘッダー・ファイルはこれを流用できる

ユーザのジオメトリ定義は必ずこの仮想クラスを継承して行う

ユーザ・クラスの名前は任意

ジオメトリの実装はこの関数の中で行う

ジオメトリ定義の実際： Geomety.ccの内容

課題：5　P02_Geometry: BGO_Oneのジオメトリを理解する

```
$ less ../source/src/Geometry.cc
```

つづく

```
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// Geometry.cc
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#include "Geometry.hh"
#include "G4Box.hh"
#include "G4Tubs.hh"
#include "G4LogicalVolume.hh"
#include "G4PVPlacement.hh"
#include "G4VPhysicalVolume.hh"
#include "G4ThreeVector.hh"
#include "G4RotationMatrix.hh"
#include "G4Transform3D.hh"
#include "G4NistManager.hh"
#include "G4VisAttributes.hh"
#include "G4SystemOfUnits.hh"

//---------------------------------------------------------------------------
  Geometry::Geometry() {}
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
  Geometry::~Geometry() {}
//---------------------------------------------------------------------------

//---------------------------------------------------------------------------
  G4VPhysicalVolume* Geometry::Construct()
//---------------------------------------------------------------------------
{
// Get pointer to 'Material Manager'
   G4NistManager* materi_Man = G4NistManager::Instance();

// Define 'World Volume'
   // Define the shape of solid
   G4double leng_X_World = 2.0*m;         // X-full-length of world
   G4double leng_Y_World = 2.0*m;         // Y-full-length of world
   G4double leng_Z_World = 2.0*m;         // Z-full-length of world
   G4Box* solid_World =
     new G4Box("Solid_World", leng_X_World/2.0, leng_Y_World/2.0, leng_Z_World/2.0);

   // Define logical volume
   G4Material* materi_World = materi_Man->FindOrBuildMaterial("G4_AIR");
   G4LogicalVolume* logVol_World =
     new G4LogicalVolume(solid_World, materi_World, "LogVol_World");
   logVol_World->SetVisAttributes (G4VisAttributes::Invisible);

   // Placement of logical volume
   G4int copyNum_World = 0;               // Set ID number of world
   G4PVPlacement* physVol_World  =
     new G4PVPlacement(G4Transform3D(), "PhysVol_World", logVol_World, 0,
                       false, copyNum_World);
```

```
// Define 'BGO Detector'
   // Define the shape of solid
   G4double radius_BGO = 2.5*cm;
   G4double leng_Z_BGO = 2.5*cm;
   G4Tubs* solid_BGO = new G4Tubs("Solid_BGO", 0., radius_BGO, leng_Z_BGO/2.0, 0., 360.*deg);

   // Define logical volume
   G4Material* materi_BGO = materi_Man->FindOrBuildMaterial("G4_BGO");
   G4LogicalVolume* logVol_BGO =
     new G4LogicalVolume( solid_BGO, materi_BGO, "LogVol_BGO", 0, 0, 0 );

   // Placement of logical volume
   G4double pos_X_LogV = 0.0*cm;          // X-location LogV
   G4double pos_Y_LogV = 0.0*cm;          // Y-location LogV
   G4double pos_Z_LogV = 0.0*cm;          // Z-location LogV
   G4ThreeVector threeVect_LogV = G4ThreeVector(pos_X_LogV, pos_Y_LogV, pos_Z_LogV);
   G4RotationMatrix rotMtrx_LogV = G4RotationMatrix();
   G4Transform3D trans3D_LogV = G4Transform3D(rotMtrx_LogV, threeVect_LogV);

   G4int copyNum_LogV = 1000;             // Set ID number of LogV
   new G4PVPlacement(trans3D_LogV, "PhysVol_BGO", logVol_BGO, physVol_World,
                     false, copyNum_LogV);

// Return the physical world
   return physVol_World;
}
```

Geometry.ccで使うクラスのヘッダー・ファイルをすべてincludeする

NISTマテリアル・マネージャ作成

ワールド・ボリュウムの作成

worldのsolid作成

worldのlogical volume作成

worldのphysical volume作成

BGO検出器の作成

BGOのsolid作成

BGOのlogical volume作成

BGOのlogical volumeをworldに設置

Physical volumeに通常は直接アクセスしないので、ポインタ変数は不要

worldのphysical volumeを返す

P02_Geometry: BGO_Two

円筒型BGO検出器ジオメトリ： Logical Volumeの複数回使用

P02_Geometry: BGO_Twoプログラムの概要

■ 演習プログラムの目的

- 前演習のP02_Geometry: BGO_Oneで学んだGeometryファイルをもとに、BGO検出器を1台追加し、プログラムをビルドして実行する

  ［注］この演習では、受講者はソースコードを書き、それを動かしてみる

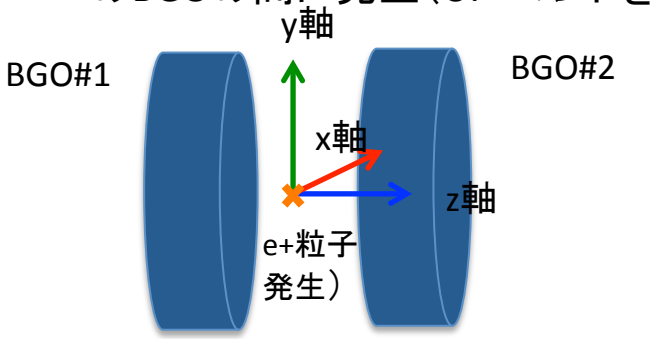
- 新たに作るプログラムはP02_Geometry: BGO_Twoとよぶ

■ プログラムの構成

- mainプログラム: 前演習と同一
- ジオメトリ: 二つの円柱BGO (下図参照)
- PhysicsList : 前演習と同一
- PrimaryGenerator: 前演習と同一

■ 組み込むジオメトリと粒子発生機能

- 静止e+粒子を二つのBGOの間に発生（UIコマンドを入れる）



円筒（BGO）:
半径: 2.5 cm
厚み: 2.5 cm
BGOの中心配置位置:
BGO#1: X=Y=0cm, Z=-2.0cm
BGO#2: X=Y=0cm, Z=+2.0cm

P02_Geometry: BGO_Twoプログラムの作成

はじめに:

P02_Geometry: BGO_Twoプログラムを作成するには:

1) プログラム作成作業はP02_Geometry: BGO_Oneで使ったファイルを変更することで行う
2) 本演習で変更が必要なファイルはGeometry.ccのみで、それ以外は全て流用する

課題:1　P02_Geometry: BGO_Twoのジオメトリを書くファイルの用意

1) 現在のGeometry.ccファイルには前課題で使ったP02_Geometry: BGO_Oneのジオメトリが書かれている。その内容をエディトして"BGO_Two"ジオメトリを作成する
2) 念のため、オリジナルのP02_Gometry: BGO_Oneを現在のGeometry.ccに再度コピーしておく
（Geometry.ccを変更した記憶がなければこのステップはスキップ可）

```
$ cd ~/Geant4Tutorial20171129/UserWorkDir/P02_Geometry
$ cd source/src
$ cp ../../util/G4Codes/Geometry.cc_P02_BGO_One  Geometry.cc
overwrite Geometry.cc? (y/n [n]) y
```

CBDir (Current Base Directory)

P02_Geomety: BGO_Twoの実装

課題：2　現在のGeometry.ccからの変更箇所を理解する ← エディット手順は次のスライド参照

［注］ 以下のソースコードは変更が完了したものを示す

```
//------------------------------------------------------------------------
  G4VPhysicalVolume* Geometry::Construct()
//------------------------------------------------------------------------
// world volume作成部は省略
// Define 'BGO Detector'   BGO検出器の作成
   // Define the shape of solid
   G4double radius_BGO = 2.5*cm;
   G4double leng_Z_BGO = 2.5*cm;
   G4Tubs* solid_BGO = new G4Tubs("Solid_BGO", 0., radius_BGO, leng_Z_BGO/2.0, 0., 360.*deg);

   // Define logical volume
   G4Material* materi_BGO = materi_Man->FindOrBuildMaterial("G4_BGO");
   G4LogicalVolume* logVol_BGO =
     new G4LogicalVolume( solid_BGO, materi_BGO, "LogVol_BGO", 0, 0, 0 );

   // Placement of logical volume - 1st BGO
   G4double pos_X_LogV = 0.0*cm;           // X-location LogV
   G4double pos_Y_LogV = 0.0*cm;           // Y-location LogV
   G4double pos_Z_LogV = -2.0*cm;          // Z-location LogV
   G4ThreeVector threeVect_LogV = G4ThreeVector(pos_X_LogV, pos_Y_LogV, pos_Z_LogV);
   G4RotationMatrix rotMtrx_LogV = G4RotationMatrix();
   G4Transform3D trans3D_LogV = G4Transform3D(rotMtrx_LogV, threeVect_LogV);

   G4int copyNum_LogV = 1000;              // Set ID number of LogV
   new G4PVPlacement(trans3D_LogV, "PhysVol_BGO", logVol_BGO, physVol_World,
                     false, copyNum_LogV);

   // Placement of logical volume - 2nd BGO
   pos_Z_LogV = +2.0*cm;                   // Z-location LogV
   threeVect_LogV = G4ThreeVector(pos_X_LogV, pos_Y_LogV, pos_Z_LogV);
   trans3D_LogV = G4Transform3D(rotMtrx_LogV, threeVect_LogV);

   copyNum_LogV = 2000;                    // Set ID number of LogV
   new G4PVPlacement(trans3D_LogV, "PhysVol_BGO", logVol_BGO, physVol_World,
                     false, copyNum_LogV);

// Return the physical world
   return physVol_World;
}
```

変更

追加

BGOのsolid作成

BGOのlogical volume作成

この部分は前演習のコードと全く同じ

1番目のBGOをworldに設置

この部分はlogical volumeのZ位置(pos_Z_LogV)の値以外は前演習のコードと同じ

上で定義されているBGOのlogical volumeを再利用して2番目のBGOを配置

BGOの中心のZ座標を
Z = + 2.0cm
となる様に配置する

P02_Geomety: BGO_Twoの実装 – つづき

課題:3　現在のGeometry.ccをエディットしてBGO_Twoのジオメトリを実装する

1)　現在のGeometry.ccとBGO_Twoジオメトリ(模範解答)の相違をcolordiffで端末に表示する

```
$ cd ~/Geant4Tutorial20171129/UserWorkDir/P02_Geometry/source/src
$ colordiff -y --width=200  Geometry.cc \
                            ../../util/G4Codes/Geometry.cc_P02_BGO_Two
```

← Geometry.ccのあるdir

BGO_Oneファイル

```
// Define 'BGO Detector'
   // Define the shape of solid
   G4double radius_BGO = 2.5*cm;
   G4double leng_Z_BGO = 2.5*cm;
   G4Tubs* solid_BGO = new G4Tubs("Solid_BGO", 0., radius_BGO, leng_Z_BGO/2.0, 0., 360.*deg);

   // Define logical volume
   G4Material* materi_BGO = materi_Man->FindOrBuildMaterial("G4_BGO");
   G4LogicalVolume* logVol_BGO =
     new G4LogicalVolume( solid_BGO, materi_BGO, "LogVol_BGO", 0, 0, 0 );

   // Placement of logical volume
   G4double pos_X_LogV = 0.0*cm;          // X-location LogV
   G4double pos_Y_LogV = 0.0*cm;          // Y-location LogV
   G4double pos_Z_LogV = 0.0*cm;          // Z-location LogV
   G4ThreeVector threeVect_LogV = G4ThreeVector(pos_X_LogV, pos_Y_LogV, pos_Z_LogV);
   G4RotationMatrix rotMtrx_LogV = G4RotationMatrix();
   G4Transform3D trans3D_LogV = G4Transform3D(rotMtrx_LogV, threeVect_LogV);

   G4int copyNum_LogV = 1000;               // Set ID number of LogV
   new G4PVPlacement(trans3D_LogV, "PhysVol_BGO", logVol_BGO, physVol_World,
                     false, copyNum_LogV);

// Return the physical world
   return physVol_World;
}
```

変更ライン印　追加ライン印

BGO_Twoファイル

```
// Define 'BGO Detector'
   // Define the shape of solid
   G4double radius_BGO = 2.5*cm;
   G4double leng_Z_BGO = 2.5*cm;
   G4Tubs* solid_BGO = new G4Tubs("Solid_BGO", 0., radius_BGO, leng_Z_BGO/2.0, 0., 360.*deg);

   // Define logical volume
   G4Material* materi_BGO = materi_Man->FindOrBuildMaterial("G4_BGO");
   G4LogicalVolume* logVol_BGO =
     new G4LogicalVolume( solid_BGO, materi_BGO, "LogVol_BGO", 0, 0, 0 );

   // Placement of logical volume - 1st BGO
   G4double pos_X_LogV = 0.0*cm;          // X-location LogV
   G4double pos_Y_LogV = 0.0*cm;          // Y-location LogV
   G4double pos_Z_LogV = -2.0*cm;         // Z-location LogV
   G4ThreeVector threeVect_LogV = G4ThreeVector(pos_X_LogV, pos_Y_LogV, pos_Z_LogV);
   G4RotationMatrix rotMtrx_LogV = G4RotationMatrix();
   G4Transform3D trans3D_LogV = G4Transform3D(rotMtrx_LogV, threeVect_LogV);

   G4int copyNum_LogV = 1000;               // Set ID number of LogV
   new G4PVPlacement(trans3D_LogV, "PhysVol_BGO", logVol_BGO, physVol_World,
                     false, copyNum_LogV);

   // Placement of logical volume - 2nd BGO
   pos_Z_LogV = +2.0*cm;                    // Z-location LogV
   threeVect_LogV = G4ThreeVector(pos_X_LogV, pos_Y_LogV, pos_Z_LogV);
   trans3D_LogV = G4Transform3D(rotMtrx_LogV, threeVect_LogV);

   copyNum_LogV = 2000;                  // Set ID number of LogV
   new G4PVPlacement(trans3D_LogV, "PhysVol_BGO", logVol_BGO, physVol_World,
                     false, copyNum_LogV);

// Return the physical world
   return physVol_World;
}
```

2)　現在のGeometry.ccをエディターで開き、colordiffの結果を参照しながらBGO_Twoを実装する

```
$ cd ~/Geant4Tutorial20171129/UserWorkDir/P02_Geometry/source/src
$ gedit Geometry.cc&
```

← Geometry.ccのあるdir

[注1]　変更をタイプするのが大変なら、以下のファイルをターミナルで表示して、変更箇所をコピペすること
　　$ less ../../util/G4Codes/Geometry.cc_P02_BGO_Two
[注2]　コピペがうまくできない場合、以下のコマンドを実行して模範解答を全コピーする
　　$ cp ../../util/G4Codes/Geometry.cc_P02_BGO_Two Geometry.cc

P02_Geometry: BGO_Twoアプリケーションのビルド

課題：4　新たに作ったGeometry.ccを用いてアプリケーションをビルドする

1) P02_Geometry作業用のディレクトリに戻る

```
$ cd ~/Geant4Tutorial20171129/UserWorkDir/P02_Geometry
```

← CBDir (Current Base Directory)

2) buildデレクトリでビルドを実行　[注1]

```
$ \rm -r bin
$ cd  build
$ \rm -r *
$ cmake ../source
$ make
$ make install
$ cd  ..
```

前のプログラムをビルドした時に作らたbinは消去（省略可）
前のプログラムをビルドした時に作られたbuildに移動
前のビルドで作られているファイル類を全て消去
cmakeの慣用句
CBDirに戻る

[注1]
buildのステップでerrorが出て、どうしても
アプリケーションが作れない場合、CBdirのもとで
以下のスクリプトを実行すれば、模範解答の
Geometry.ccをもとにbuildを自動完了する
ことができる

./util/Help/Build_P02_BGO_Two.sh

P02_Geometry: BGO_Two(Application_Main)の実行

課題：5　Application_Main を実行する

1) プログラム実行を行う作業デレクトリを新たに作成する

```
$ pwd
……/P02_Geometry
$ cd TestBench
$ ../bin/Application_Main
```

現デレクトリがCBDirで有ることを確認：
P02_Geometryでなければ、そこに移動する

作業デレクトリに移動（前演習で作られている）

TestBenchデレクトリでApplication_Main実行

2) 端末ウインドに以下のメッセージが出力され、続いてQtウインドが開く

```
*************************************************************
 Geant4 version Name: geant4-10-03-patch-03    (20-October-2017)
                      Copyright : Geant4 Collaboration
                      Reference : NIM A 506 (2003), 250-303
                            WWW : http://cern.ch/geant4
*************************************************************

<<< Geant4 Physics List simulation engine: FTFP_BERT 2.0
.....
### Adding tracking cuts for neutron  TimeCut(ns)= 10000  KinEnergyCut(MeV)= 0
Visualization Manager instantiating with verbosity "warnings (3)"...
Visualization Manager initialising...
Registering graphics systems...
.....
```

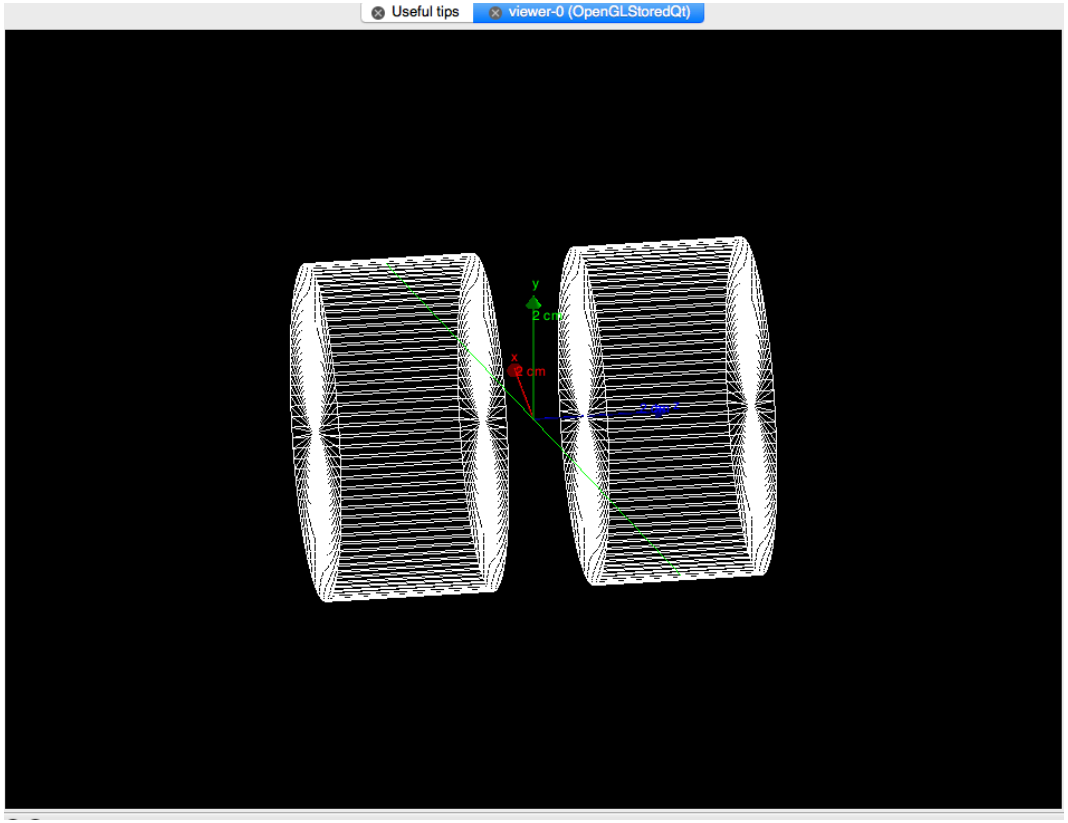
Qtウインドでアプリの動作を確認

課題：6　UI コマンドでe+発生

1) 以下のUIコマンドを先ず入力

```
/gun/particle e+
/gun/energy 0.0 MeV
/gun/position 0.0 0.0 0.0 cm
/run/beamOn 1
```

- 右図はその出力
  e+が消滅してback-to-backに2γが出ている



2) 照射回数を自由に変更してみる

3) アプリ終了

```
exit
```

P02_Geometry: Pixel_One

Pixel検出器ジオメトリ： Replicaの使用

P02_Geometry: Pixel_Oneプログラムの概要

■ 演習プログラムの目的

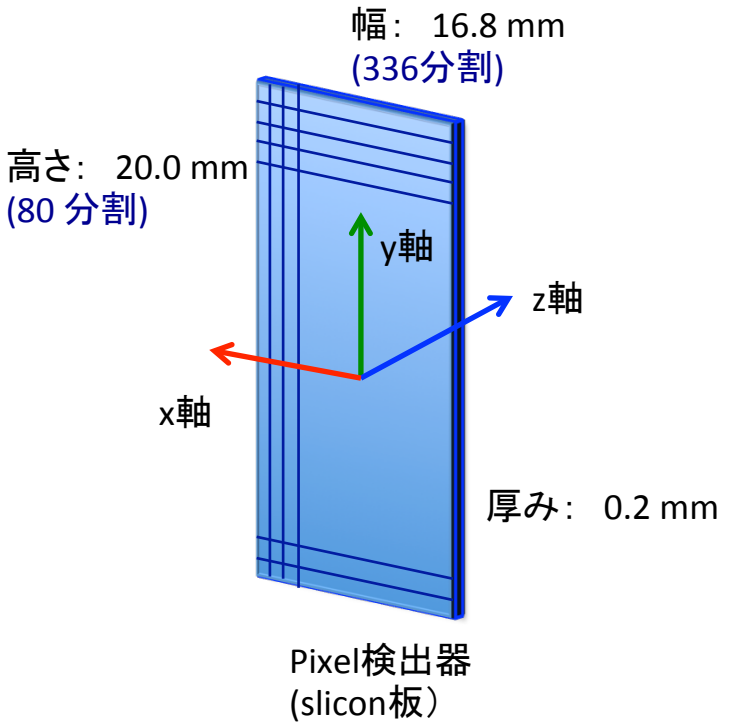
- Pixel検出器のジオメトリをレプリカ配置で記述したプログラムをビルドして実行する

  ［注］この演習では、用意されているプログラムを編集することなく、そのまま使う

- レプリカを用いると、検出器の個々の構成要素を一つづつ配置するのでなく、要素を自動的に複数個配置できることを学ぶ

■ プログラムの構成

- mainプログラム: 前演習と同一
- ジオメトリ: 一枚のPixel検出器
- PhysicsList : 前演習と同一
- PrimaryGenerator: 前演習と同一



P02_Geometry: Pixel_Oneのビルド

課題：1　P02_Geometry: Pixel_Oneをビルドして実行ファイルを作成する

1) P02_Geometry: Pixel_One用のGeometry.ccにコピー

```
$ cd ~/Geant4Tutorial20171129/UserWorkDir/P02_Geometry
$ cd source/src
$ cp ../../util/G4Codes/Geometry.cc_P02_Pixel_One  Geometry.cc
overwrite Geometry.cc? (y/n [n]) y
```

CBDir (Current Base Directory)

2) buildデレクトリでビルドを実行　[注1]

```
$ cd ../..
$ \rm -r bin
$ cd  build
$ \rm -r *
$ cmake ../source
$ make
$ make install
$ cd  ..
```

前のプログラムをビルドした時に作らたbinは消去（省略可）
前のプログラムをビルドした時に作られたbuildに移動
前のビルドで作られているファイル類を全て消去
cmakeの慣用句
CBDirに戻る

[注1]
buildを失敗したら、CBdirのもとで以下のスクリプトを実行すればbuildは自動完了
./util/Help/Build_P02_Pixel_One.sh

Application_Main (Pixel_One)の実行

課題：2　Application_Main (Pixel_One)を実行する

1) プログラム実行を行う作業デレクトリを新たに作成する

```
$ pwd
……/P02_Geometry
$ cd TestBench
$ ../bin/Application_Main
```

現レクトリがCBDirであることを確認：
P02_Geometryでなければ、そこに移動する

作業デレクトリに移動（前演習で作られている）

TestBenchデレクトリでApplication_Main実行

2) 端末ウインドに以下のメッセージが出力され、続いてQtウインドが開く

```
**************************************************************
 Geant4 version Name: geant4-10-03-patch-03    (20-October-2017)
                      Copyright : Geant4 Collaboration
                      Reference : NIM A 506 (2003), 250-303
                            WWW : http://cern.ch/geant4
**************************************************************

<<< Geant4 Physics List simulation engine: FTFP_BERT 2.0
.....
### Adding tracking cuts for neutron  TimeCut(ns)= 10000  KinEnergyCut(MeV)= 0
Visualization Manager instantiating with verbosity "warnings (3)"...
Visualization Manager initialising...
Registering graphics systems...
.....
```

Qtウインドでアプリの動作を確認

課題：3　UI コマンドでproton発生

1) 以下のUIコマンドを先ず入力

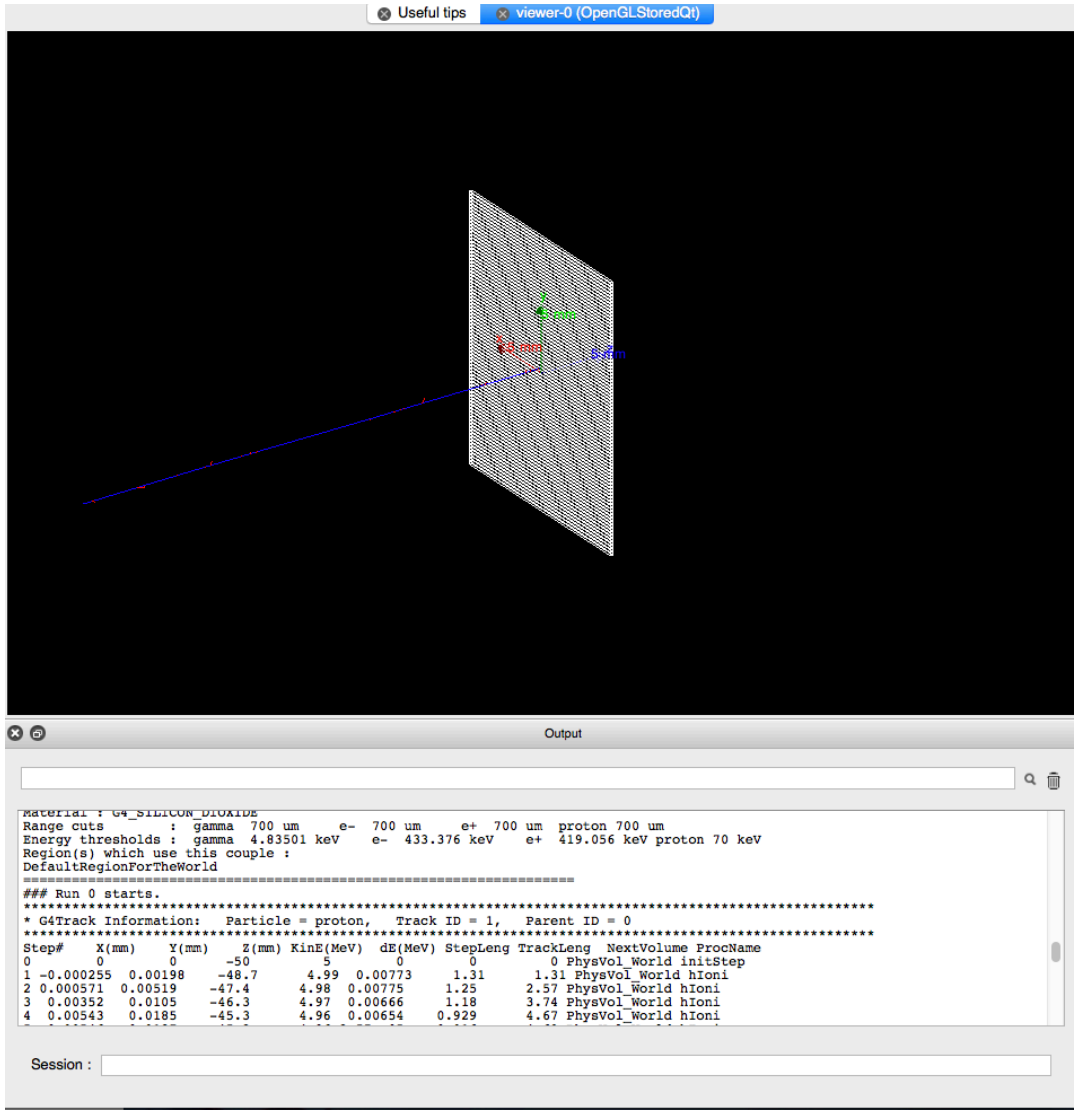
```
/gun/particle proton
/gun/energy 5.0 MeV
/run/beamOn 1
```

右図はその出力
（マウスで視点を調整してある）

2) 照射回数を自由に変更してみる

3) アプリ終了

```
exit
```

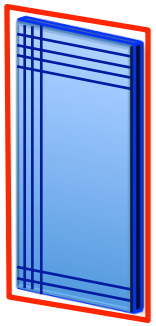


ジオメトリ定義の実際： Geomety.ccの内容

課題：4　P02_Geometry: Pixel_Oneの内容を理解する

```
$ less ../source/src/Geometry.cc
```

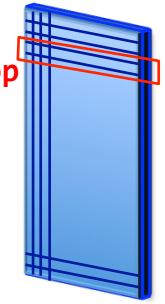
```
//---------------------------------------------------------------------------
  G4VPhysicalVolume* Geometry::Construct()
//---------------------------------------------------------------------------
{  world volume作成部は省略
// Define 'Pixel Detector' - Global Envelop    Pixel検出器の外形枠(global envelop)作成
   // Define the shape of the global envelop
   G4double leng_X_PixEnvG = 16.8*mm;     // X-full-length of pixel: global envelop
   G4double leng_Y_PixEnvG = 20.0*mm;     // Y-full-length of pixel: global envelop
   G4double leng_Z_PixEnvG =  0.2*mm;     // Z-full-length of pixel: global envelop
   G4Box* solid_PixEnvG =
     new G4Box("Solid_PixEnvG", leng_X_PixEnvG/2.0, leng_Y_PixEnvG/2.0, leng_Z_PixEnvG/2.0);

   // Define logical volume of the global envelop
   G4Material* materi_PixEnvG = materi_Man->FindOrBuildMaterial("G4_AIR");
   G4LogicalVolume* logVol_PixEnvG =
     new G4LogicalVolume(solid_PixEnvG, materi_PixEnvG, "LogVol_PixEnvG");
   logVol_PixEnvG->SetVisAttributes (G4VisAttributes::Invisible);

// Define 'Pixel Detector' - Local Envelop (divided the global envelop in Y-direction)
   // Define the shape of the local envelop
   G4int nDiv_Y = 80;                                  // Number of divisions in Y-direction
   G4double leng_X_PixEnvL = leng_X_PixEnvG;           // X-full-length of pixel: local envelop
   G4double leng_Y_PixEnvL = leng_Y_PixEnvG/nDiv_Y;    // Y-full-length of pixel: local envelop
   G4double leng_Z_PixEnvL = leng_Z_PixEnvG;           // Z-full-length of pixel: local envelop
   G4Box* solid_PixEnvL =
     new G4Box("Solid_PixEnvL", leng_X_PixEnvL/2.0, leng_Y_PixEnvL/2.0, leng_Z_PixEnvL/2.0);

   // Define logical volume of the local envelop
   G4Material* materi_PixEnvL = materi_Man->FindOrBuildMaterial("G4_AIR");
   G4LogicalVolume* logVol_PixEnvL =
     new G4LogicalVolume(solid_PixEnvL, materi_PixEnvL, "LogVol_PixEnvL");
   //logVol_PixEnvL->SetVisAttributes (G4VisAttributes::Invisible);

   // Placement of the local envelop to the global envelop using Replica
   new G4PVReplica("PhysVol_PixEnvL", logVol_PixEnvL, logVol_PixEnvG, kYAxis,
                   nDiv_Y, leng_Y_PixEnvL);
```



global envelop

global envelopのsolidと logical volume作成

Pixel検出器の外形枠のY軸方向を80分割 (local envelop)



local envelop

local envelopの solidとlogical volume作成

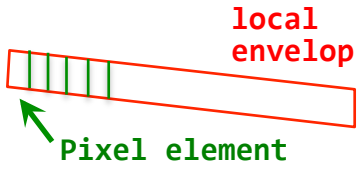
上で定義されたlocal envelop (logical volume)をレプリカを使い、80個global envelopに配置 配置はG4PVReplicaを使用

ジオメトリ定義の実際： Geomety.ccの内容 （続き）

課題：4 Geometry.cc_Pixelの内容を理解する

前のコードの続き

```
// Define 'Pixel Detector' - Pixel Element (divided the local envelop in X-direction)
   // Define the shape of the pixel element
   G4int nDiv_X = 336;                                  // Number of divisions in Y-direction
   G4double leng_X_PixElmt = leng_X_PixEnvG/nDiv_X;     // X-full-length of pixel: pixel element
   G4double leng_Y_PixElmt = leng_Y_PixEnvG/nDiv_Y;     // Y-full-length of pixel: pixel element
   G4double leng_Z_PixElmt = leng_Z_PixEnvG;            // Z-full-length of pixel: pixel element
   G4Box* solid_PixElmt =
     new G4Box("Solid_PixElmt", leng_X_PixElmt/2.0, leng_Y_PixElmt/2.0, leng_Z_PixElmt/2.0);

   // Define logical volume of the pixel element
   G4Material* materi_PixElmt = materi_Man->FindOrBuildMaterial("G4_SILICON_DIOXIDE");
   G4LogicalVolume* logVol_PixElmt =
     new G4LogicalVolume(solid_PixElmt, materi_PixElmt, "LogVol_PixElmt");
   logVol_PixElmt->SetVisAttributes (G4VisAttributes::Invisible);

   // Placement of pixel elements to the local envelop using Replica
   new G4PVReplica("PhysVol_PixElmt", logVol_PixElmt, logVol_PixEnvL, kXAxis,
                   nDiv_X, leng_X_PixElmt);

// Placement of the 'Pixel Detector' to the world: Put the 'global envelop'
   G4double pos_X_LogV_PixEnvG = 0.0*cm;       // X-location LogV_PixEnvG
   G4double pos_Y_LogV_PixEnvG = 0.0*cm;       // Y-location LogV_PixEnvG
   G4double pos_Z_LogV_PixEnvG = 0.0*cm;       // Z-location LogV_PixEnvG
   G4ThreeVector threeVect_LogV_PixEnvG =
     G4ThreeVector(pos_X_LogV_PixEnvG, pos_Y_LogV_PixEnvG, pos_Z_LogV_PixEnvG);
   G4RotationMatrix rotMtrx_LogV_PixEnvG = G4RotationMatrix();
   G4Transform3D trans3D_LogV_PixEnvG = G4Transform3D(rotMtrx_LogV_PixEnvG, threeVect_LogV_PixEnvG);

   G4int copyNum_LogV_PixEnvG = 1000;                // Set ID number of LogV_PixEnvG
   new G4PVPlacement(trans3D_LogV_PixEnvG, "PhysVol_PixEnvG", logVol_PixEnvG, physVol_World,
                     false, copyNum_LogV_PixEnvG);

// Return the physical world
   return physVol_World;
}
```



Pixelの最小エレメントのsolidと
logical volume作成

Pixelの最小エレメント（logical
volume）をレプリカを使って336個
local envelopに配置
配置はG4PVReplicaを使用

最終的にPixel全体をワールド
ボリュウムに配置

➔ Pixelのglobalvolumeを配
置すれば良い

配置はG4PVPlacementを使用

[注] Geometry.hhは前の演習で使ったものをそのまま変更せずに使用出来る

P02_Geometry: Pixel_Two

二つのPixel検出器の配置 ： 衝突チェック

P02_Geometry: Pixel_Twoプログラムの概要

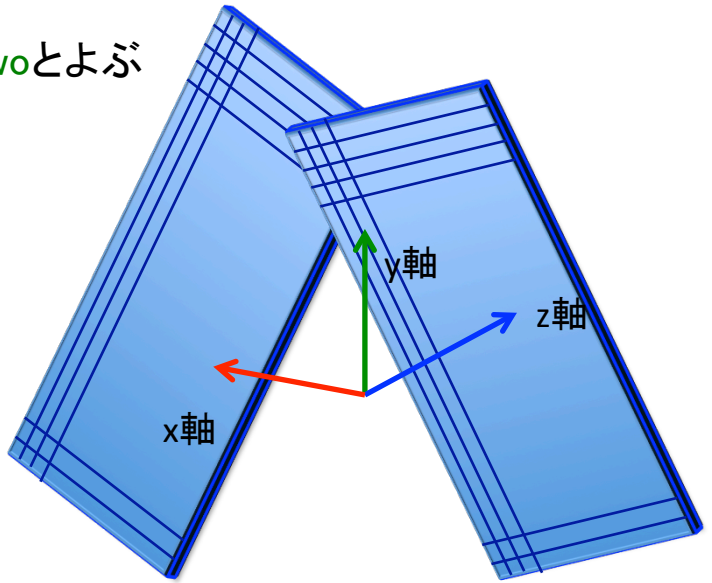
■ 演習プログラムの目的

- 前演習のP02_Geometry: Pixel_Oneで学んだGeometryファイルをもとに、Pixel検出器を1台追加し、プログラムをビルドして実行する

  ［注］この演習では、受講者はソースコードを書き、それを動かしてみる
- Pixel検出器を2台を一部が重なるように配置し、衝突チェック機能により重なりを検知できることを学ぶ
- G4Transform3Dで物体を回転させる方法を学ぶ
- 新たに作るプログラムはP02_Geometry: Pixel_Twoとよぶ

■ プログラムの構成

- mainプログラム: 前演習と同一
- ジオメトリ: 二枚のPixel検出器
- PhysicsList : 前演習と同一
- PrimaryGenerator: 前演習と同一



P02_Geometry: Pixel_Twoプログラムの作成

はじめに:

P02_Geometry: Pixel_Twoプログラムを作成するには:

1) プログラム作成作業はP02_Geometry: Pixel_Oneで使ったファイルを変更することで行う
2) 本演習で変更が必要なファイルはGeometry.ccのみで、それ以外は全て流用する

課題:1　P02_Geometry: Pixel_Twoのジオメトリを書くファイルの用意

1) 現在のGeometry.ccファイルには前課題で使ったP02_Geometry: Pixel_Oneのジオメトリが書かれている。その内容をエディトして"Pixel_Two"ジオメトリを作成する
2) 念のため、オリジナルのP02_Gometry: Pixel_Oneを現在のGeometry.ccに再度コピーしておく
   （Geometry.ccを変更した記憶がなければこのステップはスキップ可）

```
$ cd ~/Geant4Tutorial20171129/UserWorkDir/P02_Geometry
$ cd source/src
$ cp ../../util/G4Codes/Geometry.cc_P02_Pixel_One Geometry.cc
overwrite Geometry.cc? (y/n [n]) y
```

CBDir (Current Base Directory)

ジオメトリ定義の実装： P02_Geometry: Pixel_Two

課題：2　現在のGeometry.ccからの変更箇所を理解する ← エディット手順は次のスライド参照

［注］ 以下のソースコードは変更が完了したものを示す

```
//---------------------------------------------------------------------------
  G4VPhysicalVolume* Geometry::Construct()
//---------------------------------------------------------------------------
{
  Pixel検出器のlogical volumeを作るまでのコードは
  Geometry.cc_Pixel_Oneのコードと全く同じ

// Placement of the 1st 'Pixel Detector' to the world: Put the 'global envelop'
   G4double pos_X_LogV_PixEnvG = 1.0*cm;       // X-location LogV_PixEnvG
   G4double pos_Y_LogV_PixEnvG = 0.0*cm;       // Y-location LogV_PixEnvG
   G4double pos_Z_LogV_PixEnvG = 0.0*cm;       // Z-location LogV_PixEnvG
   G4ThreeVector threeVect_LogV_PixEnvG =
     G4ThreeVector(pos_X_LogV_PixEnvG, pos_Y_LogV_PixEnvG, pos_Z_LogV_PixEnvG);
   G4RotationMatrix rotMtrx_LogV_PixEnvG = G4RotationMatrix();
   rotMtrx_LogV_PixEnvG.rotateZ(45.0*deg);
   G4Transform3D trans3D_LogV_PixEnvG = G4Transform3D(rotMtrx_LogV_PixEnvG, threeVect_LogV_PixEnvG);

   G4int copyNum_LogV_PixEnvG = 1000;          // Set ID number of LogV_PixEnvG
   new G4PVPlacement(trans3D_LogV_PixEnvG, "PhysVol_PixEnvG_#1", logVol_PixEnvG, physVol_World,
                     false, copyNum_LogV_PixEnvG, true);

// Placement of the 2nd 'Pixel Detector' to the world: Put the 'global envelop'
   pos_X_LogV_PixEnvG =-1.0*cm;                // X-location LogV_PixEnvG
   threeVect_LogV_PixEnvG =
     G4ThreeVector(pos_X_LogV_PixEnvG, pos_Y_LogV_PixEnvG, pos_Z_LogV_PixEnvG);
   rotMtrx_LogV_PixEnvG = G4RotationMatrix();  // Reset the rotation matrix
   rotMtrx_LogV_PixEnvG.rotateZ(-45.0*deg);
   trans3D_LogV_PixEnvG = G4Transform3D(rotMtrx_LogV_PixEnvG, threeVect_LogV_PixEnvG);

   copyNum_LogV_PixEnvG = 2000;                // Set ID number of LogV_PixEnvG
   new G4PVPlacement(trans3D_LogV_PixEnvG, "PhysVol_PixEnvG_#2", logVol_PixEnvG, physVol_World,
                     false, copyNum_LogV_PixEnvG, true);

// Return the physical world
   return physVol_World;
}
```

一番目のPixelの配置

Z軸の周りに+45度回転させる
マトリックスを生成

一番目のPixel配置で衝突
チェック機能をonにする

二番目のPixelの配置

Z軸の周りに-45度回転させる
マトリックスを生成

二番目のPixel配置で衝突
チェック機能をonにする

P02_Geomety: Pixel_Twoの実装 – つづき

課題：3　現在のGeometry.ccをエディットしてPixel_Twoのジオメトリを実装する

1）　現在のGeometry.ccとPixel_Twoジオメトリ（模範解答）の相違をcolordiffで端末に表示する

```
$ cd ~/Geant4Tutorial20161129/UserWorkDir/P02_Geometry/source/src
$ colordiff -y -width=200  Geometry.cc \
                           ../../util/G4Codes/Geometry.cc_P02_Pixel_Two
```

Geometry.ccのあるdir

Pixel_Oneファイル　　Pixel_Twoファイル

変更ライン印

追加ライン印

```
// Placement of the 'Pixel Detector' to the world: Put the 'global envelop'
   G4double pos_X_LogV_PixEnvG = 0.0*cm;       // X-location LogV_PixEnvG
   G4double pos_Y_LogV_PixEnvG = 0.0*cm;       // Y-location LogV_PixEnvG
   G4double pos_Z_LogV_PixEnvG = 0.0*cm;       // Z-location LogV_PixEnvG
   G4ThreeVector threeVect_LogV_PixEnvG =
     G4ThreeVector(pos_X_LogV_PixEnvG, pos_Y_LogV_PixEnvG, pos_Z_LogV_PixEnvG);
   G4RotationMatrix rotMtrx_LogV_PixEnvG = G4RotationMatrix();

   G4Transform3D trans3D_LogV_PixEnvG = G4Transform3D(rotMtrx_LogV_PixEnvG, threeVect_LogV_PixEn

   G4int copyNum_LogV_PixEnvG = 1000;              // Set ID number of LogV_PixEnvG
   new G4PVPlacement(trans3D_LogV_PixEnvG, "PhysVol_PixEnvG", logVol_PixEnvG, physVol_World,
                     false, copyNum_LogV_PixEnvG);

// Return the physical world
   return physVol_World;
}
```

```
// Placement of the 1st 'Pixel Detector' to the world: Put the 'global envelop'
   G4double pos_X_LogV_PixEnvG = 1.0*cm;       // X-location LogV_PixEnvG
   G4double pos_Y_LogV_PixEnvG = 0.0*cm;       // Y-location LogV_PixEnvG
   G4double pos_Z_LogV_PixEnvG = 0.0*cm;       // Z-location LogV_PixEnvG
   G4ThreeVector threeVect_LogV_PixEnvG =
     G4ThreeVector(pos_X_LogV_PixEnvG, pos_Y_LogV_PixEnvG, pos_Z_LogV_PixEnvG);
   G4RotationMatrix rotMtrx_LogV_PixEnvG = G4RotationMatrix();
   rotMtrx_LogV_PixEnvG.rotateZ(45.0*deg);
   G4Transform3D trans3D_LogV_PixEnvG = G4Transform3D(rotMtrx_LogV_PixEnvG, threeVect_LogV_PixEn

   G4int copyNum_LogV_PixEnvG = 1000;          // Set ID number of LogV_PixEnvG
   new G4PVPlacement(trans3D_LogV_PixEnvG, "PhysVol_PixEnvG_#1", logVol_PixEnvG, physVol_World,
                     false, copyNum_LogV_PixEnvG, true);

// Placement of the 2nd 'Pixel Detector' to the world: Put the 'global envelop'
   pos_X_LogV_PixEnvG =-1.0*cm;                // X-location LogV_PixEnvG
   threeVect_LogV_PixEnvG =
     G4ThreeVector(pos_X_LogV_PixEnvG, pos_Y_LogV_PixEnvG, pos_Z_LogV_PixEnvG);
   rotMtrx_LogV_PixEnvG = G4RotationMatrix();  // Reset the rotation matrix
   rotMtrx_LogV_PixEnvG.rotateZ(-45.0*deg);
   trans3D_LogV_PixEnvG = G4Transform3D(rotMtrx_LogV_PixEnvG, threeVect_LogV_PixEnvG);

   copyNum_LogV_PixEnvG = 2000;                // Set ID number of LogV_PixEnvG
   new G4PVPlacement(trans3D_LogV_PixEnvG, "PhysVol_PixEnvG_#2", logVol_PixEnvG, physVol_World,
                     false, copyNum_LogV_PixEnvG, true);

// Return the physical world
   return physVol_World;
}
```

2）　現在のGeometry.ccをエディターで開き、colordiffの結果を参照しながらPixel_Twoを実装する

```
$ cd ~/Geant4Tutorial20171129/UserWorkDir/P02_Geometry/source/src
$ gedit Geometry.cc&
```

Geometry.ccのあるdir

[注1]　変更をタイプするのが大変なら、以下のファイルをターミナルで表示して、変更箇所をコピペすること
　　$ less ../../util/G4Codes/Geometry.cc_P02_Pixel_Two
[注2]　コピペがうまくできない場合、以下のコマンドを実行して模範解答を全コピーする
　　$ cp ../../util/G4Codes/Geometry.cc_P02_Pixel_Two Geometry.cc

P02_Geometry: Pixel_Twoアプリケーションのビルド

課題：4　新たに作ったGeometry.ccを用いてアプリケーションをビルドする

1) P02_Geometry作業用のディレクトリに戻る

```
$ cd ~/Geant4Tutorial20171129/UserWorkDir/P02_Geometry
```

← CBDir (Current Base Directory)

2) buildデレクトリでビルドを実行　[注1]

```
$ \rm -r bin
$ cd  build
$ \rm -r *
$ cmake ../source
$ make
$ make install
$ cd ..
```

- \rm -r bin ← 前のプログラムをビルドした時に作らたbinは消去（省略可）
- cd build ← 前のプログラムをビルドした時に作られたbuildに移動
- \rm -r * ← 前のビルドで作られているファイル類を全て消去
- cmake ../source / make / make install ← cmakeの慣用句
- cd .. ← CBDirに戻る

[注1]
buildのステップでerrorが出て、どうしても
アプリケーションが作れない場合、CBDirのもとで
以下のスクリプトを実行すれば、模範解答の
Geometry.ccをもとにbuildを自動完了する
ことができる

./util/Help/Build_P02_Pixel_Two.sh

P02_Geometry: Pixel_Two(Application_Main)の実行

課題:5　Application_Main を実行する

1) プログラム実行を行う作業デレクトリを新たに作成する

```
$ pwd
……/P02_Geometry
$ cd TestBench
$ ../bin/Application_Main
```

現デレクトリがCBDirであることを確認:
P02_Geometryでなければ、そこに移動する

作業デレクトリに移動（前演習で作られている）

TestBenchデレクトリでApplication_Main実行

2) 端末ウインドに以下のメッセージが出力され、続いてQtウインドが開く

```
**************************************************************
 Geant4 version Name: geant4-10-03-patch-03    (20-October-2017)
                      Copyright : Geant4 Collaboration
                      Reference : NIM A 506 (2003), 250-303
                            WWW : http://cern.ch/geant4
**************************************************************

<<< Geant4 Physics List simulation engine: FTFP_BERT 2.0

Checking overlaps for volume PhysVol_PixEnvG_#1 ... OK!
Checking overlaps for volume PhysVol_PixEnvG_#2 ...
-------- WWWW ------- G4Exception-START -------- WWWW -------
*** G4Exception : GeomVol1002
      issued by : G4PVPlacement::CheckOverlaps()
Overlap with volume already placed !
        Overlap is detected for volume PhysVol_PixEnvG_#2
        with PhysVol_PixEnvG_#1 volume's
        local point (-4.14214,7.09471,-0.0854881), overlapping by at least: 14.5119 um
NOTE: Reached maximum fixed number -1- of overlaps reports for this volume !
*** This is just a warning message. ***
-------- WWWW -------- G4Exception-END --------- WWWW -------

.....
```

衝突チェック機能からのメッセージ

Qtウインドでアプリの動作を確認

課題:6　UI コマンドでproton発生

1) 以下のUIコマンドを先ず入力

```
/gun/particle proton
/gun/energy 5.0 MeV
/run/beamOn 1
```

右図はその出力
（マウスで視点を調整してある）

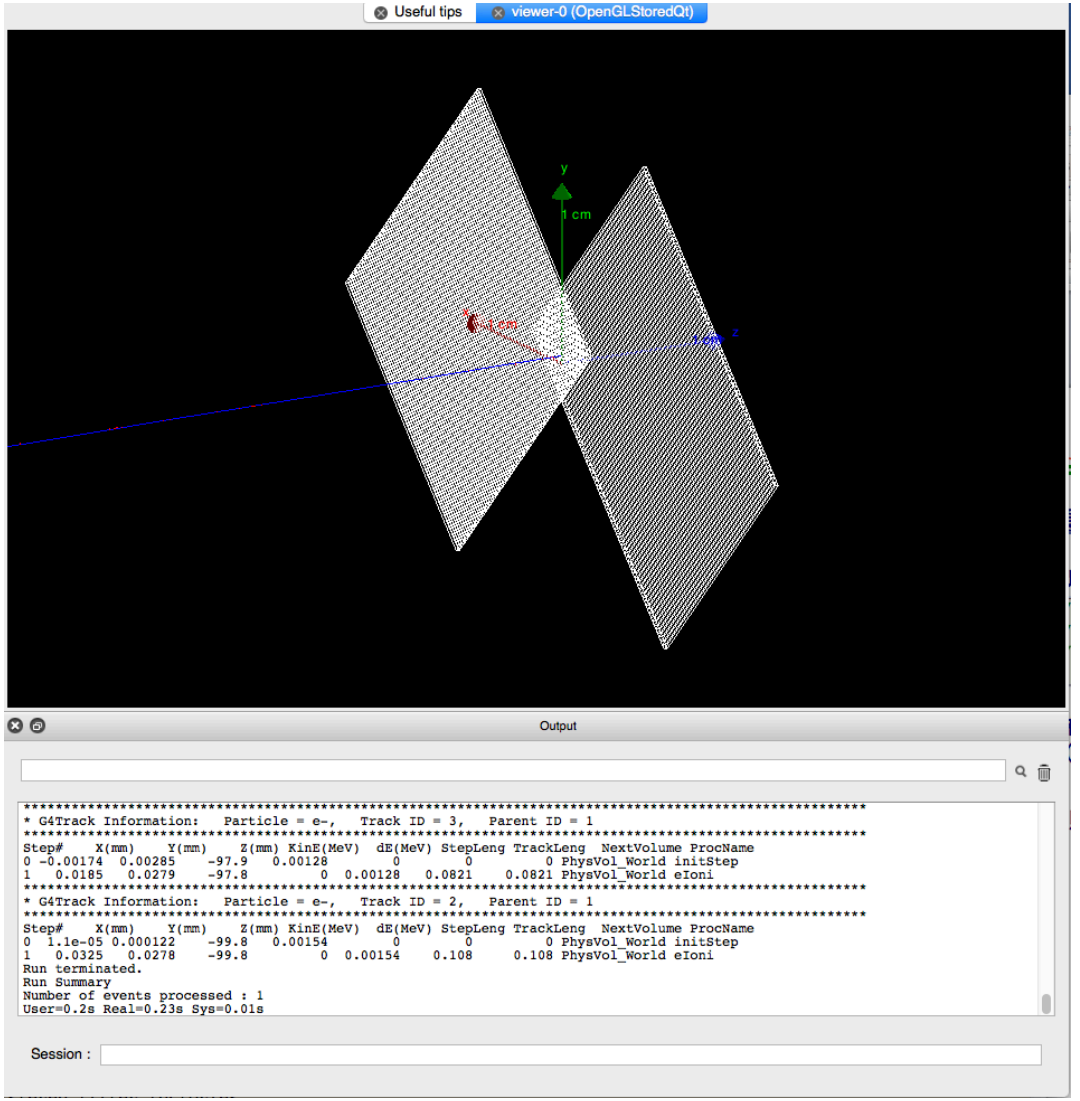
2) 照射回数を自由に変更してみる

ジオメトリに重なり部分があっても粒子は輸送される

ただし、シミュレーション結果は保障されない

3) アプリ終了

```
exit
```



終了