Geant4: A Simulation toolkit

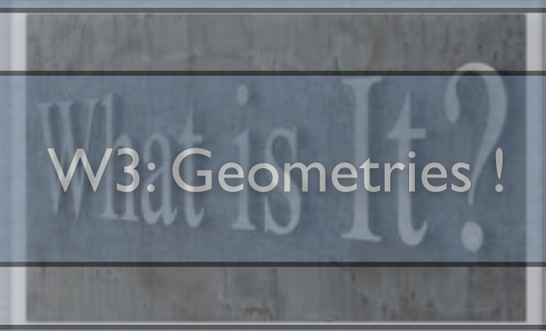
O. Stézowski and M. Pinto

With many thanks to the Geant4 community !!!!

The roadmap of the week

W1: installation / running a G4 application

W2: Primary generator, GPS, physics list



W3: Geometries !

W4: Sensitive detectors / user's actions

w1: 3:00, Monday
w2: 3:00, Tuesday
w3: 4:30, Wednesday
w4: 3:00, Thursday

NOW, HOW does it really work ?

W1: installation / running a G4 application

Geant4 installation, the cmake tool

The user's application

the bricks to build an application

compilation using cmake, requirements

playing with the simulation

W1: installation / running a G4 application

Geant4 installation, the cmake tool

User's application

the bricks to build an application

compilation using cmake, requirements

playing with the simulation

G4 installation, the cmake tool

- **Linux systems**
  - Scientific Linux CERN SLC5, with gcc 4.1.2 or 4.3.X, 32/64bit
  - Scientific Linux CERN 6 with gcc 4.6.X, 64bit

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

- **MacOSX systems**
  - Mac OS X 10.7 (Lion) and 10.8 (Mountain Lion) with gcc 4.2.1 (Apple), 64bit

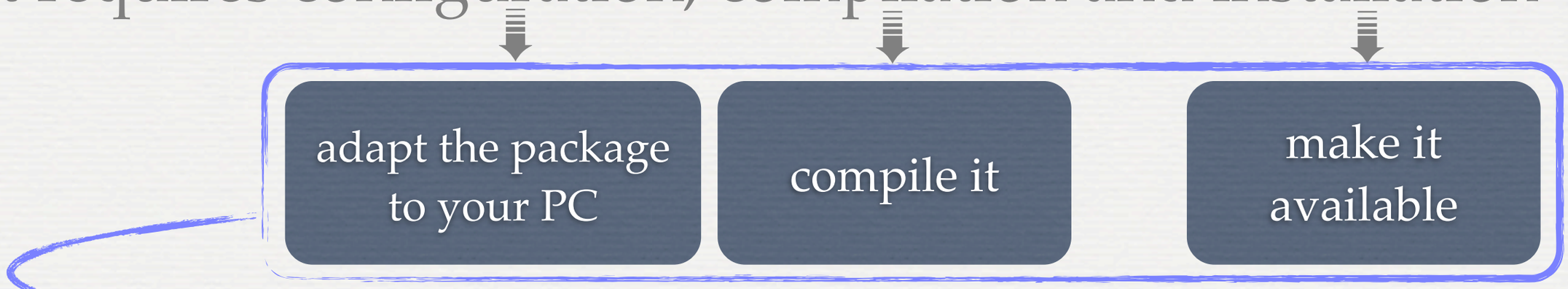
Geant4 has also been successfully compiled on Mac OS X 10.6.8 (Snow Leopard) with gcc 4.2.1 (Apple), (not officially supported)

- **Windows systems**
  - Windows 7 with Visual Studio 10 (VS2010).

G4 installation, the cmake tool

Installation from sources*:

- no need to be super-user, root, admin ☛ autonomy
- help to customize the installation to match needs
- it requires configuration, compilation and installation



CMake do the job

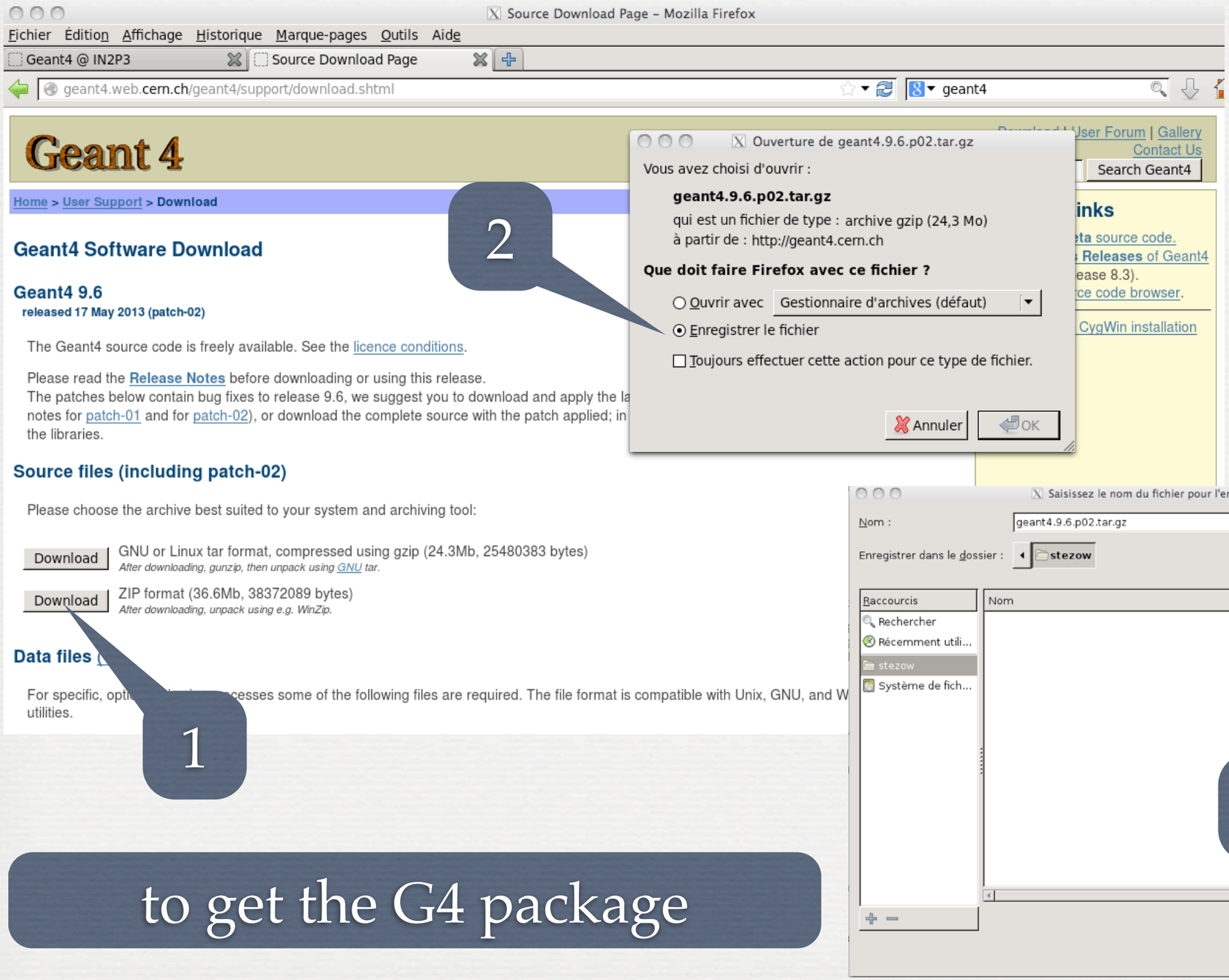
http://www.cmake.org

[G4 recommended and officially supported]

You have to have it installed on you machine !

* *pre-compiled package are also available on the G4 site* — *not covered here*

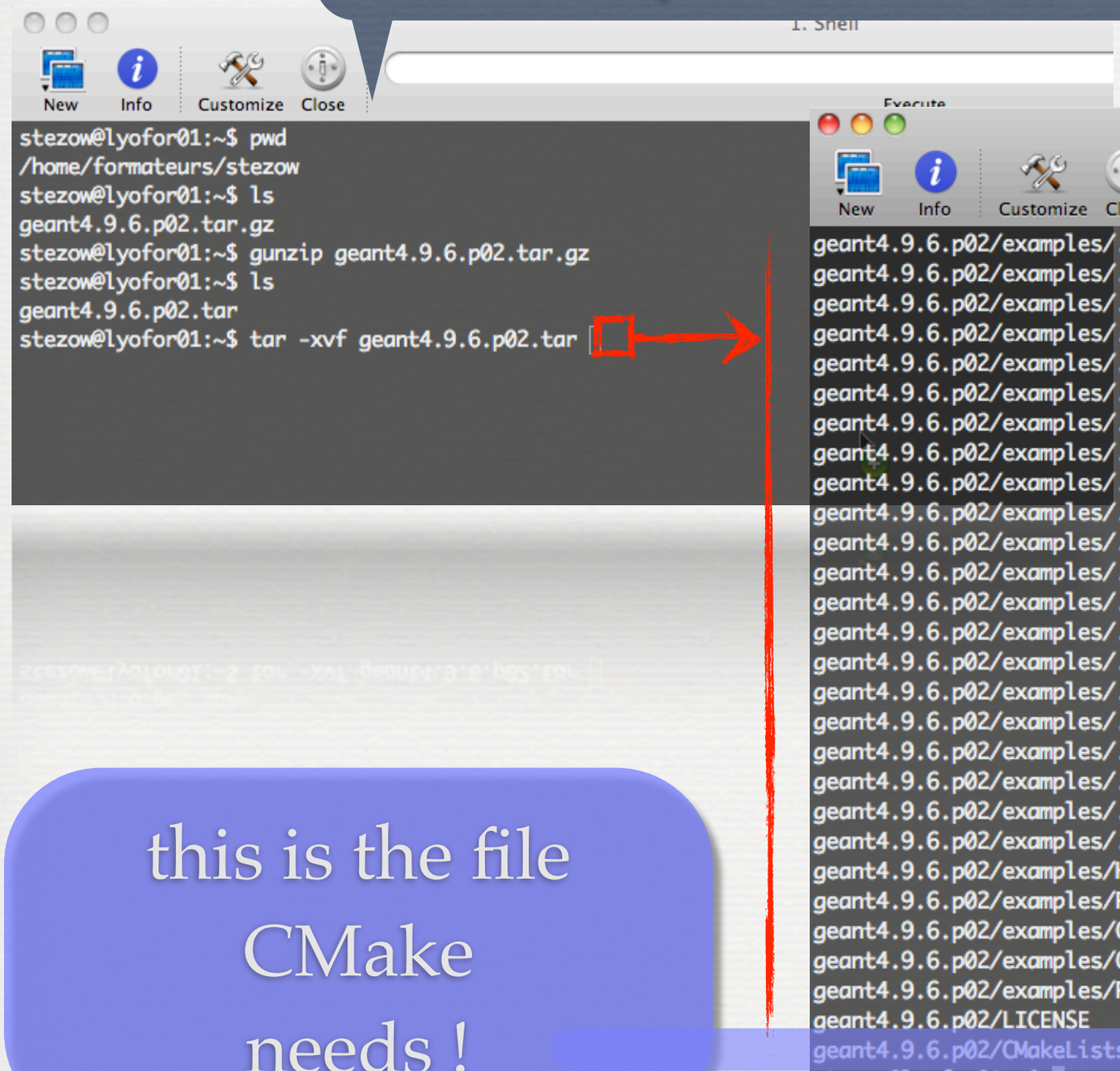
G4 installation, the cmake tool

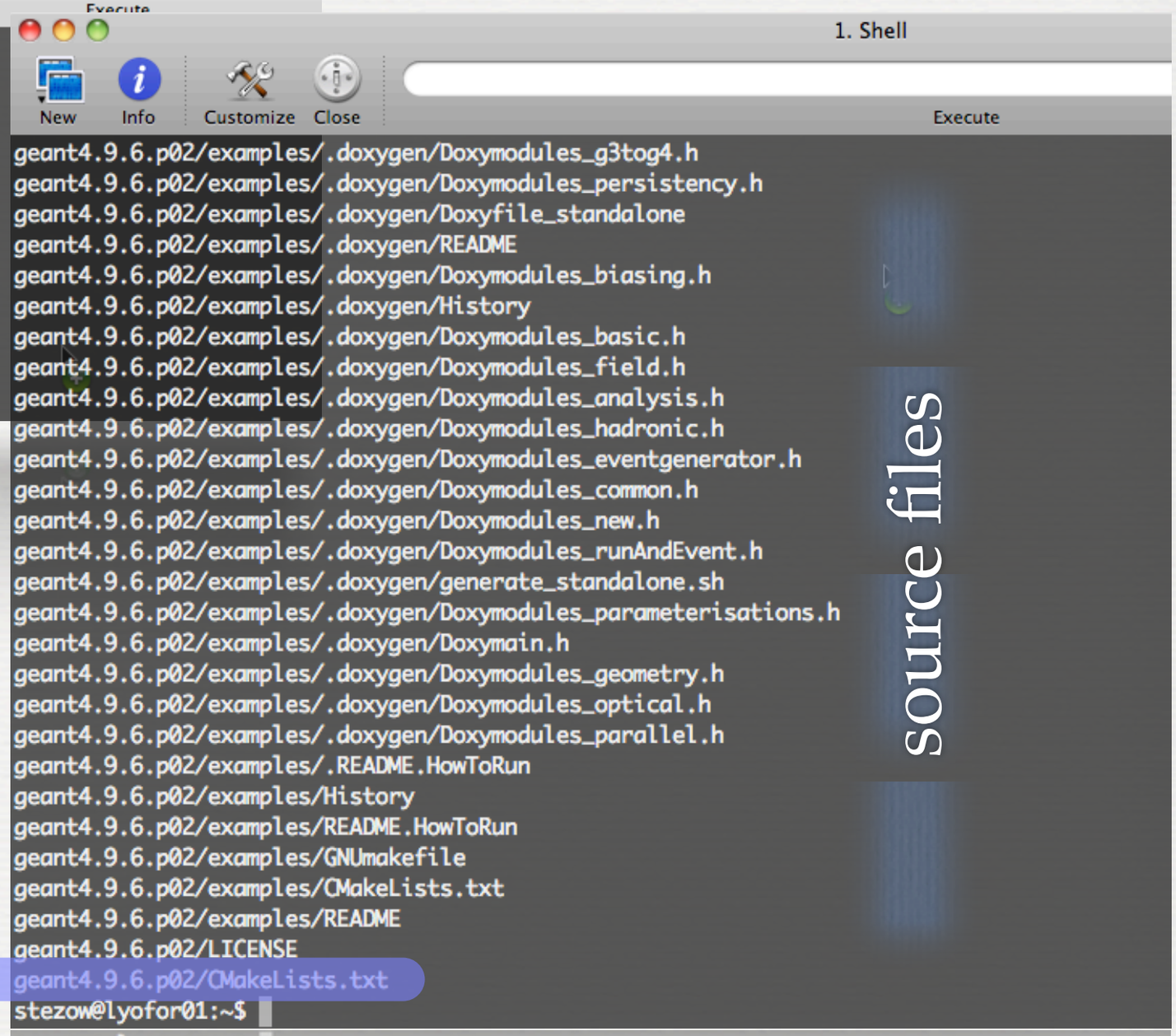


to get the G4 package

G4 installation, the cmake tool

unzip, untar ... of course in /home/





source files

this is the file CMake needs !

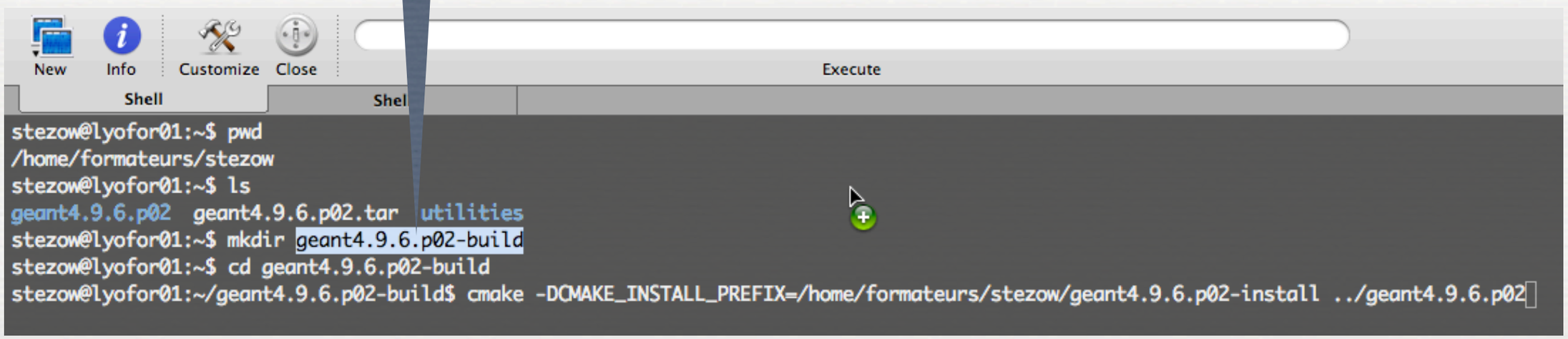
G4 installation, the cmake tool

And now, full G4 installation in three steps

1. Configuration

Out of source building

- keep sources clean
- allows several installations



G4 installation, the cmake tool

And now, full G4 installation in three steps

1. Configuration

Out of source building

- keep sources clean
- allows several installations

```
Shell   Shell
*WARNING*
  Geant4 has been pre-configured to look for datasets
  in the directory:

  /home/formateurs/stezow/geant4.9.6.p02-install/share/Geant4-9.6.2/data

  but the following datasets are NOT present on disk at
  that location:

  G4NDL (4.2)
  G4EMLOW (6.32)
  PhotonEvaporation (2.3)
  RadioactiveDecay (3.6)
  G4NEUTRONXS (1.2)
  G4PII (1.3)
  RealSurface (1.0)
  G4SAIDDATA (1.1)

  If you want to have these datasets installed automatically
  simply re-run cmake and set the GEANT4_INSTALL_DATA
  variable to ON. This will configure the build to download
  and install these datasets for you. For example, on the
  command line, do:

  cmake -DGEANT4_INSTALL_DATA=ON <otherargs>

  The variable can also be toggled in ccmake or cmake-gui.
  If you're running on a Windows system, this is the best
  solution as CMake will unpack the datasets for you
```

New Info Customize Close Execute

Shell Shell

```
stezow@lyofor01:~$ pwd
/home/formateurs/stezow
stezow@lyofor01:~$ ls
geant4.9.6.p02  geant4.9.6.p02.tar  utilities
stezow@lyofor01:~$ mkdir geant4.9.6.p02-build
stezow@lyofor01:~$ cd geant4.9.6.p02-build
stezow@lyofor01:~/geant4.9.6.p02-build$ cmake -DCMAKE_INSTALL_PREFIX=/home/formateurs/stezow/geant4.9.6.p02-install ../geant4.9.6.p02
```

G4 installation, the cmake tool

And now, full G4 installation in three steps

1. Configuration

Out of source building

- keep sources clean
- allows several installations

```
Shell | Shell
*WARNING*
  Geant4 has been pre-configured to look for datasets
  in the directory:

  /home/formateurs/stezow/geant4.9.6.p02-install/share/Geant4-9.6.2/data

  but the following datasets are NOT present on disk at
  that location:

  G4NDL (4.2)
  G4EMLOW (6.32)
  PhotonEvaporation (2.3)
  RadioactiveDecay (3.6)
  G4NEUTRONXS (1.2)
  G4PII (1.3)
  RealSurface (1.0)
  G4SAIDDATA (1.1)

  If you want to have these datasets installed automatically
  simply re-run cmake and set the GEANT4_INSTALL_DATA
  variable to ON. This will configure the build to download
  and install these datasets for you. For example, on the
  command line, do:

  cmake -DGEANT4_INSTALL_DATA=ON <otherargs>

  The variable can also be toggled in ccmake or cmake-gui.
  If you're running on a Windows system, this is the best
  solution as CMake will unpack the datasets for you
```

G4 is made of modules !

New | Info | Customize | Close | Execute

Shell | Shell

```
stezow@lyofor01:~$ pwd
/home/formateurs/stezow
stezow@lyofor01:~$ ls
geant4.9.6.p02  geant4.9.6.p02.tar  utilities
stezow@lyofor01:~$ mkdir geant4.9.6.p02-build
stezow@lyofor01:~$ cd geant4.9.6.p02-build
stezow@lyofor01:~/geant4.9.6.p02-build$ cmake -DCMAKE_INSTALL_PREFIX=/home/formateurs/stezow/geant4.9.6.p02-install ../geant4.9.6.p02
```

G4 installation, the cmake tool

And now, full G4 installation in three steps

1. Configuration

Out of source building

- keep sources clean
- allows several installations

```
Shell | Shell
*WARNING*
  Geant4 has been pre-configured to look for datasets
  in the directory:

  /home/formateurs/stezow/geant4.9.6.p02-install/share/Geant4-9.6.2/data

  but the following datasets are NOT present on disk at
  that location:

  G4NDL (4.2)
  G4EMLOW (6.32)
  PhotonEvaporation (2.3)
  RadioactiveDecay (3.6)
  G4NEUTRONXS (1.2)
  G4PII (1.3)
  RealSurface (1.0)
  G4SAIDDATA (1.1)

  If you want to have these datasets installed automatically
  simply re-run cmake and set the GEANT4_INSTALL_DATA
  variable to ON. This will configure the build to download
  and install these datasets for you. For example, on the
  command line, do:

  cmake -DGEANT4_INSTALL_DATA=ON .

  The variable can also be toggled in ccmake or cmake-gui.
  If you're running on a Windows system, this is the best
  solution as CMake will unpack the datasets for you
```

G4 is made of modules !

Data needed @ running time

New | Info | Customize | Close | Execute

Shell | Shel

```
stezow@lyofor01:~$ pwd
/home/formateurs/stezow
stezow@lyofor01:~$ ls
geant4.9.6.p02  geant4.9.6.p02.tar  utilities
stezow@lyofor01:~$ mkdir geant4.9.6.p02-build
stezow@lyofor01:~$ cd geant4.9.6.p02-build
stezow@lyofor01:~/geant4.9.6.p02-build$ cmake -DCMAKE_INSTALL_PREFIX=/home/formateurs/stezow/geant4.9.6.p02-install ../geant4.9.6.p02
```

G4 installation, the cmake tool

-DOPTION=VALUE -DGEANT4_INSTALL_DATA=ON -DGEANT4_USE_QT=ON ...

Additional modules:
options [external packages]

Core components:
all needed and built

```
stezow@lyofor01:~/geant4.9.6.p02-build$
stezow@lyofor01:~/geant4.9.6.p02-build$ cmake -DCMAKE_INSTALL_PREFIX=/home/formateurs/stezow/geant4.9.6.p02-install -DGEANT4_INSTALL_DATA=ON ../geant4.9.6.p02
-- Configuring download of missing dataset G4NDL (4.2)
-- Configuring download of missing dataset G4EMLOW (6.32)
-- Configuring download of missing dataset PhotonEvaporation (2.3)
-- Configuring download of missing dataset RadioactiveDecay (3.6)
-- Configuring download of missing dataset G4NEUTRONXS (1.2)
-- Configuring download of missing dataset G4PII (1.3)
-- Configuring download of missing dataset RealSurface (1.0)
-- Configuring download of missing dataset G4SAIDDATA (1.1)
-- The following Geant4 features are enabled:
GEANT4_BUILD_CXXSTD: Compiling against C++ Standard 'c++98'
GEANT4_USE_SYSTEM_EXPAT: Use system EXPAT library
```

2. Compilation

```
stezow@lyofor01:~/geant4.9.6.p02-build$ make -j2
Scanning dependencies of target G4EMLOW
Scanning dependencies of target G4NDL
[  0%] Creating directories for 'G4EMLOW'
[  0%] Creating directories for 'G4NDL'
[  0%] Performing download step (download, verify and extract) for 'G4EMLOW'
[  0%] -- downloading...
     src='http://geant4.cern.ch/support/source/G4EMLOW.6.32.tar.gz'
     dst='/home/formateurs/stezow/geant4.9.6.p02-build/Externals/G4EMLOW-6.32/src/G4EMLOW.6.32.tar.gz'
     timeout='1500 seconds'
```

3. Installation

NOT mandatory, the building directory could be enough

```
[100%] Building CXX object source/physics_lists/CMakeFiles/G4physicslists.dir/list
[100%] Building CXX object source/physics_lists/CMakeFiles/G4physicslists.dir/list
[100%] Building CXX object source/physics_lists/CMakeFiles/G4physicslists.dir/list
[100%] Building CXX object source/physics_lists/CMakeFiles/G4physicslists.dir/list
[100%] Building CXX object source/physics_lists/CMakeFiles/G4physicslists.dir/list
Linking CXX shared library ../../outputs/library/Linux-g++/libG4physicslists.so
[100%] Built target G4physicslists
stezow@lyofor01:~/geant4.9.6.p02-build$ make install
```

Note: Modules are also shared libraries

For this workshop, two versions installed

```
cmake -DGEANT4_INSTALL_DATA=ON -DGEANT4_USE_OPENGL_X11=ON -DGEANT4_USE_RAYTRACER_X11=ON ../geant4.9.6.p02
-- The C compiler identification is GNU
-- The CXX compiler identification is GNU
.
-- Found X11: /usr/lib/i386-linux-gnu/libX11.so
-- Found OpenGL: /usr/lib/i386-linux-gnu/libGL.so
-- Configuring download of missing dataset G4NDL (4.2)
-- Configuring download of missing dataset G4EMLOW (6.32)
.
-- The following Geant4 features are enabled:
GEANT4_BUILD_CXXSTD: Compiling against C++ Standard 'c++98'
GEANT4_USE_SYSTEM_EXPAT: Use system EXPAT library
GEANT4_USE_RAYTRACER_X11: Build RayTracer driver with X11 support
GEANT4_USE_OPENGL_X11: Build Geant4 OpenGL driver with X11 support

-- Configuring done
-- Generating done
-- Build files have been written to: /group/formateurs/stezowski/geant4.9.6.p02-build
```

```
cmake -DGEANT4_INSTALL_DATA=ON -DGEANT4_USE_OPENGL_X11=ON -DGEANT4_USE_RAYTRACER_X11=ON
      -DGEANT4_USE_GDML=ON -DGEANT4_USE_QT=ON ../geant4.9.6.p02

-- The following Geant4 features are enabled:
GEANT4_BUILD_CXXSTD: Compiling against C++ Standard 'c++98'
GEANT4_USE_SYSTEM_EXPAT: Use system EXPAT library
GEANT4_USE_GDML: Build Geant4 with GDML support
GEANT4_USE_QT: Build Geant4 with Qt support
GEANT4_USE_RAYTRACER_X11: Build RayTracer driver with X11 support
GEANT4_USE_OPENGL_X11: Build Geant4 OpenGL driver with X11 support

-- Configuring done
-- Generating done
-- Build files have been written to: /group/formateurs/stezowski/geant4.9.6.p02-build-full
```

GDML ☛ W3

QT ☛ W1

G4 installation, the cmake tool

TODO List

Install Geant4 the same way in your home directory !

- first, the ‘core’ version
- then the more complete one*

*see here for a full description of the available options
http://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/InstallationGuide/html/ch02s03.html

W1: installation / running a G4 application

Geant4 installation, the cmake tool

User's application

the bricks to build an application

compilation using cmake, requirements

playing with the simulation

The user's application

C++ (Object Oriented) into the game - *ex: classes that transform objects*

include file

VBase.hh

```
#ifndef _VBase_hh
#define _VBase_hh

class VBase
{
protected:
    float _X;
    float _Y;
    AnObject _O;

public:
    VBase(float x, float y, AnObject o);
    virtual ~VBase();

    //! Reset VBase
    virtual void Reset();

    //! pure virtual method, HAS to be implemented
    virtual void Transform(float xnew, float ynew) = 0;
};

#endif
```

data members

constructor, to init

methods to change data values

inherits from

MyBase.hh

```
#ifndef _MyBase_hh
#define _MyBase_hh

#include "VBase.hh"

class MyBase : public VBase
{
protected:
    AnotherObject _OO;

public:
    MyBase(float x, float y, AnObject o, AnotherObject oo);
    virtual ~ MyBase();

    // Reset MyBase
    virtual void Reset();

    // Really do the job of transforming _O into _OO
    // moving it at a different position
    virtual void Transform(float xnew, float ynew);
};

#endif
```

source file

VBase.cc

```
#include "VBase.hh"

VBase::VBase(float x, float y, AnObject o)
{
    _X = x;
    _Y = y;
    _O = o;
}

void VBase::Reset()
{
    _X = _Y = 0;
    _O = 0;
}
```

MyBase.cc

```
#include MyBase.hh"

MyBase::MyBase(float x, float y, AnObject o, AnotherObject oo) :
       VBase(x,y,o)
{
    _OO = oo;
}

void MyBase::Reset()
{
    VBase::Reset(); _OO = 0;
}

void MyBase::Transform(float xnew, float ynew)
{
    AnObject o_tmp = _O;

    _O  = _OO;
    _OO = o_tmp;

    _X = xnew; _Y = ynew;

    // ... something like _O.Show() and _OO.Hide()
}
```

The user's application

C++ (Object Oriented) into the game - *ex: classes that transform objects*

include file

VBase.hh

```
#ifndef _VBase_hh
#define _VBase_hh

class VBase
{
protected:
    float _X;
    float _Y;
    AnObject _O;

public:
    VBase(float x, float y, AnObject o);
    virtual ~VBase();

    //! Reset VBase
    virtual void Reset();

    //! pure virtual method, HAS to be implemented
    virtual void Transform(float xnew, float ynew) = 0;
};

#endif
```

data members

constructor, to init

methods to change data values

inherits from

MyBase.hh

```
#ifndef _MyBase_hh
#define _MyBase_hh

#include "VBase.hh"

class MyBase : public VBase
{
protected:
    AnotherObject _OO;

public:
    MyBase(float x, float y, AnObject o, AnotherObject oo);
    virtual ~ MyBase();

    // Reset MyBase
    virtual void Reset();

    // Really do the job of transforming _O into _OO
    // moving it at a different position
    virtual void Transform(float xnew, float ynew);
};

#endif
```

source file

VBase.cc

```
#include "VBase.hh"

VBase::VBase(float x, float y, AnObject o)
{
    _X = x;
    _Y = y;
    _O = o;
}

void VBase::Reset()
{
    _X = _Y = 0;
    _O = 0;
}
```

MyBase.cc

```
#include MyBase.hh"

MyBase::MyBase(float x, float y, AnObject o, AnotherObject oo) :
       VBase(x,y,o)
{
    _OO = oo;
}

void MyBase::Reset()
{
    VBase::Reset(); _OO = 0;
}

void MyBase::Transform(float xnew, float ynew)
{
    AnObject o_tmp = _O;

    _O  = _OO;
    _OO = o_tmp;

    _X = xnew; _Y = ynew;

    // ... something like _O.Show() and _OO.Hide()
}
```

- Knowing VBase interface enough to play with **all kind of objects** inheriting from VBase
- At running time the 'right' methods are called

The user's application

Building an application requires to put together 3 mandatory bricks*

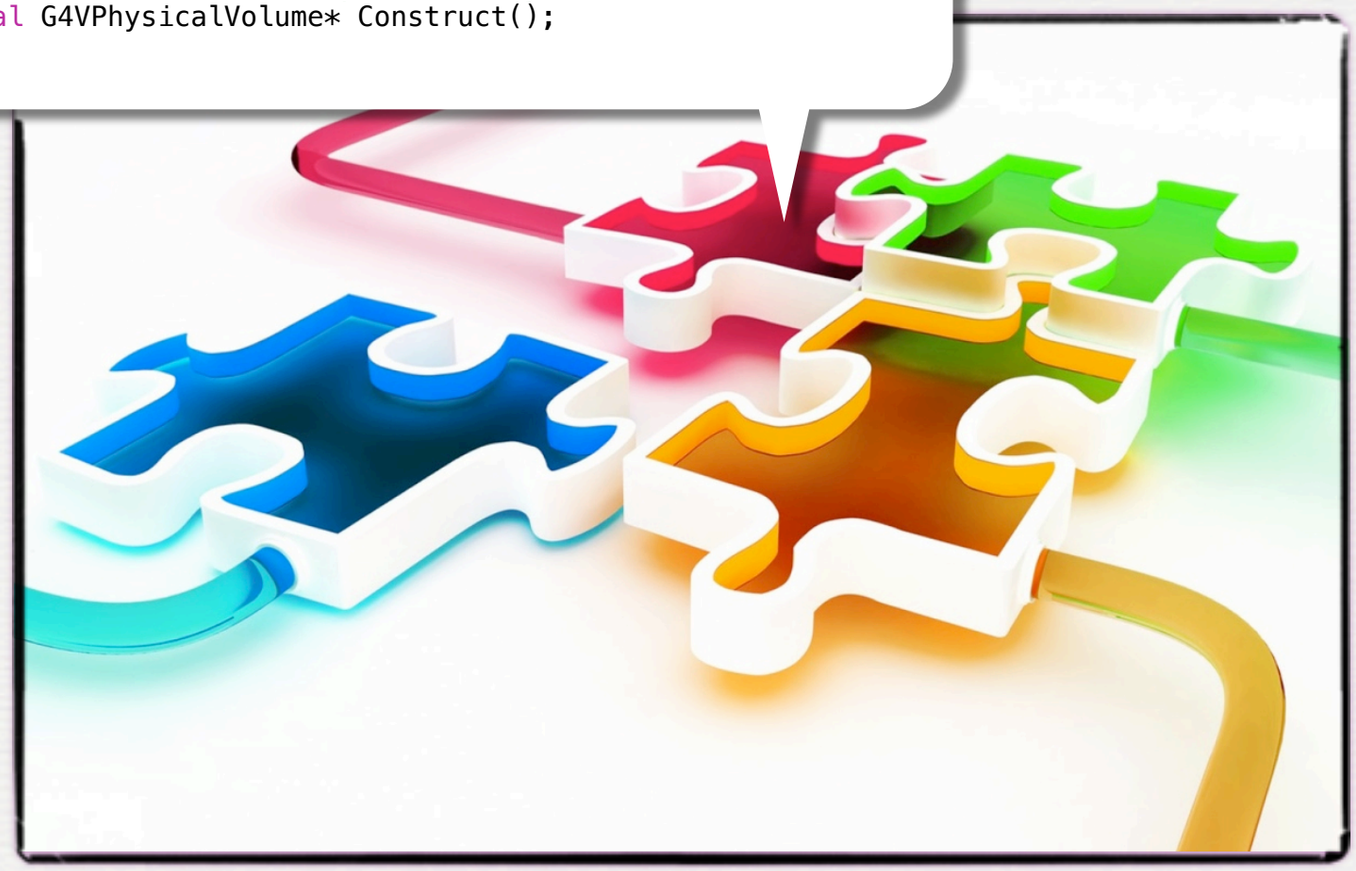
the detector construction - the description of the physics - the primary generator

The user's application

Building an application requires to put together 3 mandatory bricks*

the detector construction - the description of the physics - the primary generator

```
class ARedSphereConstruction : public G4VUserDetectorConstruction
{
// the virtual method to be implemented by the user
    virtual G4VPhysicalVolume* Construct();
};
```

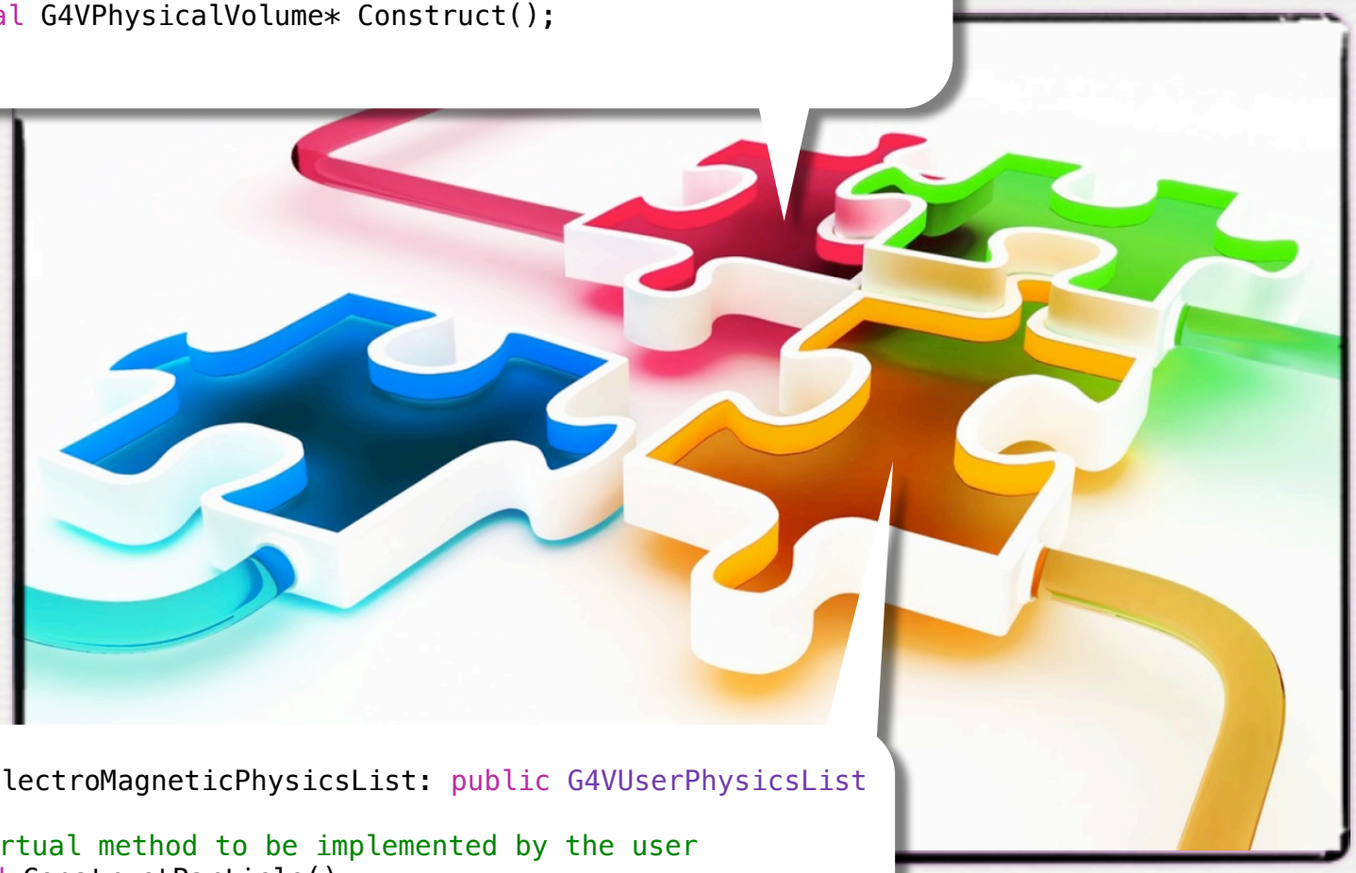


The user's application

Building an application requires to put together 3 mandatory bricks*

the detector construction - the description of the physics - the primary generator

```
class ARedSphereConstruction : public G4VUserDetectorConstruction
{
// the virtual method to be implemented by the user
    virtual G4VPhysicalVolume* Construct();
};
```



```
class AnElectroMagneticPhysicsList: public G4VUserPhysicsList
{
// the virtual method to be implemented by the user
     void ConstructParticle();
// the virtual method to be implemented by the user
     void ConstructProcess();
// the virtual method to be implemented by the user
     void SetCuts();
};
```

The user's application

Building an application requires to put together 3 mandatory bricks*

the detector construction - the description of the physics - the primary generator

```
class ARedSphereConstruction : public G4VUserDetectorConstruction
{
// the virtual method to be implemented by the user
    virtual G4VPhysicalVolume* Construct();
};
```

```
class AGammaGun : public G4VUserPrimaryGeneratorAction
{
// the virtual method to be implemented by the user
    virtual void GeneratePrimaries(G4Event* anEvent);
};
```

```
class AnElectroMagneticPhysicsList: public G4VUserPhysicsList
{
// the virtual method to be implemented by the user
     void ConstructParticle();
// the virtual method to be implemented by the user
     void ConstructProcess();
// the virtual method to be implemented by the user
     void SetCuts();
};
```

The user's application

Building an application requires to put together 3 mandatory bricks*

the detector construction - the description of the physics - the primary generator

```
class ARedSphereConstruction : public G4VUserDetectorConstruction
{
// the virtual method to be implemented by the user
    virtual G4VPhysicalVolume* Construct();
};
```

```
class AGammaGun : public G4VUserPrimaryGeneratorAction
{
// the virtual method to be implemented by the user
    virtual void GeneratePrimaries(G4Event* anEvent);
};
```

```
class AnElectroMagneticPhysicsList: public G4VUserPhysicsList
{
// the virtual method to be implemented by the user
     void ConstructParticle();
// the virtual method to be implemented by the user
     void ConstructProcess();
// the virtual method to be implemented by the user
     void SetCuts();
};
```

```
// The User's main program to control / run simulations
int main(int argc, char** argv)
{
// Construct the run manager, necessary for G4 kernel to control everything
    G4RunManager *theRunManager = new G4RunManager();

// Then add mandatory initialization G4 classes provided by the USER
    // detector construction
    // physics  list
    // initialisation of the generator

    theRunManager->SetUserInitialization( new ARedSphereConstuction() );
    theRunManager->SetUserInitialization( new AnElectroMagneticPhysicsList() );
    theRunManager->SetUserAction( new AGammaGun() );
      .
      .
    return 0;
}
```

The user's application

Building an application requires to put together 3 mandatory bricks*

the detector construction - the description of the physics - the primary generator

```
class ARedSphereConstruction : public G4VUserDetectorConstruction
{
// the virtual method to be implemented by the user
    virtual G4VPhysicalVolume* Construct();
};
```

```
class AGammaGun : public G4VUserPrimaryGeneratorAction
{
// the virtual method to be implemented by the user
    virtual void GeneratePrimaries(G4Event* anEvent);
};
```

+
many other hooks
but
not mandatory

```
class AnElectroMagneticPhysicsList: public G4VUserPhysicsList
{
// the virtual method to be implemented by the user
     void ConstructParticle();
// the virtual method to be implemented by the user
     void ConstructProcess();
// the virtual method to be implemented by the user
     void SetCuts();
};
```

```
// The User's main program to control / run simulations
int main(int argc, char** argv)
{
// Construct the run manager, necessary for G4 kernel to control everything
    G4RunManager *theRunManager = new G4RunManager();

// Then add mandatory initialization G4 classes provided by the USER
    // detector construction
    // physics  list
    // initialisation of the generator

    theRunManager->SetUserInitialization( new ARedSphereConstuction() );
    theRunManager->SetUserInitialization( new AnElectroMagneticPhysicsList() );
    theRunManager->SetUserAction( new AGammaGun() );
      .
      .
    return 0;
}
```

W1: installation / running a G4 application

Geant4 installation, the cmake tool

User's application

the bricks to build an application

compilation using cmake, requirements

playing with the simulation

The user's application

your CMakelists.txt

```
# Setup the project
project(W1_LIO)

#----------------------------------------------------------------------------
# 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
include(${Geant4_USE_FILE})
include_directories(${PROJECT_SOURCE_DIR}/csrc)

#----------------------------------------------------------------------------
# Locate sources and headers for this project.

set(PROJECT_SRC

    )

set(PROJECT_HEADER

    )

#----------------------------------------------------------------------------
# Add the executable, and link it to the Geant4 libraries
add_executable(LIO_W1 LIO_W1.cc ${PROJECT_SRC} ${PROJECT_HEADER})
#
target_link_libraries(LIO_W1 ${Geant4_LIBRARIES} ${EXTRA_LIB})

#----------------------------------------------------------------------------
# Install the executable to 'bin' directory under CMAKE_INSTALL_PREFIX
#
install(TARGETS LIO_W1 DESTINATION bin)
```

your application's name

to be sure what is installed is enough to build your application

where is the G4 version used

this is the place where you tell cmake what files are part of your application

it fully defines the main/exe

place to install your application (if required)

The user's application

```
# Setup the project
project(W1_LIO)

#----------------------------------------------------------------------------
# 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
include(${Geant4_USE_FILE})
include_directories(${PROJECT_SOURCE_DIR}/csrc)

#----------------------------------------------------------------------------
# Locate sources and headers for this project.

set(PROJECT_SRC

    )

set(PROJECT_HEADER

    )

#----------------------------------------------------------------------------
# Add the executable, and link it to the Geant4 libraries
add_executable(LIO_W1 LIO_W1.cc ${PROJECT_SRC} ${PROJECT_HEADER})
#
target_link_libraries(LIO_W1 ${Geant4_LIBRARIES} ${EXTRA_LIB})

#----------------------------------------------------------------------------
# Install the executable to 'bin' directory under CMAKE_INSTALL_PREFIX
#
install(TARGETS LIO_W1 DESTINATION bin)
```

+ add the source files

+ add the header files

your application's name

to be sure what is installed is enough to build your application

where is the G4 version used

this is the place where you tell cmake what files are part of your application

it fully defines the main/exe

place to install your application (if required)

The user's application

To build your application

mkdir build9.6-p02
cd build9.6-p02
cmake -DGeant4_DIR=*/path/to/the/G4buildingDirYouWant* ../
make -j2
cd ..

To run it

./build9.6-p02/*the_exe_you_have_defined_its_name*

The user's application

TODO List

Copy the first example in your directory

cp -r /group/formateurs/xxxx/LIO_W1 LIO_W1_MyWork

Have a look in the directory, identify the various files

Build the application [*in a sub-directory called build9.6-p02*]:

- using the 'core' G4 installed
- you may need to modify some files !
- run a GeantinoGun in a Red Sphere [*./build9.6-p02/LIO_W1*]
- run a GammaGun in a Red Sphere
- run a ProtonGun in a Blue Cube

30 minutes

W1: installation / running a G4 application

Geant4 installation, the cmake tool

User's application

the bricks to build an application

compilation using cmake, requirements

playing with the simulation

The user's application

TODO List

Play with the simulation using the command line:

- run the application and type help
- have a look at the commands, try for instance:

  /units/list

  /process/list and */process/dump* -

  /run/setCut 0.1 mm and */run/setCutForAGivenParticle e- 10 um*

  /material/g4/printElement and */material/g4/printMaterial*

  /particle/list and */gun/List*

  ...
- check geometry with */vis/drawTree*
- all commands could be in a file - see visGL.mac
- run it with */control/execute visGL.mac*
- to start a run with 100 particles */run/beamOn 100*

The user's application

Advanced features to check geometry, see and interact

TODO List

HepRep:

- Requires to dump the geometry and traces in a .heprep[.gz] file
- No need of specific G4 modules
- Requires a java program HepRApp.jar to read back the file:
  - */group/tmp_softs/jre1.6.0_33/bin/java -Xms512M -Xmx1024M -jar HepRApp.jar*
  - *There is a version of HepRApp.jar* in */group/formateurs/xxxx/utilities*
- Run the visHepRep.mac macro in the application
- Browse the file using HepRApp

The user's application

Advanced features to check geometry, see and interact

TODO List

Qt:

- It allows to see geometries, traces and run simulations
- It requires to build G4 with Qt. In you application, create a new directory (*mkdir build9.6-p02-full)* build and run !
- try also with G4 standard examples:

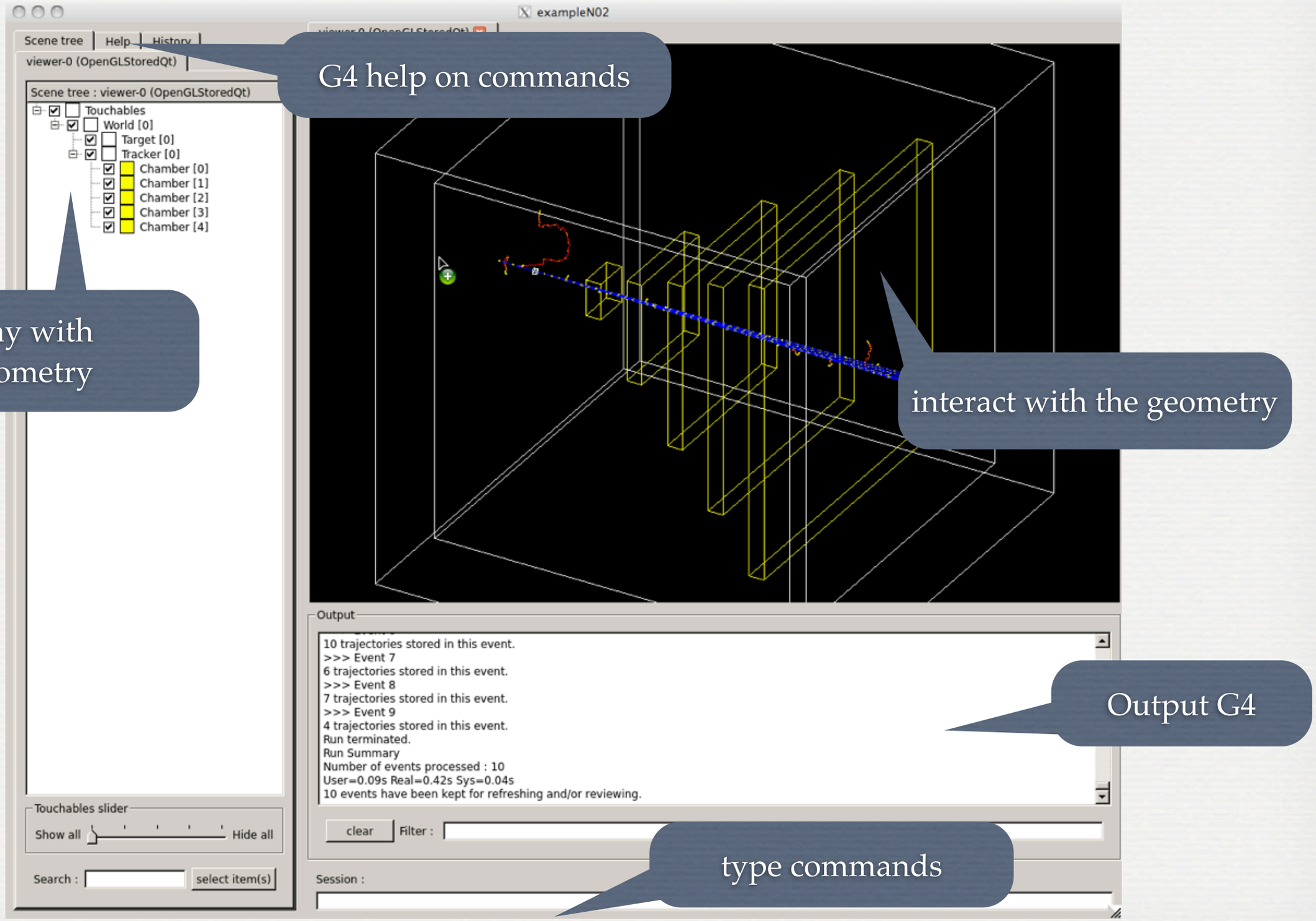
```
ExampleN05
Simplified BaBar calorimeter with EM shower parametrisation
run and execute in Qt vis.mac

ExampleB3
Schematic Positron Emitted Tomography system + Radioactive source
run + /run/beamOn 10

extended/optical/Lxe
examples of generic optical processes simulation setups
/run/initialize then /run/beamOn 10

....
```

The user's application



Conclusions of W1

We have seen

- How to install G4 using CMake
- How to customize / build / run the user's application
- The commands called C++ methods using Messengers
  - ➥ see W2 to know how to do it