# Geant4 を使ってみよう。

まずは、サンプルを動かしてみましょう。
ここからは、特に記載がない限り、/home/****/++++(Win:C:¥cygwin¥home¥++++)を基準ディレクトリとして扱います。(****は学年のディレクトリ、++++は自分の名前です。)

まずは、サンプルを動かすための下準備です。
xterm(ウィンドウ)を開いて下さい。(Windows 版の場合は、XWin server を起動してください。X server と共に xterm が開きます。)

ウィンドウが開いたらまず、Geant4 を使い始める前に必ず以下を実行してください。

[****@lnx* ++++]$ **source /usr/local/Geant4/geant4.9.1.p03/env.sh**
(Win: **source /cygdrive/c/Geant4/geant4_9_1_p03/env.sh**)

これで、Geant4 を動かす上で必要な環境変数が設定されました。この操作は、Geant4 のプログラムをコンパイルするウィンドウ毎に行ってください。そのウィンドウを閉じるまで、この環境変数の設定は有効です。

いよいよ、サンプルを動かします。
/usr/local/Geant4/geant4.9.1.p03/examples/novice(Win:/cygdrive/c/Geant4/geant4_9_1_p03/examples/novice)以下にある N0*ディレクトリ以下のものはすべてサンプルです。N01 から N07 まですべてをコンパイルして実行してみましょう。

まずは自分のディレクトリにサンプルをコピーします。
ここでは N02 をコピーした場合を示します。

[****@lnx* ++++]$ **cp -r /usr/local/Geant4/geant4.9.1.p03/examples/novice/N02 .**
(Win: **cp -r /cygdrive/c/Geant4/geant4_9_1_p03/examples/novice/N02 .** )

-r はディレクトリごとコピーするときのオプションです。
.(ドット)は、今自分がいるディレクトリ(カレント ディレクトリ)を指し、..は今自分がいるディレクトリの一つ上のディレクトリ(ペアレント ディレクトリ)を指します。

N02 ディレクトリに入ります。

[****@lnx* ++++]$ **cd N02**
(Win: **cd N02**)

N02 ディレクトリの中にある makefile を使って exampleN02.cc をコンパイルします。

[****@lnx* N02]$ **make**
(Win: **make**)

コンパイルが始まると以下のような画面出力がされます。

**Making dependency for file exampleN02.cc ...**
**Making dependency for file src/ExN02TrackerSD.cc ...**
**Making dependency for file src/ExN02TrackerHit.cc ...**
**Making dependency for file src/ExN02SteppingVerbose.cc ...**
**Making dependency for file src/ExN02SteppingAction.cc ...**
**Making dependency for file src/ExN02RunAction.cc ...**
**Making dependency for file src/ExN02PrimaryGeneratorAction.cc ...**
**Making dependency for file src/ExN02PhysicsList.cc ...**
**Making dependency for file src/ExN02MagneticField.cc ...**
**Making dependency for file src/ExN02EventAction.cc ...**
**Making dependency for file src/ExN02DetectorMessenger.cc ...**
**Making dependency for file src/ExN02DetectorConstruction.cc ...**
**Making dependency for file src/ExN02ChamberParameterisation.cc ...**
**Compiling ExN02ChamberParameterisation.cc ...**
**Compiling ExN02DetectorConstruction.cc ...**
**Compiling ExN02DetectorMessenger.cc ...**
**Compiling ExN02EventAction.cc ...**
**Compiling ExN02MagneticField.cc ...**
**Compiling ExN02PhysicsList.cc ...**
**Compiling ExN02PrimaryGeneratorAction.cc ...**
**Compiling ExN02RunAction.cc ...**
**Compiling ExN02SteppingAction.cc ...**
**Compiling ExN02SteppingVerbose.cc ...**
**Compiling ExN02TrackerHit.cc ...**

Compiling ExN02TrackerSD.cc ...
Creating/replacing object files in ./tmp/Linux-g++/exampleN02/libexampleN02.a ...
ar: ./tmp/Linux-g++/exampleN02/libexampleN02.a を作成します
Compiling exampleN02.cc ...
Using granular libraries ...
Linking exampleN02 ...
... Done!

Done!が出たら成功です。

*********Windows 版を使っている人のみ以下のことを行ってから次に進みます。*******
まず、emacs でカレントディレクトリにある vis.mac を開いてください。

$ emacs vis.mac

別ウィンドウで emacs が開きます。vis.mac のうちの以下の部分を赤字のように修正して保存します。

```
#
# Macro file for the initialization phase of "exampleN02.cc"
# when runing in interactive mode
#
# Sets some default verbose
#
/control/verbose 2
/run/verbose 2
#
# Create a scene handler for a specific graphics system
# (Edit the next line(s) to choose another graphics system)
#
#/vis/open OGLIX 600x600-0+0
/vis/open OGLSWin32
#
####/vis/open DAWNFILE
```

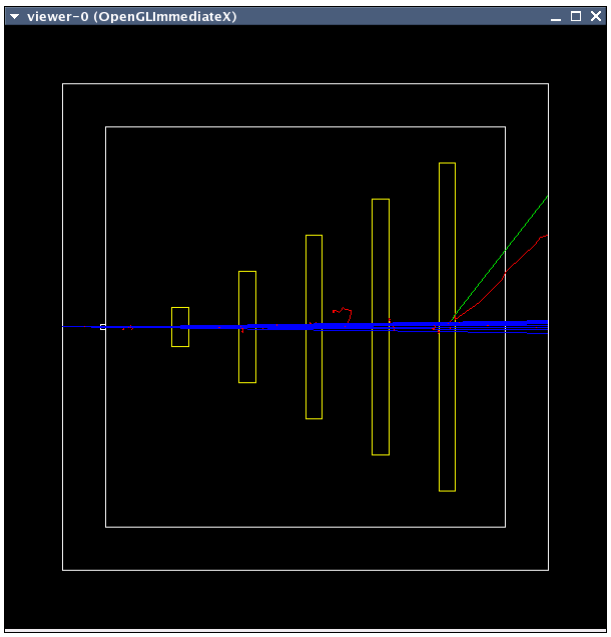
（続く…）

**********

コンパイルに成功したら、実行ファイルを起動します。
コンパイルが正しく進むと、カレントディレクトリに bin/Linux-g++(Win: bin/WIN32-VC)ディレクトリが自動的に作られて、実行ファイルはその中に生成されます。

[****@lnx* N02]$ **./bin/Linux-g++/exampleN02**
(Win: **./bin/WIN32-VC/exampleN02.exe**)

このプログラムは自動的に可視化ツールが立ち上がるようにプログラムされているので、以下のような画面が出てきたら成功です。




Idle> **exit**

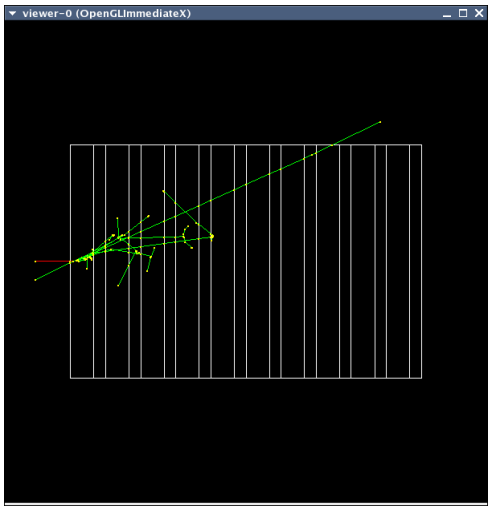
コマンドラインで exit と入力してプログラムを終了させます。

その他の N01、N03～N07 も同様にコピー、コンパイル、実行と順にやってみましょう。
Windows 版の人は、すべての vis.mac の同じ場所を上記の通り修正してください。
次のサンプルへ移る時には、自分の今いる場所を確認します。

[****@lnx* N02]$ **pwd**
(Win: **pwd**)

すると、自分が今いる場所が画面に出力されます。
/home/****/++++/N02
と出力された場合には自分の名前のディレクトリ(++++)の下の N02 ディレクトリにいます。別のサンプルをコピーする時には、自分の名前のディレクトリ(++++)に行ってから行います。上の例の場合は、
[****@lnx* N02]$ **cd ..**
(Win: **cd ..**)
として、一つ上のディレクトリに移動します。

N01 は実行してもテキストのアウトプットを出すだけなので、絵は出ません。
N03 は実行ファイルを起動してコマンドプロンプトが出たら以下のように/run/beamOn と実行して仮想ビームを出します。

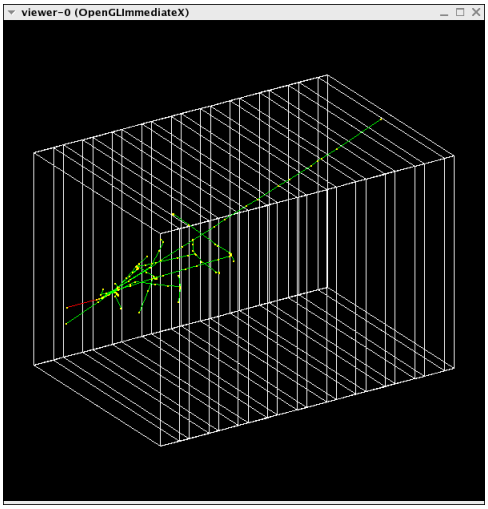



Idle> **/run/beamOn**

すると、左のような絵が描かれます。
白い線で示されているのがカロリメーターです。では、この検出器の向きを回転させて別の方向からこの反応を見てみましょう。

コマンドラインで、

Idle> **/vis/viewer/set/viewpointThetaPhi 40 140**




と打ち、視点をθ 方向に 40 度、φ 方向に 140 度回転させます。すると、左のような絵を見ることができます。
Windows 版の人は、プログラムを実行すると別の小さなウィンドウが立ち上がってきますので、(Idle> のようなプロンプトは出ませんが)その立ち上がった小さなウィンドウで同様にコマンドを打つと同じことができます。

N04 も、コマンドプロンプトが出たら/run/beamOn と仮想ビームを出してみましょう。また、検出器を見る視点を上記のように回転させてみましょう。
N05 は、まず最初にコマンドプロンプトで

Idle> **/control/execute vis.mac**

と実行して検出器の構造を可視化します。それから/run/beamOn と仮想ビームを出します。
N06 と N07 もコマンドプロンプトが出たら、仮想ビームを出してみましょう。また検出器を見る視点を回転してみましょう。

次は、/usr/local/Geant4/geant4.9.1.p03/examples/extended/analysis にある A01 ディレクトリをコピーしてきます。

**$ cp -r /usr/local/Geant4/geant4.9.1.p03/examples/extended/analysis/A01 .**
(Win: **cp -r /cygdrive/c/Geant4/geant4.9.1.p03/examples/extended/analysis/A01 .**)

コンパイルをして実行してみましょう。
コマンドプロンプトが出てくるので反応を可視化できるようにコマンドを打ちます。

Idle> **/vis/open OGLIX**
Idle> **/vis/drawVolume**
Idle> **/vis/scene/add/trajectories**
Idle> **/vis/scene/add/hits**
Idle> **/run/beamOn**

絵が描けることを確認したら

Idle> **exit**

と終了します。
これを毎回打つのは面倒なので、自分でマクロファイルを作ります。*filename* の部分を自分の好きな名前にして ***filename*.mac** というファイルを開きます。できるだけ、後で見直したときに何のファイルだか分からなくならないような名前をつけてください。ここでは、*filename* を **visualtest** とします。

[****@lnx* A01]$ **emacs visualtest.mac &**
(Win: **emacs visualtest.mac &**)

最後に&をつけることによって、今作業をしているウィンドウとは独立にemacsを立ち上げることができます。emacs が立ち上がったら上でコマンドライン上で入力した/vis/open OGLIX から /run/beamOn までを 1 行ずつ書いて保存します。保存して emacs を閉じたら、

[****@lnx* A01]$ **ls**
(Win: **ls**)

で、visualtest.mac が作られていることを確認します。
それから再び、実行ファイルを起動します。コマンドプロンプトが出たら

Idle> **/control/execute visualtest.mac**

と打つと、visualtest.mac に書いたことをやってくれて絵が同じように描けることを確認してください。できたら

Idle> **exit**

で終了します。
ここまでできたらサンプル(Geant4 の動作確認)は終了です。