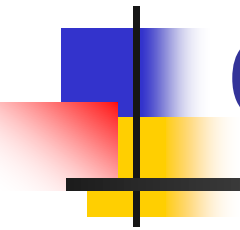
Geant4: basic concept

Luciano Pandola

INFN



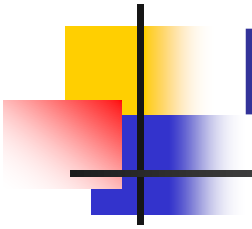
The (conceptual) recipe for a Geant4-based application

Interaction with the Geant4 kernel - 1

- Geant4 design provides **tools** for a user application
  - To tell the kernel about your simulation configuration
  - To interact with Geant4 kernel itself

- Geant4 tools for user interaction are **base classes**
  - You create **your own concrete class** derived from the base classes → interface to the Geant4 kernel
  - Geant4 kernel handles your own derived classes transparently through their base class interface (polymorphism)

Interaction with the Geant4 kernel - 2



Two types of Geant4 base classes:

- **Abstract base classes** for user interaction (classes starting with G4V)
  - User derived concrete classes are **mandatory**
  - User to implement the purely virtual methods
- **Concrete base classes** (with **virtual** dummy default methods) for user interaction
  - User derived classes are **optional**

User Classes (from 10.0)

Initialisation classes

Invoked at the initialization

- G4VUserDetectorConstruction
- G4VUserPhysicsList

Global: only one instance of them exists in memory, shared by all threads (**readonly**). Managed only by the master thread.

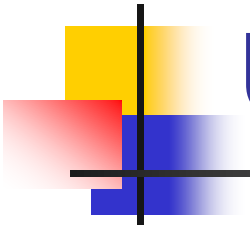
Action classes

Invoked during the execution loop

- G4VUserActionInitialization
  - G4VUserPrimaryGeneratorAction
  - G4UserRunAction (*)
  - G4UserEventAction
  - G4UserTrackingAction
  - G4UserStackingAction
  - G4UserSteppingAction

Local: an instance of each action class exists **for each thread**.

(*) Two RunAction's allowed: one for master and one for threads

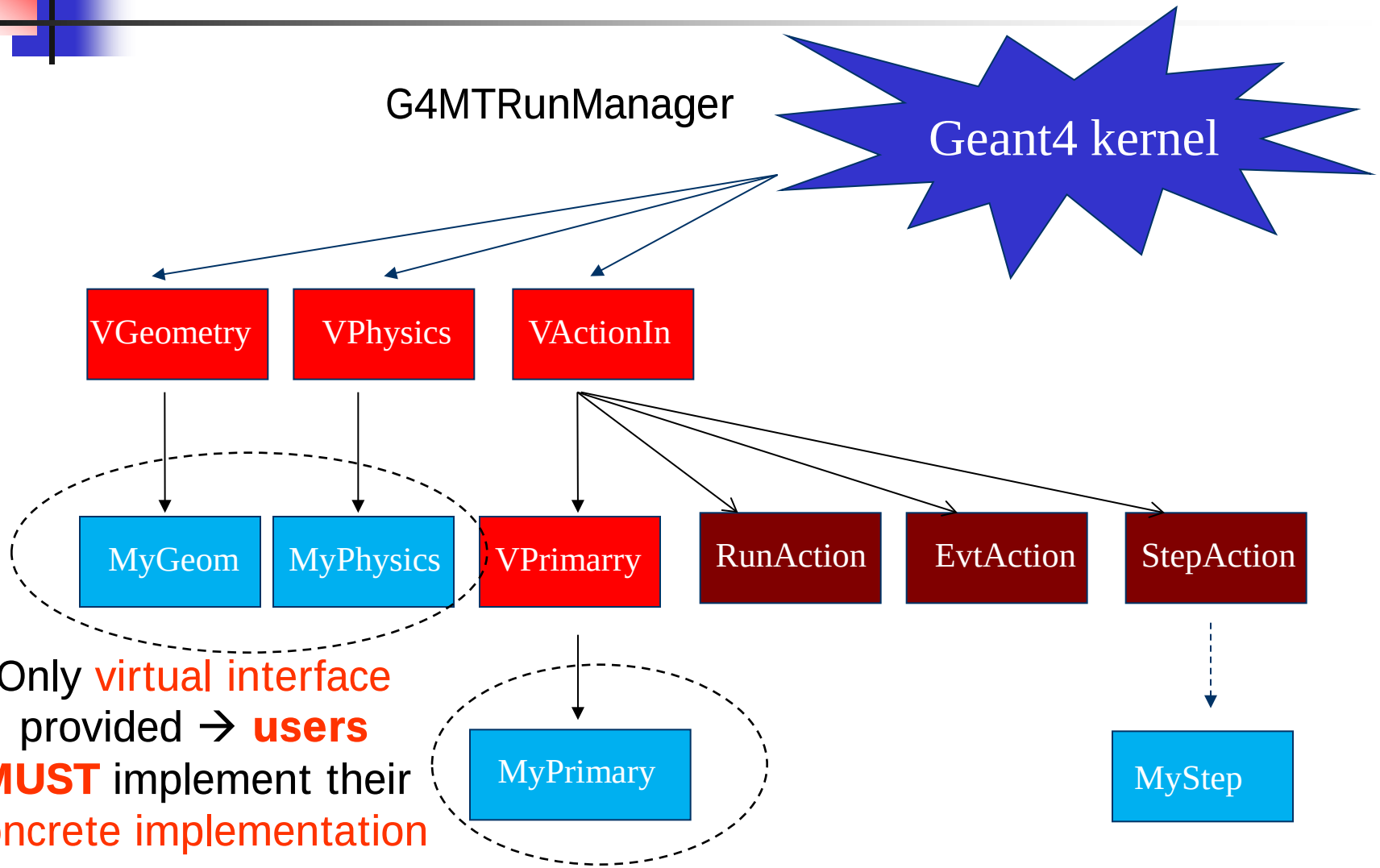


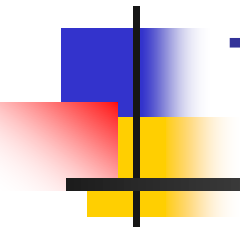
User Classes - 2

Mandatory classes in ANY Geant4 User Application

- G4VUserDetectorConstruction
  describe the experimental set-up
- G4VUserPhysicsList
  select the physics you want to activate
- G4VUserActionInitialization
  takes care of the user initializations
  G4VUserPrimaryGeneratorAction

Geant4 concept (MT)





The mandatory user classes

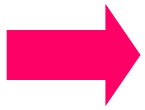
The geometry

- User class which describes the geometry must inherit from **`G4VUserDetectorConstruction`** and registered in the Run Manager
- Virtual base class: the purely virtual method must be implemented
  - **`G4VPhysicalVolume* Construct() = 0;`**
    - Must return the pointer to the world volume: all other volumes are contained in it
- Optionally, implement the virtual method
  - **`void ConstructSDandField();`**
    - Defines sensitive volumes and EM fields

Select physics processes

- Geant4 doesn't have any default particles or processes
- Derive your own concrete class from the **`G4VUserPhysicsList`** abstract base class
  - define all necessary particles
  - define all necessary processes and assign them to proper particles
  - define γ/δ production thresholds (in terms of range)
- Pure virtual methods of **`G4VUserPhysicsList`**

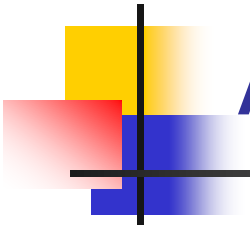
ConstructParticles()
ConstructProcesses()
SetCuts()



must be implemented by the user in his/her concrete derived class

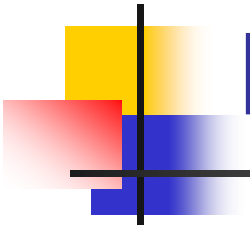
Physics Lists

- Geant4 doesn't have any default particles or processes
- *Partially true*: there is no default, but there are a set of **"ready-for-use" physics lists** released with Geant4, tailored to different use cases. Mix and match:
  - Different sets of hadronic models (depending on the energy scale and modeling of the interactions)
  - Different options for neutron tracking
    - Do we need (CPU-intensive) description of thermal neutrons, neutron capture, etc?
  - Different options for EM physics
    - Do you need (CPU-intensive) precise description at the low-energy scale (< 1 MeV)? E.g. fluorescence, Doppler effects in the Compton scattering, Auger emission, Rayleigh diffusion
    - Only a waste of CPU time for LHC, critical for many low-background experiments



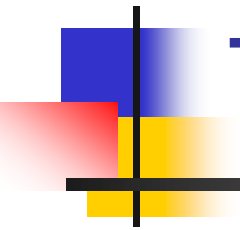
Action Initialization

- New in Geant4 10.0 (supports multi-thread)
- User class must inherit from **`G4VUserActionInitialization`** and registered in the Run Manager
- Implement the purely virtual method
  - **`void Build() = 0;`**
  - Invoked in sequential mode and in MT mode by all workers
  - Must instantiate at least the primary generator
- Optional virtual method
  - **`void BuildForMaster();`**
  - Invoked by the master in MT mode. Applies **only** to Run Action (all other user actions are thread-local)



Primary generator

- User class must inherit from **`G4VUserPrimaryGeneratorAction`**
  - Registered to the Run Manager via the ActionInizialitation (MT mode)
    - Register directly to the RunManager in seq-mode
- Implement the purely virtual method
  - **`void GeneratePrimaries(G4Event*)=0;`**
  - Called by the RunManager during the event loop, to generate the primary vertices/particles
- Uses internally a concrete instance of **`G4VPrimaryGenerator`** (e.g. **`G4ParticleGun`**) to do the job



The optional user classes

Optional user classes - 1

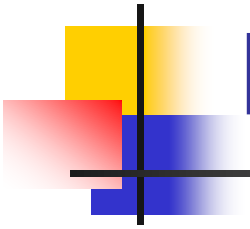
- Five concrete base classes whose virtual member functions the user may override to gain **control** of the simulation at various stages
  - G4User**Run**Action
  - G4User**Event**Action
  - G4User**Tracking**Action
  - G4User**Stacking**Action
  - G4User**Stepping**Action

e.g. actions to be done at the beginning and end of each event

- Each member function of the base classes has a dummy implementation (**not** purely virtual)
  - Empty implementation: does nothing

Optional user classes - 2

- The user may implement the member functions he desires in his/her derived classes
  - E.g. one may want to perform some action at each tracking step
- Objects of user action classes must be **registered** to the Run Manager via the **`G4VActionInizialization`**
  - Notice: in the old-style sequential mode, the user action classes can be registered directly to the Run Manager



Methods of user classes - 1

G4UserRunAction

- **`BeginOfRunAction(const G4Run*)`** // book histos
- **`EndOfRunAction(const G4Run*)`** //store histos

G4UserEventAction

- **`BeginOfEventAction(const G4Event*)`** //initialize event
- **`EndOfEventAction (const G4Event*)`** // analyze event

G4UserTrackingAction

- **`PreUserTrackingAction(const G4Track*)`**
  //decide to store/not store a given track
- **`PostUserTrackingAction(const G4Track*)`**

Methods of user classes - 2

G4UserSteppingAction

- **`UserSteppingAction(const G4Step*)`**

  //kill, suspend, pospone the track, draw the step, ...

G4UserStackingAction

- **`PrepareNewEvent()`** //reset priority control
- **`ClassifyNewTrack(const G4Track*)`**

  // Invoked when a new track is registered (e.g. kill, pospone)
- **`NewStage()`**

  // Invoked when the Urgent stack becomes empty (re-classify, abort event)

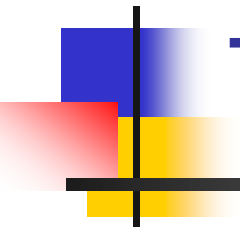
MyActionInitialization (MT mode)

- Register thread-local user actions

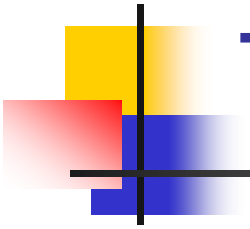
```
void MyActionInitialization::Build() const
{
  //Set mandatory classes
  SetUserAction(new MyPrimaryGeneratorAction());
  // Set optional user action classes
  SetUserAction(new MyEventAction());
  SetUserAction(newMyRunAction());
}
```

- Register RunAction for the master

```
void MyActionInitialization::BuildForMaster() const
{
  // Set optional user action classes
SetUserAction(newMyMasterRunAction());
}
```

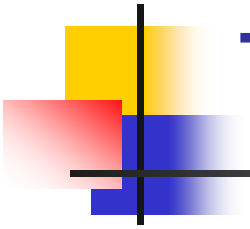


The main() program



The main() program - 1

- Geant4 does **not** provide the **main()**
  - Geant4 is a toolkit!
  - The main() is part of the user application
- In his/her main(), the user **must**
  - construct **`G4RunManager`** (or his/her own derived class)
  - notify the **`G4RunManager`** mandatory user classes derived from
    - **`G4VUserDetectorConstruction`**
    - **`G4VUserPhysicsList`**
    - **`G4VUserActionInitialization`** (takes care of Primary)
  - In MT mode, use **`G4MTRunManager`**



The main() program - 2

- The user may define in his/her main()
  - optional user action classes
  - VisManager, (G)UI session
- The user also has to take care of retrieving and saving the relevant information from the simulation (Geant4 will not do that by default)
- Don’t forget to delete the G4RunManager at the end

Sequential vs. MT main()

- The MT vs. sequential mode can be chosen in the main() by picking the appropriate RunManager:
  - **G4RunManager** for sequential
  - **G4MTRunManager** for multi-thread

```
  // Construct the default run manager. Pick the proper run
  // manager depending if the multi-threading option is
  // active or not.
#ifdef G4MULTITHREADED
  G4MTRunManager* runManager = new G4MTRunManager;
#else
  G4RunManager* runManager = new G4RunManager;
#endif
```

An example of (MT) main()

```
{
  …
  // Construct the default run manager
  G4MTRunManager* runManager = new G4MTRunManager;

 // Set mandatory user initialization classes
  MyDetectorConstruction* detector = new MyDetectorConstruction;
  runManager->SetUserInitialization(detector);
  MyPhysicsList* physicsList = new MyPhysicsList;
  runManager->SetUserInitialization(myPhysicsList);

  // Set mandatory user action classes
  runManager->SetUserAction(new MyActionInitialization);
  …
}
```

General recipe for novice users

Experienced users may do **much more**, but the conceptual process is still the same...

- Design your application... requires some preliminar **thinking** (what is it supposed to do?)
- Create your derived **mandatory** user classes
  - My**DetectorConstruction**
  - My**PhysicsList**
  - My**ActionInitialization** (must register My**PrimaryGenerator**)
- Create optionally your derived user action classes
  - My**UserRunAction**, My**UserEventAction**, ...
- Create your **main()**
  - Instantiate **`G4RunManager`** or your own derived MyRunManager
  - Notify the RunManager of your mandatory and optional user classes
  - Optionally initialize your favourite User Interface and Visualization
- That's all!

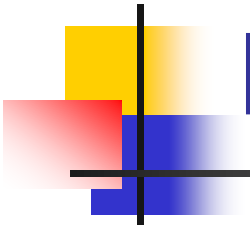
http://geant4.org

Documentation

- A few manuals available in the Geant4 webpage
  - Application developer manual
  - Physics manual
- Other tools available
  - LXR code repository
  - User forum
  - Bugzilla

User Support

1. Getting started
2. Training courses and materials
3. Source code
   a. Download page
   b. LXR code browser -or- draft doxygen documentation
4. Frequently Asked Questions (FAQ)
5. Bug reports and fixes
6. User requirements tracker
7. User Forum
8. Documentation
   a. Introduction to Geant4
   b. Installation Guide
   c. Application Developers Guide
   d. Toolkit Developers Guide
   e. Physics Reference Manual
9. Examples
10. Physics lists
    a. Electromagnetic
    b. Hadronic
11. User Aids
    a. Tips for improving CPU performance

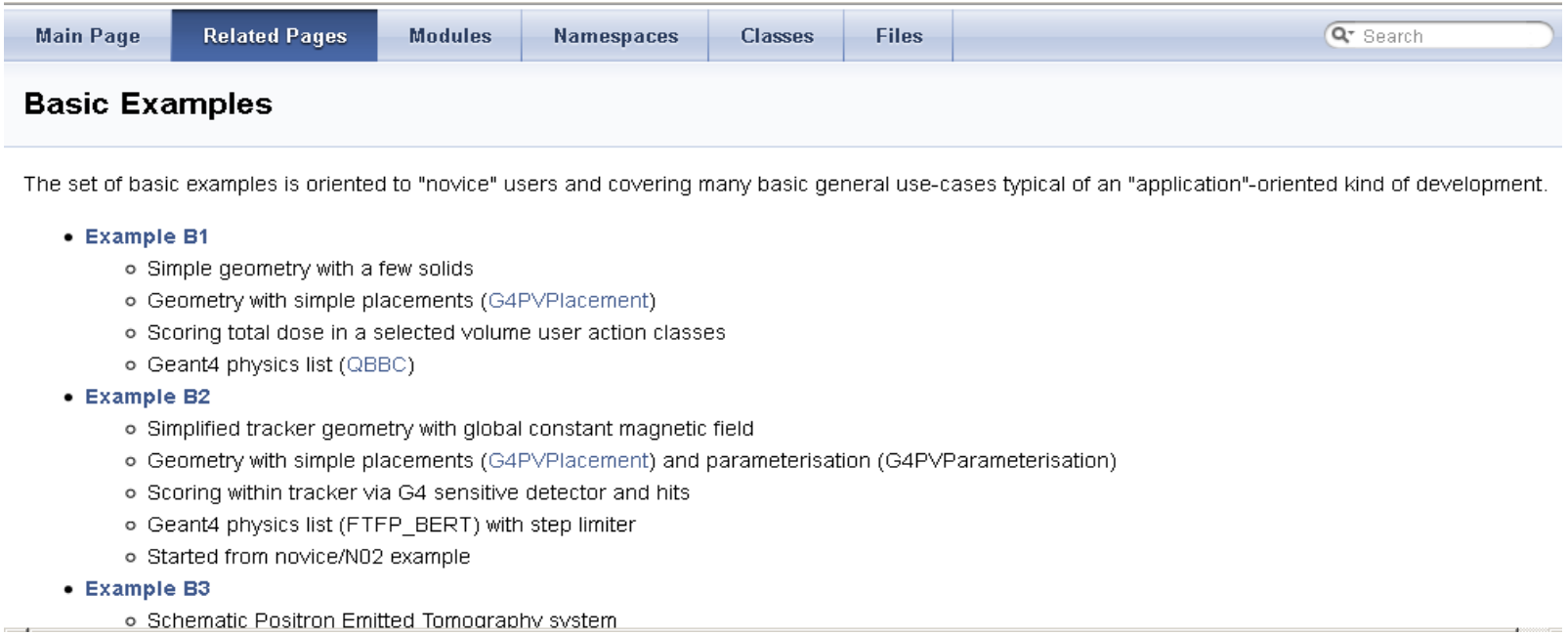


Examples

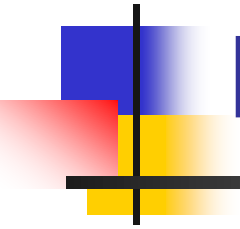
- Ready-for-the-use Geant4 applications (examples) are distributed with Geant4
  - Very good starting point for new users
- Three suites of examples:
  - **"basic"**: oriented to novice users and covering the most typical use-cases of a Geant4 application with keeping simplicity and ease of use.
  - **"extended"**: covers many specific use cases for actual detector simulation.
  - **"advanced"**: where real-life complete applications for different simulation studies are provided
- The exercises of this course are based on the basic example **B3** (pretty much revisited,though)

Examples

- A webpage with doxygen documentation is available for the basic/extended examples



http://cern.ch/geant4/UserDocumentation/Doxygen/examples_doc/html



Build Geant4

If the System Manager did not already do for you

Supported platforms and compilers

- Linux systems
  - Scientific Linux CERN 6 with gcc 4.7.X, 64 bit

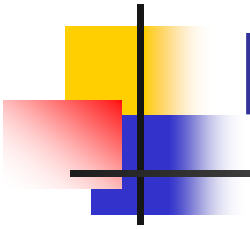
    Geant4 has also been successfully compiled on other Linux distributions, including Debian, Ubuntu and openSUSE (not officially supported)

- MacOSX systems
  - Mac OS X 10.8 (Lion with clang 3.3), 64bit

    Geant4 has also been successfully compiled on Mac OS X 10.7 (Lion) with clang 3.1 (Apple), (not officially supported)

- Windows systems
  - Windows 7 with Visual Studio 11 (VS2010).

Check Geant4 supported platforms in **http://geant4.org**

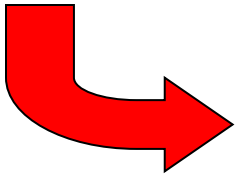


Required items and software

- The Geant4 toolkit source code (10.00)
- C++ compiler
  - It is usually installed on your Linux. If not, you need to install it (not shown here)
- CMake 2.6.4 or higher
  - for clang, Cmake 2.8.2 or higher is required
- The Geant4 data files
  - an automatic procedure can retrieve them (with CMake)

Where to download the Geant4 source code

- **Geant4**



Geant4 10.0
released 6 December 2013

The Geant4 source code is freely available. See the licence conditions.

Please read the **Release Notes** before downloading or using this release.
It is required to apply a full rebuild of the libraries.

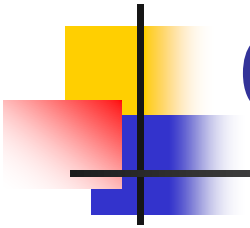
Source files

Please choose the archive best suited to your system and archiving tool:

Download — GNU or Linux tar format, compressed using gzip (29.4Mb, 30780131 bytes)
After downloading, gunzip, then unpack using GNU tar.

Download — ZIP format (41.4Mb, 43365939 bytes)
After downloading, unpack using e.g. WinZip.

http://geant4.cern.ch/support/download.shtml



Geant4 installation (10.0)

- Working area & installation area
- Why two different areas ?
  - To allow centralized installation of the Geant4 kernel libraries and related sources in a multi-user environment
  - To decouple user-developed code and applications from the kernel
  - To allow an easy integration of the Geant4 software in an existing software framework

Two ways to proceed:

- Manually installing by env variables (deprecated)
- Using CMake (recommended and officially supported)

Geant4 installation with cmake

- Unpack the geant4 source package geant4.10.00.tar.gz to a location of your choice:

  ex.: /path/to/geant4.10.00 → source directory

- **Create** a directory in which to configure and run the build and store the build products (not inside the source dir!)

  ex.: /path/to/geant4.10.10-build → build directory

```
$ cd /path/to
$ mkdir geant4.10.0-build
$ ls
geant4.10.00  geant4.10.0-build
```

Geant4 installation with cmake

- To configure, change into the build directory and run **CMake**:

```
$ cd /path/to/geant4.10.0-build
$ cmake -DCMAKE_INSTALL_PREFIX=/path/to/geant4.10.0-install /path/to/geant4.10.00
```

  - **CMAKE_INSTALL_PREFIX** option is used to set the install directory
  - The second argument to CMake is the path to the source directory
- So: three conceptually-different directories
  - Source
  - Build (you can also delete it, when everything done)
  - **Install** (contains libs and includes)

Geant4 installation with cmake

- CMake configures the build and **generates Unix Makefiles** to perform the actual build:

```
$ cmake -DCMAKE_INSTALL_PREFIX=/path/to/geant4.10.0-install /path/to/geant4.10.00
-- The C compiler identification is GNU
-- The CXX compiler identification is GNU
-- Check for working C compiler: /usr/bin/gcc
-- Check for working C compiler: /usr/bin/gcc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- setting default compiler flags for CXX
-- 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
-- Found EXPAT: /usr/lib64/libexpat.so
-- Looking for sys/types.h
-- Looking for sys/types.h - found
-- Looking for stdint.h
-- Looking for stdint.h - fo
-- Looking for stddef.h
-- Looking for stddef.h - fo
```

If you see that, you are successful !!!

```
-- Configuring done
-- Generating done
-- Build files have been written to: /path/to/geant4.10.0-build
```

Geant4 installation with cmake

- After the configuration has run, CMake has generated Unix Makefiles for building Geant4. To run the build, simply execute make in the build directory:

```
$ make -jN
```

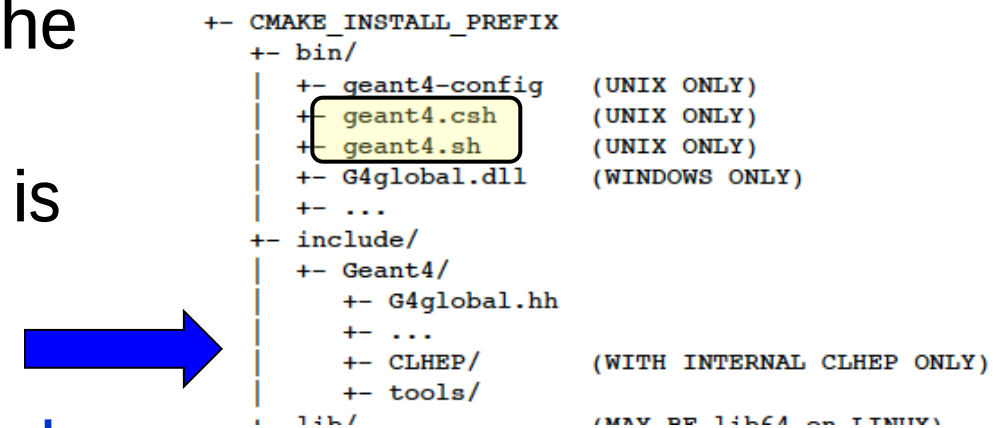
- When build has completed, you can install Geant4 to the directory you specified earlier in **CMAKE_INSTALL_PREFIX** by running:

```
$ make install
```

Geant4 installation with cmake

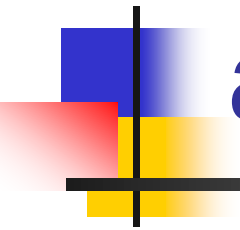
- If you want to *activate* additional options, simply re-run CMake in the build directory, passing it the extra options
- The install of Geant4 is contained under the directory chosen
- To get everything ready for Geant4 (env variable, path to libraries, etc.)

```
$ cd /path/to/geant4.10.0-build
$ cmake -DGEANT4_INSTALL_DATA=ON .
```




```
$ . geant4.sh
```

Building Geant4 user-applications with cmake



Building an application with cmake

- To build an application that uses the Geant4 toolkit, it is necessary to include Geant4 headers in the application sources and link the application to the Geant4 libraries:
  - using CMake → Geant4Config.cmake → writing a **`CMakeLists.txt`** script
- For instance:

  `examples/basic/B1`:

```
+- B1/
   +- CMakeLists.txt
   +- exampleB1.cc
   +- include/
   |  ... headers.hh ...
   +- src/
      ... sources.cc ...
```

- **`CMakeLists.txt`** file has to be located in the root directory of the application.

Building an application with cmake

```
# (1)
cmake_minimum_required(VERSION 2.6 FATAL_ERROR)
project(B1)

# (2)
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()

# (3)
include(${Geant4_USE_FILE})
include_directories(${PROJECT_SOURCE_DIR}/include)

# (4)
file(GLOB sources ${PROJECT_SOURCE_DIR}/src/*.cc)
file(GLOB headers ${PROJECT_SOURCE_DIR}/include/*.hh)

# (5)
add_executable(exampleB1 exampleB1.cc ${sources} ${headers})
target_link_libraries(exampleB1 ${Geant4_LIBRARIES})

# (6)
set(EXAMPLEB1_SCRIPTS
  exampleB1.in
  exampleB1.out
  init.mac
  init_vis.mac
  run1.mac
  run2.mac
  vis.mac
  )

foreach(_script ${EXAMPLEB1_SCRIPTS})
  configure_file(
    ${PROJECT_SOURCE_DIR}/${_script}
        ${PROJECT_BINARY_DIR}/${_script}
        COPYONLY
        )
    endforeach()

    # (7)
    install(TARGETS exampleB1 DESTINATION bin)
```

- The text file CMakeLists.txt is the CMake script containing commands which describe how to build the exampleB1 application
- Example of structure:
  1. Cmake minimum version and set the project name
  2. Find and configure G4
  3. Configure the project to use G4 and B1 headers
  4. List the sources
  5. Define and link the executable
  6. Copy any runtime script to the build directory
  7. Install the executable

Building an application with cmake

- First step: **create** a build directory for the specific application
- Change to this build directory and **run CMake** to generate the Makefiles needed to build the B1 application. Pass CMake two arguments:

```
$ cd $HOME
$ mkdir B1-build
```

```
$ cd $HOME/B1-build
$ cmake -DGeant4_DIR
```

```
$ cmake -DGeant4_DIR=/home/you/geant4-install/lib64/Geant4-10.0.0 $HOME/B1
-- The C compiler identification is GNU
-- The CXX compiler identification is GNU
-- Check for working C compiler: /usr/bin/gcc
-- Check for working C compiler: /usr/bin/gcc -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - 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
-- Configuring done
-- Generating done
-- Build files have been written to: /home/you/B1-build
```

- CMake will now configure the generate Make

Building an application with cmake

- The following files have been generated:

```
$ ls
CMakeCache.txt       exampleB1.in   init_vis.mac  run2.mac
CMakeFiles           exampleB1.out  Makefile      vis.mac
cmake_install.cmake  init.mac       run1.mac
```

- Once the Makefile is available we can do:

```
$ make -jN
```

```
$ make
Scanning dependencies of target exampleB1
[ 16%] Building CXX object CMakeFiles/exampleB1.dir/exampleB1.cc.o
[ 33%] Building CXX object CMakeFiles/exampleB1.dir/src/B1PrimaryGeneratorA
ction.cc.o
[ 50%] Building CXX object CMakeFiles/exampleB1.dir/src/B1EventAction.cc.o
[ 66%] Building CXX object CMakeFiles/exampleB1.dir/src/B1RunAction.cc.o
[ 83%] Building CXX object CMakeFiles/exampleB1.dir/src/B1DetectorConstruct
ion.cc.o
[100%] Building CXX object CMakeFiles/exampleB1.dir/src/B1SteppingAction.cc
.o
Linking CXX executable exampleB1
[100%] Built target exampleB1
```

Building an application with cmake

- List again the content of the build directory, you see the **executable**

```
$ ls
CMakeCache.txt        exampleB1       init.mac       run1.mac
CMakeFiles            exampleB1.in    init_vis.mac   run2.mac
cmake_install.cmake   exampleB1.out   Makefile       vis.mac
```

- Run the application, simply with **`./exampleB1`**

```
$ ./exampleB1

************************************************************
Geant4 version Name: geant4-10-00-ref-00 [MT]   (6-December-2013)
<< in Multi-threaded mode >>
Copyright : Geant4 Collaboration
Reference : NIM A 506 (2003), 250-303
WWW : http://cern.ch/geant4
************************************************************
```

For more info/help

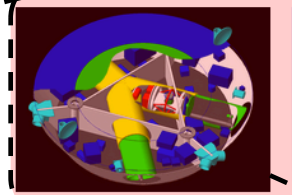
- For further details have a **guide:**

Geant 4

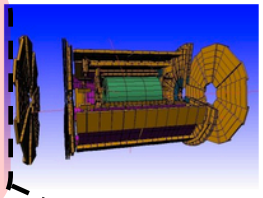
Geant4 is a toolkit for the simulation of the passage of particles through matter. Its areas
energy, nuclear and accelerator physics, as well as studies in medical and space science
papers for Geant4 are published in *Nuclear Instruments and Methods in Physics Researc*
IEEE Transactions on Nuclear Science 53 No. 1 (2006) 270-278.

Applications | **User Support** | **Publications** | **Coll**

A sampling of applications, technology transfer and other uses of Geant4



Getting started, guides and information for users and developers



Validation of Geant4, results from experiments and publications

Who
memb
inform

Events

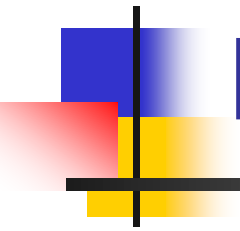
- Geant4 Beginners Course, Belfast (Northern Ireland), **20-24 January 2014**.
- SLAC Geant4 Tutorial Course, Jen-Hsun Huang Engineering Center, Stanford (US), **3-6 Marc**
- 19th Geant4 Collaboration Meeting, Okinawa (Japan), **29 September - 4 October 2014**.
- **Past events**

Geant 4

Home > User Support

User Support

1. Getting started
2. Training courses and materials
3. Source code
   a. Download page
   b. LXR code browser -or- draft doxygen documentation
4. Frequently Asked Questions (FAQ)
5. Bug reports and fixes
6. User requirements tracker
7. User Forum
8. Documentation
   a. Introduction to Geant4
   b. Installation Guide
   c. Application Developers Guide
   d. Toolkit Developers Guide
   e. Physics Reference Manual
9. Examples
10. Physics lists
    a. Electromagnetic
    b. Hadronic
11. User Aids
    a. Tips for improving CPU performance



Hands-on session (task0)

http://geant4.lngs.infn.it/TRISEP2014/introduction/index.html