ユーザ・アプリケーション作成の基礎

演習パッケージ： P01_FirstStep

Geant4 10.3.P3準拠

Geant4　HEP/Space/Medicine 講習会資料

KEK 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

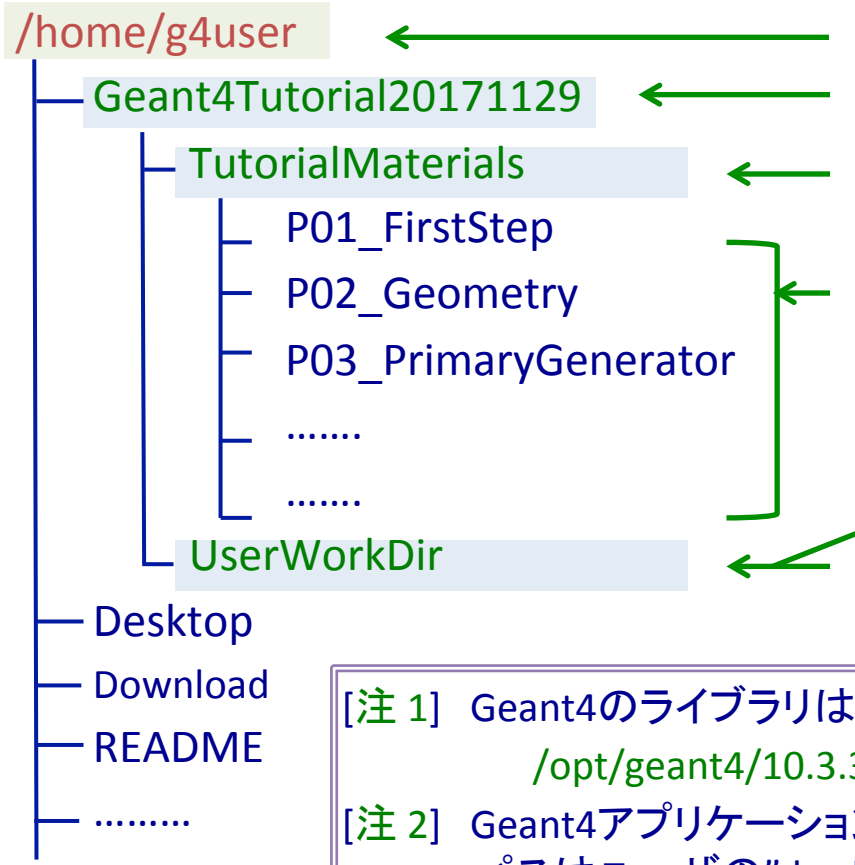
演習の目標

1. 最初の演習ではユーザがGeant4のアプリケーションを作成する際に、必ず用意しなければならないメイン・プログラム(main.cc)の構造について学ぶ
2. 演習は以下の2種類のメイン・プログラムの構造をあつかう:
    1. グラフィック機能を持たない対話型アプリケーション用
    2. 上のメインプログラムにグラフィック機能を追加
3. これらのプログラムをcmakeを使いビルドし、アプリケーションを作る方法を学ぶ
4. 簡単なマクロ・コマンドを使ったアプリケーション実行方法を実習する

提供されている演習プログラムのファイル構成

仮想マシン上でのディレクトリ構造

■ 以下の構造を仮想マシン上で確認する



仮想マシンのユーザのルート・ディレクトリ

Tutorialのルート・ディレクトリ

Tutorialで使う資料はすべてここにある

各演習用ファイル保存ディレクトリ:

- これらのディレクトリでは絶対に作業しないこと
- 必ずUserWorkDirにコピーして演習する

ユーザが作業する演習ワーク・ディレクトリ:

[注 1] Geant4のライブラリは以下にインストールされている

/opt/geant4/10.3.3/lib64/

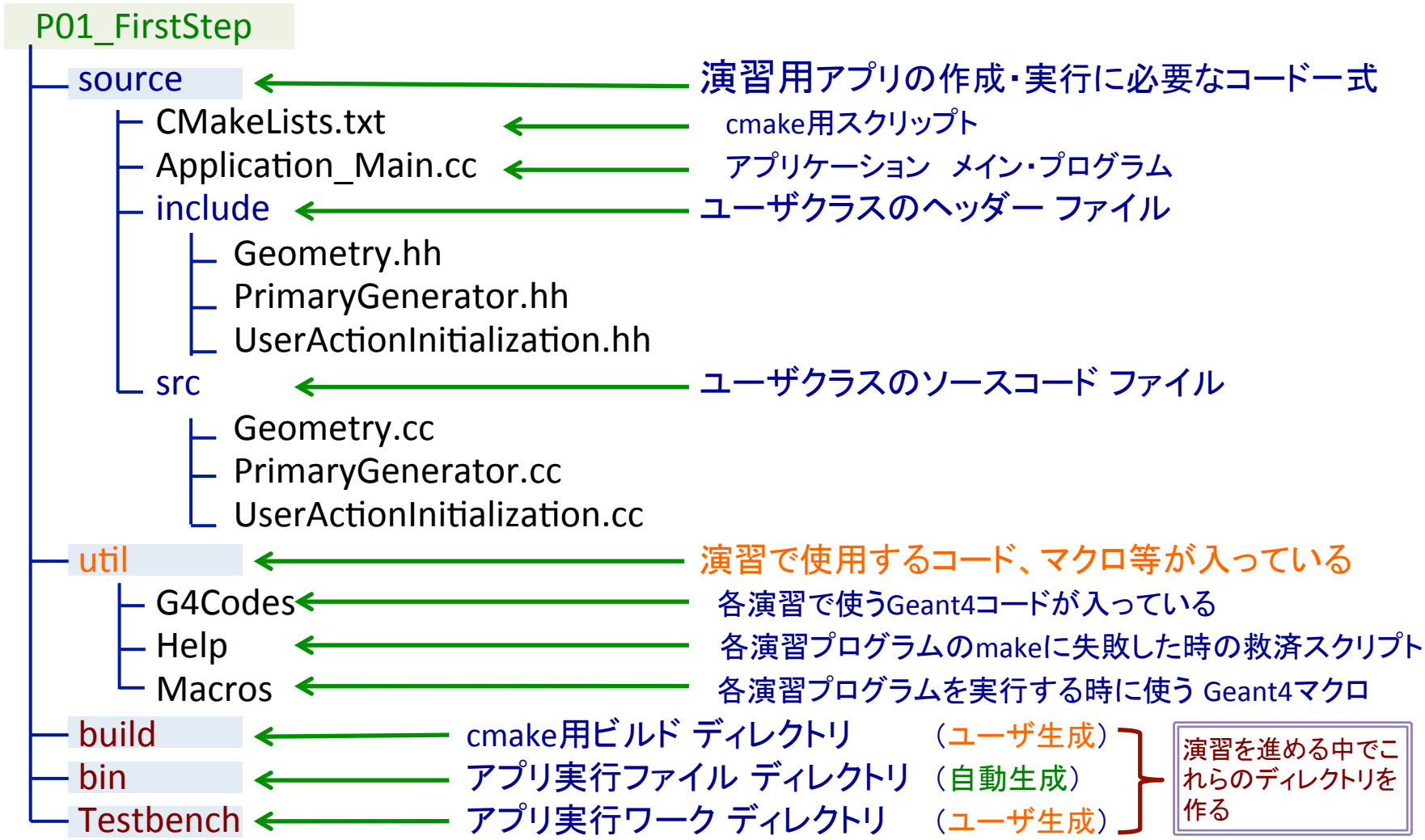
[注 2] Geant4アプリケーション実行時に必要な物理過程反応断面積データへのパスはユーザの".bashrc"で自動的に環境変数へ設定済み

source /opt/geant4/10.3.3/bin/geant4.sh

!!! これが設定されていないとアプリが実行時にエラーとなることに注意

演習プログラムのファイル構成

■ 演習プログラムのファイル構造



演習の準備

P01_FirstStep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P01_FirstStep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ィレクトリにコピー

```
$ cd ~/Geant4Tutorial20171129
$ cd UserWorkDir
$ cp -r ../TutorialMaterials/P01_FirstStep .
```

ユーザ作業root dir

先ず、演習のルート・ディレクトリに行く

P01_FirstStep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 P01_FirstStep
source/    util/
$ ls P01_FirstStep/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プログラム名は任意だが、演習では常にこの名前を使う

```
$ ls P01_FirstStep/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] 三つのファイル名は任意だが、演習では常にこの名前を使う

```
$ ls P01_FirstStep/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

P01_FirstStep

対話型アプリケーションの構造

P01_FirstStep プログラムの概要

■ 演習プログラムの目的

- 対話型アプリケーションの作成方法を学ぶ
  ［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

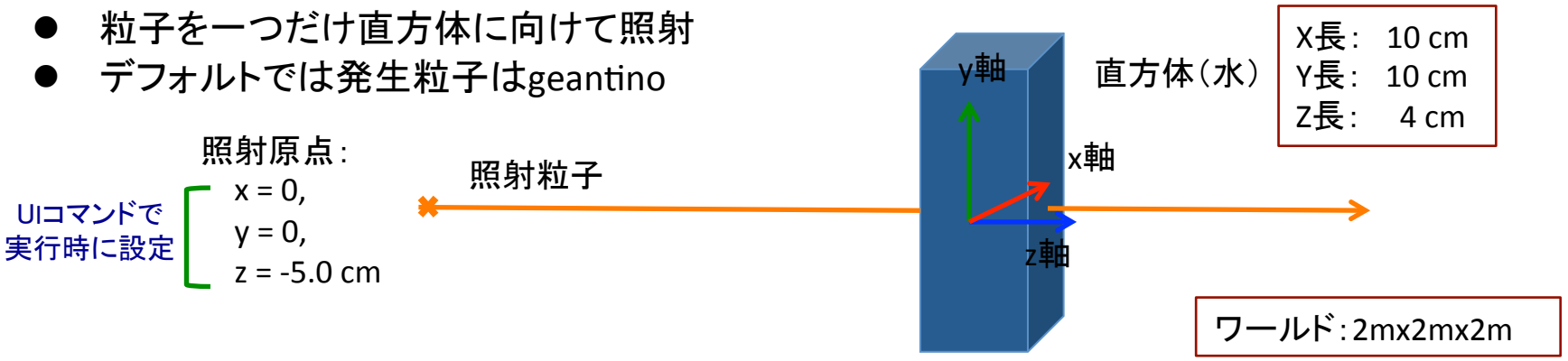
- mainプログラム： グラフィク機能がない対話型アプリケーション
- ジオメトリ： 直方体（物質は水）が一つだけ置かれている
- PhysicsList : 高エネルギー領域で汎用的な‘FTFP_BERT’とよばれるもの
- PrimaryGenerator: 標準ツールとして提供される’ParticleGun’とよばれるもの

■ プログラムの機能

- tcshスタイルの対話型アプリケーション
- 事象のグラフィカル表示機能は含まれていない（グラフィク機能追加の手法は後述）
- mainプログラムは大量の事象をシミュレーションする時のテンプレートとなる

■ 組み込まれているジオメトリと粒子発生機能

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



P01_FirstStepのビルド

課題：1　P01_FirstStep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P01_FirstStepにすでに用意されている)

1) P01_FirstStepのディレクトリに移行する（UserWorkDirの下にある）

```
$ cd ~/Geant4Tutorial20171129/UserWorkDir/P01_FirstStep
```

[注意] コマンド入力には必ずtcsh補完機能を使う

以後CBDir (Current Base Dir)とよぶ

2) ビルドするための作業ディレクトリを作成（P01_FirstStepの下に）

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

ビルド作業のディレクトリ名は習慣に従いbuildとする

3) buildディレクトリでビルドを実行 [注1]

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

sourceディレクトリにあるCMakeLists.txtを実行
(CMakeLists.txtの中でmain program名前がApplication_Mainであることが指定されている)

自動生成されたMakefileを実行 （-jオプション使用可能）

binディレクトリが自動作成され実行ファイルを収納
(binディレクトリはbuildディレクトリと同じレベルに作成される)

- 'cmake; make; make install'はcmakeでビルドを実行する時の慣用句 ('cmake; make install'のみでもOK)
- cmake実行でbuildディレクトリの中に多くのファイルが自動作成されるが、その内容は無視してOK

4) ビルド実行後のディレクトリ構造確認

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

CBDirディレクトリへ移動

binディレクトリが自動生成

実行ファイルが生成

[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_P01_FirstStep.sh

P01_FirstStepのビルド

課題：1　P01_FirstStep:をビルドして実行ファイルを作成する

[注意] コマンド入力には必ず tcsh補完機能を使う

1) P01_FirstStepのディレクトリに移行する（UserWorkDirの下にある）

```
$ cd ~/Geant4Tutorial20171129/UserWorkDir/P01_FirstStep
```

以後CBDir（Current Base Dir）とよぶ

2) ビルドするための作業ディレクトリを作成（P01_FirstStepの下に）

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

ビルド作業のディレクトリ名は習慣に従いbuildとする

3) buildディレクトリでビルドを実行 [注1]

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

sourceディレクトリにあるCMakeLists.txtを実行
（CMakeLists.txtの中でmain program名前が Application_Mainであることが指定されている）

自動生成されたMakefileを実行 （-jオプション使用可能）

binディレクトリが自動作成され実行ファイルを収納
（binディレクトリはbuildディレクトリと同じレベルに作成される）

- ‘cmake; make; make install’はcmakeでビルドを実行する時の慣用句 (‘cmake; make install’のみでもOK)
- cmake実行でbuildディレクトリの中に多くのファイルが自動作成されるが、その内容は無視してOK

4) ビルド実行後のディレクトリ構造確認

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

CBDirディレクトリへ移動

binディレクトリが自動生成

実行ファイルが生成

[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_P01_FirstStep.sh

P01_FirstStep: Application_Mainの実行

課題:2　Application_Main を実行する

1) プログラム実行を行う作業ディレクトリを新たに作成する

```
$ pwd
……/P01_FirstStep
$ mkdir TestBench
$ ls
TestBench/  bin/   build/   source/  util/
$ cd TestBench
$ ../bin/Application_Main
```

現在のディレクトリがCBDirであることを確認:
P01_FirstStepでなければ、そこに移動する

作業ディレクトリをつくる
(名前は自由だが、演習を通してこの名前を使う)

TestBenchディレクトリで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ることを確認

```
*************************************************************
 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

FTFP_BERT : New threshold …….
……

### Adding tracking cuts for neutron  TimeCut(ns)= 10000  KinEnergyCut(MeV)= 0
Available UI session types: [ Qt, GAG, tcsh, csh ]
Idle>
```

使用するGeant4のバージョン情報

使用するPhysics Listの名前

'Idle'状態でコマンドを待っている

helpコマンド

課題：3　helpコマンドを実行し、Geant4が標準で用意しているコマンドを概観する

[注]

- コマンドとはアプリの実行を対話的に制御するために用いるもので、UI コマンドとよばれる
- 用意されているコマンドは多岐にわたるので、良く使うものから徐々に知るのでOK
- 複数のコマンドを入力するのは間違いが生じやすいので、実際の使用ではマクロファイルを用意し、それを実行させる　ー　マクロファイルの導入は先の課題で行う

1) コマンド・デレクトリのルート

```
Idle> help
Command directory path : /
 Sub-directories :
  1) /control/   UI control commands.
 2) /units/   Available units.
 3) /process/   Process Table control commands.
 4) /analysis/   ...Title not available...
 5) /particle/   Particle control commands.
 6) /geometry/   Geometry control commands.
 7) /tracking/   TrackingManager and SteppingManager control ...
 8) /event/   EventManager control commands.
 9) /cuts/   Commands for G4VUserPhysicsList.
 10) /run/   Run control commands.
 11) /random/   Random number status control commands.
 12) /material/   Commands for materials
 13) /physics_lists/   commands related to the physics simulation..
 14) /gun/   Particle Gun control commands.
 15) /heptst/   Controls for the hadronic energy/momentum test
 16) /physics_engine/   ...Title not available...
 Commands :

Type the number ( 0:end, -n:n level back ) :
10
```

← runコマンドの内容を見る

2) runコマンドの内容

```
Command directory path : /run/

Guidance :
Run control commands.

 Sub-directories :
 1) /run/particle/   Commands for G4VUserPhysicsList.
 Commands :
 2) initialize * Initialize G4 kernel.
 3) beamOn * Start a Run.
 4) verbose * Set the Verbose level of G4RunManager.
 5) printProgress * Display begin_of_event information.....
    .....
    .....

Type the number ( 0:end, -n:n level back ) :
0
Exit from HELP.

Idle>
```

← help終了

粒子を照射してみる

課題：4　デフォルトの仮想粒子(geantino)を直方体（水）に照射してみる

[注]

- source/srcデレクトリ内のGeometry.ccとPrimaryGenerator.ccに粒子と直方体が実装されている

1) シミュレーション開始は/run/beamOnコマンド

```
Idle> /run/beamOn 1
phot:   for  gamma    SubType= 12  BuildTable= 0
     LambdaPrime table from 200 keV to 100 TeV in 61 bins
     ===== EM models for the G4Region  DefaultRegionForTheWorld ======
      PhotoElectric :  Emin=       0 eV    Emax=     100 TeV   AngularGenSauterGavrila...

compt:   for  gamma    SubType= 13  BuildTable= 1
     Lambda table from 100 eV  to 1 MeV, 7 bins per decade, spline: 1
     LambdaPrime table from 1 MeV to 10 TeV in 49 bins
     ===== EM models for the G4Region  DefaultRegionForTheWorld ======
      Klein-Nishina :  Emin=       0 eV    Emax=      100 TeV
.........
.........
.........

============================================================
Idle>
```

粒子（デフォルトはgeantino）を1コだけ照射してみる
(コマンドの引数は粒子数)

初めて/run/beamOnを実行すると、粒子相互作用に関して設定された条件が出力される

内容に関しては当面は無視してOK

geantinoが照射されたが、上の相互作用初期設定条件の情報以外は何も出力されず'Idle'状態に戻る

2) 再度、/run/beamOnコマンドを実行

[注１] コマンド名を全てタイプしなくても、tcsh流のコマンド補完機能が使える
[注２] コマンドの頭にある'/'は省略可能

```
Idle> /run/beamOn 5
Idle>
```

geantinoを5回照射したが、何も出力されず'Idle'状態に戻る

粒子tracking過程を見る

課題:５　geantinoがどのようにtrackingされているかの詳細情報の取得法を知る

[注]

- 前の課題でgeantinoを複数回照射したが、geant4は何の出力も出さなかった。これはアプリでシミュレーション結果の出力に関してユーザアクションで何も指定していないため
- しかし舞台裏ではGeant4は粒子を指定された相互作用を考慮しながらtrackingしている

1) Geant4が粒子をtrackingしている過程をtrackingコマンドで確認してみる

```
Idle> /gun/position  0.0  0.0  -5.0 cm
Idle> /gun/direction 0.0  0.0  +1.0
Idle> /tracking/verbose 1
Idle> /run/beamOn 1
**********************************************************************************************
* G4Track Information:   Particle = geantino,   Track ID = 1,   Parent ID = 0
**********************************************************************************************

Step#    X(mm)    Y(mm)    Z(mm) KinE(MeV)  dE(MeV) StepLeng TrackLeng  NextVolume ProcName
    0        0        0      -50     1e+03        0        0          0 PhysVol_World initStep
    1        0        0      -20     1e+03        0       30         30 PhysVol_WaterBox Transportation
    2        0        0       20     1e+03        0       40         70 PhysVol_World Transportation
    3        0        0    1e+03     1e+03        0      980  1.05e+03  OutOfWorld Transportation
```

粒子発生条件を先ず設定する（粒子はデフォルトでgeanatino）

tracking過程の各ステップでの情報を出力させる

trackingの各ステップでの情報

[注]

- Geant4は粒子をstep単位で進ませるが、構造物の境界では必ず一つのstepが終了する
- 上記出力で、geantinoは物質と相互作用はしないので、step終了点は直方体(水)およびワールドの境界でのみで生じていることがわかる

粒子tracking過程を見る – 続き

課題:６　粒子の種類とkinematicsを変え、trackingの詳細情報を見てみる

1)　入射粒子をkinetic energy 25MeVの陽子に設定　（発生点と方向は前に設定したもののまま）

```
Idle> /gun/particle proton
Idle> /gun/energy 25 MeV
```

[注]　得られる出力内容は環境により
このスライドと異なることがある

2) 上記設定の陽子を1コ照射

```
Idle> run/beamOn 1
**********************************************************************************************************
* G4Track Information:   Particle = proton,   Track ID = 1,   Parent ID = 0
**********************************************************************************************************
Step#    X(mm)    Y(mm)    Z(mm) KinE(MeV)  dE(MeV) StepLeng TrackLeng  NextVolume ProcName
    0        0        0      -50        25        0        0         0 PhysVol_World initStep
    1  -0.0005  0.00387    -44.2        25  0.00865     5.81      5.81 PhysVol_World hIoni
    2  0.00118     0.01    -38.6        25  0.00913     5.56      11.4 PhysVol_World hIoni
    3  -0.0031   0.0159    -31.7        25   0.0116     6.91      18.3 PhysVol_World hIoni
..... 省略
.....
**********************************************************************************************************
* G4Track Information:   Particle = e-,   Track ID = 3,   Parent ID = 1
**********************************************************************************************************
Step#    X(mm)    Y(mm)    Z(mm) KinE(MeV)  dE(MeV) StepLeng TrackLeng  NextVolume ProcName
    0  0.00118     0.01    -38.6   0.00103        0        0         0 PhysVol_World initStep
    1   0.0247   0.0145    -38.6         0  0.00103   0.0596    0.0596 PhysVol_World eIoni

**********************************************************************************************************
* G4Track Information:   Particle = e-,   Track ID = 2,   Parent ID = 1
**********************************************************************************************************
Step#    X(mm)    Y(mm)    Z(mm) KinE(MeV)  dE(MeV) StepLeng TrackLeng  NextVolume ProcName
    0  -0.0005  0.00387    -44.2   0.00212        0        0         0 PhysVol_World initStep
    1  -0.0329   0.0727    -44.2         0  0.00212    0.178     0.178 PhysVol_World eIoni
```

一次粒子(陽子)のステップ情報

二次粒子(電子)のステップ情報

二次粒子(電子)のステップ情報

3) アプリケーションの終了

```
Idle> exit
```

マクロファイルを使ってみる

課題:7　マクロファイルの基本的な作り方、使い方を知る

- 今までの課題ではUIコマンドを一つ一つ手で入力したが、これは間違いやすいし手間が大変
- Geant4はUIコマンドを一括して実行できるマクロ機能を用意している
- アプリケーションを実行するにはマクロファイルを作成して実行するのが標準手法
- 以下では、今までの課題で使ったUIコマンドをマクロファイルに書きそれを実行してみる

1) TestBenchディレクトリに以下の内容のマクロファイルを適当なエディターで作成

```
$ pwd          ← 課題6を終了時点で、以下のdirにいるはず – もし異なる場合、このdir移動
/home/g4user/Geant4Tutorial20171129/UserWorkDir/P01_FirstStep/TestBench

$ gedit myFirstMacro.mac   ← マクロの名前は拡張子を含めて任意
```

```
##  My first Geant4 macro file

/gun/position  0.0  0.0  -5.0 cm
/gun/direction 0.0  0.0  +1.0
/gun/particle proton
/gun/energy 25 MeV
/tracking/verbose 1
/run/beamOn 1
```

← マクロファイルにこの内容を入力し、エディターを終了

2) TestBenchディレクトリで先に作ったアプリケーションを実行して、マクロファイルを実行

```
$ ../bin/Application_Main          ← アプリケーションの実行

Geant4の初期メッセージ出力
……
Idle>  /control/execute myFirstMacro.mac   ← 作ったマクロを実行
```

3) 課題6で得られたtackingの詳細情報と同じことを確認する

mainプログラムの構造 (P01_FirstStep)

課題:7　mainプログラムの基本構造を知る

```
$ less ../source/Application_Main.cc
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// Geant4 Application: Tutorial course for Hep/Medicine Users
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
#include "Geometry.hh"
#include "UserActionInitialization.hh"

#include "G4RunManager.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 );

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

// Start interactive session
   G4UIExecutive* uiExec = new G4UIExecutive(argc, argv, "tcsh");
   uiExec->SessionStart();

// Job termination
   delete uiExec;
   delete runManager;

   return 0;
}
```

mainプログラムで使用するクラスのヘッダーファイルをすべてincludeする

RunManagerを作る

ジオメトリの登録
(ユーザが定義するクラス)

粒子と物理相互作用モデルを登録
(ツールキット提供クラス)

primary generatorをUserActionを通して登録
(ユーザが定義するクラス)

以上の設定で初期化実行

tcshタイプの対話型ユーザインタフェース設定

対話セッション開始

アプリケーション終了処理

P01_FirstStep_Vis

対話型アプリケーションにグラフィク機能を追加

P01_FirstStep_Vis: グラフィク機能を持つプログラム

■ 演習プログラムの目的

- Geant4アプリケーションの開発には、グラフィク機能が不可欠 – この機能追加法を学ぶ
  ［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

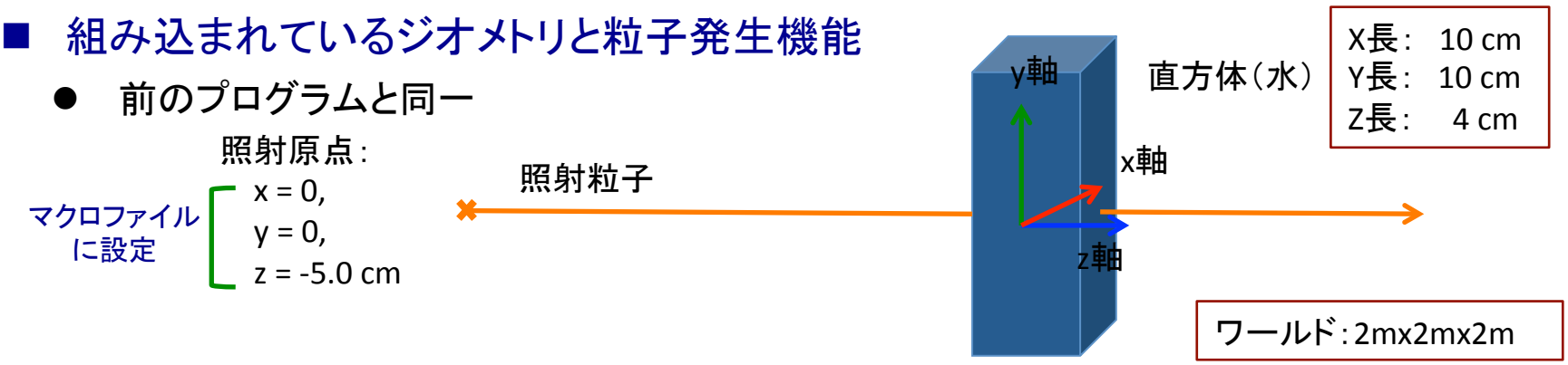
- mainプログラム: グラフィク機能を持つ対話型アプリケーション
- ジオメトリ: 前のプログラムと同一
- PhysicsList : 前のプログラムと同一
- PrimaryGenerator: 前のプログラムと同一

■ プログラムの機能

- Qtスタイルの対話型アプリケーション
- OpenGLベースのグラフィカルな環境
- グラフィク機能のセットアップなどの標準的マクロファイルを使う

■ 組み込まれているジオメトリと粒子発生機能

- 前のプログラムと同一



P01_FirstStep_Visプログラムの構造

課題:1　mainプログラムの基本構造を知る

```
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
// Geant4 Application: Tutorial course for Hep/Space 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 );

// 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;
}
```

P01_FirstStepとの違い

- includeファイルの追加
- 三つのユーザ定義クラスは前と同じまま
- グラフィック・ツールを設定
- /control/executeコマンドをプログラム内で実行し、講習会用に用意されているマクロファイル(GlobalSetup.mac)を読み込む
- グラフィック・ツール使用の後始末

P01_FirstStep_Visのビルド

課題：2　P01_FirstStep_Visをビルドして実行ファイルを作成する

[注意]
コマンド入力には必ず
tcsh補完機能を使う

1）　P01_FirstStep_VisのmainをApplication_Main.ccにコピー

```
$ cd ~/Geant4Tutorial20171129/UserWorkDir/P01_FirstStep/
$ cd source
$ cp ../util/G4Codes/AppliMain.cc_P01_FirstStep_Vis Application_Main.cc
overwrite Application_Main.cc? (y/n [n]) y
```

CBDir
(Current Base Directory)

前の演習で使ったP01_FirstStepのメインプログラムがグラフィック付きのものに置き換えられた

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 ..
```

CBDirに戻る
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
[重要] 必ずbuildの中身は全てクリアしてから
cmakeすることを習慣にする
初めのmakeコマンドを省略し、make installのみでもOK
CBdir(P01_FirstStep)に戻っておく

[注1]
buildを失敗したたら、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_P01_FirstStep_Vis.sh

P01_FirstStep_Visの実行

課題：3　Application_Mainを実行する

1) プログラム実行デレクトリ(TestBench)に前もって用意されている環境設定マクロをコピーする

```
$ pwd
……/P01_FirstStep
$ cd TestBench
$ rm *
$ cp ../util/Macros/* . ; ls
GlobalSetup.mac
primaryGeneratorSetup.mac
verboseSetup.mac
visSetup_Simplex.mac
$ ../bin/Application_Main
```

現在のデレクトリ確認：　P01_FirstStepでなければ移動

前プログラム作成でつくられているTestBenchに移動

前プログラムで使ったマクロを消す（消さなくてもOKだが）

用意されているマクロファイルをTestBenchにコピーし、‘ls’でコピーされたファイル確認

4つのマクロファイルがコピーされている（詳細は後述）

TestBenchでアプリを実行

2) 端末に以下のメッセージが出力されることを確認

```
**************************************************************
 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

FTFP_BERT : New threshold …….
……


You have successfully registered the following graphics systems.
Current available graphics systems are:
  ASCIITree (ATree)
  DAWNFILE (DAWNFILE)
  ……..
```

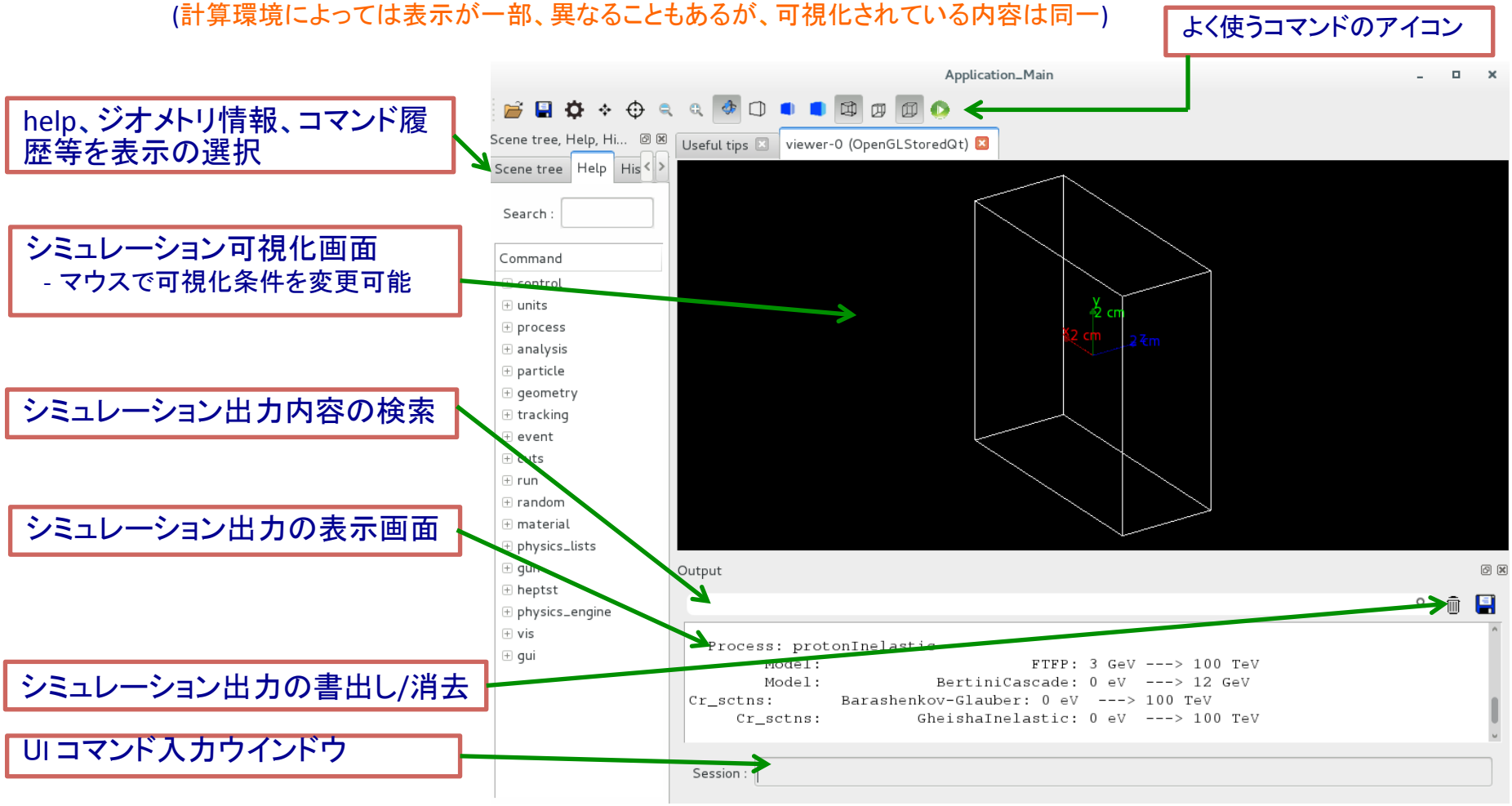
可視化機能(Visualization)が追加されたので、出力メッセージ内容がP01_FirstStepの時に比べて増加している

メッセージ出力が終了すると次のスライドで示すQtウインドが開く

初期Qtウィンドウ

課題：4　Qtウィンドウの初歩的な使い方を学ぶ

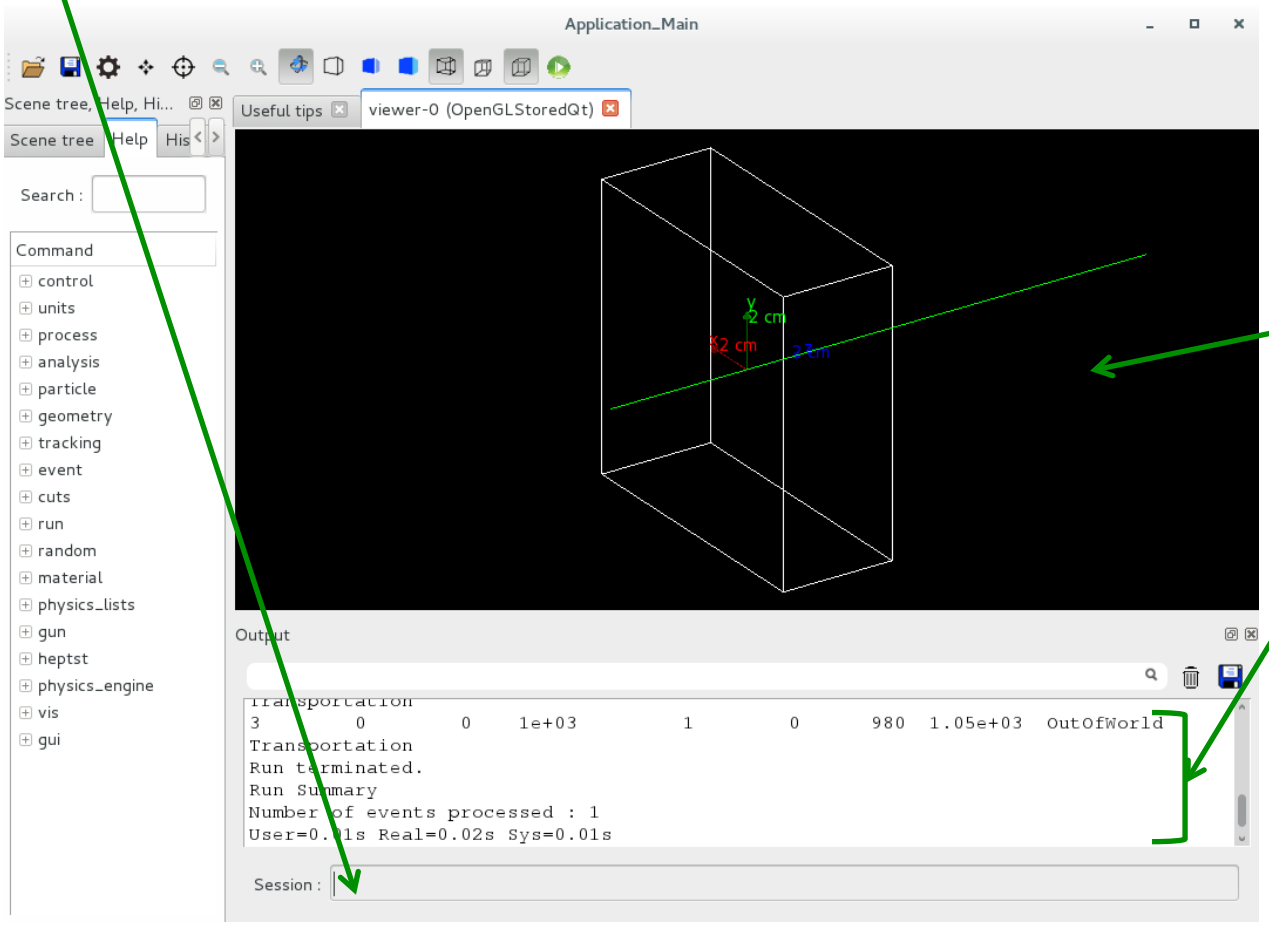
[注]　P01_FirstStepで設定されている直方体のボリュウムが初期画面で表示される
(計算環境によっては表示が一部、異なることもあるが、可視化されている内容は同一)



Qtウインドウを使ってみる

課題:5　簡単な組込みコマンドを用いて、粒子を構造体に照射する

以下のUIコマンドを入力して画面図のように粒子が走ることを確認する
`/run/beamOn 1`



geantino(仮想粒子)が発生点から＋z方向に照射される

マウスの右クリックで表示されるメニュからview条件を変更可能

- 移動、回転、拡大
- 構造体のsurface表示など

また、左クリック、wheelによるコントロールも可能」

出力表示画面に初期情報及びgeantinoのステップ情報が出力されている

[注]
粒子の発生条件コマンド及び
‘/tracking/verbose 1’
はマクロファイルで設定済
(マクロファイルについては後述)

Qtウインドで入射粒子の種類を変えて照射してみる

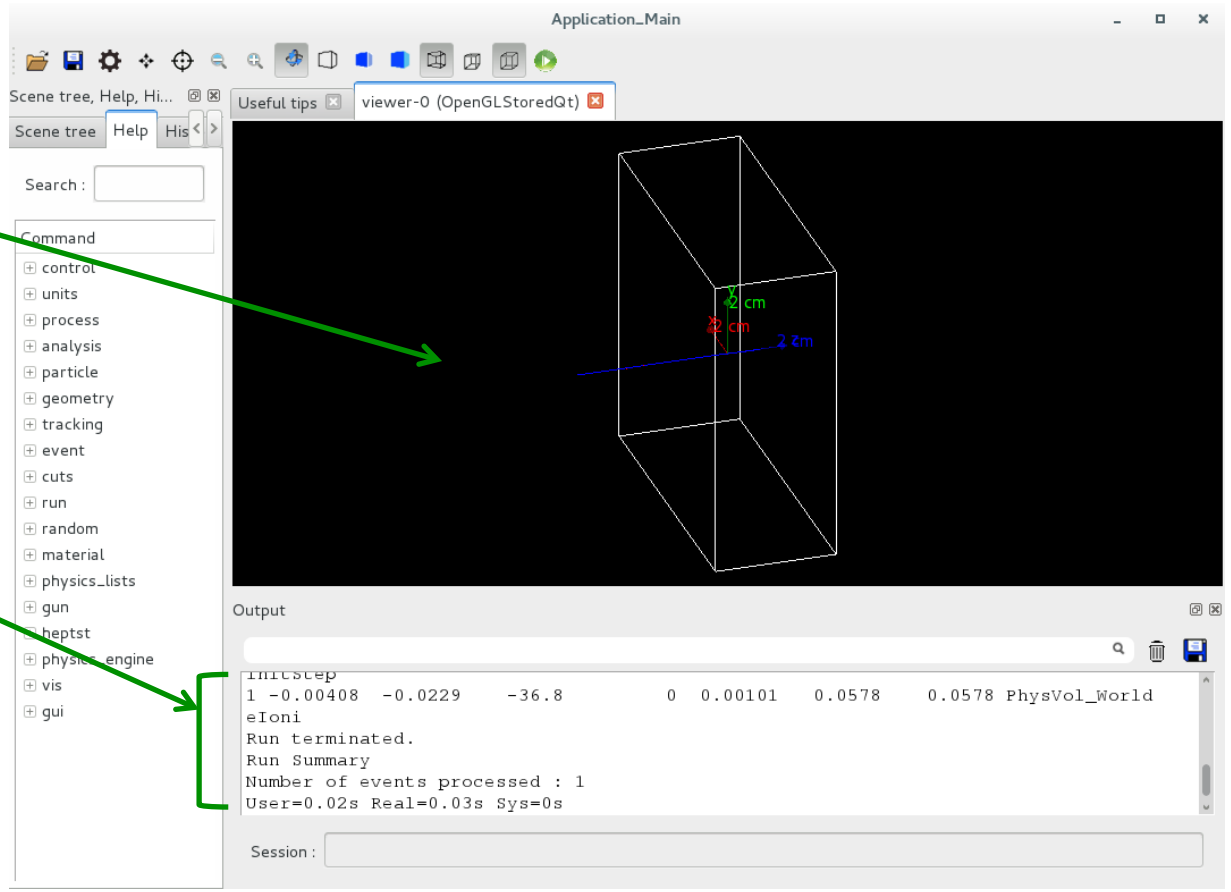
課題：6　以下のUI コマンドを入力して、geantino以外の粒子を照射する

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

proton及び物質との相互作用で作られた二次粒子の飛跡が表示される

- 青：＋電荷の粒子
- 赤：－電荷の粒子
- 緑：0 電荷の粒子

proton及び物質との相互作用で作られた二次粒子のステップ情報が出力されている



課題：7

粒子の種類、kinematical条件をいろいろ変え、結果を見る

最後に以下のコマンドでセッションを終了する

```
exit
```

マクロファイルの構造

課題:8　マクロファイルの基本構造を知る

- 定型的なUIコマンド入力をまとめて処理するためにはマクロファイルを用いるのが便利
- マクロファイルを実行するUIコマンドは以下のとおり（前スライドのmain()の中で実行されている）

```
/control/execute マクロファイル名
```

- 本演習で使われているマクロファイルの構造をみる

```
$ cd TestBench
$ ls
GlobalSetup.mac     primaryGeneratorSetup.mac     visSetup_Simplex.mac     verboseSetup.mac
```

［注］この構造は現演習用に用意したもので、ユーザは自らのアプリでこの構造を踏襲する必要はない

マクロの親ファイル
アプリのmain.ccで実行され、内部で右の3つのマクロを実行

照射粒子のキネマテックス等を設定するマクロ

可視化環境の設定マクロ
複雑なので、このまま使うことを推奨

Geant4カーネルからの種々の情報出力の制御マクロ

- GlobalSetupマクロファイルの内容をみる

```
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# GlobalSetup.mac: Top level macro to set up the initial environment to run "appli"
#  [Note] Geant4 Tutorial for Hep/Space Users
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

## Set up primary generaotr
/control/execute primaryGeneratorSetup.mac

## Set up visulalization ennrionment
/control/execute visSetup_Simplex.mac

## Set up verbosity
/control/execute verboseSetup.mac

## Invoke the application
/run/beamOn 0        # arg = 0; Only invokes initialization - no event generation
                     # arg > 0; Specified number of events will be executed
```

マクロファイルの使用例

課題：9　GlobalSetup.macとPrimaryGeneratorSetup.macの内容を変更してアプリを走らせてみる

```
$ cd TestBench
$ gedit GlobalSetup.mac    [注] gedit以外にエディターとしてatom, emacs, viが使用可能

  最後の行の'/run/beamOn'のパラメタを0から5に変更
    ー アプリを走らせると、自動的に5回初期粒子を照射してから'Idle'状態になる

$ gedit primaryGeneratorSetup.mac

  粒子の種類、エネルギーなどのkinematicsを自由に設定してみる

$ gedit verboseSetup.mac

  tracking verboseをゼロにする （/Tracking/verbose 0）
   [注]
     /run/beamOnで多数のprimary particlesの照射を指定した場合、
      tracking verboseがnon-zeroだとG4coutへの大量出力で実行時間が
     遅くなることに注意


$ ../bin/Application_Main
```

“.g4session”を使ったユーザ・インターフェイスの選択

課題：10　“.g4session”を使ったユーザ・インターフェイスの選択方法を知る

- この演習では今までに以下のユーザインターフェイスを使った
  - tcsh　　P01_FirstStep
  - Qt　　P01_FirstStep_Vis
- これらのユーザ・インターフェイスに加えて、以下のものも使うことが可能
  - csh, xm, win32, gag
- これらのユーザ・インターフェイスを選択するには、いくつかの手法が用意されている
  1) G4UIExecutiveを使う:　　G4UIExecutiveで直接指定:　P01_FirstStep:のmainを参照
  2) ユーザ環境変数を使う:　　G4UI_USE_QT, G4UI_USE_TCSH, などを定義
  3) .g4sessionファイルを使う　　これが最も手軽で推奨される手法　（以下参照）

  ［注］1)の手法と両立はしない
- ”.g4session”の使い方を以下の手順で試してみる
  1) ユーザのホームデレクトリにすでに”.g4session”というファイルが作られているので、その内容を確認

```
g4vm: less ~/.g4session

# - .g4session
#
#tcsh
qt
```

  2) エディタで”.g4session”を開き、その内容を以下のように変えたのち、Application_MainをTestBench ディレクトリで実行してみる

［注］これを実行した後は、必ず.g4sessionの内容を1）の内容に戻すこと ← 以後の演習の実行に影響があるので……

```
# - .g4session
#
tcsh
#qt
```

qtをコメントアウトしてtcshのコメントを外す

終了