超

Geant4 4.9.5 短期講習

K.Hoshina

6/1/2012

CHEER, Earthquake Research Institute

イントロダクション

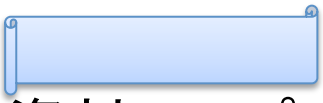
この講習会のサポート範囲

- Geant4の基本概念を学ぶ
- Geant4の一番簡単な例題(Example Novice)を実行しながら、基本的な使い方を学ぶ
- 各自のやりたい事に合わせて、どのように例題を変更したら良いかを学ぶ

- C++、UNIXの知識はあるものと仮定していますが、分からない場合はどんどん質問して下さい。時間が許す限り説明します（でないと講習の内容が分からなくなるので…）

参考サイト

- 2010年の福井大学で行われたGeant4講習会資料（ここの資料で初心者用の内容は殆ど網羅されている）
  http://wiki.kek.jp/display/geant4/Geant4+School+2010

-  この形のバナーがあるスライドは上記講習会資料のコピーです（ファイル名を参照）。詳細は講習会資料を見て下さい。

- ただし、最新版のG4（version4.9.5)はMake system がCmakeになっているので、コンパイル等の面で若干勝手が違います（本講習を参考）。

環境について

- 本講習ではインストールはやりません。
  - インストールの際の変数などを見たい人はAppendixを参照
- Linux, Windows, MacOSXの全てで動きます。
  - http://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/InstallationGuide/html/ を参照
- 以下のソフトを予めインストールすること。
  - Cmake(必須）
  - X11、Qt4 (visualization用にあった方がよい）
    - Linuxの場合、yumなどでインストールしても、パスが通っていないことがある。パスが通っていないと、geant4が見つけられないので、コンパイルをかける前にqtへのpathを通す。
    - export PATH=$PATH:/usr/lib64/qt4/bin など

Geant4の例題を動かしてみよう

まず、とにかく例題を見てみよう！

- 殆どの簡単なシミュレーションは、Exampleを変更するだけで事足ります。
- Geant4は膨大なライブラリ群なので、最初から全部理解しようと思わず、「習うより慣れろ」の精神が大事
- 複雑な検出器シミュレーションをやりたくなったら、Exampleの改変ではなく、自分でアプリケーションのフレームワークを書く方が、のちのちのメンテがやりやすい（上級者向け）
  - 複雑な検出器シミュレーションの例：高エネルギー加速器実験の検出器、ターゲットの数が大量でなおかつそれぞれのターゲットのアライメントが複雑な場合など
- 赤文字で書かれているコマンドは自分で打って下さい。

演習0:ログインと環境変数の設定

$ ssh –Y muon02　(guest以外はmuon01~06のどれでもよい、-Y オプション必須)

#　以下のコマンドを実行

$ source /data/g4/4.9.5p01/g4env.sh

(.bashrcに書いても良い)

G4を走らせる時は、常にこの作業が必要

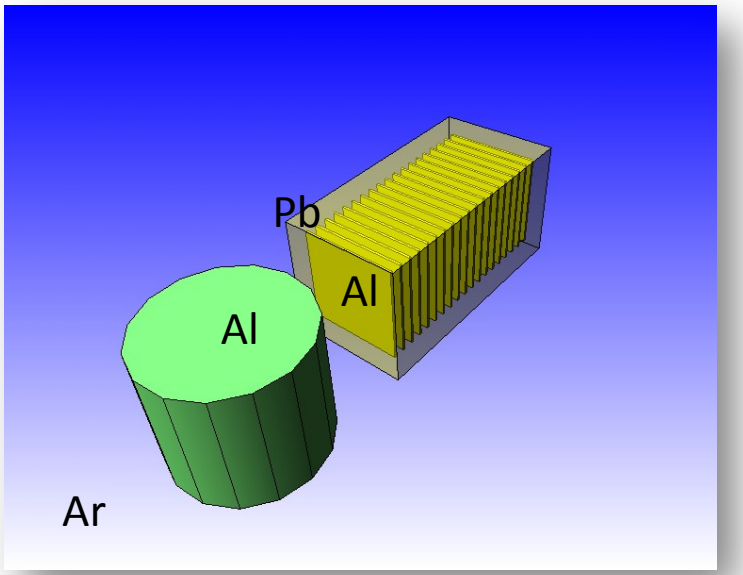
```
# add QT4 path
export PATH=$PATH:/usr/lib64/qt4/bin:/data/g4/

# add CMake path
export PATH=$PATH:/data/g4/cmake-2.8.8-Linux-i386/bin

# source G4 env
source /data/g4/install/bin/geant4.sh
```

演習1: exampleディレクトリを見つけよう

```
[hoshina@muon02 ~]$ cd /data/g4/     ←共有ディレクトリ、全てのノードから見える
[hoshina@muon02 g4]$ ls
4.9.5p01  cmake-2.8.8-Linux-i386  cmake-2.8.8-Linux-i386.tar.gz
HOWTO_INSTALL_G4.txt     ←自分でインストールしたい人はAppendixを参照

[hoshina@muon02 g4]$ cd 4.9.5p01/
[hoshina@muon02 4.9.5p01]$ ls
geant4.9.5.p01  geant4.9.5.p01.tar.gz  install  RHEL5
↑sourceファイル                    バイナリ↑     ↑ビルドディレクトリ

[hoshina@muon02 4.9.5p01]$ cd install/share/Geant4-9.5.1/examples/
[hoshina@muon02 examples]$ ls
advanced  basic  CMakeLists.txt  extended  GNUmakefile  History  novice  README

[hoshina@muon02 examples]$ cd novice/N01/
[hoshina@muon02 N01]$ ls
CMakeLists.txt  exampleN01.err  exampleN01.out  History  README
exampleN01.cc   exampleN01.in   GNUmakefile     include  src
```

Novice Example N01

- アルゴンガスで満たされた実験室中にアルミニウムの板等を並べたセットアップ
- geantino という相互作用をしない仮想粒子を一個発射する
  - これ（イベント）を3回繰り返す
- 物理プロセスと粒子とカット値の定義
  - ジアンティーノという仮想粒子
  - 運ぶだけ（物質との相互作用は未定義）
  - デフォルトのカット値を使う
- しばらくの間、プログラムの中身（実装）はブラックボックスでよい
  - 順に説明していきます



Go Iwai (KEK/CRC)
2010_12_07_0920_1050_Environment_Set-up_and_Geant4_Installation.pptx

演習2:N01をコンパイル

#workディレクトリの作成とソースのコピー
[guest@muon06 users]$ mkdir /data/users/ (ユーザー名）
[guest@muon06 users]$ cd /data/users/(ユーザー名）
[guest@muon06 guest]$ cp -rp /data/g4/4.9.5p01/install/share/Geant4-9.5.1/examples/novice/N01 .
[guest@muon06 guest]$ mkdir N01-build
[guest@muon06 guest]$ ls
N01 N01-build
#ソースのコンパイル
[guest@muon06 guest]$ cd N01-build/
[guest@muon06 N01-build]$ cmake ../N01
-- The C compiler identification is GNU 4.1.2
…
[guest@muon06 N01-build]$ make
Scanning dependencies of target exampleN01
…
#プログラム実行
[guest@muon06 N01-build]$ ./exampleN01

N01のmainプログラム

$ less /data/users/(ユーザー名)/N01/exampleN01.cc

```
int main()
{
  // Construct the default run manager
  //
  G4RunManager* runManager = new G4RunManager;

  // set mandatory initialization classes
  //
  G4VUserDetectorConstruction* detector = new ExN01DetectorConstruction;
  runManager->SetUserInitialization(detector);
  //
  G4VUserPhysicsList* physics = new ExN01PhysicsList;
  runManager->SetUserInitialization(physics);
 // set mandatory user action class
  //
  G4VUserPrimaryGeneratorAction* gen_action = new ExN01PrimaryGeneratorAction;
  runManager->SetUserAction(gen_action);
```

3つのクラス

- G4VUserDetectorConstruction から派生した ExN01DetectorConstruction
  - 物体の配置
- G4VUserPrimaryGeneratorAction から派生した ExN01PrimaryGeneratorAction
  - geantino の発射
- G4VUserPhysicsList から派生した ExN01PhysicsList
  - 物理プロセスや粒子の定義

Go Iwai (KEK/CRC)
2010_12_07_0920_1050_Environment_Set-up_and_Geant4_Installation.pptx

なぜか

- Geant4（の開発者）はみなさんが
  - どのような空間に
  - なんの粒子を入れて
  - どのような物理プロセスを使って
- シミュレーションしたいか知らない
  - みなさんが定義して知らせて（登録）する必要があります
- Geant4（の開発者）はみなさんがシミュレーションを行うための枠組みや便利な部品を提供する
  - ツールキット

Go Iwai (KEK/CRC)
2010_12_07_0920_1050_Environment_Set-up_and_Geant4_Installation.pptx

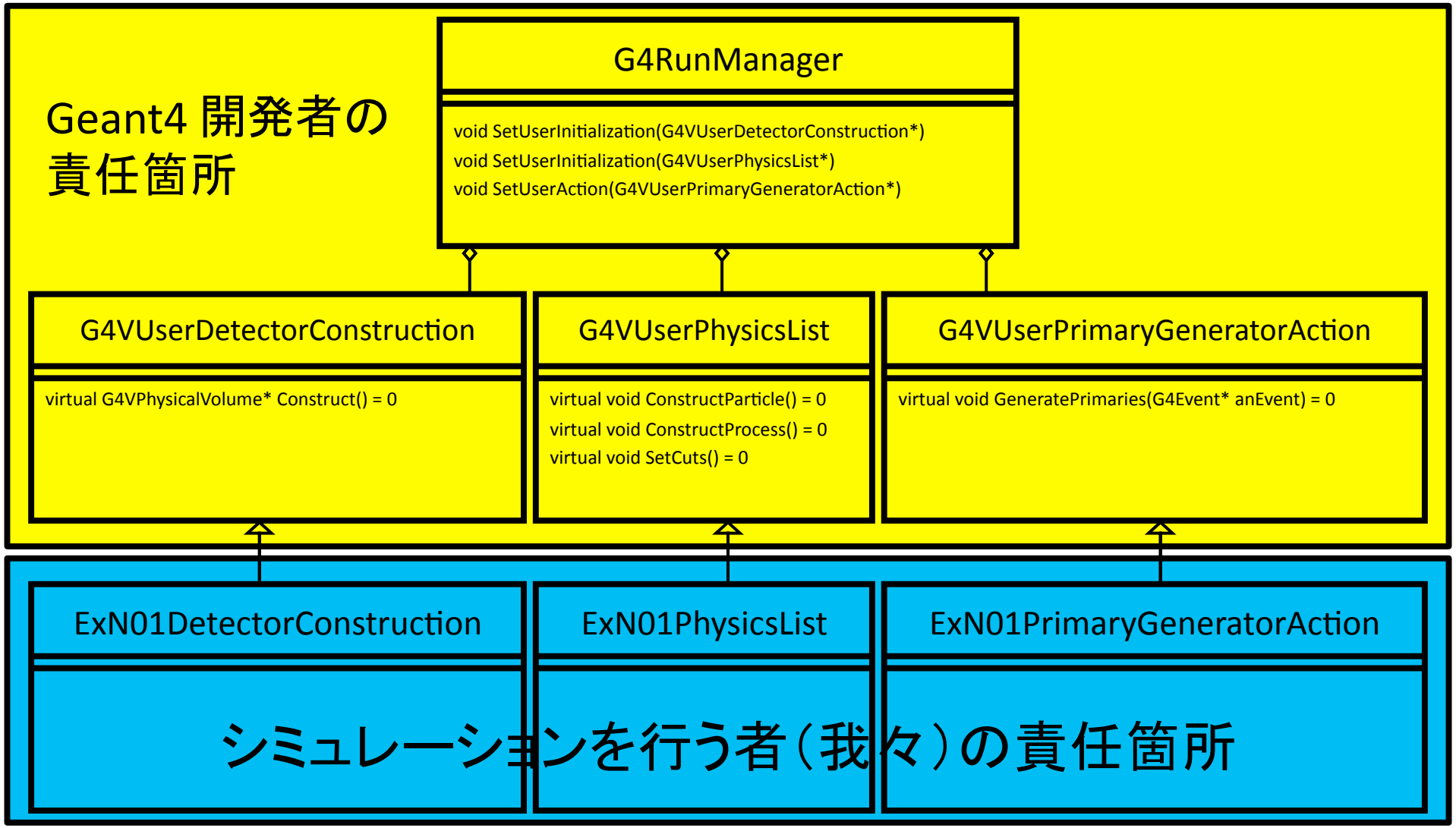
登録するために

- （通常は）exampleN01.cc 中の main() 関数内で行います

```
// ランマネージャーの構築、最初に作って最後に消す
G4RunManager* runManager = new G4RunManager();

// 必須！ジオメトリを定義したクラスの登録
runManager->SetUserInitialization(new ExN01DetectorConstruction());

// 必須！使用する物理プロセスを定義したクラスの登録
runManager->SetUserInitialization(new ExN01PhysicsList());

// 必須！イベントの始まりを定義したクラスの登録
runManager->SetUserAction(new ExN01PrimaryGeneratorAction());
```

Go Iwai (KEK/CRC)
2010_12_07_0920_1050_Environment_Set-up_and_Geant4_Installation.pptx

Geant4が提案するフレームワーク



登録後

```
// 一通り、登録が終わったら、初期化してやる
runManager->Initialize();

// ユーザーインターフェースへのポインタをとってくる
G4UImanager* UI = G4UImanager::GetUIpointer();

// コマンドを実行する、これは本来の使い方ではない
UI->ApplyCommand("/run/verbose 1");
UI->ApplyCommand("/event/verbose 1");
UI->ApplyCommand("/tracking/verbose 1");
UI->ApplyCommand("/run/beamOn 3");

// 最後にランマネージャーを消してやる、このとき登録したクラスも消される
delete runManager;
```

Go Iwai (KEK/CRC)
2010_12_07_0920_1050_Environment_Set-up_and_Geant4_Installation.pptx

- GNUmakefile
  - gmake 用の Makefile
  - 文法が拡張されている
  - いまどきの人間は gmake と make の違いを気にしなくてよい
- exampleN01.cc
  - main() 関数作成
  - RunManagerへのクラスオブジェクト登録
- include
  - クラスのヘッダファイルを置く
  - ExN01DetectorConstruction.hh
  - ExN01PrimaryGeneratorAction.hh
  - ExN01PhysicsList.hh
- src
  - クラスの実装ファイルを置く
  - ExN01DetectorConstruction.cc
  - ExN01PrimaryGeneratorAction.cc
  - ExN01PhysicsList.cc

Version 4.9.5からCmakeに変更

Go Iwai (KEK/CRC)

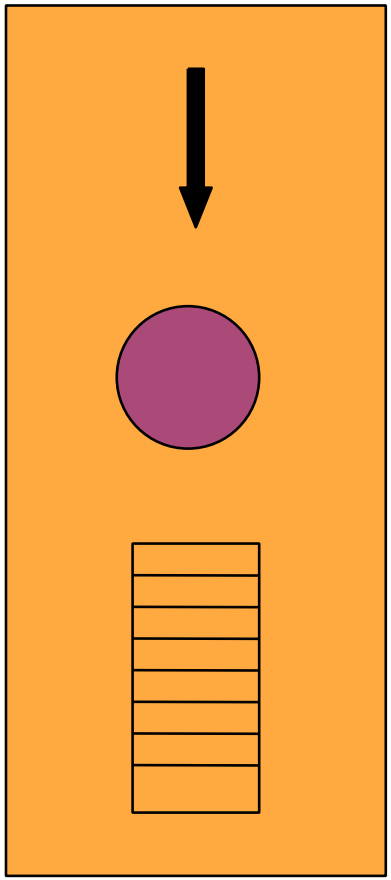
その他の例題

- Novice以下
  - 古くからある初心者用例題、そこそこ実際のシミュレーションにも使える
- Basic 以下
  - 新しくできた初心者用例題、pre-definedのphysics listなど、新しい機能も使われている
- 全ての例題にはソースディレクトリにREADMEがついており、そこに何が出来るか書いてあるので、まずREADMEに目を通してやりたい事を探す

Peter Gumplinger (TRIUMF)
2010_12_07_1340_1410_OnlineDocExamples.ppt

Novice Example N01

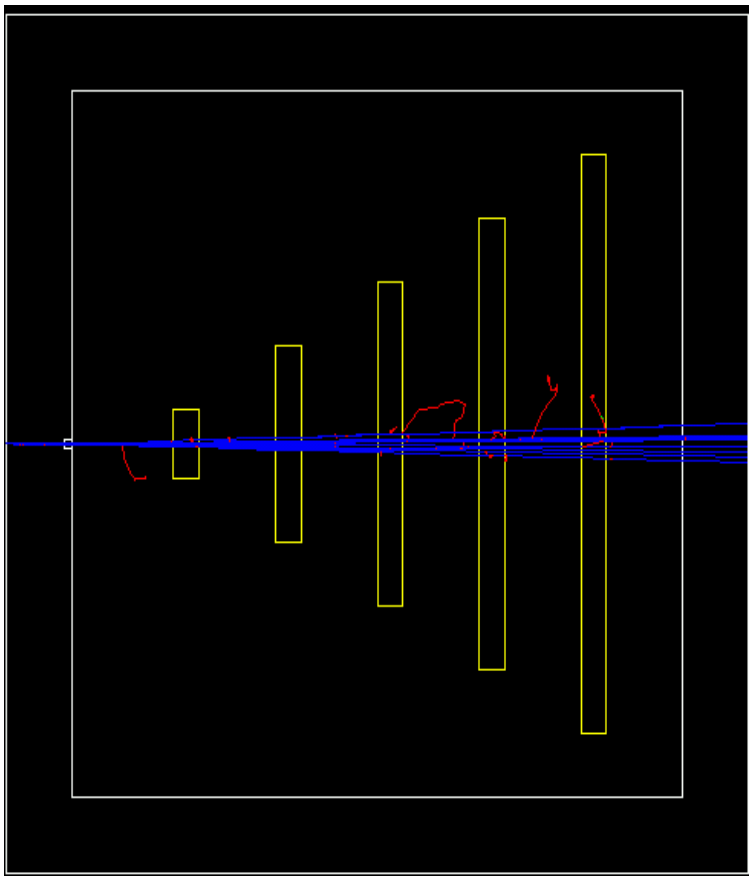
- hard coded batch, only mandatory user classes
- single element material, fixed geometry (GSG solids, G4PVPlacement without rotation)
- incident particle is a geantino
- only Transporation and no physics interactions, *G4ParticleGun*
- no magnetic field



Peter Gumplinger (TRIUMF)
2010_12_07_1340_1410_OnlineDocExamples.ppt

Novice Example N02

- interactive/batch mode
- simplified tracker geometry (parameterized volumes), mixtures and compound elements
- all EM processes + decay included for γ, charged leptons and charged hadrons
- uniform magn. Field, stack control
- detector response (SD/Hits)
  - trajectories and chamber hit collections may be stored
- Visualization of detector and event
- Command interface introduced

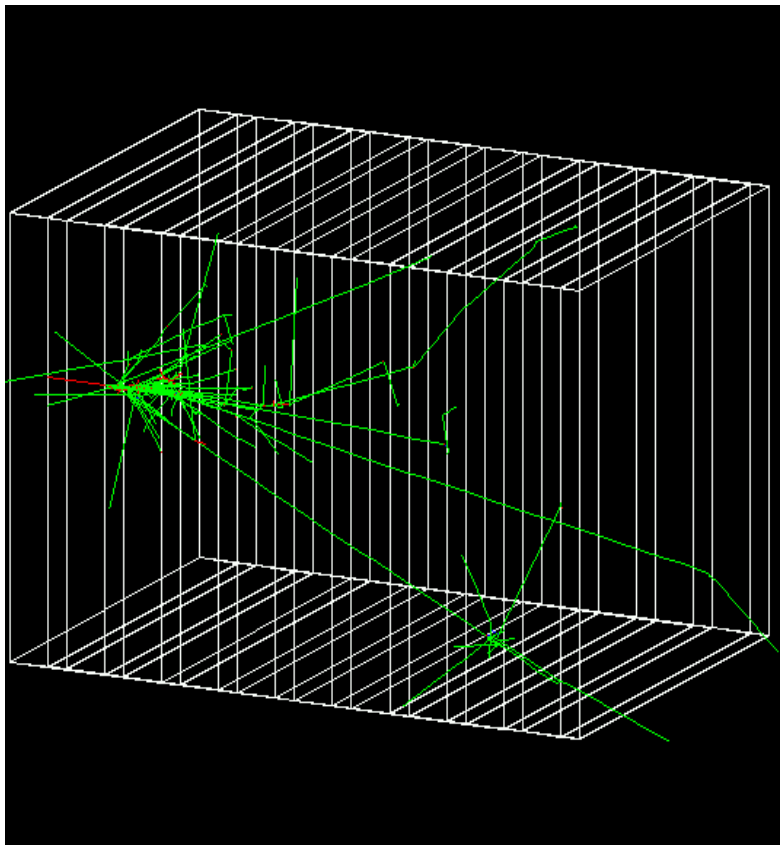


Peter Gumplinger (TRIUMF)
2010_12_07_1340_1410_OnlineDocExamples.ppt

Novice Example N03

- Sampling calorimeter with layers of Pb absorber and liquid Ar detection gaps (replicas)
- Automatic initialization of geometry via macro file (command interface)
- Exhaustive material definitions
- Randomization of incident beam

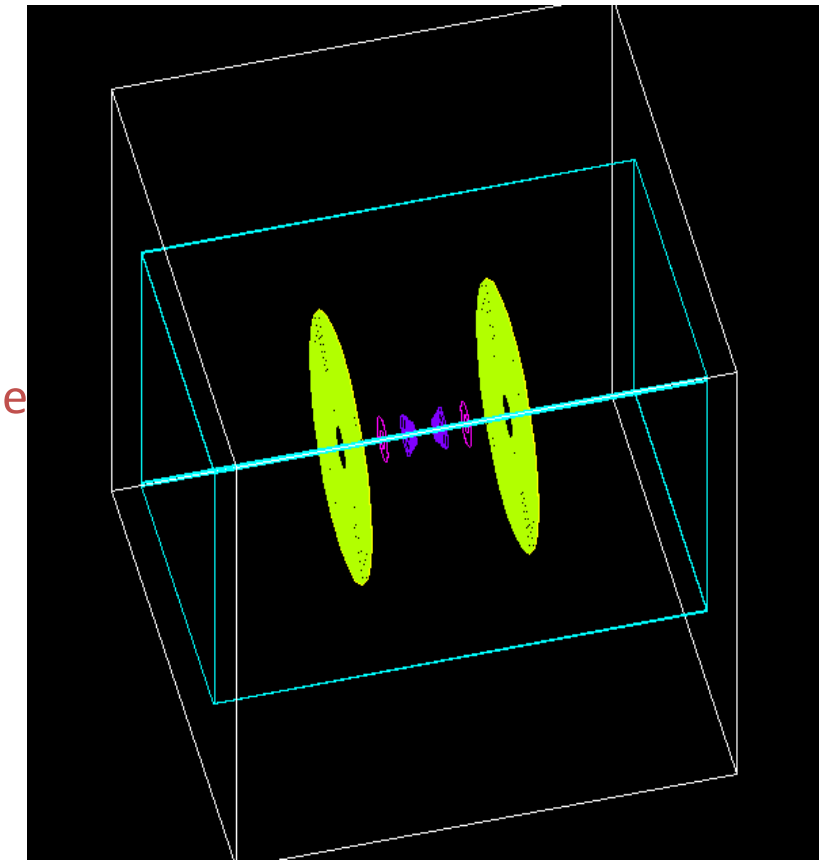
- All EM processes + decay, with separate production cuts for γ, e+, e- (use for shower studies)
- Detector response: E deposit, track length in absorber and gap
- Visualization tutorial
- Random number seed handling



Peter Gumplinger (TRIUMF)
2010_12_07_1340_1410_OnlineDocExamples.ppt

Novice Example N04

- simplified collider detector
  - all kinds of volume definitions
  - Readout geometry
- events from PYTHIA primary generator (HEPEvtInterface)
- full set of EM + hadronic processe
- non-uniform magn. field
- event filtering by using stacking mechanism
- defined user commands



Peter Gumplinger (TRIUMF)
2010_12_07_1340_1410_OnlineDocExamples.ppt

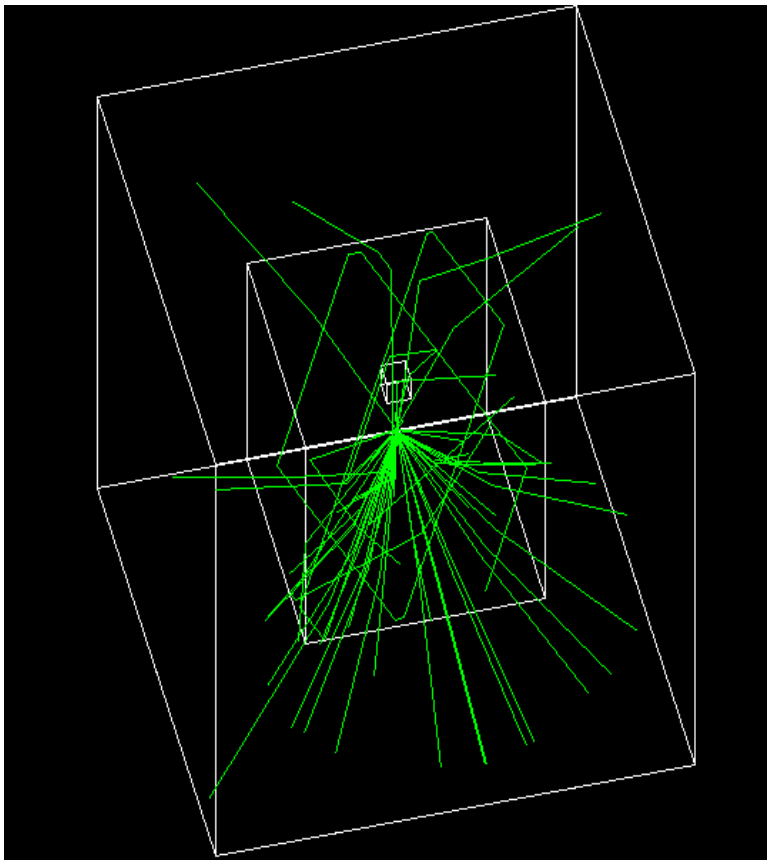
Novice Example N05

- Fast simulation with parameterized showers
  - ghost volume for shower parametrisation
  - EM showers (derived from G4VFastSimulationModel)
- EM physics only
  - Use of G4FastSimulationManagerProcess
- simplified collider detector geometry
  - drift chamber
  - EM, hadronic calorimeter
- sensitive detector

Peter Gumplinger (TRIUMF)
2010_12_07_1340_1410_OnlineDocExamples.ppt

Novice Example N06

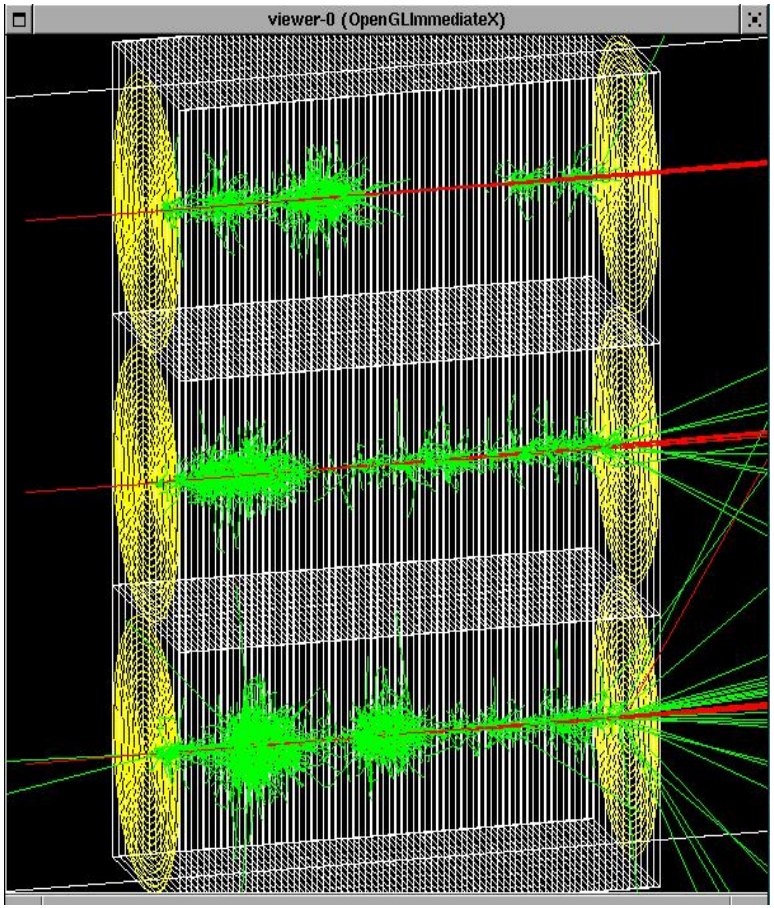
- Water Cerenkov detector with air "bubble"
- Materials
  - specification of optical properties
  - specification of scintillation spectra
- Physics
  - Optical processes
  - Generation of Cerenkov radiation, energy loss collected to produce scintillation photons
- Random number engine



Peter Gumplinger (TRIUMF)
2010_12_07_1340_1410_OnlineDocExamples.ppt

Novice Example N07

- 3 simplified sandwich calorimeters
- Derived run class and run action
- cylindrical ghost volume for scoring (primitive scorer and filters)
- Run-based (as opposed to event-based) hit accumulation
- changing geometries without rebuilding world (dynamic geometry setups between runs)
- setting different secondary production cuts for each geometrical region using G4Region



演習3:自力でN03をコンパイル

- いままでの作業を繰り返して、自力でN03をコピー、コンパイルしてみて下さい。
- コンパイルできたら、以下のコマンドを打ってみよう。

```
$ ./exampleN03
(ウィンドウが開くので、画面下のSessionのところに以下のコマンドを入力

/control/execute run2.mac
（終わったらexitと入力すればプログラム終了）

$ ./exampleN03 run2.mac  (batchモードで動かす方法）
```

Geant4のジオメトリ

プログラムを読む/書く時の注意

- 残念ながらGeant4はもっとも読みにくいコードの一つ
  - Naming Convention が統一されておらず、何がデータメンバで、何が定数なのか定数名だけでは判別不能
  - 自分で書くときは、Naming Conventionを決めよう！
    - 自分でルールを作る必要はなく、既にあるものを用いるのがお手軽
  - protectedメンバーは極力定義しない（カプセル化）
- 赤の他人が読んで分かるプログラムを書く事が、バグを減らす鉄則です（一年経ったら自分でも何書いたか忘れます）。コメントも大事ですが、Coding Conventionは多分それ以上に大事。(プログラムの汚さはコメントでカバーできない！）

Naming Conventionの例

- ROOT方式（Quasi-Hungarian）
  - ローカル変数は全て小文字始まり
  - データメンバは全てf(fieldの意)で始まり、その次の文字は必ず大文字
  - グローバル変数はgで始まり、その次は必ず大文字
  - 定数はkで始まり、その次は必ず大文字
  - 詳しくは以下を参照http://root.cern.ch/drupal/content/c-coding-conventions

- C++ Standard Library
  - ローカル変数は小文字のみ、語句をアンダースコアで繋ぐ
  - データメンバは変数名の最後に_（アンダースコア）をつける
  - ただし、最後に__（ダブルアンダースコア）、頭に_＋大文字、は予約語なので使用しないこと（例：value__, _Valiue)

- C++の場合、ユーザーが作る関数は常に大文字始まりにするケースが多い（Javaは必ず小文字始まりでないといけない、という言語の決まりがあるので、Java出身の人のコードはこの限りではない。）

- （経験上）ローカル変数、データメンバ、定数またはグローバル変数、の三種類くらいは変数名だけでわかるようにしておいた方が、いろいろ便利です。

ファイル、変数の検索

- Webで検索してもよいが、ネットワークが繋がらない時のためにお手製検索スクリプトが使える。

```
$ export G4SOURCES=/data/g4/4.9.5p01/geant4.9.5.p01/
$ /data/g4/utils/findG4file [検索ファイル文字列]
$ /data/g4/utils/findG4text [検索文字列]
```

- ＊は使えないが、いずれも部分マッチをするので、検索文字列の一部を指定してもよい。
- G4SOURCES環境変数を設定する必要がある。検索は、$G4SOURCES以下しか行わない。
- 自分の$HOME/binなどにコピーしても使用可。

N03を例題に…

- 検出器を定義するクラス
  - N03/include/DetectorConstruction.hh

```
#include "G4VUserDetectorConstruction.hh"
#include "globals.hh”
…
class DetectorConstruction : public G4VUserDetectorConstruction
{
```

- G4VUserDetectorConstructionを継承しなくてはならない！

G4VUserDetectorConstruction

```
//
// $Id: G4VUserDetectorConstruction.hh,v 1.4 2001/07/11 10:08:33 gunter Exp $
// GEANT4 tag $Name: geant4-08-00-patch-01 $
//

#ifndef G4VUserDetectorConstruction_h
#define G4VUserDetectorConstruction_h 1

class G4VPhysicalVolume;

// class description:
//
//  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 it's Initialize() method is invoked.
//  The Construct() method must return the G4VPhysicalVolume pointer which represents
// the world volume.
//

class G4VUserDetectorConstruction
{
  public:
    G4VUserDetectorConstruction();
    virtual ~G4VUserDetectorConstruction();

  public:
    virtual G4VPhysicalVolume* Construct() = 0;
};

#endif
```

Makoto Asai (SLAC)
2010_12_08_0910_1050_Geometry1.ppt

Construct() should return the pointer of the world physical volume. The world physical volume represents all of your geometry setup.

Describe your detector

- Derive your own concrete class from G4VUserDetectorConstruction abstract base class.
- Implement the method Construct()
  1) Construct all necessary materials
  2) Define shapes/solids
  3) Define logical volumes
  4) Place volumes of your detector geometry
  5) Associate (magnetic) field to geometry *(optional)*
  6) Instantiate sensitive detectors / scorers and set them to corresponding volumes *(optional)*
  7) Define visualization attributes for the detector elements *(optional)*
  8) Define regions *(optional)*
- Set your construction class to G4RunManager
- It is suggested to modularize Construct() method w.r.t. each component or sub-detector for easier maintenance of your code.

Makoto Asai (SLAC)
2010_12_08_0910_1050_Geometry1.ppt

N03の関数(一部）

```
public:
   void SetAbsorberMaterial (G4String);
   void SetAbsorberThickness(G4double);
   void SetGapMaterial (G4String);
   void SetGapThickness(G4double);
   void SetCalorSizeYZ(G4double);
   void SetNbOfLayers (G4int);
   void SetMagField(G4double);

   G4VPhysicalVolume* Construct();

   void UpdateGeometry();

private:
   void DefineMaterials();
   void ComputeCalorParameters();
   G4VPhysicalVolume* ConstructCalorimeter();
```

あとでジオメトリをコマンドライン等で変えたい場合のための布石（変えないならハードコードでも良し）

この中でジオメトリの定義は全部終える

Construct()の中で呼ばれる

Materialを定義しよう(1)

- N03/src/DetectorConstruction.ccを見る
- 数値代入するときは必ず単位をつける

```
void DetectorConstruction::DefineMaterials()
{
G4String symbol;            //a=mass of a mole;
G4double a, z, density;     //z=mean number of protons;
G4int iz, n;                //iz=number of protons  in an isotope;
                            // n=number of nucleons in an isotope;

// Elementの定義
G4Element* H  = new G4Element("Hydrogen",symbol="H" , z= 1., a= 1.01*g/mole);
G4Element* C  = new G4Element("Carbon"  ,symbol="C" , z= 6., a= 12.01*g/mole);
G4Element* N  = new G4Element("Nitrogen",symbol="N" , z= 7., a= 14.01*g/mole);
G4Element* O  = new G4Element("Oxygen"  ,symbol="O" , z= 8., a= 16.00*g/mole);
G4Element* Si = new G4Element("Silicon",symbol="Si" , z= 14., a= 28.09*g/mole);
```

単位について

- Geant4の内部単位に合わせるため、数値を代入するときは必ず単位とセットで。
- 単位の付け方は、数字*単位。
- 使える単位はG4UnitsTable.ccに定義。
- G4UnitsTable.ccを探してみよう。

```
$ export G4SOURCES=/data/g4/4.9.5p01/geant4.9.5.p01/
$ /data/g4/utils/findG4file G4UnitsTable.cc
```

Materialを定義しよう(2)

- Elementから分子を作る
- G4Materialをnewで作ると、自動的にG4内部のMaterialのリストに登録される。

```
G4Material* SiO2 =    //化合物
new G4Material("quartz",density= 2.200*g/cm3, ncomponents=2);
SiO2->AddElement(Si, natoms=1);
SiO2->AddElement(O , natoms=2);


G4Material* Air =       // 混合物
new G4Material("Air"  , density= 1.290*mg/cm3, ncomponents=2);
Air->AddElement(N, fractionmass=0.7);
Air->AddElement(O, fractionmass=0.3);
```

この他はコードを参照。

演習３

- 鉄(Fe)を定義する
  - Z = 26
  - A = 55.847
  - density = 7.87 g/mol

- CompileしてrunしたらFeが見えるか？

Detectorを作ろう

- まず形　→　G4VSolidの子クラス
- 次に材質　→　G4LogicalVolume
- 最後に位置　→　G4VPhysicalVolumeの子クラス
- 位置の決定は、常にその外側にあるジオメトリのLogicalVolumeの内部座標系で定義される
- 一番外側のVolumeは”World”
- これ以降１５ページ

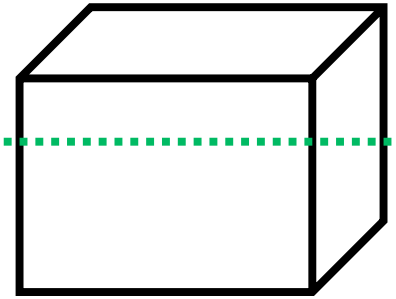
Makoto Asai (SLAC)
2010_12_08_0910_1050_Geometry1.ppt

Define detector geometry

- Basic strategy

```
G4VSolid* pBoxSolid =
  new G4Box("aBoxSolid",
      1.*m, 2.*m, 3.*m);
G4LogicalVolume* pBoxLog =
  new G4LogicalVolume( pBoxSolid,
    pBoxMaterial, "aBoxLog", 0, 0, 0);
G4VPhysicalVolume* aBoxPhys =
  new G4PVPlacement( pRotation,
    G4ThreeVector(posX, posY, posZ),
    pBoxLog, "aBoxPhys", pMotherLog,
    0, copyNo);
```

Logical volume :
Solid : shape and size
+ material, sensitivity, etc.

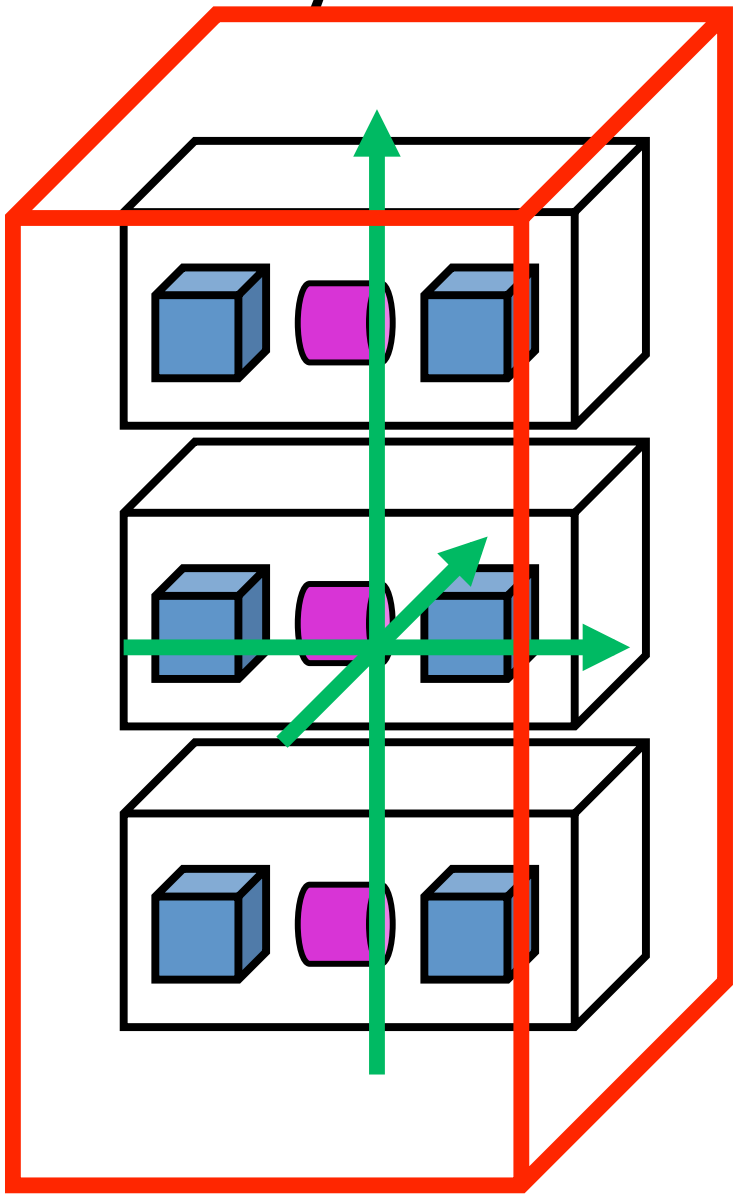


Physical volume :
+ rotation and position

- A volume is placed in its mother volume. Position and rotation of the daughter volume is described with respect to the local coordinate system of the mother volume. The origin of mother volume's local coordinate system is at the center of the mother volume.
  - Daughter volume cannot protrude from mother volume.

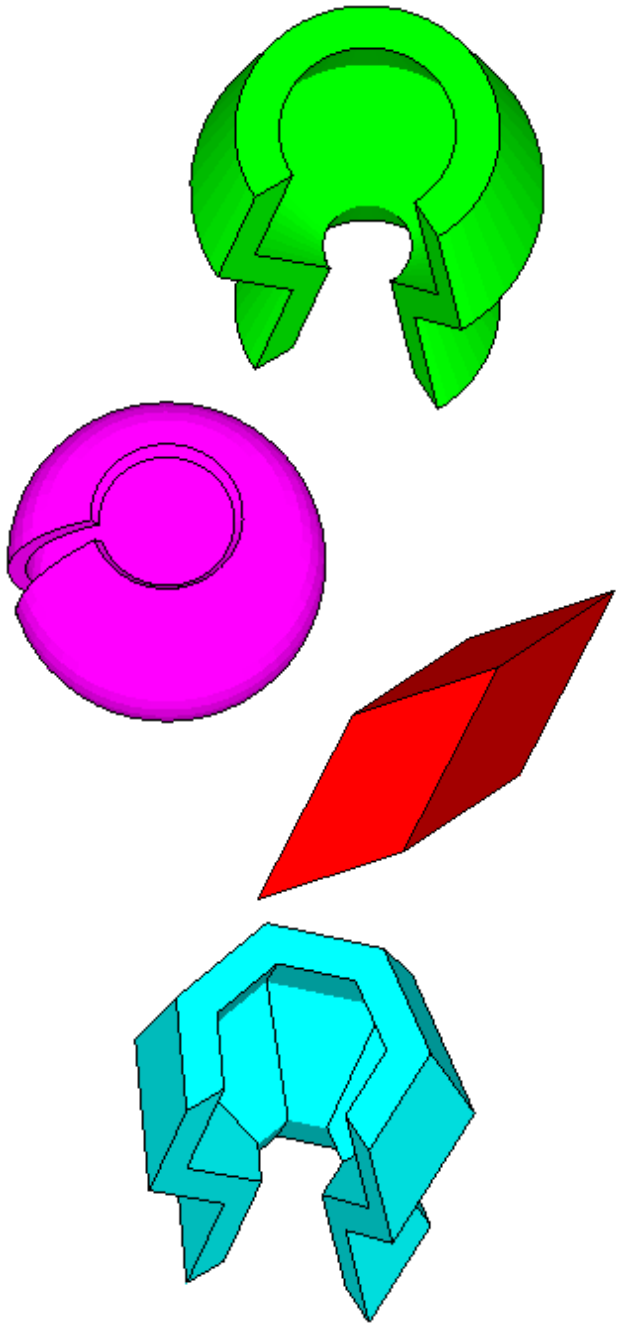
Geometrical hierarchy

- One logical volume can be placed more than once. One or more volumes can be placed to a mother volume.
- Note that the mother-daughter relationship is an information of G4LogicalVolume.
  - If the mother volume is placed more than once, all daughters are by definition appear in all of mother physical volumes.
- The world volume must be a unique physical volume which fully contains all the other volumes.
  - The world volume defines the global coordinate system. The origin of the global coordinate system is at the center of the world volume.
  - Position of a track is given with respect to the global coordinate system.



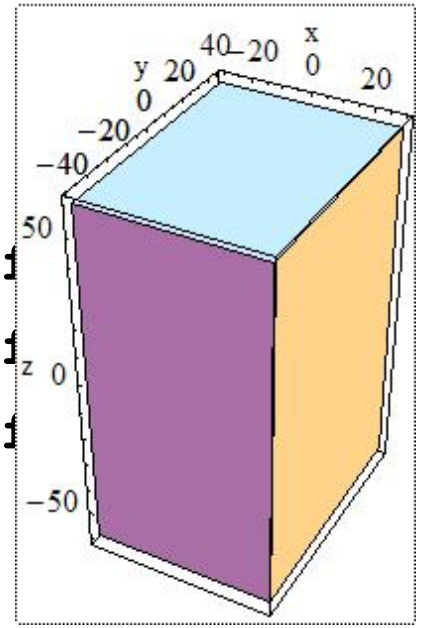
Solids

- Solids defined in Geant4:
  - CSG (Constructed Solid Geometry) solids
    - G4Box, G4Tubs, G4Cons, G4Trd, ...
    - Analogous to simple GEANT3 CSG solids
  - Specific solids (CSG like)
    - G4Polycone, G4Polyhedra, G4Hype, ...
  - BREP (Boundary REPresented) solids
    - G4BREPSolidPolycone, G4BSplineSurface, ...
    - Any order surface
  - Boolean solids
    - G4UnionSolid, G4SubtractionSolid, ...

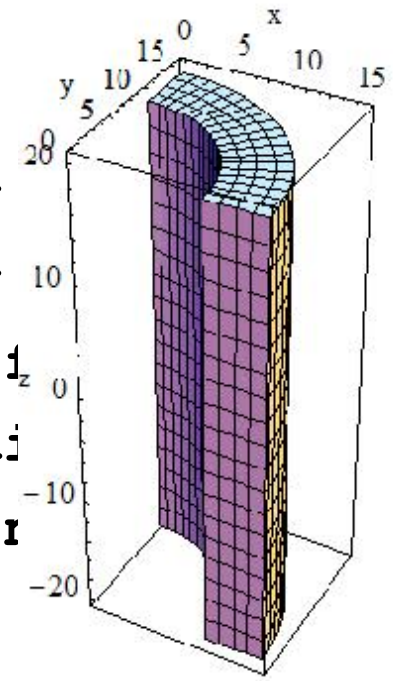


CSG: G4Box, G4Tubs

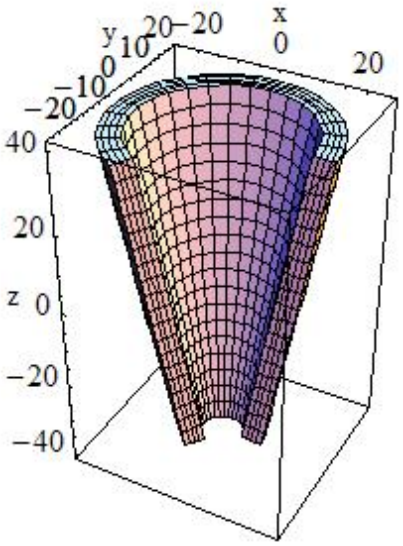
```
G4Box(const G4String &pname,    // name
            G4double half_x,    // X half
            G4double half_y,    // Y half
            G4double half_z);   // Z half
```



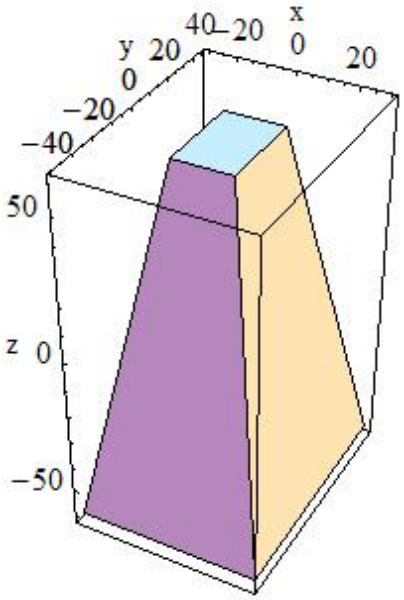

```
G4Tubs(const G4String &pname,   // name
             G4double  pRmin,   // inner
             G4double  pRmax,   // outer
             G4double  pDz,     // Z half
             G4double  pSphi,   // starti
             G4double  pDphi); // segmer
```



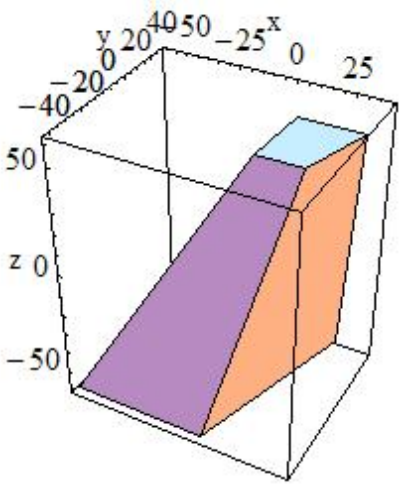
Other CSG solids



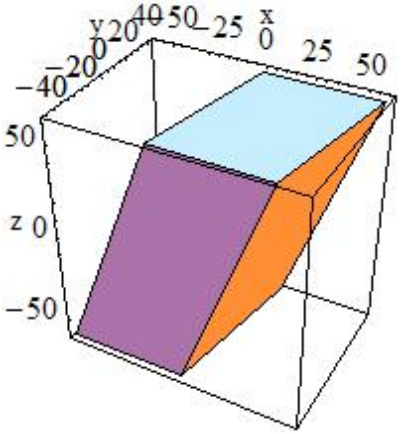
G4Cons



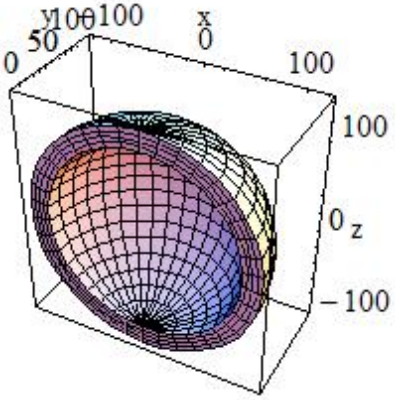
G4Trd



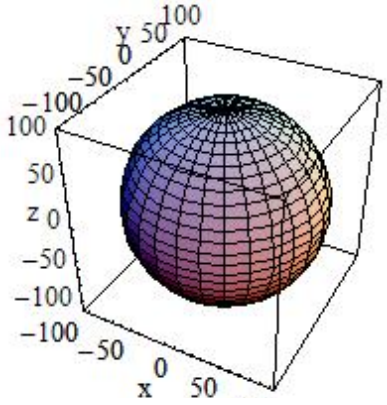
G4Trap



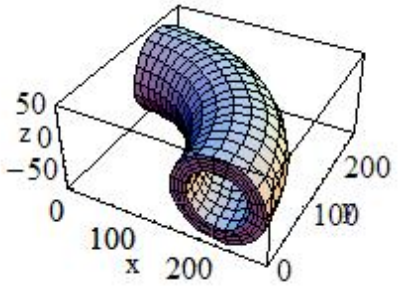
G4Para
(parallelepiped)



G4Sphere



G4Orb
(full solid sphere)

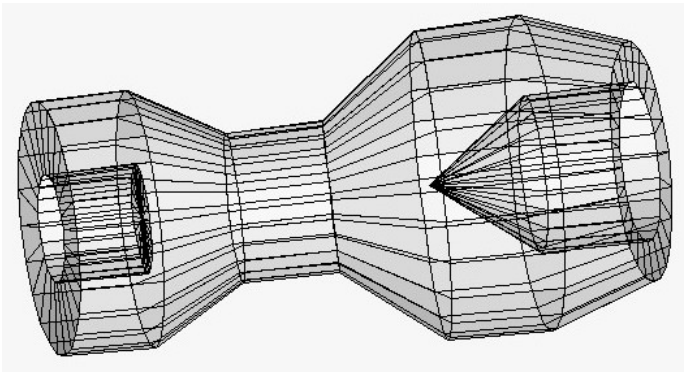


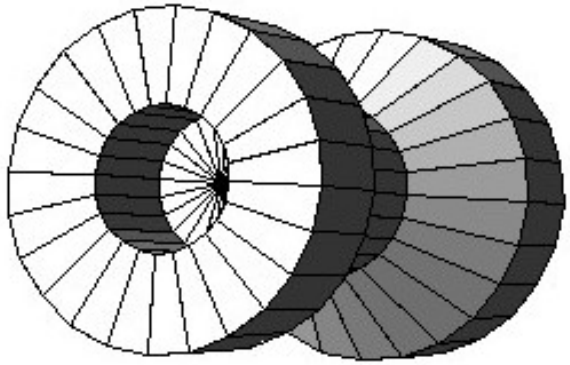
G4Torus

Consult to
[Section 4.1.2 of Geant4 Application Developers Guide](#) for all available shapes.

Specific CSG Solids: G4Polycone

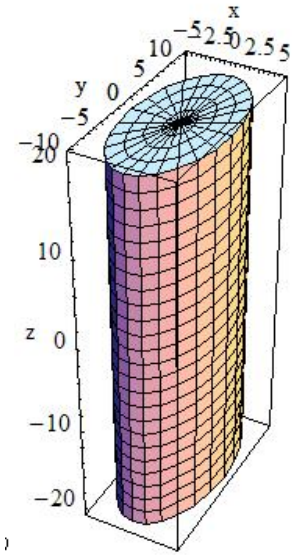
```
G4Polycone(const G4String&
   pName,
              G4double phiStart,
              G4double phiTotal,
              G4int numRZ,
              const G4double r[],
              const G4double z
   []);
```



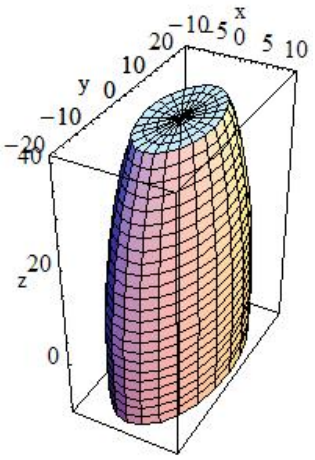


- **numRZ** - numbers of corners in the **r,z** space
- **r, z** - coordinates of corners

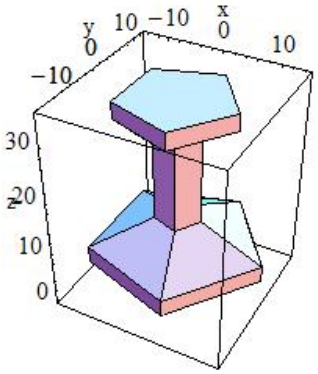
Other Specific CSG solids



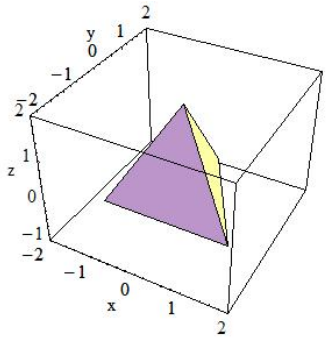
G4EllipticalTube



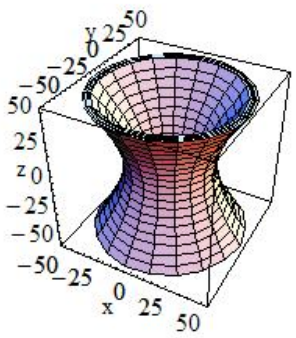
G4Ellipsoid



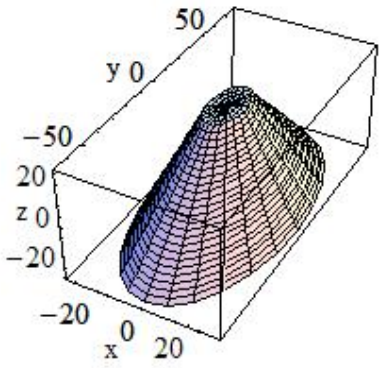
G4Polyhedra



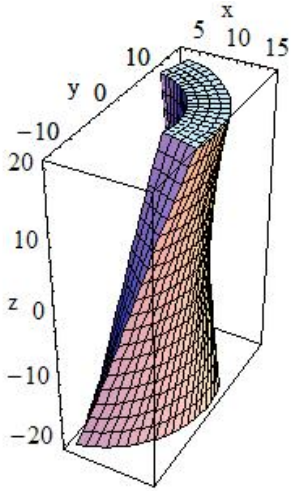
G4Tet
(tetrahedra)



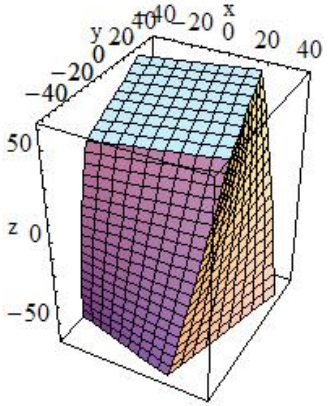
G4Hype



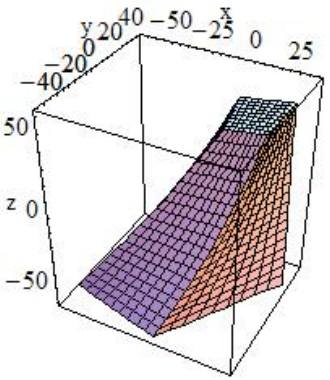
G4EllipticalCone



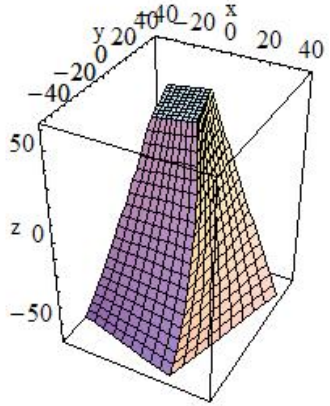
G4TwistedTubs



G4TwistedBox



G4TwistedTrap



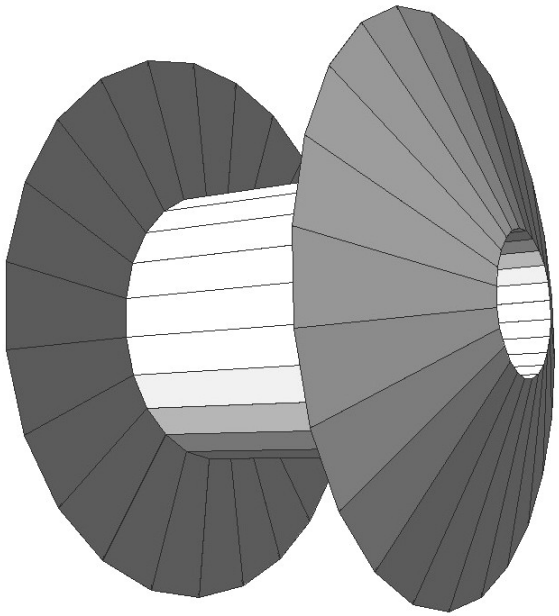
G4TwistedTrd

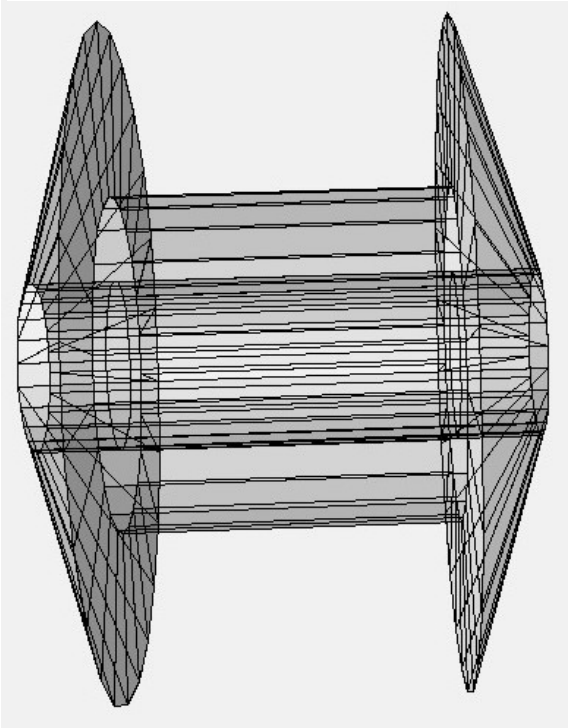
Consult to
[Section 4.1.2 of Geant4 Application Developers Guide](#) for all available shapes.

BREP Solids

- *BREP = Boundary REPresented Solid*
- Listing all its surfaces specifies a solid
  - e.g. 6 planes for a cube
- Surfaces can be
  - planar, 2nd or higher order
    - elementary BREPS
  - Splines, B-Splines,

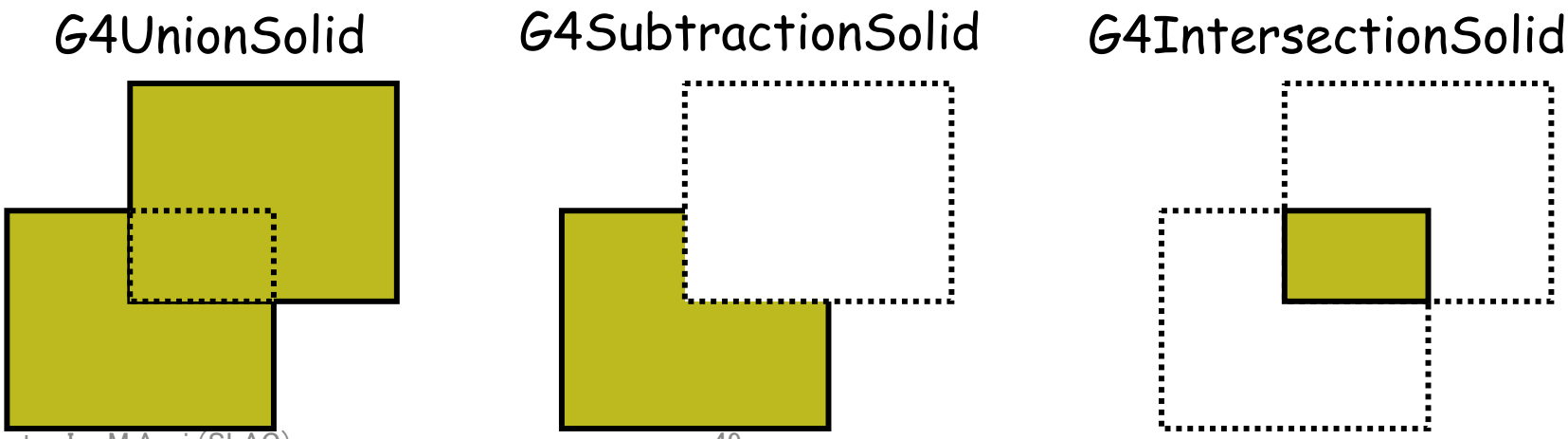
    NURBS (Non-Uniform B-Splines)
    - advanced BREPS
- Few elementary BREPS pre-defined
  - box, cons, tubs, sphere, torus, polycone, polyhedra
- Advanced BREPS built through CAD systems

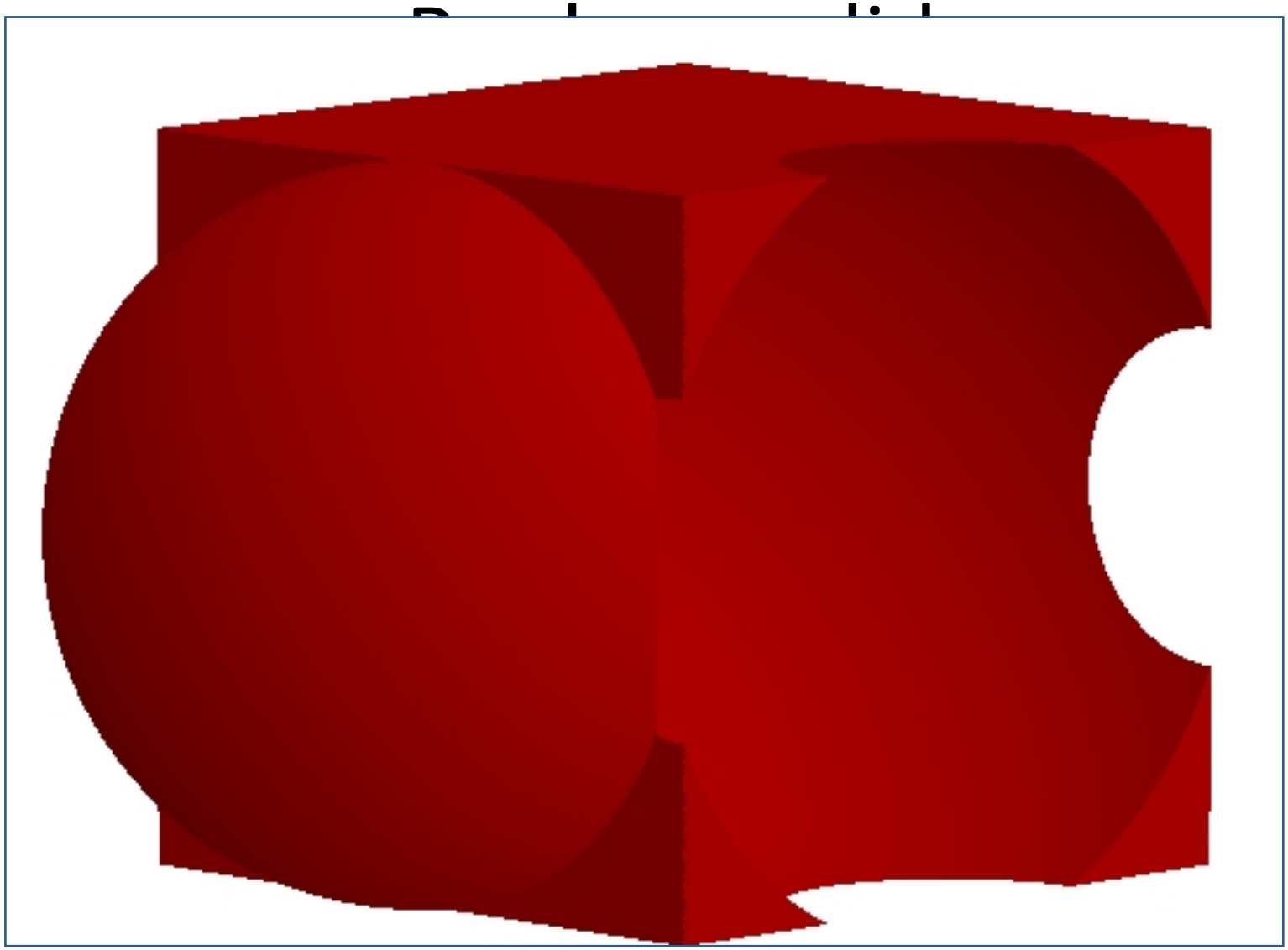




Boolean Solids

- Solids can be combined using boolean operations:
  - **`G4UnionSolid, G4SubtractionSolid, G4IntersectionSolid`**
  - Requires: 2 solids, 1 boolean operation, and an (optional) transformation for the 2nd solid
  - 2nd solid is positioned relative to the coordinate system of the 1st solid
  - Result of boolean operation becomes a solid. Thus the third solid can be combined to the resulting solid of first operation.
- Solids to be combined can be either CSG or other Boolean solids.
- Note: tracking cost for the navigation in a complex Boolean solid is proportional to the number of constituent CSG solids





Boolean Solids - example

```
G4VSolid* box = new G4Box("Box",50*cm,60*cm,40*cm);
G4VSolid* cylinder
 = new G4Tubs("Cylinder",0.,50.*cm,50.*cm,0.,2*M_PI*rad);
G4VSolid* union
 = new G4UnionSolid("Box+Cylinder", box, cylinder);
G4VSolid* subtract
 = new G4SubtractionSolid("Box-Cylinder", box, cylinder,
       0, G4ThreeVector(30.*cm,0.,0.));
G4RotationMatrix* rm = new G4RotationMatrix();
rm->RotateX(30.*deg);
G4VSolid* intersect
   = new G4IntersectionSolid("Box&&Cylinder",
       box, cylinder, rm, G4ThreeVector(0.,0.,0.));
```

- The origin and the coordinates of the combined solid are the same as those of the first solid.

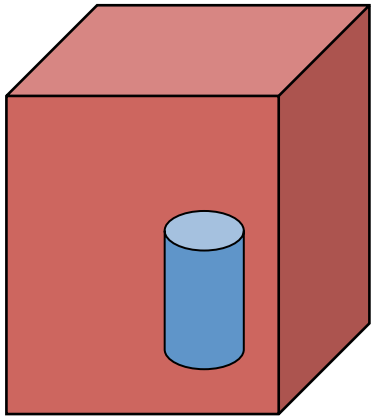
G4LogicalVolume

```
G4LogicalVolume(G4VSolid* pSolid,
                G4Material* pMaterial,
                const G4String &name,
                G4FieldManager* pFieldMgr=0,
                G4VSensitiveDetector* pSDetector=0,
                G4UserLimits* pULimits=0);
```

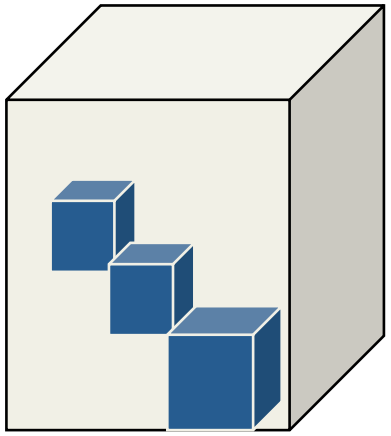
- Contains all information of volume except position and rotation
  - Shape and dimension (G4VSolid)
  - Material, sensitivity, visualization attributes
  - Position of daughter volumes
  - Magnetic field, User limits, Region
- Physical volumes of same type can share the common logical volume object.
- The pointers to solid must NOT be null.
- The pointers to material must NOT be null for tracking geometry.
- It is not meant to act as a base class.

Physical Volumes

- Placement volume : it is one positioned volume
  - One physical volume object represents one “real” volume.
- Repeated volume : a volume placed many times
  - One physical volume object represents any number of “real” volumes.
  - reduces use of memory.
  - Parameterised
    - repetition w.r.t. copy number
  - Replica and Division
    - simple repetition along one axis
- A mother volume can contain either
  - many placement volumes
  - or, one repeated volume



placement



repeated

Physical volume

- G4PVPlacement 1 Placement = One Placement Volume
  - A volume instance positioned once in its mother volume
- G4PVParameterised 1 Parameterized = Many Repeated Volumes
  - Parameterized by the copy number
    - Shape, size, material, sensitivity, vis attributes, position and rotation can be parameterized by the copy number.
    - You have to implement a concrete class of G4VPVParameterisation.
  - Reduction of memory consumption
  - Currently: parameterization can be used only for volumes that either

    a) have no further daughters, or

    b) are identical in size & shape (so that grand-daughters are safely fit inside).
  - By implementing G4PVNestedParameterisation instead of G4VPVParameterisation, material, sensitivity and vis attributes can be parameterized by the copy numbers of ancestors.

Physical volume

- G4PVReplica 1 Replica = Many Repeated Volumes
  - Daughters of same shape are aligned along one axis
  - Daughters fill the mother completely without gap in between.
- G4PVDivision 1 Division = Many Repeated Volumes
  - Daughters of same shape are aligned along one axis and fill the mother.
  - There can be gaps between mother wall and outmost daughters.
  - No gap in between daughters.
- G4ReflectionFactory 1 Placement = a pair of Placement volumes
  - generating placements of a volume and its reflected volume
  - Useful typically for end-cap calorimeter
- G4AssemblyVolume 1 Placement = a set of Placement volumes
  - Position a group of volumes

PVPlacementの注意点

- G4PVReplica 、G4PVDivision は、セグメント同士が隙間無く接しているものしかダメ(どちらかというと、1つのものを等分に分ける感覚)
- G4PVParameterised は、段々大きくなる、小さくなるなどの簡単なパラメトライズしかできない。
- G4AssemblyVolume はドローソフトのグループ化みたいな機能で、配置が終わったらグルーピングは消える(グループ解除)。
- 結局、自由度の高いG4PBPlacementを使う機会がもっとも多いと思われる。

演習４

- 検出器の大きさを変えてみる
- 検出器の数を増やしてみる

入射ビームの定義

どんなビームを入射するか？

- 粒子の種類
- ビームの大きさや数、方向など
- G4VUserPrimaryGeneratorActionを継承して作る

```
$less N03/src/PrimaryGeneratorAction.cc
```

- G4ParticleGunは、G4VPrimaryGeneratorを継承して作られたユーザークラスの一つなので、必ずしもこれを使わなくてもよい。（自分で一からビームクラスを書くこともできる）
- 一発ずつ打ち込むような場合は、G4ParticleGunでも十分使える（UIもお仕着せのものがある）

- GeneratePrimary関数は、1イベントに1回呼ばれる
- 最後にparticleGun->GeneratePrimaryVertex(anEvent)を呼ぶ事で、入射粒子がセットされる(実装はG4ParticleGun.ccを見よ)
- ただし、1イベント内で異なった粒子の重ね合わせをやりたい場合は、G4ParticleGunでは対処できないので、自分でビームクラスを作らなければならない。

```
void PrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
{
  //this function is called at the begining of event
  //
  G4double x0 = -0.5*(Detector->GetWorldSizeX());
  G4double y0 = 0.*cm, z0 = 0.*cm;
  if (rndmFlag == "on")
     {y0 = (Detector->GetCalorSizeYZ())*(G4UniformRand()-0.5);
      z0 = (Detector->GetCalorSizeYZ())*(G4UniformRand()-0.5);
     }
  particleGun->SetParticlePosition(G4ThreeVector(x0,y0,z0));

  particleGun->GeneratePrimaryVertex(anEvent);
}
```

この部分で粒子の種類やエネルギー、方向、入射位置を(イベントごとに)変えたりすれば、大気ミューオンビームも作れる

PhysicsList

Geant4内で起こさせる反応の定義

- 何も定義しないと、Geant4はビームを素通りさせて終わり
- どんな物理を組み込むかはユーザーの自由だが、よく使うセットはExampleでも使われている
- PhysicsListの選定はG4を使用する上で大変重要だが、時間の都合で今回の講習では割愛。各自、2010年の講習のDay3, Day4**のスライドで自習のこと。**
- EMをやりたいなら、N03のPhysicsListでほとんどそのまま使える。（低エネルギー側は注意）

検出器の反応

どのタイミングで情報を記録するか？

- １イベント中にたまったエネルギー損失の総量を知りたい
  - EventAction, RunActionで対処可能(N03)
- それぞれの検出モジュールで、どの位置でどれだけのエネルギー損失があったか知りたい
  - SensitiveDetectorとHitクラスを作成（N02)

EventAction

```
class EventAction : public G4UserEventAction
{
public:
 …
  void AddAbs(G4double de, G4double dl) {EnergyAbs += de; TrackLAbs += dl;};
  void AddGap(G4double de, G4double dl) {EnergyGap += de; TrackLGap += dl;};

  void SetPrintModulo(G4int    val)  {printModulo = val;};

private:
   RunAction*  runAct;

   G4double  EnergyAbs, EnergyGap;
   G4double  TrackLAbs, TrackLGap;
```

EventAction

```
void EventAction::BeginOfEventAction(const G4Event* evt)
{
…
 // initialisation per event
 EnergyAbs = EnergyGap = 0.;
 TrackLAbs = TrackLGap = 0.;
}

void EventAction::EndOfEventAction(const G4Event* evt)
{
…
G4cout << "---> End of event: " << evtNb << G4endl;

   G4cout
     << "   Absorber: total energy: " << std::setw(7)
                                << G4BestUnit(EnergyAbs,"Energy")
     << "       total track length: " << std::setw(7)
…
```

RunActionなどでファイルオープンしておき、coutの変わりにそこに書き出せば、ASCIIファイルに書き出せる

SteppingAction

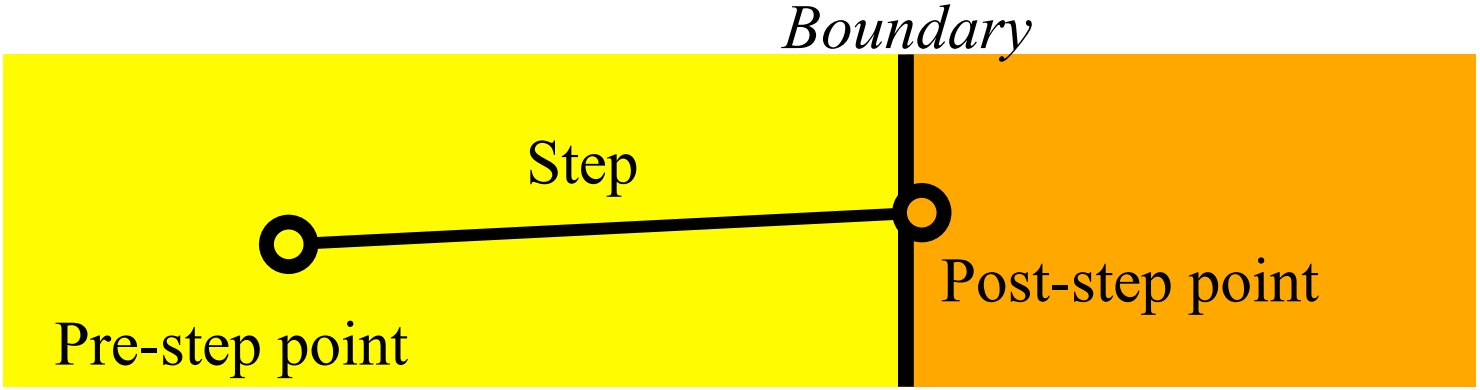
- Geant4が乱数をふって1ステップするごとに呼ばれる

```
void SteppingAction::UserSteppingAction(const G4Step* aStep) {
  // get volume of the current step
  G4VPhysicalVolume* volume
  = aStep->GetPreStepPoint()->GetTouchableHandle()->GetVolume();

  // collect energy and track length step by step
  G4double edep = aStep->GetTotalEnergyDeposit();

  G4double stepl = 0.;
  if (aStep->GetTrack()->GetDefinition()->GetPDGCharge() != 0.)
    stepl = aStep->GetStepLength();

  if (volume == detector->GetAbsorber()) eventaction->AddAbs(edep,stepl);
  if (volume == detector->GetGap())      eventaction->AddGap(edep,stepl);
```

Step

Makoto Asai (SLAC)
2010_12_09_1110_1150_Scoring2.ppt

- Step has two points and also “delta” information of a particle (energy loss on the step, time-of-flight spent by the step, etc.).
- Each point knows the volume (and material). In case a step is limited by a volume boundary, the end point physically stands on the boundary, and it logically belongs to the next volume.
- Note that you must get the volume information from the “PreStepPoint”.



Sensitive Detectorを作る場合

- N02に例題がある
  - SensitiveDetector(SD)の作成
  - 記録する情報をまとめたHitクラスを作成
- SDの作成はほぼおまじない
  - 下手に弄ると危険、N02のExN02TrackerSDクラスをコピーし、適当にリネーム、ProcessHits関数だけ書き換えるのが無難。
  - HitクラスもExN02TrackerHitクラスをコピー＆リネーム、保存したい情報をデータメンバに書き足す。
  - SDも1イベントの最後に情報を書き出すことが出来る(EndOfEvent関数）

SDを登録する

- 例題は ExN02DetectorConstruction.cc
- SDはLogicalVolumeに対して登録する

```
#include "ExN02TrackerSD.hh”
#include "G4SDManager.hh"

G4VPhysicalVolume* ExN02DetectorConstruction::Construct()
{
…
  G4SDManager* SDman = G4SDManager::GetSDMpointer();

  G4String trackerChamberSDname = "ExN02/TrackerChamberSD";
  ExN02TrackerSD* aTrackerSD = new ExN02TrackerSD( trackerChamberSDname );
  SDman->AddNewDetector( aTrackerSD );
  logicChamber->SetSensitiveDetector( aTrackerSD );

…
}
```

LogicalVolumeへのポインタ（シンチレータのLVなど）

演習２

- ３）自分のやりたいシミュレーションに従って、どこを変更したらよいか書き出してみよう。
- ４）どこをどうしたら良いかわからなかったら、質問して下さい。今日中に道筋が立てば、目標達成！
- 後日の質問は、hoshina@eri.u-tokyo.ac.jpへ。

Appendix

How To Install G4

```
# install qt4 and cmake in advance

$ mkdir /data/g4/4.9.5p01
$ cd /data/g4/4.9.5p01

(download geant4 source file)

$ mkdir RHEL5
$ mkdir install
$ cd RHEL5

$ ../../cmake-2.8.8-Linux-i386/bin/cmake -DGEANT4_INSTALL_EXAMPLES=ON -
DCMAKE_INSTALL_PREFIX=../install -DGEANT4_INSTA
LL_DATA=ON -DGEANT4_USE_QT=ON -DGEANT4_USE_RAYTRACER_X11=ON -
DGEANT4_USE_OPENGL_X11=ON -DBUILD_SHARED_LIBS=OFF -DBUILD
_STATIC_LIBS=ON ../geant4.9.5.p01

$ make install
```