Physics Listの使用方法

演習パッケージ： P04_PhysicsLists

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. 今までの演習ではHEP/Space用に適したPhysics ListとしてFTFP_BERTを使ってきたが、それを他のPhysics Listsに変える方法を学ぶ
2. 具体例としてRadioactive Sourceが扱えるPhysics Listに切り替える方法を学ぶ
3. 既存のPhysics Listsをカストマイズする手法としてG4PhysListFactoryの使い方を学ぶ
4. ユーザ独自のPhysics Listsの作成方法を学ぶ

演習の準備

P04_PhysicsLists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P04_PhysicsLists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/P04_PhysicsLists .
```

先ず、演習のルート・ディレクトリに行く

ユーザ作業root dir

P04_PhysicsLists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 P04_PhysicsLists
source/    util/
$ ls P04_PhysicsLists/source
Application_Main.cc     CMakeLists.txt     include/   src/
```

[注1] mainプログラムは以前の演習(P01_FirstStep_Vis)で使用したものと同じ

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

```
$ ls P04_PhysicsLists/source/src
Geometry.cc          PrimaryGenerator.cc      UserActionInitialization.cc
```

[注2] ジオメトリファイルは以前の演習で使ったBGO_One

入射粒子ファイルは以前の演習で使ったGun_One

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]

```
$ ls P04_PhysicsLists/util
G4Codes/          Help/          Macros/    Macros_PrimaryGen/
```

本演習で使う様々なGeant4マクロが入っている

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

P04_PhysicsLists: Shielding

Radioactive Sourceの扱い

P04_PhysicsLists: Shieldingプログラムの概要

■ 演習プログラムの目的

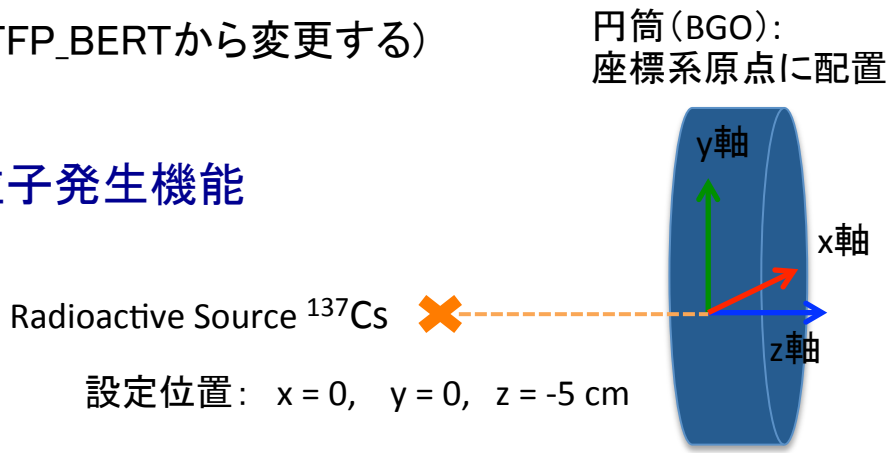
- 今までの演習ではPhysics Listとして’FTFP_BERT’を使ってきた。これは、中高エネルギーのHEP/Spaceの多くのシミュレーションに応用できるものである。しかし、シミュレーションで’activation’を考慮したい場合などに必要となる放射性崩壊が組み込まれていない。この演習ではPhysics Listの選択方法の学習として、’FTFP_BERT’を放射性崩壊が扱える’Shielding’に切り替える方法を学ぶ
- Geant4が配布している標準的なPhysics List同士を切り替えることは簡単であることを学ぶ

  ［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

■ プログラムの構成

- mainプログラム： P01_FirstStep_Visをエディットして使用する
- ジオメトリ： P02_Geometryで使ったBGO検出器
- PhysicsList： Shielding（FTFP_BERTから変更する）
- PrimaryGenerator: Particle Gun

■ 組み込まれているジオメトリと粒子発生機能



P04_PhysicsLists: Shielding プログラムの作成

はじめに:

P04_PhysicsLists: Shieldingプログラムを作成するには:

1) プログラム作成作業は今まで使ってきた標準のmainプログラム(*P01_FirstStep_Vis*)を変更することで進める
2) 本演習で変更が必要なファイルはこのmainファイルのみで、それ以外は変更不要

課題:1　現在のmainファイル(Application_Main.cc)を確認する

1) 前のステップでコピーしたP04_PhysicsListsに用意されているApplication_Main.ccの内容を見てみる

   ← この'Application_Main.cc'は'*FTFP_BERT*'が使われていることを確認すること

```
$ cd ~/Geant4Tutorial20171129/UserWorkDir/P04_PhysicsLists     ← CBDir (Current
$ cd source                                                      Base Directory)
$ less Application_Main.cc
```

P04_PhysicsLists: Shieldingのmainプログラムの実装

課題:2　現mainプログラムをShielding用にするために必要な変更箇所

```
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// Geant4 Application: Tutorial course for Hep/Medicine Users
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#include "Geometry.hh"
#include "UserActionInitialization.hh"

#include "G4RunManager.hh"
#include "G4UImanager.hh"
#include "G4VisExecutive.hh"
#include "G4UIExecutive.hh"
#include "Shielding.hh"

//------------------------------------------------------------------------------
int main( int argc, char** argv )
//------------------------------------------------------------------------------
{
// Construct the default run manager
   G4RunManager * runManager = new G4RunManager;

// Set up mandatory user initialization: Geometry
   runManager->SetUserInitialization( new Geometry );

// Set up mandatory user initialization: Physics-List
   runManager->SetUserInitialization( new Shielding );

// Set up user initialization: User Actions
   runManager->SetUserInitialization( new UserActionInitialization );

// Initialize G4 kernel
   runManager->Initialize();

// Create visualization environment
   G4VisManager* visManager = new G4VisExecutive;
   visManager->Initialize();

// Start interactive session
   G4UIExecutive* uiExec = new G4UIExecutive(argc, argv);
   G4UImanager*   uiManager = G4UImanager::GetUIpointer();
   uiManager->ApplyCommand("/control/execute GlobalSetup.mac");
   uiExec->SessionStart();

// Job termination
   delete uiExec;
   delete visManager;
   delete runManager;

   return 0;
}
```

Shielding用Mainプログラム

P01_FirstStep_Visとの違い

もとは'FTFP_BERT.hh'であった:

もとは'FTFP_BERT'であった:

Physic Listを変更するには:

上の2ヶ所のPhysics List名前を変更するだけで完了

P04_PhysicsLists: Shieldingのmainプログラムの実装 – つづき

課題:3 現在のApplication_Main.ccをエディットしてShieldingを実装する

1) 現在のApplication_Main.ccとShielding用(模範解答)の相違をcolordiffで端末に表示する

```
$ cd ~/Geant4Tutorial20171129/UserWorkDir/P04_PhysicsLists/source
$ colordiff -y --width=200  Application_Main.cc \
                            ../util/G4Codes/P04_PhysicsLists.cc_Shielding
```

現在のファイル

```
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// Geant4 Application: Tutorial course for Hep/Medicine Users
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#include "Geometry.hh"
#include "UserActionInitialization.hh"

#include "G4RunManager.hh"
#include "G4UImanager.hh"
#include "G4VisExecutive.hh"
#include "G4UIExecutive.hh"
#include "FTFP_BERT.hh"

//------------------------------------------------------------------------------
  int main( int argc, char** argv )
//------------------------------------------------------------------------------
{
// Construct the default run manager
   G4RunManager* runManager = new G4RunManager;

// Set up mandatory user initialization: Geometry
   runManager->SetUserInitialization( new Geometry );

// Set up mandatory user initialization: Physics-List
   runManager->SetUserInitialization( new FTFP_BERT );

// Set up user initialization: User Actions
   runManager->SetUserInitialization( new UserActionInitialization );
```

変更ライン印

Shielding用ファイル

```
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// Geant4 Application: Tutorial course for Hep/Medicine Users
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#include "Geometry.hh"
#include "UserActionInitialization.hh"

#include "G4RunManager.hh"
#include "G4UImanager.hh"
#include "G4VisExecutive.hh"
#include "G4UIExecutive.hh"
#include "Shielding.hh"

//------------------------------------------------------------------------------
  int main( int argc, char** argv )
//------------------------------------------------------------------------------
{
// Construct the default run manager
   G4RunManager* runManager = new G4RunManager;

// Set up mandatory user initialization: Geometry
   runManager->SetUserInitialization( new Geometry );

// Set up mandatory user initialization: Physics-List
   runManager->SetUserInitialization( new Shielding );

// Set up user initialization: User Actions
   runManager->SetUserInitialization( new UserActionInitialization );
```

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

```
$ cd ~/Geant4Tutorial20171129/UserWorkDir/P04_PhysicsLists/source
$ gedit Application_Main.cc&
```

[注] もしエディットがうまくできない場合、以下のコマンドを実行
$ cp ../util/G4Codes/P04_PhysicsLists.cc_Shielding Application_Main.cc

P04_PhysicsLists: Shieldingプログラムアプリケーションのビルド

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

1） 現在のベースデレクトリに移動

```
$ cd ~/Geant4Tutorial20171129/UserWorkDir/P04_PhysicsLists
```

← CBDir (Current Base Directory)

2) buildデレクトリでビルドを実行　[注]

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

← ビルド作業のデレクトリ名は習慣的にbuildとする

← 作られたbuildに移動

cmakeの慣用句

[注]
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、buildを自動
完了することができる

./util/Help/Build_P04_Shielding.sh

Application_Main (Shielding)の実行

課題：5　Application_Main (Shielding)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
……/P04_PhysicsLists
$ mkdir TestBench
$ cd TestBench
$ cp ../util/Macros/* .
```

現在いるデレクトリを確認： P04_PhysicsLists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

2) 上でコピーしたprimaryGeneratorSetup.macを変更して^{137}Cs以下が使えるようにする

（以下のようにcolordiffを使って変更箇所を表示したのち、マクロファイルをエディトする）

```
$ colordiff -y --width=150 primaryGeneratorSetup.mac \
                ../util/Macros_PrimaryGen/primaryGeneratorSetup.mac_Cs137
$ gedit primaryGeneratorSetup.mac
```

現在のファイル

```
#------------------------------------------------------------------
# primaryGeneratorSetup.mac: Set up the primary generator environment
#  [Note] Geant4 Tutorial for Hep/Medicine Users
#------------------------------------------------------------------

## Particle type
/gun/particle geantino


## Kinematical information
/gun/energy    1.0 MeV
/gun/position  0.0  0.0  -5.0 cm
/gun/direction 0.0  0.0  +1.0
```

LBE用ファイル

```
#------------------------------------------------------------------
# primaryGeneratorSetup.mac: Set up the primary generator environment
#  [Note] Geant4 Tutorial for Hep/Medicine Users
#------------------------------------------------------------------

## Particle type
/gun/particle ion
/gun/ion 55 137

## Kinematical information
/gun/energy 0.0 MeV                # For stationary decay
/gun/position 0.0 0.0 -5.0 cm
```

^{137}Csが設定される

追加ライン印 (>)

削除ライン印 (<)

[注]　もしエディットがうまくできない場合、以下のコマンドを実行

```
$ cp ../util/Macros_PrimaryGen/primaryGeneratorSetup.mac_Cs137 primaryGeneratorSetup.mac
```

3) アプリケーション実行してQtウインドを開く

```
$ ../bin/Application_Main
```

TestBenchデレクトリでApplication_Main実行

Qtウインドでアプリの動作を確認

課題：6　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

```
/run/beamOn 1
```

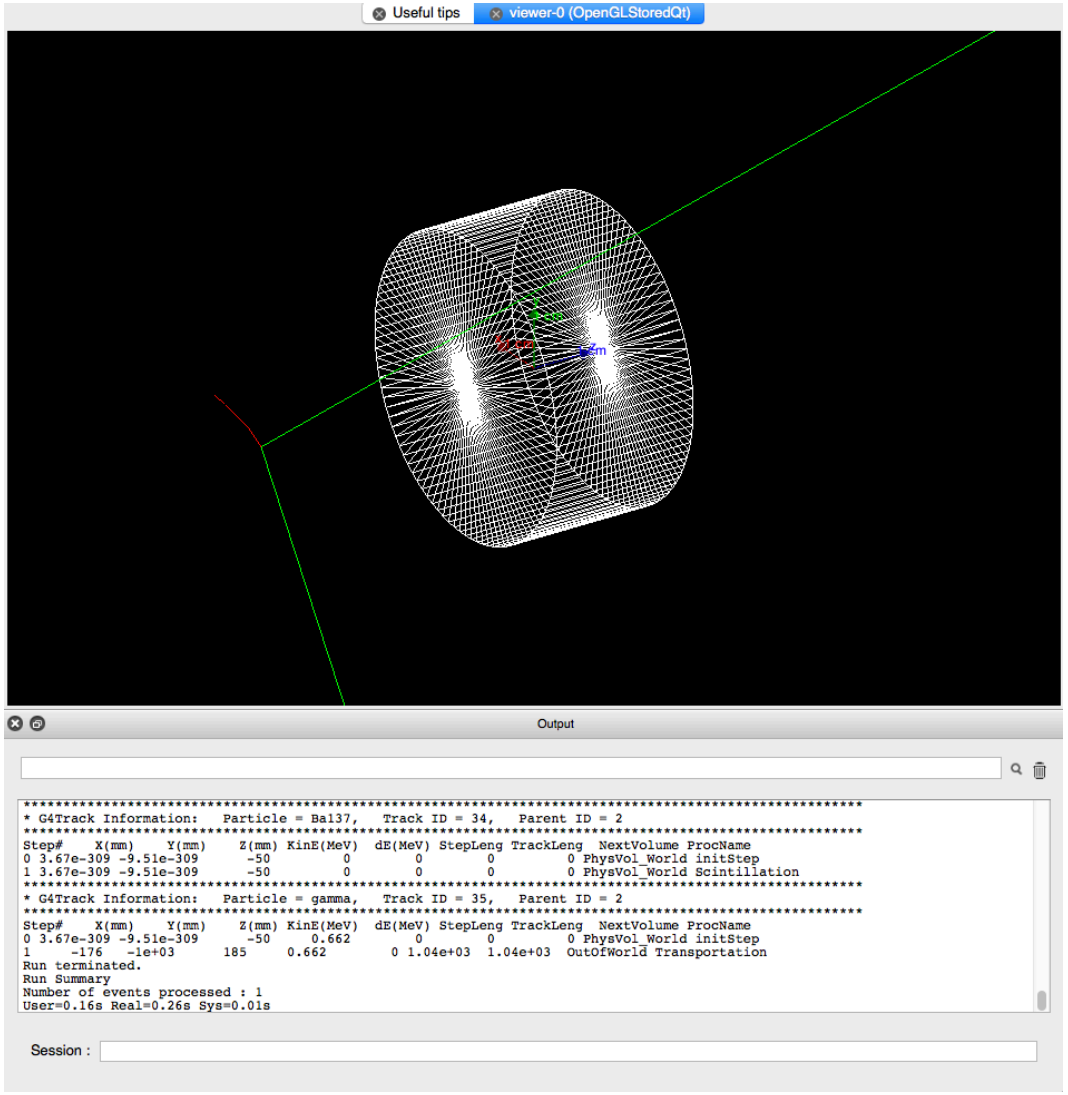
右図はその出力
（マウスで視点を移動させ見た目を調整している）

2) 事象発生数を増やしてみる

```
/run/beamOn 50
```

3) アプリ終了

```
exit
```



P04_PhysicsLists: Factory

Physics Listsのカストマイズ: G4PhysListFactoryを使う

P04_PhysicsLists: Factoryプログラムの概要

■ 演習プログラムの目的

- 前の演習では放射性崩壊を扱うために’FTFP_BERT’を’Shielding’に切り替た。
  この演習では’FTFP_BERT’を別のものに切り替えるのでなく、’FTFP_BERT’に
  放射性崩壊過程を追加する方法を学ぶ
- Physics Listの編集（この演習では物理過程の追加）には
  ’G4PhysListFactory’を使う
- この方法により、ユーザは physics listsを自由にカスタマイズできる

  ［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

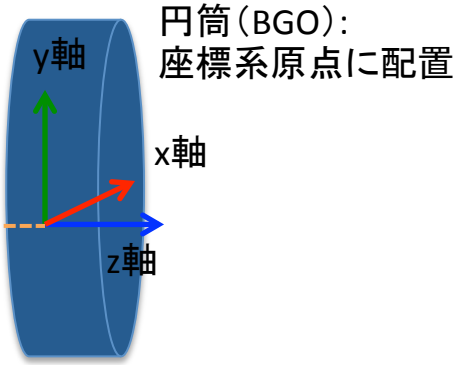
■ プログラムの構成

- mainプログラム： P04_PhysicsLists.cc_Factory
- ジオメトリ： P02_Geometryで使ったBGO検出器
- PhysicsList： FTFP_BERT + G4RadioactiveDecayPhysics

■ 組み込まれているジオメトリと粒子発生機能

Radioactive Source ^{137}Cs

設定位置： x = 0, y = 0, z = -5 cm



円筒（BGO）：
座標系原点に配置

P04_PhysicsLists: Factoryを使ったアプリケーションのビルド

課題：1　P04_PhysicsLists.cc_Factoryをmainプログラムとしてアプリケーションをビルド

1）ソースファイルP04_PhysicsLists.cc_FactoryをApplication_Main.ccにコピー

```
$ cd ~/Geant4Tutorial20171129/UserWorkDir/P04_PhysicsLists
$ cd source
$ cp ../util/G4Codes/P04_PhysicsLists.cc_Factory Application_Main.cc
overwrite Application_Main.cc? (y/n [n]) y
```

CBDir (Current Base Directory)

2) buildデレクトリでビルドを実行　[注]

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

CBDir (Current Base Directory)へ戻る

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

[注]
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_P04_Factory.sh

Application_Main (Factory)の実行

課題：2　Application_Main (Factory)を実行する

1) プログラム実行を行う作業デレクトリに移動

```
$ pwd
……/P04_PhysicsLists
$ cd TestBench
$ ../bin/Application_Main
```

現在いるデレクトリを確認： P04_PhysicsLists
でなければ、そこに移動する
作業デレクトリに移動
TestBenchデレクトリでApplication_Main実行

Qtウインドでアプリの動作を確認

課題:3　UI コマンドで事象発生

1) 以下のUIコマンドを先ず入力

```
/run/beamOn 1
```

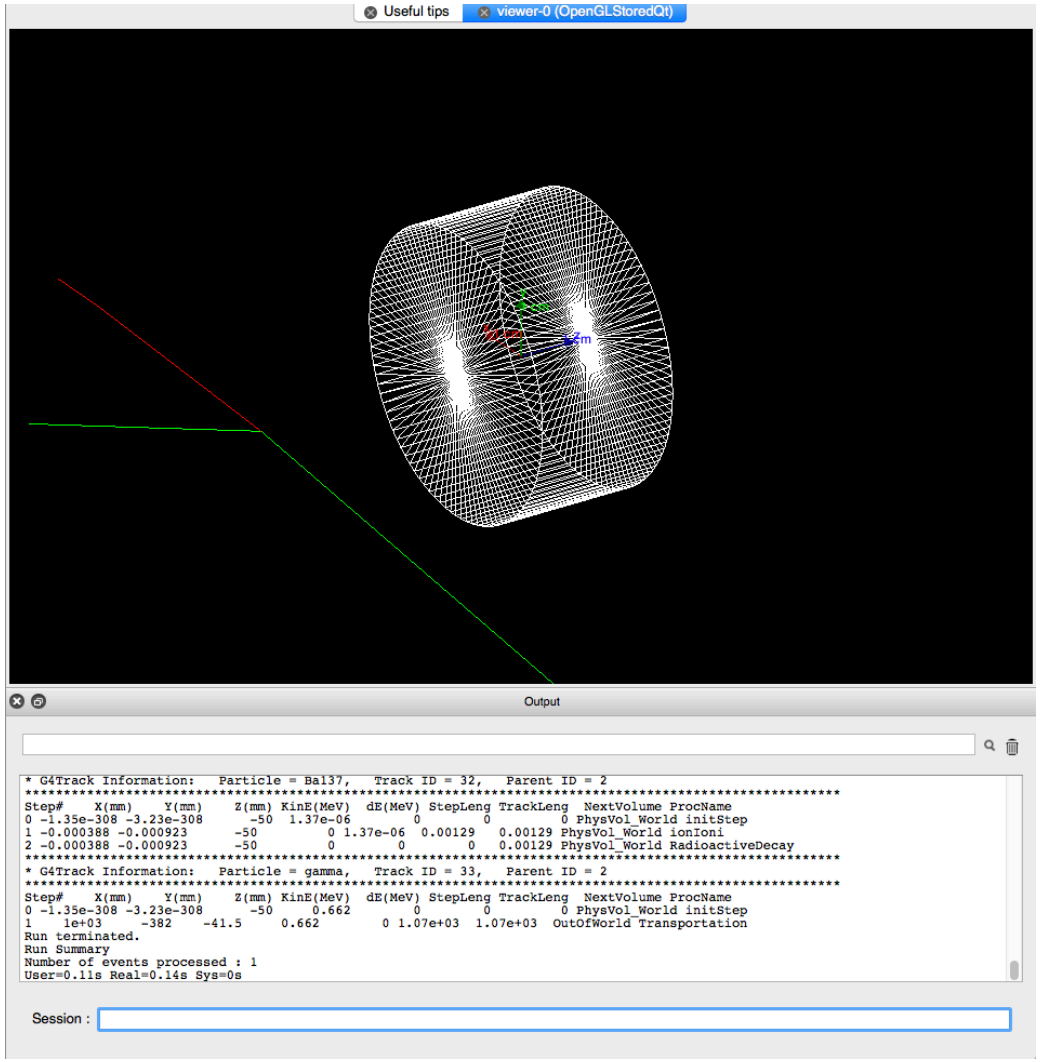
右図はその出力
（マウスで視点を調整してある）

[注]
前演習の'P04_PhysicsLists: Shielding'
と同じ動作をしている

2) 照射回数を自由に変更してみる

3) アプリ終了

```
exit
```



P04_PhysicsLists: Factoryのmainプログラムの構造

課題：4　G4PhysListFactoryを使ったmainプログラムの基本構造を知る

```
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// Geant4 Application: Tutorial course for Hep/Medicine Users
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#include "Geometry.hh"
#include "UserActionInitialization.hh"

#include "G4RunManager.hh"
#include "G4UImanager.hh"
#include "G4VisExecutive.hh"
#include "G4UIExecutive.hh"
#include "G4PhysListFactory.hh"
#include "G4VModularPhysicsList.hh"
#include "G4RadioactiveDecayPhysics.hh"

//-------------------------------------------------------------------------------
int main( int argc, char** argv )
//-------------------------------------------------------------------------------
{
// Construct the default run manager
   G4RunManager * runManager = new G4RunManager;

// Set up mandatory user initialization: Geometry
   runManager->SetUserInitialization( new Geometry );

// Set up mandatory user initialization: Physics-List
   G4PhysListFactory factory;
   G4VModularPhysicsList* physicsList = factory.GetReferencePhysList("FTFP_BERT");
   physicsList->RegisterPhysics( new G4RadioactiveDecayPhysics );
   runManager->SetUserInitialization( physicsList );

// Set up user initialization: User Actions
   runManager->SetUserInitialization( new UserActionInitialization );

// Initialize G4 kernel
   runManager->Initialize();

// Create visualization environment
   G4VisManager* visManager = new G4VisExecutive;
   visManager->Initialize();

// Start interactive session
   G4UIExecutive* uiExec = new G4UIExecutive(argc, argv);
   G4UImanager*   uiManager = G4UImanager::GetUIpointer();
   uiManager->ApplyCommand("/control/execute GlobalSetup.mac");
   uiExec->SessionStart();

// Job termination
   delete uiExec;
   delete visManager;
   delete runManager;

   return 0;
}
```

```
$ cd ../source
$ less Application_Main.cc
```

P04_PhysicsLists_Shieldingとの違い

もとは以下であった：
#include ”Shielding.hh”

これらの4行が以下の1行の代わりに追加された：

```
runManager->
    SetUserInitialization( new Shielding );
```

[追加4行で行われていること]
1. まずG4PhysListFactoryを作る
2. そこから’FTFP_BERT’のPhysics Listを取得
3. ‘new G4RadioactivePhysics()’を実行して放射性崩壊過程を生成
4. 過程をRegisterPhysics()でPhysics Listに追加
5. 変更したPhysics ListをRunManagerに渡す

P04_PhysicsLists: MyPhysList

独自のPhysics Listsの作成

P04_PhysicsLists: MyPhysListプログラムの概要

■ 演習プログラムの目的

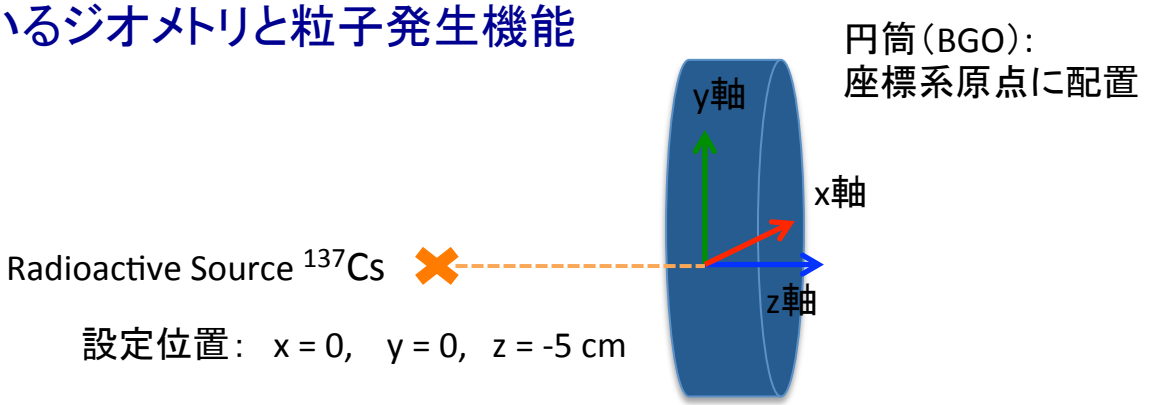
- この演習ではG4VModularPhysicsListクラスを使って、ユーザが独自のPhysics Listを作成する方法を学ぶ

［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

■ プログラムの構成

- mainプログラム： P04_PhysicsLists.cc_MyPhysList
- ジオメトリ： P02_Geometryで使ったBGO検出器
- PhysicsList： MyPhysicsList.hh/MyPhysicsList.cc
  (Geant4が提供している例題B3に基づいている)

■ 組み込まれているジオメトリと粒子発生機能



P04_PhysicsLists: MyPhysListを使ったアプリケーションのビルド

課題：1　P04_PhysicsLists.cc_MyPhysListをmainプログラムとしてアプリケーションをビルド

1）事前に用意されているP04_PhysicsLists.cc_MyPhyisListをApplication_Main.ccにコピー

```
$ cd ~/Geant4Tutorial20171129/UserWorkDir/P04_PhysicsLists
$ cd source
$ cp ../util/G4Codes/P04_PhysicsLists.cc_MyPhysList Application_Main.cc
overwrite Application_Main.cc? (y/n [n]) y
```

← CBDir (Current Base Directory)

2）事前に用意されているMyPhysicsList.hh/.ccをincludeとsrcディレクトリにコピー

```
$ cd include
$ cp ../../util/G4Codes/PhysicsList.hh_MyPhysList MyPhysicsList.hh
$ cd ../src
$ cp ../../util/G4Codes/PhysicsList.cc_MyPhysList MyPhysicsList.cc
```

← 名前を間違えないこと

← 名前を間違えないこと

3) buildデレクトリでビルドを実行　[注]

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

← CBDir (Current Base Directory)へ戻る

← 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

[注]
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_P04_MyPhysList.sh

ただし、buildの失敗がMyPhysicsListの名前を間違えたために生じたなら、それを消去したのち、スクリプトを実行

Application_Main (MyPhysList)の実行

課題：2　Application_Main (MyPhysList)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
……/P04_PhysicsLists
$ cd TestBench
$ ../bin/Application_Main
```

現在いるデレクトリを確認：
P04_PhysicsListsでなければ、そこに移動する

作業デレクトリに移動

TestBenchデレクトリでApplication_Main実行

Qtウインドでアプリの動作を確認

課題：3　UI コマンドで事象発生

1) 以下のUIコマンドを先ず入力

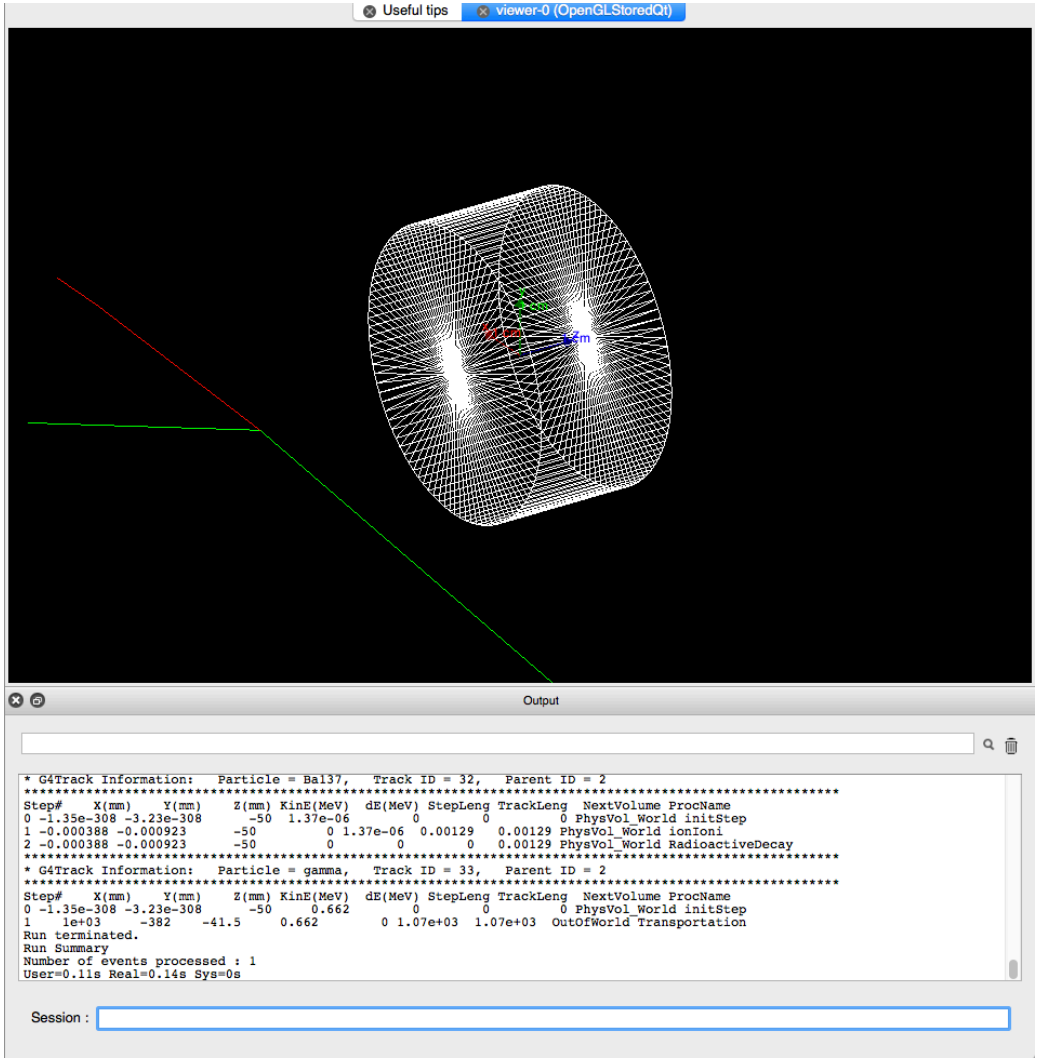
```
/run/beamOn 1
```

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
```



P04_PhysicsLists: MyPhysListのプログラムの構造

課題：4　mainプログラムの基本構造を知る

```
$ less Application_Main.cc
```

```
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// Geant4 Application: Tutorial course for Hep/Medicine Users
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#include "Geometry.hh"
#include "MyPhysicsList.hh"
#include "UserActionInitialization.hh"

#include "G4RunManager.hh"
#include "G4UImanager.hh"
#include "G4VisExecutive.hh"
#include "G4UIExecutive.hh"

//-------------------------------------------------------------------------------
int main( int argc, char** argv )
//-------------------------------------------------------------------------------
{
// Construct the default run manager
   G4RunManager * runManager = new G4RunManager;

// Set up mandatory user initialization: Geometry
   runManager->SetUserInitialization( new Geometry );

// Set up mandatory user initialization: Physics-List
   runManager->SetUserInitialization( new MyPhysicsList );

// Set up user initialization: User Actions
   runManager->SetUserInitialization( new UserActionInitialization );

// Initialize G4 kernel
   runManager->Initialize();

// Create visualization environment
   G4VisManager* visManager = new G4VisExecutive;
   visManager->Initialize();

// Start interactive session
   G4UImanager*   uiManager = G4UImanager::GetUIpointer();
   G4UIExecutive* uiExec = new G4UIExecutive(argc, argv);
   uiManager->ApplyCommand("/control/execute " + nameMainMacro);
   uiExec->SessionStart();

// Job termination
   delete uiExec;
   delete visManager;
   delete runManager;

   return 0;
}
```

P04_PhysicsLists: Factoryとの違い

ユーザが独自に定義したPhysics Listをincludeする

[注]
前にあったG4PhysListFactory関連の#include 3行が不要なので消去されている

ユーザが独自に定義したPhysics Listをnewする

[注]
前にあったG4PhysListFactory関連の3行が不要なので消去されている

P04_PhysicsLists: MyPhysListのプログラムの構造 (つづき)

課題:5　ユーザのPhyiscs Listの基本構造を知る

```
$ cd source/include ; less MyPhysicsList.hh
$ cd ../src ; less MyPhysicsList.cc
```

MyPhysicList.hh

```
//---------------------------------------------------
// MyPhysicsList.hh
//---------------------------------------------------
#ifndef MyPhysicsList_h
#define MyPhysicsList_h 1

#include "G4VModularPhysicsList.hh"

//---------------------------------------------------
  class MyPhysicsList : public G4VModularPhysicsList
//---------------------------------------------------
{
  public:
    MyPhysicsList();
   ~MyPhysicsList();

  public:
    void SetCuts();
};
#endif
```

MyPhysicList.cc

```
//---------------------------------------------------------------
// MyPhysicsList.cc
//  [Note] Based on "G4 Basic Example: B3"
//---------------------------------------------------------------
#include ”MyPhysicsList.hh"
#include "G4DecayPhysics.hh"
#include "G4RadioactiveDecayPhysics.hh"
#include "G4EmStandardPhysics.hh"

//---------------------------------------------------------------
  MyPhysicsList::MyPhysicsList()
  : G4VModularPhysicsList()
//---------------------------------------------------------------
{
// Default physics
   RegisterPhysics(new G4DecayPhysics());

// Radioactive decay
   RegisterPhysics(new G4RadioactiveDecayPhysics());

// EM physics
   RegisterPhysics(new G4EmStandardPhysics());
}

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

//---------------------------------------------------------------
  void MyPhysicsList::SetCuts()
//---------------------------------------------------------------
{
    G4VUserPhysicsList::SetCuts();
}
```

終了