Introductory training on Geant4
ANSTT5 Workshop 15-19 April 2024

iThemba LABS

1 Introduction

Geant4 is a toolkit for simulating the interaction of radiation or particles with matter using Monte Carlo methods. It was developed by CERN and is maintained by the international Geant4 collaboration, that is, many specialized working groups (around the globe) that are responsible for the various components of the toolkit (`https://geant4.web.cern.ch/`). It is widely used today in high energy, nuclear, medical and accelerator physics, as well as in space science. It is built using object-oriented C++ programming language.

2 Installing Geant4

Geant4 can be installed on many platforms. Depending on the operating system there may be slight variations in the install process. Follow the Geant4 installation guide here `https://geant4-userdoc.web.cern.ch/UsersGuides/InstallationGuide/html/index.html`. In general, to install Geant4, the following are the steps:

1. Install the pre-requisites or Geant4 pre-install packages.
2. Install the right (recommended) version of C++ compiler.
3. Download and install the recommended CMake version. `https://cmake.org/download/`
4. Download the Geant4 installer at `https://geant4.web.cern.ch/download/11.2.1.html`

In this workshop, we are mainly going to use Geant4.11.2 on Linux based (Ubuntu to be specific) platform.

2.1 Geant4 on virtual machines

- For training purposes, Geant4 can be installed on a virtual machine with an operating system of choice to circumvent installation problems on individual computers and on different operating systems.

- One can either install their preferred version of Geant4 on a pre-configured VM, or can download a portable pre-configured VM, with all the Geant4 prerequisite software and Geant4 installation already on it.
- CENBG and CNRS have, since 2004, developed and keep updating the GEANT4 VIRTUAL MACHINE (see here `https://geant4.lp2ib.in2p3.fr/`), which they avail to the public for free. Virtual Machines hosting various versions of Geant4 over the years are available here `https://extra.lp2ib.in2p3.fr/G4/download/` for download.
- Organisers of the annual course on Geant4 at CERN have put together clear instructions on how to install and use the Geant4 Virtual Machine here `https://indico.cern.ch/event/1370034/page/32238-geant4-virtual-machine`.
- In preparation for this component (Geant4 training) of the ANSTT5 Workshop we encourage attendees to follow these instructions to setup the Geant4 VM on their computers and test it to ensure that it works perfectly days ahead of the training.

2.2 On Linux: Ubuntu, etc.

For a very long time now, Linux environments have been among the most convenient operating systems to install and run Geant4. The steps given above are followed here.

1. Some of the software/packages needed to install Geant4 or to successfully run Geant4 applications.

```
# prerequisites:
sudo apt-get -y install build-essential openssl libssl-dev libssl1.0 libgl1-mesa-dev
libqt5x11extras5

# qt:
sudo apt-get install qtbase5-dev
sudo apt-get install qtdeclarative5-dev

# On Debian platforms
sudo apt-get install -y qt3d5-dev

# libxmu
sudo apt-get install -y libxmu-dev
```

   You may also follow this link to build Coin3D SoXt from the source `https://sourceforge.net/p/coin3d-soxt/wiki/SoXt_Build_Environment/`. And also `https://wiki.qt.io/Install_Qt_5_on_Ubuntu` for installing Qt5 on Ubuntu.

2. Download and install Geant4
   Assuming you have downloaded geant4 in the folder called Geant4, extract (unzip) it. Along side this folder (the source), create another folder and name it geant4-build.

```
# In the terminal
$ ls
 geant4-v11.2.0
$ mkdir geant4-build
$ ls
geant4-build geant4-v11.2.0
```

```
# Go into the folder geant4-build
$ cd geant4-build

# Now run ccmake:
$ ccmake ../geant4-v11.2.0

# ccmake-curses-gui will open as in the Fig 1 below
```

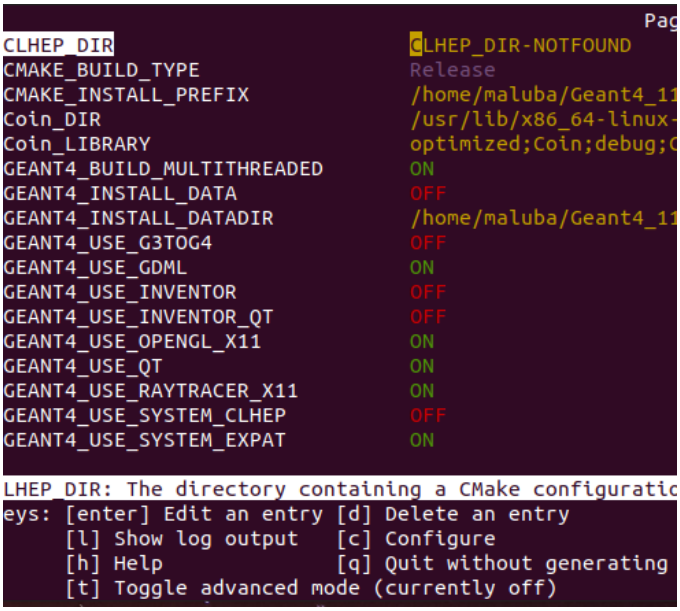

Figure 1: ccmake gui.

```
# If you run into compiler errors, you can try
$ ccmake -DCMAKE_CXX_COMPILER=/usr/bin/g++
 -DCMAKE_C_COMPILER=/usr/bin/gcc ../geant4-v11.2.1/

# Replace /usr/bin/g++ or gcc with the path to the location of these
compilers on your system.
```

3. When ccmake-curses-gui is open as in Fig. 1,
   - you can navigate up and down using the up/down keyboard arrows and press the "Enter" key to switch ON/OFF some required software.
   - You also need to set up the "install" directory following the "CMAKE_INSTALL_PREFIX" line, and this should be within the parent directory Geant4, for example.
   - Typically, this directory is created using the absolute path like so "/home/yourusername/Geant4/geant4-install".
   - Similarly, a path to the folder containing the datasets can be given as "/home/yourusername/Geant4/geant4-install/share/Geant4/data" against the line "GEANT4_INSTALL_DATADIR".
   - As stated in the Geant4 installation guide, the datasets can be downloaded and placed in the this folder. Make sure to extract (unzip) them.
   - Note that if the software you have switched on is not installed or has some missing dependencies it may give an error when you try to compile. You should either ensure

all required software is installed or just leave it OFF is not mandatory.

4. Finally, once the above is done, you can press keyboard character "c" (see the bottom of the ccmake screenshot in Fig. 1) to configure, if everything goes well (it doesn't complain about anything), then you can press "g" to generate the make file. Can press "e" when it is done to exit ccmake.

5. Next you type *make -j4* and enter to compile,

6. When it is done, type and enter *make install* to install Geant4. When it has finished running, you have installed Geant4 successfully, but you need to test it.

7. Before compiling an example to test whether your Geant4 installation is working properly, you need to run one of these scripts in the terminal:
   (1) *./geant4.sh* which is located in the directory
   /home/yourusername/Geant4-parent-directory/geant4-install/bin/
   Alternatively, you can source it like so
   source /home/yourusername/Geant4-parent-directory/geant4-install/bin/geant4.sh
   in the .bashrc file so that everytime you start a new terminal Geant4 is ready for use.
   or
   (2) *./geant4make.sh*, located in the directory
   /home/yourusername/Geant4-parent-directory/geant4-install/share/geant4make/
   Similarly, you may want to put the line
   source /home/name/Geant4-parent-directory/geant4-install/share/geant4make/geant4make.sh
   in the .bashrc file.
   Either of these scripts will set up the needed environmental variables required for execution of Geant4 programs.

8. To test your Geant4 installation, create a buildb1 folder within the parent Geant4 folder, in the terminal

```
# navigate to the Geant4 parent directory, if you list, ls command
$ ls
 geant4-build  geant4-install geant4-source

# make the buildb1 directory
$ mkdir buildb1

# If you list again, you have
 buildb1 geant4-build  geant4-install geant4-source

# Now cd into buildb1
$ cd buildb1
$ cmake -DCMAKE_PREFIX_PATH=/home/yourusername/Geant4-parent-directory
/geant4-install/lib/cmake/Geant4/
 /home/yourusername/Geant4-parent-directory/geant4-source/examples/basic/B1/
$ make -j4
```

9. At this point cmake will have generated a lot of files in the buildb1 directory, including an executable (in green) "exampleB1" which you can run in the terminal by simply **./exampleB1**.

2.3 On Mac Os.

In addition to the official Geant4 installation guide (https://geant4-userdoc.web.cern.ch/UsersGuides/InstallationGuide/html/index.html), there are plenty other resources online, for example, Youtube videos that can help you install Geant4 on macOs. For example, the following Youtube videos by Jing (aka Physino):

1. Geant4.10.7– https://www.google.com/search?client=ubuntu&sca_esv=236e917ef57ab2a8&channel=fs&cs=0&q=How+to+install+Geant4+on+Mac&sa=X&ved=2ahUKEwixvcSLu5GFAxXMW0EAbiw=1540&bih=807&dpr=1.2#fpstate=ive&vld=cid:088160b8,vid:Qk34s9xIF_4,st:0
2. Geant4.11.0– https://www.google.com/search?channel=fs&client=ubuntu&q=how+to+install+Geant4+11.2+on+macOs#fpstate=ive&vld=cid:b93c8f0b,vid:PrRFIhoT9Bw,st:0

2.4 On Windows.

Similarly, there are Youtube videos that one can follow to install Geant4 on Windows operating systems, such as

1. a Youtube video by John Francis, which can be followed when attempting to build Geant4 from the source: https://www.google.com/search?channel=fs&client=ubuntu&q=Geant4+installation#fpstate=ive&vld=cid:a1f68088,vid:w7k9PK1Ipv8,st:0, and
2. another Youtube video by Jing (aka Physino) that one can follow when installing a pre-compile Geant4 (Binary file) on Windows OS https://www.youtube.com/watch?v=XdzFsVGTbLc.

3 Developing a Geant4 simulation app

1. The structure of the a geant4 application:
   In general, a Geant4 program will be placed in a folder consisting of the cmake files, macros, header and source files placed in separate folders.
   - CMakeLists.txt-is the file giving cmake information to build the program.

```
#----------------------------------------------------------------------------
# Setup the project
#
cmake_minimum_required(VERSION 3.16...3.27)
project(B4c)

#----------------------------------------------------------------------------
# Find Geant4 package, activating all available UI and Vis drivers by default
# You can set WITH_GEANT4_UIVIS to OFF via the command line or ccmake/cmake-gui
# to build a batch mode only executable
#
option(WITH_GEANT4_UIVIS "Build example with Geant4 UI and Vis drivers" ON)
if(WITH_GEANT4_UIVIS)
  find_package(Geant4 REQUIRED ui_all vis_all)
else()
  find_package(Geant4 REQUIRED)
```

```
endif()

#----------------------------------------------------------------------------
# Setup Geant4 include directories and compile definitions
# Setup include directory for this project
#
include(${Geant4_USE_FILE})
include_directories(${PROJECT_SOURCE_DIR}/include)

#----------------------------------------------------------------------------
# Locate sources and headers for this project
# NB: headers are included so they will show up in IDEs
#
file(GLOB sources ${PROJECT_SOURCE_DIR}/src/*.cc)
file(GLOB headers ${PROJECT_SOURCE_DIR}/include/*.hh)

#----------------------------------------------------------------------------
# Add the executable, and link it to the Geant4 libraries
#
add_executable(exampleB4c exampleB4c.cc ${sources} ${headers})
target_link_libraries(exampleB4c ${Geant4_LIBRARIES})

#----------------------------------------------------------------------------
# Copy all scripts to the build directory, i.e. the directory in which we
# build B4c. This is so that we can run the executable directly because it
# relies on these scripts being in the current working directory.
#
set(EXAMPLEB4C_SCRIPTS
  exampleB4c.out
  exampleB4.in
  gui.mac
  init_vis.mac
  plotHisto.C
  plotNtuple.C
  run1.mac
  run2.mac
  vis.mac
  )

foreach(_script ${EXAMPLEB4C_SCRIPTS})
  configure_file(
    ${PROJECT_SOURCE_DIR}/${_script}
    ${PROJECT_BINARY_DIR}/${_script}
    COPYONLY
    )
endforeach()

#----------------------------------------------------------------------------
# Install the executable to 'bin' directory under CMAKE_INSTALL_PREFIX
#
install(TARGETS exampleB4c DESTINATION bin)
```

- The main() method which is implemented by two toolkit classes, namely *G4RunManager*

and *G4UImanager*, and three classes, *DetectorConstruction*, *PhysicsList* and *ActionInitialization*, which are derived from the toolkit classes.

```
/// \file exampleB4c.cc
/// \brief Main program of the B4c example

#include "DetectorConstruction.hh"
#include "ActionInitialization.hh"

#include "G4RunManagerFactory.hh"
#include "G4SteppingVerbose.hh"
#include "G4UIcommand.hh"
#include "G4UImanager.hh"
#include "G4UIExecutive.hh"
#include "G4VisExecutive.hh"
#include "FTFP_BERT.hh"
#include "Randomize.hh"

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo.....

namespace {
  void PrintUsage() {
    G4cerr << " Usage: " << G4endl;
    G4cerr << " exampleB4c [-m macro ] [-u UIsession] [-t nThreads] [-vDefault]"
           << G4endl;
    G4cerr << "   note: -t option is available only for multi-threaded mode."
           << G4endl;
  }
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo.....

int main(int argc,char** argv)
{
  // Evaluate arguments
  //
  if ( argc > 7 ) {
    PrintUsage();
    return 1;
  }

  G4String macro;
  G4String session;
  G4bool verboseBestUnits = true;
#ifdef G4MULTITHREADED
  G4int nThreads = 0;
#endif
  for ( G4int i=1; i<argc; i=i+2 ) {
    if      ( G4String(argv[i]) == "-m" ) macro = argv[i+1];
    else if ( G4String(argv[i]) == "-u" ) session = argv[i+1];
#ifdef G4MULTITHREADED
    else if ( G4String(argv[i]) == "-t" ) {
      nThreads = G4UIcommand::ConvertToInt(argv[i+1]);
```

```
    }
#endif
    else if ( G4String(argv[i]) == "-vDefault" ) {
      verboseBestUnits = false;
      --i;  // this option is not followed with a parameter
    }
    else {
      PrintUsage();
      return 1;
    }
  }

  // Detect interactive mode (if no macro provided) and define UI session
  //
  G4UIExecutive* ui = nullptr;
  if ( ! macro.size() ) {
    ui = new G4UIExecutive(argc, argv, session);
  }

  // Optionally: choose a different Random engine...
  // G4Random::setTheEngine(new CLHEP::MTwistEngine);

  // Use G4SteppingVerboseWithUnits
  if ( verboseBestUnits ) {
    G4int precision = 4;
    G4SteppingVerbose::UseBestUnit(precision);
  }

  // Construct the default run manager
  //
  auto runManager =
    G4RunManagerFactory::CreateRunManager(G4RunManagerType::Default);
#ifdef G4MULTITHREADED
  if ( nThreads > 0 ) {
    runManager->SetNumberOfThreads(nThreads);
  }
#endif

  // Set mandatory initialization classes
  //
  auto detConstruction = new B4c::DetectorConstruction();
  runManager->SetUserInitialization(detConstruction);

  auto physicsList = new FTFP_BERT;
  runManager->SetUserInitialization(physicsList);

  auto actionInitialization = new B4c::ActionInitialization();
  runManager->SetUserInitialization(actionInitialization);

  // Initialize visualization
  auto visManager = new G4VisExecutive;
  // G4VisExecutive can take a verbosity argument - see /vis/verbose guidance.
  // auto visManager = new G4VisExecutive("Quiet");
```

```
  visManager->Initialize();

  // Get the pointer to the User Interface manager
  auto UImanager = G4UImanager::GetUIpointer();

  // Process macro or start UI session
  //
  if ( macro.size() ) {
    // batch mode
    G4String command = "/control/execute ";
    UImanager->ApplyCommand(command+macro);
  }
  else  {
    // interactive mode : define UI session
    UImanager->ApplyCommand("/control/execute init_vis.mac");
    if (ui->IsGUI()) {
      UImanager->ApplyCommand("/control/execute gui.mac");
    }
    ui->SessionStart();
    delete ui;
  }

  // Job termination
  // Free the store: user actions, physics_list and detector_description are
  // owned and deleted by the run manager, so they should not be deleted
  // in the main() program !

  delete visManager;
  delete runManager;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo.....
```

- Other classes derived for example from *G4VUserPrimaryGeneratorAction*, *G4VSensitiveDetector*, *G4VHit*, *G4UserRunAction*, *G4UserEventAction*, and availed via the *G4VUserActionInitialization* class.
- Macros-files containing commands that can be used to change the way the simulation proceeds without necessarily editting the sourced code.

2. Detector Construction
   - Define the Solid, the Logic Volume and the Physical Volume, which is the positioning or placement of the volume.

```
class G4VPhysicalVolume
class G4LogicalVolume
class G4Material
class G4Box
class G4Tubs
class G4Cons
```

   - Define materials...

```
        G4double z, a, density;
        density = 1.390*g/cm3;
        a = 39.95*g/mole;
        G4Material* lAr = new G4Material(name="liquidArgon", z=18., a, density);
```

```
G4NistManager* man = G4NistManager::Instance();
G4Material* H2O = man->FindOrBuildMaterial("G4_WATER");
G4Material* Air= man->FindOrBuildMaterial("G4_AIR");
```

- Define particles...

```
// G4VUserPhysicsList consists of the following methods
ConstructParticle(); // construction of particles
ConstructProcess(); // construct processes and register them to particles
```

3. Make the detector sensitive

```
  auto absoSD
    = new CalorimeterSD("AbsorberSD", "AbsorberHitsCollection"); //, fNofLayers);
  G4SDManager::GetSDMpointer()->AddNewDetector(absoSD);
  SetSensitiveDetector("AbsoLV",absoSD);
```

4. Extracting particle interaction data

```
      G4int         fTrackID;
      G4int         fVolumeNo;
      G4String      fVolume;
      G4double      fEdep;
      G4double      fKinEnergy;
      G4ThreeVector fPos;
      G4ThreeVector fMomentum;
      G4double fTrackLength;
      G4ParticleDefinition* fParticle;
```

```
inline void* CalorHit::operator new(size_t)
{
  if (!CalorHitAllocator) {
    CalorHitAllocator = new G4Allocator<CalorHit>;
  }
  void *hit;
  hit = (void *) CalorHitAllocator->MallocSingle();
  return hit;
}

inline void CalorHit::operator delete(void *hit)
{
  if (!CalorHitAllocator) {
    CalorHitAllocator = new G4Allocator<CalorHit>;
  }
  CalorHitAllocator->FreeSingle((CalorHit*) hit);
}

inline void CalorHit::Add(G4double de, G4double dl) {
```

```
  fEdep += de;
  fTrackLength += dl;
}

inline G4double CalorHit::GetEdep() const {
  return fEdep;
}

inline G4double CalorHit::GetTrackLength() const {
  return fTrackLength;
}
```

5. Hands on experience...