Introdução ao GEANT4

Maurício Moralles
Instituto de Pesquisas Energéticas e Nucleares (IPEN/CNEN)

Técnicas Experimentais em Física de Partículas Elementares (IFUSP - 2018)

História

GEANT de GEometry ANd Tracking

Simulações para Física de Altas Energias

- 1974 - GEANT em FORTRAN (CERN)
- Até 2000: GEANT3.21
- 1994-1998 colaboração para novo pacote em C++

História

- 1998: GEANT4.0.0
- Dez/2000: GEANT4 3.0; o pacote passou a se chamar GEANT4
- Inclusão de processos adequados para baixa energia (< 1 keV)
- A cada 6/12 meses uma nova versão
- Dez/2017: última versão GEANT4 10.4
- Versão estável atual: 10.4.p02

O que é o GEANT4

- Não é um programa executável
- É um pacote de ferramentas que permite a criação de aplicativos para simulações com método de Monte Carlo que envolvam o transporte de partículas na matéria
- É composto de classes escritas em C++, uma linguagem orientada a objetos

- É desenvolvido por uma colaboração internacional de diversos institutos, envolvendo mais de 400 desenvolvedores
- É distribuído gratuitamente pela internet:
  http://geant4.web.cern.ch
- Distribuição e redistribuição é regida pela licença descrita na página
  http://geant4.web.cern.ch/geant4/license/LICENSE.html

O que contém o GEANT4

Ferramentas básicas para:

- Descrição de geometrias
- Descrição de materiais
- Escolha da fonte de partículas que estarão presentes na simulação
- Escolha de processos físicos a serem usados para a interação das partículas com os materiais
- Visualização da geometria e dos caminhos das partículas

Ferramentas avançadas:

- Funções para definir tipos de fontes de partículas: General Particle Source (GPS)
- Pacote para definir a geometria (GDML)
- Funções para registrar dados na forma de histogramas e n-tuplas (ROOT)
- Interfaces gráficas avançadas
- Opção de executar em Multithread
- outras ...

- O pacote contém grande número de exemplos que podem ser usados como ponto de partida para criar uma nova aplicação

- Há exemplos em diversas áreas: dosimetria, braquiterapia, terapia com prótons, radioproteção, radiação cósmica, fluorescência, altas energias, etc.

→ Cada exemplo contém uma documentação breve que descreve as ferramentas utilizadas no problema simulado

→ Convém ler a documentação de diversos exemplos para escolher o que mais se aproxima do problema que se quer simular

Suporte: documentação

- BookIntroToGeant4.pdf (10 páginas)
  - História e funcionalidades do GEANT4
- BookInstalGuide.pdf (52 páginas)
  - Manual de instalação (Linux/UNIX e Windows)
- BookForAppliDev.pdf (391 páginas)
  - Manual para o usuário criar seus programas
- PhysicsReferenceManual.pdf (561 páginas)
  - Descrição da implementação dos processos físicos incluídos no GEANT4

Suporte: User Forum

- Há uma página web dedicada à solução de problemas que os usuários encontram na utilização das ferramentas
  http://hypernews.slac.stanford.edu/HyperNews/geant4/cindex
- Normalmente as questões são respondidas em pouco tempo
- Há uma página web dedicada à comunicação de bugs

BookIntroToGeant4, cap. 5

Chapter 5. Software Knowledge Required to Use the Geant4 Toolkit

In general, there are three types of users:

- the **end user**,
- the **application programmer**,

and for large simulation tasks:

- the **framework provider**.

The **end user** runs the simulation program by controlling run time parameters. The interface with the program may be a graphical user interface, an interactive command line interface, or the macro-based system for batch. The end user needs a basic knowledge of how to control the program flow but does not necessarily have to know object-oriented programming or C++.

The **application programmer** is central to any simulation task. A firm knowledge of C++ is required to implement code in user action classes to specify, at a minimum, the detector description, the relevant particles and physics processes, and the initial event kinematics. A manual for the application programmer is found in the User's Guide: For Application Developers .

Using standard components of Geant4, a **framework provider** would add interfaces to external tools such as for example, to Computer Aided Design (CAD) programs, Object-Oriented Data Base Management Systems (ODBMS) and graphics systems. This requires the development of new classes overloading standard Geant4 functionality and hence a solid understanding of object-oriented Programming. A manual for the framework provider is found in the User's Guide: For Toolkit Developers.

Instalação (Linux, versão 10.4)

- Baixar o arquivo geant4.10.04.tar.gz da página http://geant4.web.cern.ch/support/download
- Escolher uma pasta para a instalação, e mover o arquivo para esta pasta. Exemplo “/usr/local/share”
- Descompactar o arquivo nesta pasta:
  → cd /usr/local/share
  → tar -zxvf geant4.10.04.tar.gz
- Criar a pasta de montagem com extensão “-build”
  → mkdir geant4.10.04-build
- → cd geant4.10.04-build
- Utilizar o “cmake” para configurar a instalação:
  → cmake -DCMAKE_INSTALL_PREFIX=/usr/local/share/geant4.10.04-install /usr/local/share/geant4.10.04 -DGEANT4_INSTALL_DATA=ON -DGEANT4_USE_OPENGL_X11=ON -DGEANT4_USE_QT=ON

- Informações sobre as opções do cmake podem ser encontradas no manual de instalação
- O cmake verificará se o seu sistema possui os pacotes necessários ao Geant4

```
-- The C compiler identification is GNU 6.3.0
-- The CXX compiler identification is GNU 6.3.0
-- Check for working C compiler: /usr/bin/cc
-- Check for working C compiler: /usr/bin/cc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Detecting C compile features
-- Detecting C compile features - done
-- Check for working CXX compiler: /usr/bin/c++
-- Check for working CXX compiler: /usr/bin/c++ -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found EXPAT: /usr/lib/x86_64-linux-gnu/libexpat.so (found version "2.2.0")
CMake Warning at /usr/share/cmake-3.7/Modules/FindQt4.cmake:618 (message):
  /usr/bin/qmake reported QT_INSTALL_LIBS as "/usr/lib/x86_64-linux-gnu" but
  QtCore could not be found there.  Qt is NOT installed correctly for the
  target build environment.
Call Stack (most recent call first):
  cmake/Modules/Geant4InterfaceOptions.cmake:115 (find_package)
  cmake/Modules/G4CMakeMain.cmake:64 (include)
  CMakeLists.txt:56 (include)
```

Caso algum pacote não esteja instalado, o cmake indica erros:

```
CMake Error at /usr/share/cmake-3.7/Modules/FindQt4.cmake:626 (message):
  Could NOT find QtCore.  Check
  /usr/local/share/geant4.10.04-build/CMakeFiles/CMakeError.log for more
  details.
Call Stack (most recent call first):
  cmake/Modules/Geant4InterfaceOptions.cmake:115 (find_package)
  cmake/Modules/G4CMakeMain.cmake:64 (include)
  CMakeLists.txt:56 (include)


-- Configuring incomplete, errors occurred!
```

O erro foi corrigido após instalação do pacote qt4-default, que estava faltando:

→ apt-get install qt4-default

```
root@eletron:/usr/local/share/geant4.10.04-build# cmake -DCMAKE_INSTALL_PREFIX=/usr/
local/share/geant4.10.04-install /usr/local/share/geant4.10.04 -DGEANT4_INSTALL_DATA=ON
-DGEANT4_USE_OPENGL_X11=ON -DGEANT4_USE_QT=ON
-- Looking for Q_WS_X11
-- Looking for Q_WS_X11 - found
-- Looking for Q_WS_WIN
-- Looking for Q_WS_WIN - not found
-- Looking for Q_WS_QWS
-- Looking for Q_WS_QWS - not found
-- Looking for Q_WS_MAC
-- Looking for Q_WS_MAC - not found
-- Found Qt4: /usr/bin/qmake (found version "4.8.7")
-- Found OpenGL: /usr/lib/x86_64-linux-gnu/libGL.so
-- Looking for XOpenDisplay in /usr/lib/x86_64-linux-gnu/libX11.so;/usr/lib/x86_64-
linux-gnu/libXext.so
-- Looking for XOpenDisplay in /usr/lib/x86_64-linux-gnu/libX11.so;/usr/lib/x86_64-
linux-gnu/libXext.so - found
-- Looking for gethostbyname
-- Looking for gethostbyname - found
-- Looking for connect
-- Looking for connect - found
-- Looking for remove
-- Looking for remove - found
-- Looking for shmat
-- Looking for shmat - found
-- Looking for IceConnectionNumber in ICE
-- Looking for IceConnectionNumber in ICE - found
-- Found X11: /usr/lib/x86_64-linux-gnu/libX11.so
-- Looking for sys/types.h
-- Looking for sys/types.h - found
-- Looking for stdint.h
-- Looking for stdint.h - found
-- Looking for stddef.h
-- Looking for stddef.h - found
-- Check size of off64_t
-- Check size of off64_t - done
-- Looking for fseeko
-- Looking for fseeko - found
```

```
-- Looking for unistd.h
-- Looking for unistd.h - found
-- Configuring download of missing dataset G4NDL (4.5)
-- Configuring download of missing dataset G4EMLOW (7.3)
-- Configuring download of missing dataset PhotonEvaporation (5.2)
-- Configuring download of missing dataset RadioactiveDecay (5.2)
-- Configuring download of missing dataset G4NEUTRONXS (1.4)
-- Configuring download of missing dataset G4PII (1.3)
-- Configuring download of missing dataset RealSurface (2.1)
-- Configuring download of missing dataset G4SAIDDATA (1.1)
-- Configuring download of missing dataset G4ABLA (3.1)
-- Configuring download of missing dataset G4ENSDFSTATE (2.2)
-- The following Geant4 features are enabled:
GEANT4_BUILD_CXXSTD: Compiling against C++ Standard '11'
GEANT4_USE_SYSTEM_EXPAT: Using system EXPAT library
GEANT4_USE_QT: Build Geant4 with Qt support
GEANT4_USE_OPENGL_X11: Build Geant4 OpenGL driver with X11 support

-- Configuring done
-- Generating done
-- Build files have been written to: /usr/local/share/geant4.10.04-build
```

A opção “-DGEANT4_INSTALL_DATA=ON” solicita que as bases de dados (seções de choque, informações sobre isótopos, etc) sejam baixadas durante a instalação.
Se não for feita, será preciso instalar cada base de dados manualmente.

Após a configuração da instalação, compilar os pacotes do Geant4:

→ make

... isso pode demorar dezenas de minutos ou horas, dependendo do computador

e instalar o Geant4 no sistema:

→ make install

Agora o Geant4 está disponível para qualquer usuário o sistema.

Para usá-lo num terminal, o usuário necessita apenas configurar as variáveis de ambiente daquele terminal, com o comando:

source /usr/local/share/geant4.10.04-install/bin/geant4.sh

Depende da pasta onde foi instalado

Testando a instalação do Geant4

Para testar a instalação, vamos compilar e rodar o exemplo mais simples.

1) Na sua pasta de usuário (*/home/usuario*), crie uma pasta com o nome da distribuição instalada. Exemplo:

→ mkdir geant4.10.04

Então você deve ter agora uma pasta /home/*usuario*/geant4.10.04

2) Entre na pasta onde serão criados os aplicativos do Geant4

→ cd geant4.10.04

3) Dê o comando para configurar as variáveis de ambiente do Geant4:

→ source /usr/local/share/geant4.10.04-install/bin/geant4.sh

4) Copie a pasta do exemplo B1 para este local:

→ cp -r /usr/local/share/geant4.10.04/examples/basic/B1 .

5) Crie a pasta onde será montado o executável, e mude para esta pasta:

→ mkdir B1-build

→ cd B1-build

6) Dê o comando cmake para configurar a montagem do exemplo B1 (ver manual):

→ cmake -DGeant4_DIR=/usr/local/share/geant4.10.04-install/lib/Geant4-10.0.2 /home/moralles/geant4.10.04/B1

7) Compile o exemplo B1:

→ make

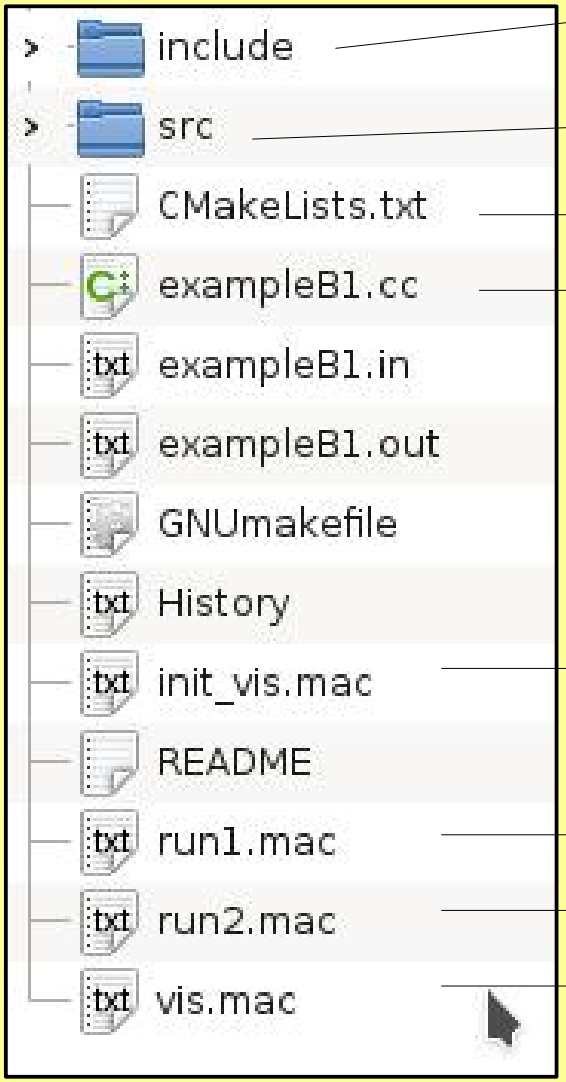
Agora a sua pasta deve ter um arquivo executável com nome

exampleB1

Para rodar o exemplo, dê o comando:

./exampleB1

A estrutura do exemplo B1



Pasta dos arquivos tipo “Headers”

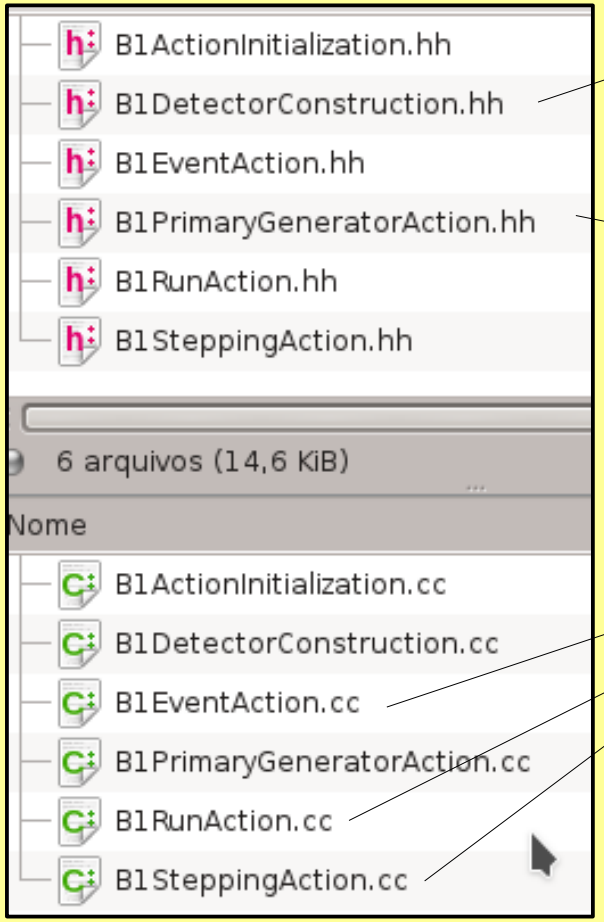
Pasta dos arquivos tipo “Source”

Instruções para o cmake

Arquivo com o main()

Arquivos com comandos macro

A estrutura do exemplo B1



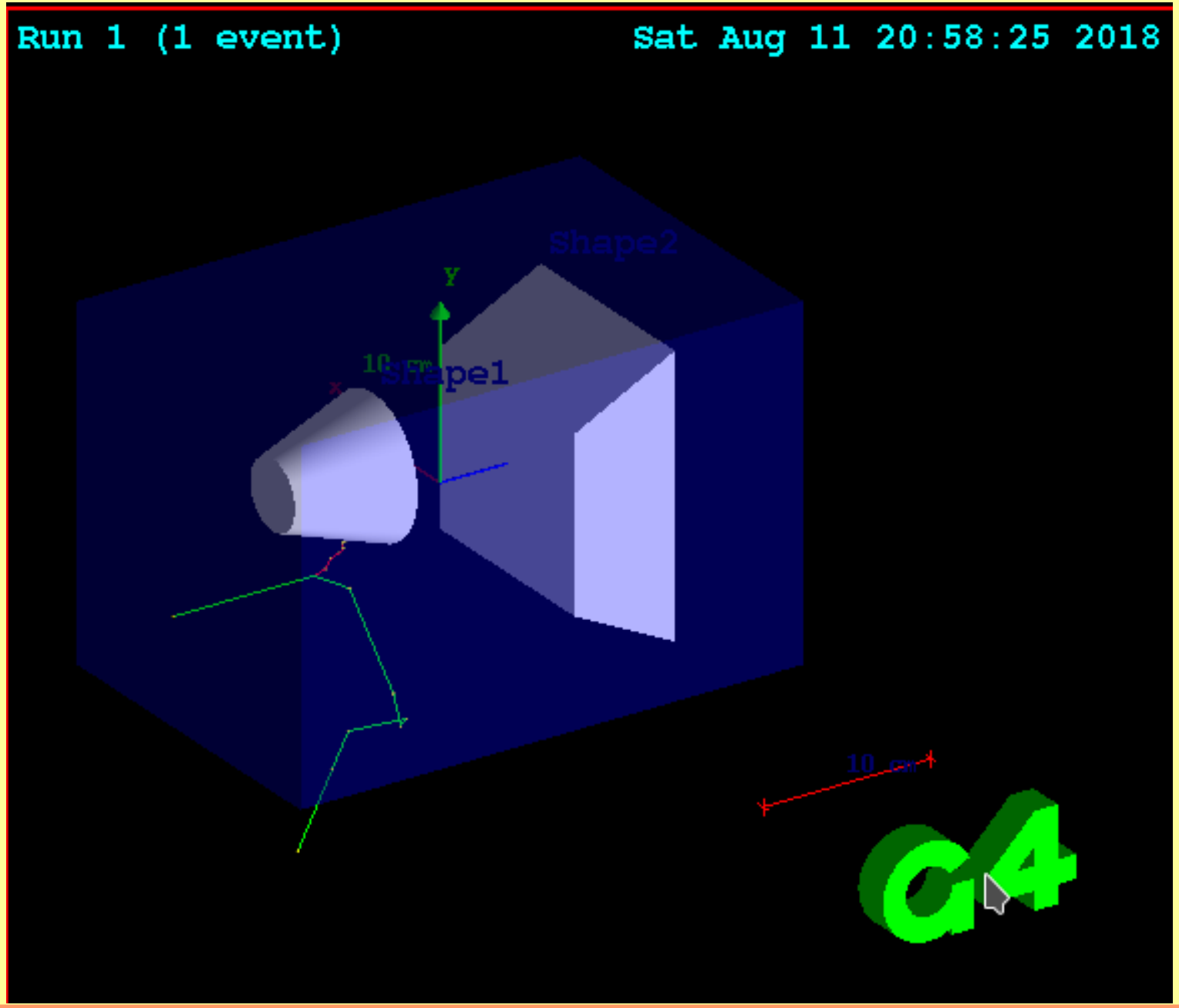
Definição do mundo, objetos (sólidos), materiais e posições

Definição das partículas primárias (feixe, fonte radioativa) suas geometrias, energias, direções de propagação e posições

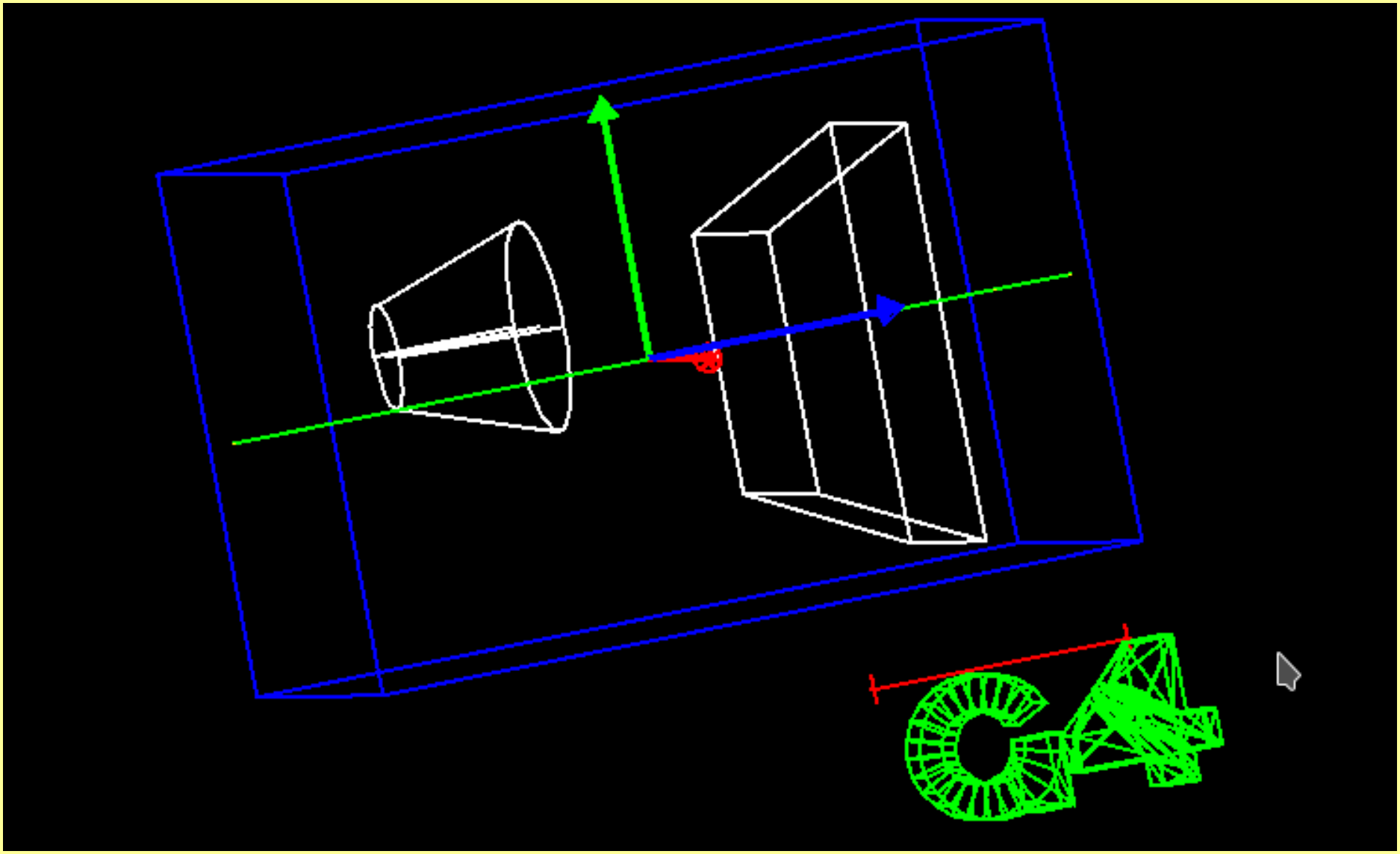
Definições e controles para a execução: inicialização da execução, cálculos das grandezas de interesse, impressão dos resultados, etc.

Outros exemplos podem ter mais arquivos nestas pastas, como um "PhysicsList" por exemplo, que define os modelos físicos para serem usados.
Neste exemplo B1 o PhysicsList está definido no main().

Visualização com OpenGL



Visualização com HepRep



Uso do Geant4 na sala 201

→ Usaremos o Geant4 instalado no grid “sampa”

→ Cada aluno usará apenas uma conta por conexão ssh:

ssh -Y alunoX@sampassh.if.usp.br

→ Cada aluno usará um número X indicado pelo professor com uma senha que deve ser trocada na primeira conexão.

→ Cada aluno usará o mesmo número X até o término da disciplina.

→ Usaremos o visualizador gráfico será o HepRep.

Copiar o exemplo B1 para sua área:

→ cp – r $G4INSTALL/share/Geant4-10.1.2/examples/basic/B1 .

Criar a pasta de montagem (build)

→ cd B1
→ mkdir build

Configurar o ambiente usando cmake

→ cd build
→ cmake ../

Compilar o exemplo B1:

→ make

Executar o exemplo B1:

→ ./exampleB1

Comandos macro

→ O Geant4 tem ferramentas para definir comandos macro que facilitam a mudança de parâmetros sem que seja necessário alterar o código e recompilar o programa

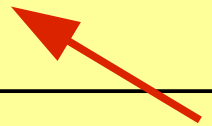
→ Alguns comandos macro já são definidos nas próprias ferramentas do Geant4, para mudanças na visualização, em processos físicos, etc

→ Os comandos macro específicos da aplicação devem ser criados pelo programador

→ Para ver os comandos macro do exemplo B1, pode-se usar o comando “help”

Comandos macro do arquivo “init_vis.mac”

```
# Macro file for the initialization of example B1
# in interactive session
#
# Set some default verbose
/control/verbose 2
/control/saveHistory
/run/verbose 2
#
# Change the default number of threads (in multi-threaded mode)
#/run/numberOfThreads 4
#
# Initialize kernel
/run/initialize
#
# Visualization setting
/control/execute vis.mac
```



Executa o macro de visualização

Comandos macro do arquivo “vis.mac”

```
# Macro file for the visualization setting in the initialization phase
# of the B1 example when running in interactive mode
#

# Use these open statements to open selected visualization
#
# Use this open statement to create an OpenGL view:
#/vis/open OGL 600x600-0+0
#
# Use this open statement to create an OpenInventor view:
#/vis/open OI
#
# Use this open statement to create a .prim file suitable for
# viewing in DAWN:
#/vis/open DAWNFILE
#
# Use this open statement to create a .heprep file suitable for
# viewing in HepRApp:
/vis/open HepRepFile
#
# Use this open statement to create a .wrl file suitable for
# viewing in a VRML viewer:
#/vis/open VRML2FILE
```

Comandos macro do arquivo “vis.mac”

```
# Disable auto refresh and quieten vis messages whilst scene and
# trajectories are established:
/vis/viewer/set/autoRefresh false
/vis/verbose errors
#
# Draw geometry:
/vis/drawVolume
#
# Specify view angle:
/vis/viewer/set/viewpointVector -1 0 0
/vis/viewer/set/lightsVector -1 0 0
#
# Specify style (surface, wireframe, auxiliary edges,...)
/vis/viewer/set/style wireframe
/vis/viewer/set/auxiliaryEdge true
/vis/viewer/set/lineSegmentsPerCircle 100
#
# Draw smooth trajectories at end of event, showing trajectory points
# as markers 2 pixels wide:
/vis/scene/add/trajectories smooth
/vis/modeling/trajectories/create/drawByCharge
/vis/modeling/trajectories/drawByCharge-0/default/setDrawStepPts true
/vis/modeling/trajectories/drawByCharge-0/default/setStepPtsSize 2
```

Comandos macro do arquivo “vis.mac”

```
# To superimpose all of the events from a given run:
/vis/scene/endOfEventAction accumulate

# Decorations
# Name
/vis/set/textColour green
/vis/set/textLayout right
/vis/scene/add/text2D 0.9 -.9 24 ! ! exampleB1
# or, if your system does not support right-adjustment
#/vis/scene/add/text2D 0 -.9 24 ! ! exampleB1
/vis/set/textLayout    # Revert to normal (left adjusted) layout
/vis/set/textColour    # Revert to default text colour (blue)
#
# Axes, scale, etc.
/vis/scene/add/scale   # Simple scale line
/vis/scene/add/axes    # Simple axes: x=red, y=green, z=blue.
/vis/scene/add/eventID # Drawn at end of event
/vis/scene/add/date    # Date stamp
/vis/scene/add/logo2D  # Simple logo
/vis/scene/add/logo    # 3D logo
#
```

Comandos macro do arquivo “vis.mac”

```
# Frame
/vis/set/colour red
/vis/set/lineWidth 2
/vis/scene/add/frame   # Simple frame around the view
/vis/set/colour        # Revert to default colour (white)
/vis/set/lineWidth     # Revert to default line width (1.)
#
# Attach text to one edge of Shape1, with a small, fixed offset
/vis/scene/add/text 0 6 -4 cm 18 4 4 Shape1
# Attach text to one corner of Shape2, with a small, fixed offset
/vis/scene/add/text 6 7 10 cm 18 4 4 Shape2
#
# To get nice view
/vis/geometry/set/visibility World 0 false
/vis/geometry/set/visibility Envelope 0 false
/vis/viewer/set/style surface
/vis/viewer/set/hiddenMarker true
/vis/viewer/set/viewpointThetaPhi 120 150
#
# Re-establish auto refreshing and verbosity:
/vis/viewer/set/autoRefresh true
/vis/verbose warnings
#
# For file-based drivers, use this to create an empty detector view:
#/vis/viewer/flush
```