# Outline 2017/05/12

- GEANT4 introduction (v 4.10.01.p01)
- Why the Monte-Carlo approach
- Basic GEANT4 concepts
- Defining materials and volumes
- Defining detectors
- Exercises

# Monte-Carlo in physics

Physics Detectors:

– a particle interacts with the detector material, depositing energy (e.g. ionization, Landau distributed)

– energy deposit → analog electrical signal

*noise, non-linearities of amplifiers, other statistical effects*

– analog signal → digital signal

*noise, limited dynamic range*

# Advantages of Monte-Carlo methods

We can break down a complex problem into many simple problems that are treated separately for a random initial state:

- break down a long particle track into many small steps
- look at each detector component and each particle separately
- treat physics processes consecutively instead of having to worry about everything at once
- the output of our simulated detector is qualitatively the same as of the real detector

# Why are simulations needed?

- to determine the experimental setup
- to optimize the detector/shielding locations to obtain decent
- to study the expected background
- to get physics results (risky!)

# BEWARE!

- A simulation will only produce what you put it

– A simulation will help you understand your detector but it's a tool without scientific value by itself.

- Simulated detectors are based to a large part on “Educated Guesses”

– Use set of parameters to be able to adjust your detector to match measurements.

- Actually simulating all physical processes accurately is not possible

– You have to make a lot of approximations: this **WILL** distort your results.

# GEANT4

**GEANT4 is a *toolkit* for the simulation of the passage of particles through matter.**

- Often used in physics (nuclear, high-energy, accelerator, and medical) as well as space science
- Toolkit: Using GEANT4, users need to build a simulation program
- Developed from GEANT3 (fortran), GEANT4 is written in C++
- Developed and maintained by GEANT4 Collaboration

http://www.cern.ch/geant4

http://www-geant4.kek.jp/Reference/

# GEANT?

What does GEANT stand for?

Two possible answers:

- **GE**neration **AN**d **T**racking (End of 1970's, at the beginning of GEANT1)

- **GE**ometry **AN**d **T**racking (Nowadays)

# Basic GEANT4 concepts

- GEANT4 is a framework with an interface for integration in C++ programs
- It does not offer a command-line interface or a GUI to define the geometry
- Every simulation is an executable C++ program
- The user supplies the geometry, the list of physics processes, the initial states
- GEANT4 supplies the information about particles passing through user-defined inspection points called detectors

# Memory management (like ROOT)

GEANT4 does **not** have any consistent memory management

For every class that you instantiate you will ultimately have to check whether GEANT4 cleans it up or whether you have to do it – and no, it's not in the documentation...

E.g.:

handled by GEANT4

- G4VSolids
- G4VPhysicalVolumes
- G4LogicalVolumes
- ...

handled by user

- G4Materials
- G4MaterialPropertiesTables
- G4Elements
- ...

# Conventions and types

GEANT4 class names start with “G4”:

- G4LogicalVolumes
- G4Materials
- ...

GEANT4 virtual classes names start with “G4V”:

- G4VSolids
- G4VPhysicalVolumes
- ....

G4cout and G4cerr (and G4endl) are ostream objects deffned by Geant4:

G4cout << ”Hello Geant4!” << G4endl;

| | | |
|---|---|---|
| int | → | G4int |
| long | → | G4long |
| float | → | G4float |
| double | → | G4double |
| bool | → | G4bool |
| string | → | G4String |

# System of units

Internal unit system used in Geant4 is completely hidden not only from user's code but also from Geant4 source code implementation.

Each given hard-coded number must be *multiplied* by its proper unit:

```
radius = 10.0 * cm;
Ekin   = 1.0 * GeV;
```

To get a number, it must be *divided* by a proper unit:

```
G4cout << eDep / MeV << " [MeV]" << G4endl;
```

Most of commonly used units are provided and user can add his/her own units.

Complete list of units is given in G4SystemOfUnits.hh

(`/usr/local/geant4.10.01.p01-install/include/Geant4/G4SystemOfUnits.hh`)

# GEANT4 as toolkit

The framework part in Geant4 has been reduced to a minimum, so as to allow the user to implement his/her own program structure.

Bits and pieces of Geant4 can be used independently from the rest.

Libraries have been made as granular as possible, in order to reduce (re-) compilation time and to allow the user to only link against those parts of G4 which are needed.

# What does GEANT4 do for you

A particle is transported through matter for every step until the particle:

- goes out of the world volume
- loses its kinetic energy to zero
- it disappears by an interaction or decay

Users can access the transportation process to get the simulated results (*User Actions*):

- at the beginning and end of transportation
- at the end of each transportation step
- when the particle enters the sensitive volume of the detector

# Run in GEANT4

As an analogy of the real experiment, a run of Geant4 starts with “Beam On”.

Conceptually a run is a collection of events which share the same detector and physics conditions.

Hence, within a run, the user cannot change:

- detector setup
- settings of physics processes

Usually a run consist of one event loop.

Beginning of a run:

geometry is optimized for navigation, cut-off values are defined, and cross-section tables are calculated according to materials appearing in the geometry description.

**G4RunManager** class manages processing a run.

# Event in GEANT4

An event is the basic unit of simulation in Geant4.

At beginning of processing, primary tracks are generated. These primary tracks are pushed into a stack.

A track is popped up from the stack one by one and “tracked”. Resulting secondary tracks are pushed into the stack.

This “tracking” lasts as long as the stack has a track.

When the stack becomes empty, processing of one event is over.

**G4Event** class represents an event. It has following objects at the end of its (successful) processing.

- List of primary vertices and particles (as input)
- Hits and Trajectory collections (as output)

# Track in GEANT4

Track is a snapshot of a particle. It has physical quantities of current instance only. It does not record previous quantities.

Step is a “delta” information to a track. **Track is not a collection of steps**. Instead, a track is being updated by steps.

Track object is deleted when

- it goes out of the world volume,
- it disappears (by e.g. decay, inelastic scattering),
- it goes down to zero kinetic energy and no “AtRest” additional process is required, or
- the user decides to kill it artificially.

No track object persists at the end of event, for the record of tracks, use trajectory class objects.

A track is represented by **G4Track** class.

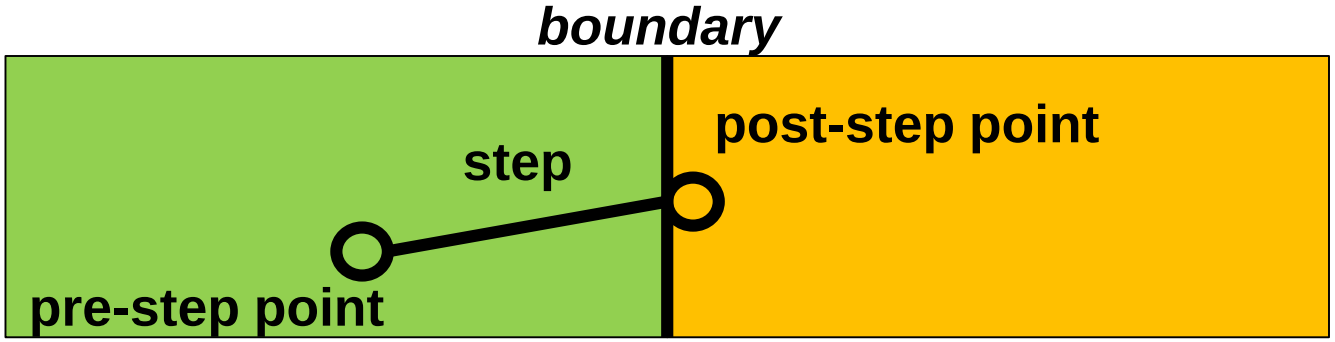
# Step in GEANT4

Step has two points and also “delta” information of a particle (energy loss on the step, time-of-flight spent by the step, etc.).

Each point knows the volume (and material). In case a step is limited by a volume boundary, the end point physically stands on the boundary, and it logically belongs to the next volume.

Because one step knows materials of two volumes, boundary processes such as transition radiation or refraction could be simulated.

A step is represented by **G4Step** class.



# Step in GEANT4

GEANT4 breaks down particle track into small steps and stores particle momentum, positions before and after last step, polarization...

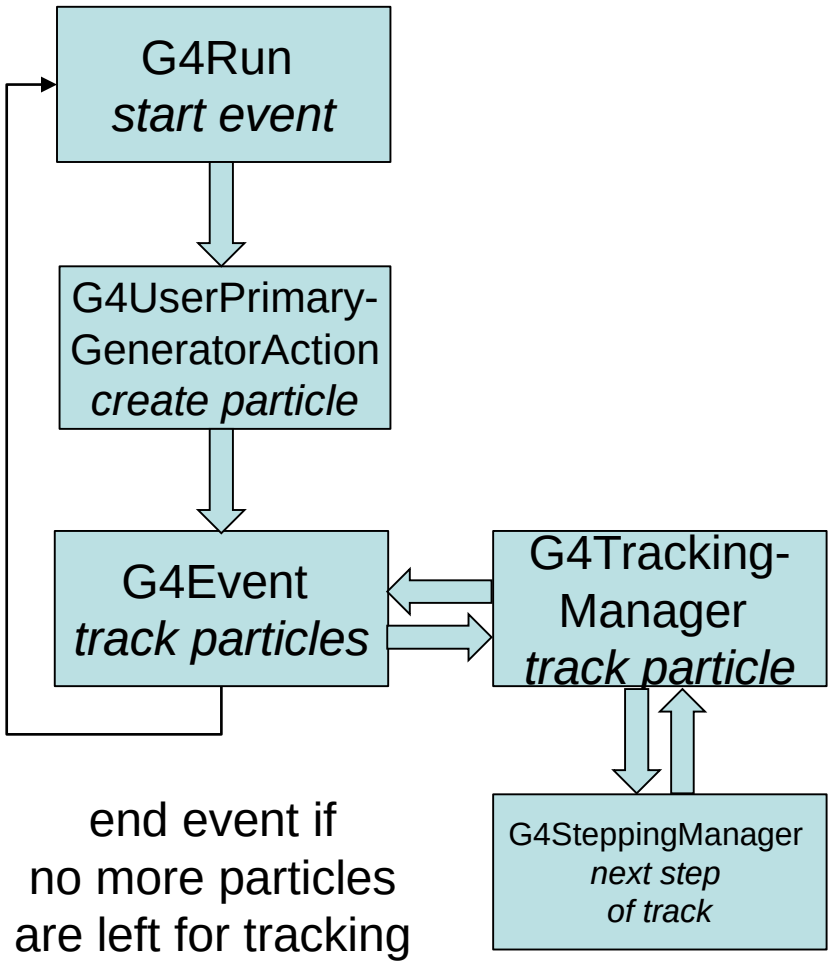
GEANT4 applies processes between steps.

- processes may modify particle momentum, direction, polarization and/or create new particles
- processes may also modify current volume (e.g. store an energy deposit)
- newly created particles are also tracked

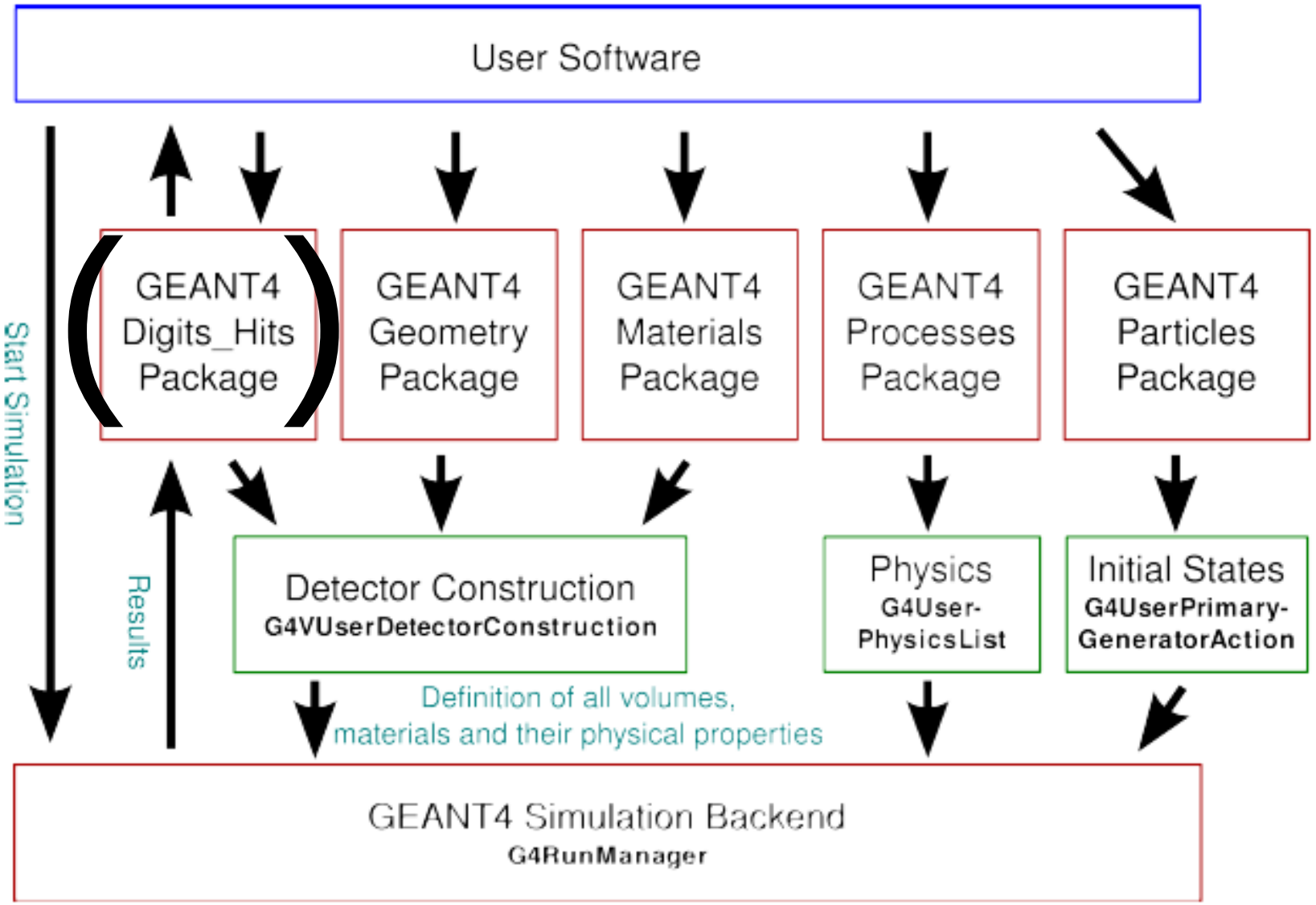
# Process summary

- Each event starts with a single created particle
- Initial particle and all secondaries are tracked
- After each event data for the particle trajectories can be accessed

GEANT4 allows intervening at any given point of the simulation



# Basic GEANT4 workflow



# Needed inputs to GEANT4

The user must:

1) define materials and geometry of target and detectors (and add EM fields whenever needed)
2) define particles and physics interactions
3) decide how a primary event should be generated

# Needed inputs to GEANT4

The user must:

1) define materials and geometry of target and detectors (and add EM fields whenever needed) **G4VUserDetectorConstruction** class
2) define particles and physics interactions
3) decide how a primary event should be generated

# Needed inputs to GEANT4

The user must:

1) define materials and geometry of target and detectors (and add EM fields whenever needed) **G4VUserDetectorConstruction** class

2) define particles and physics interactions **G4VUserPhysicsList** class

3) decide how a primary event should be generated

# Needed inputs to GEANT4

The user must:

1) define materials and geometry of target and detectors (and add EM fields whenever needed) **G4VUserDetectorConstruction** class

2) define particles and physics interactions **G4VUserPhysicsList** class

3) decide how a primary event should be generated **G4VUserPrimaryGeneratoraAction** class

# Initialization

**G4VUserDetectorConstruction**

the detector set-up must be described

(Materials, geometry of the detector, definition of sensitive detectors, ...)

**G4VUserPhysicsList**

Particles and processes to be used in the simulation + cutoff parameters

**G4VUserPrimaryGeneratorAction**

Primary event kinematics

# Physics lists

A physics list is a class which collects all the particles, physics processes, and production thresholds needed to GEANT application.

It tells the run manager how and when to invoke physics.

# Physics lists

Physics is physics, shouldn't GEANT4 provide, as a default, a physics list that everyone can use?

No...

- many different physics models and approximations
- computation speed can be an issue (a user may want a less-detailed but faster approximation)
- no application requires all the physics and particles GEANT4 can offer (e.g., most medical applications do no require multi-GeV physics)

# Geometry and materials

The simulated detector geometry and materials are defined by a class extending G4VUserDetectorConstruction

- Use predefined volumes from GEANT4
- Each volume can have any number of sub-volumes of different materials
- You can create unions & intersections of volumes

# Materials definition

In nature, general materials (compounds, mixtures) are made by elements and elements are made by isotopes. These are the three main classes designed in Geant4:

- The **G4Element** class describes the properties of the atoms: atomic number, number of nucleons, atomic mass, shell energy…
- The **G4Isotope** class permits to describe the isotopic composition of a material
- The **G4Material** class describes the macroscopic properties of the matter: density, state, temperature, pressure, radiation length, mean free path, dE/dx…

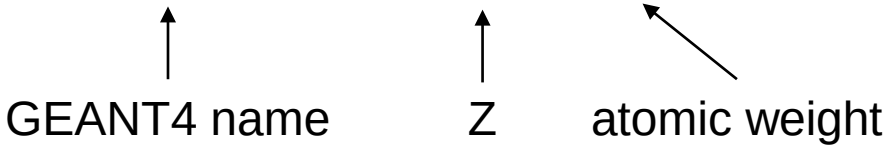
G4Material is the class visible to the rest of the toolkit and it is used by the tracking, the geometry and physics.

# Defining Elements

```
G4Element *H = new G4Element("Hydrogen","H",1,1.008*g/mole);
```

# Defining Elements

```
G4Element *H = new G4Element("Hydrogen","H",1,1.008*g/mole);
```



# Defining Elements

```
G4Element *H = new G4Element("Hydrogen","H",1,1.008*g/mole);
```

GEANT4 name  Z  atomic weight

```
new G4Element("Carbon","C",6,12*g/mole);
```

since materials are stored by GEANT4 in memory we can also omit the pointer...

# Defining Elements

```
G4Element *H = new G4Element("Hydrogen","H",1,1.008*g/mole);
```

GEANT4 name — Z — atomic weight

```
new G4Element("Carbon","C",6,12*g/mole);
```

since materials are stored by GEANT4 in memory we can also omit the pointer...

```
G4Isotope *isoU235 = new G4Isotope("U235",92,235,235.04*g/mole);
G4Isotope *isoU238 = new G4Isotope("U238",92,238,238.05*g/mole);

G4Element *enrichedU = new G4Element("enrichedU","U",2);
```

number of components

# Defining Elements

```
G4Element *H = new G4Element("Hydrogen","H",1,1.008*g/mole);
```

GEANT4 name — Z — atomic weight

```
new G4Element("Carbon","C",6,12*g/mole);
```

since materials are stored by GEANT4 in memory we can also omit the pointer...

```
G4Isotope *isoU235 = new G4Isotope("U235",92,235,235.04*g/mole);
G4Isotope *isoU238 = new G4Isotope("U238",92,238,238.05*g/mole);

G4Element *enrichedU = new G4Element("enrichedU","U",2);
enrichedU->AddIsotope(isoU235,0.8);
enrichedU->AddIsotope(isoU238,0.2);
```

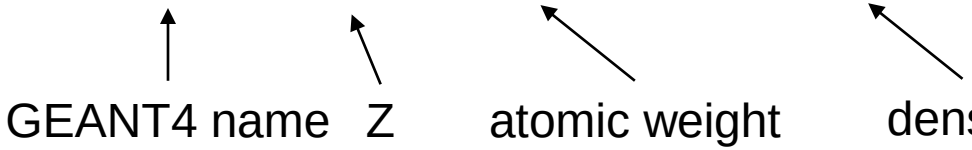
number of components

abundances (percentage)

# Defining Materials

Single element material:

```
G4Material *gasH = new
        G4Material("gasH",1,1.008*g/mole,0.001*g/cm3,kStateGas);
```



# Defining Materials

Single element material:

```
G4Material *gasH = new
        G4Material("gasH",1,1.008*g/mole,0.001*g/cm3,kStateGas);
```

GEANT4 name  Z  atomic weight  density

Molecule (composition by number of atoms):

```
G4Material *H2O = new G4Material("water",1.0*g/cm3,2);
```

density  number of components (must be integer!)

# Defining Materials

Single element material:

```
G4Material *gasH = new
        G4Material("gasH",1,1.008*g/mole,0.001*g/cm3,kStateGas);
```

GEANT4 name  Z  atomic weight  density

Molecule (composition by number of atoms):

```
G4Material *H2O = new G4Material("water",1.0*g/cm3,2);
H2O->AddElement(H, 2);
H2O->AddElement(O, 1);
```

density  number of components (must be integer!)

number of atoms (must be integer!)

H and O defined previously as G4Element ptrs

# Defining Materials

Compound material (composition by fraction of mass):

```
G4Material *air = new G4Material("air",1.3*mg/cm3,2);
```

density

number of components (must be integer!)

# Defining Materials

Compound material (composition by fraction of mass):

```
G4Material *air = new G4Material("air",1.3*mg/cm3,2);
air->AddElement(N, 0.77);
air->AddElement(O, 0.23);
```

density

number of components (must be integer!)

fraction of mass (must be double!)

N and O defined previously as G4Element ptrs

# Defining Materials

Mixture material (composition by fraction of mass):

```
G4Material *aerogel = new G4Material("aerogel",0.2*g/cm3,3);
```

density

number of components (must be integer!)

# Defining Materials

Mixture material (composition by fraction of mass):

```
G4Material *aerogel = new G4Material("aerogel",0.2*g/cm3,3);
aerogel->AddElement(C, 0.001);
```

density

number of components (must be integer!)

fraction of mass (must be double!)

C defined previously as G4Element

# Defining Materials

Mixture material (composition by fraction of mass):

```
G4Material *aerogel = new G4Material("aerogel",0.2*g/cm3,3);
aerogel->AddElement(C, 0.001);
aerogel->AddMaterial(H2O, 0.374);
aerogel->AddMaterial(SiO2, 0.625);
```

density

number of components (must be integer!)

fraction of mass (must be double!)

C defined previously as G4Element, H2O and SiO2 defined previously as G4Material ptrs

# NIST material database


NIST database for materials is imported in GEANT4.

Class for handling the database is G4NistManager

```
GNistManager *nist = G4NistManager::Instance();

G4Element *H = nist->FindOrBuildElement("H");
G4Material *teflon = nist->FindOrBuildMaterial("G4_TEFLON");
```

# NIST elements and isotopes

| Z | | A | m | error | (%) | $A_{eff}$ |
|---|---|---|---|---|---|---|
| 14 | Si | 22 | 22.03453 | (22) | | 28.0855(3) |
| | | 23 | 23.02552 | (21) | | |
| | | 24 | 24.011546 | (21) | | |
| | | 25 | 25.004107 | (11) | | |
| | | 26 | 25.992330 | (3) | | |
| | | 27 | 26.98670476 | (17) | | |
| | | 28 | 27.9769265327 | (20) | 92.2297 (7) | |
| | | 29 | 28.97649472 | (3) | 4.6832 (5) | |
| | | 30 | 29.97377022 | (5) | 3.0872 (5) | |
| | | 31 | 30.97536327 | (7) | | |
| | | 32 | 31.9741481 | (23) | | |
| | | 33 | 32.978001 | (17) | | |
| | | 34 | 33.978576 | (15) | | |
| | | 35 | 34.984580 | (40) | | |
| | | 36 | 35.98669 | (11) | | |
| | | 37 | 36.99300 | (13) | | |
| | | 38 | 37.99598 | (29) | | |
| | | 39 | 39.00230 | (43) | | |
| | | 40 | 40.00580 | (54) | | |
| | | 41 | 41.01270 | (64) | | |
| | | 42 | 42.01610 | (75) | | |

- Natural isotope compositions
- More than 3000 isotope masses are used for definition

# NIST materials

```
=====================================
###  Elementary Materials from the NIST Data Base
=====================================
 Z Name  ChFormula       density(g/cm^3)  I(eV)
=====================================
 1  G4_H    H_2          8.3748e-05       19.2
 2  G4_He                0.000166322      41.8
 3  G4_Li                0.534            40
 4  G4_Be                1.848            63.7
 5  G4_B                 2.37             76
 6  G4_C                 2                81
 7  G4_N    N_2          0.0011652        82
 8  G4_O    O_2          0.00133151       95
 9  G4_F                 0.00158029       115
10  G4_Ne                0.000838505      137
11  G4_Na                0.971            149
12  G4_Mg                1.74             156
13  G4_Al                2.6989           166
14  G4_Si                2.33             173
```

```
=====================================
###  Compound Materials from the NIST Data Base
=====================================
 N Name     ChFormula       density(g/cm^3)  I(eV)
=====================================
13  G4_Adipose_Tissue                 0.92          63.2
                1    0.119477
                6    0.63724
                7    0.00797
                8    0.232333
               11    0.0005
               12    2e-05
               15    0.00016
               16    0.00073
               17    0.00119
               19    0.00032
               20    2e-05
               26    2e-05
               30    2e-05
 4  G4_Air                            0.00120479    85.7
                6    0.000124
                7    0.755268
                8    0.231781
               18    0.012827
 2  G4_CsI                            4.51          553.1
               53    0.47692
               55    0.52308
```

- NIST Elementary Materials
  - H to Cf
- NIST Compounds
- HEP and Nuclear Materials
  - Ex. liquid Ar $PbWO_4$

# GEANT4: geometry

A detector geometry in GEANT4 is made of a number of volumes.

The largest volume is called the ***World volume***. It must contain all other volumes in the detector geometry.

The World volume:

- defines the global coordinates system (origin = center of world volume)
- fully contains all other volumes

# GEANT4: geometry

The other volumes are created and placed inside previous volumes, including the World.

Each volume is created by describing its shape and its physical characteristics and then placing it inside a containing volume.

The coordinate system used to specify where the daughter volume is placed is the one of the mother.

# Geometry: three steps

To implement a volume in GEANT4 one has to go through three steps:

1) a **Solid** is used to describe a volume's shape. A solid is a geometrical object that has a shape and specific values for each of that shape's dimensions

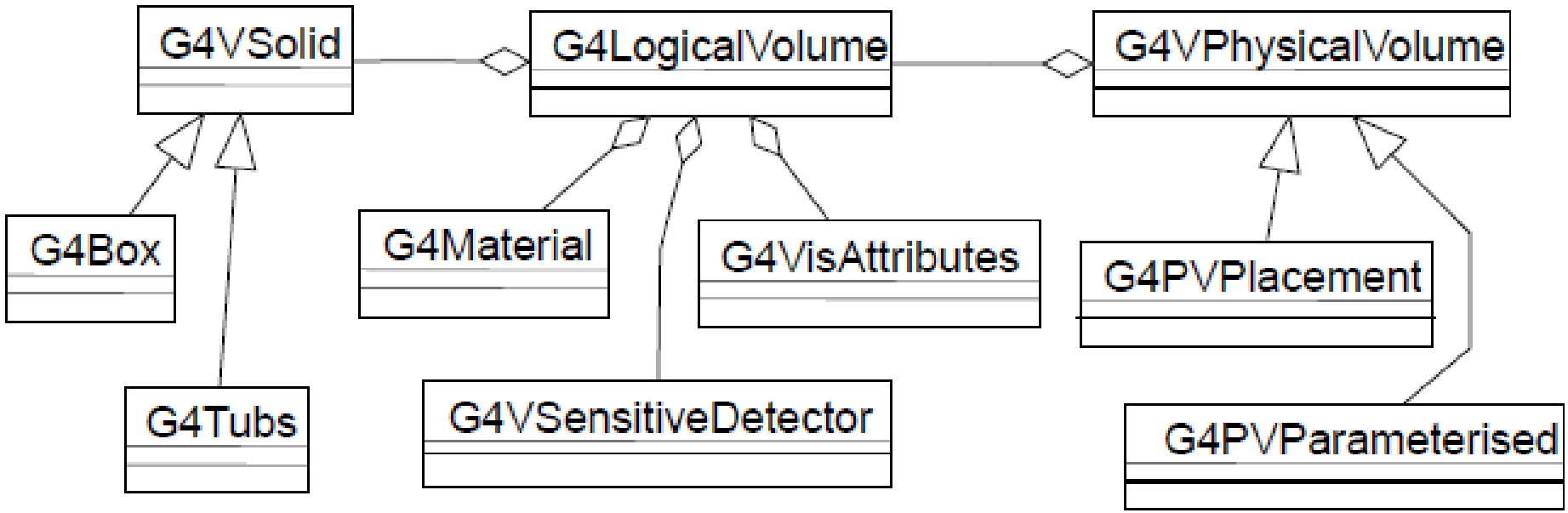
# Geometry: three steps

2) a **Logical Volume** is use for describing a volume's full properties. It starts from its geometrical properties (the solid) and adds physical characteristics, like the material, the sensitivity, the magnetic field, the color…

# Geometry: three steps

3) What remains to describe is the position of the volume. For doing that, one creates a **Physical volumes**, which places a copy of the logical volume inside a larger, containing volume.

# Geometry: three steps




G4VSolid : shape, size

G4LogicalVolume : material, fields, sensitivity

G4VPhysicalVolume : position, rotation

# Geometry hierarchy

A volume is placed in its mother volume

- Position and rotation of the daughter volume is described with respect to the local coordinate system of the mother volume
- The origin of the mother's local coordinate system is at the center of the mother volume
- Daughter volumes cannot protrude from the mother volume
- Daughter volumes cannot overlap
- One or more volumes can be placed in a mother volume

# Geometry, example

```
G4Box *solidWorld = new G4Box("World",1.*m,2.*m,3.*m);
```

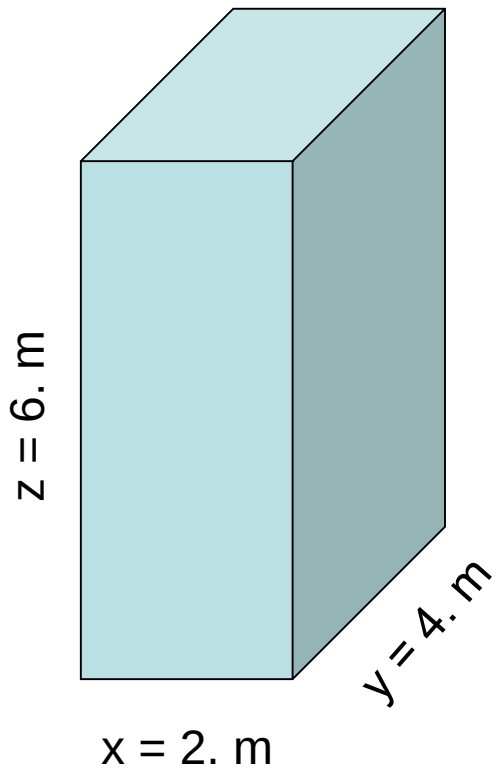

http://www-geant4.kek.jp/Reference/10.03/classG4Box.html

# Geometry, example

```
G4Box *solidWorld = new G4Box("World",1.*m,2.*m,3.*m);
```

name

HALF size of x, y, and z dimensions



# Geometry, example

```
G4Box *solidWorld = new G4Box("World",1.*m,2.*m,3.*m);
G4LogicalVolume *logicWorld =
                  new G4LogicalVolume(solidWorld,air,"World");
```

http://www-geant4.kek.jp/Reference/10.03/classG4LogicalVolume.html

# Geometry, example

```
G4Box *solidWorld = new G4Box("World",1.*m,2.*m,3.*m);
G4LogicalVolume *logicWorld =
                  new G4LogicalVolume(solidWorld,air,"World");
```

name

G4Material ptr

http://www-geant4.kek.jp/Reference/10.03/classG4LogicalVolume.html

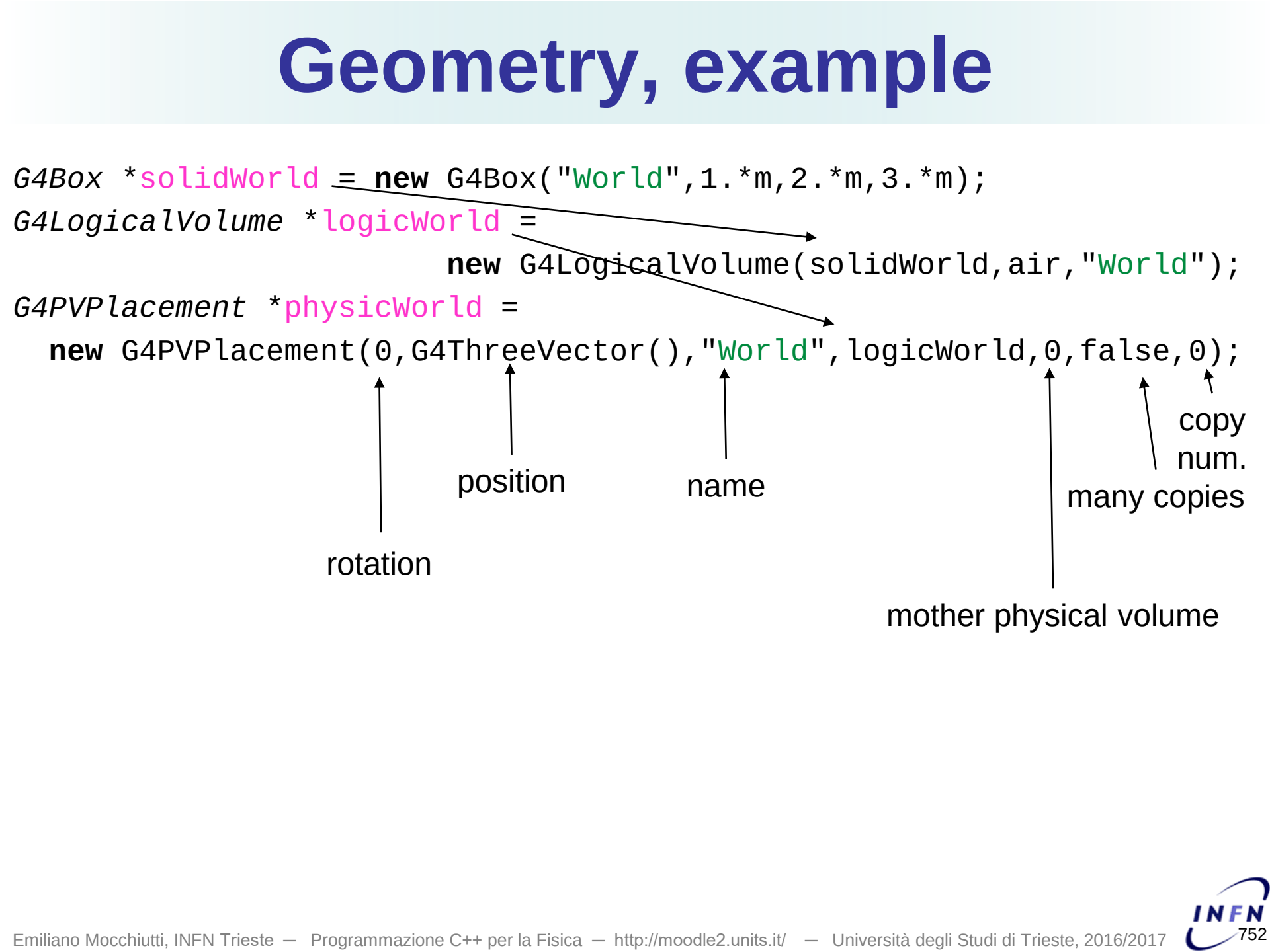
# Geometry, example

```
G4Box *solidWorld = new G4Box("World",1.*m,2.*m,3.*m);
G4LogicalVolume *logicWorld =
                          new G4LogicalVolume(solidWorld,air,"World");
G4PVPlacement *physicWorld =
  new G4PVPlacement(0,G4ThreeVector(),"World",logicWorld,0,false,0);
```

http://www-geant4.kek.jp/Reference/10.03/classG4PVPlacement.html

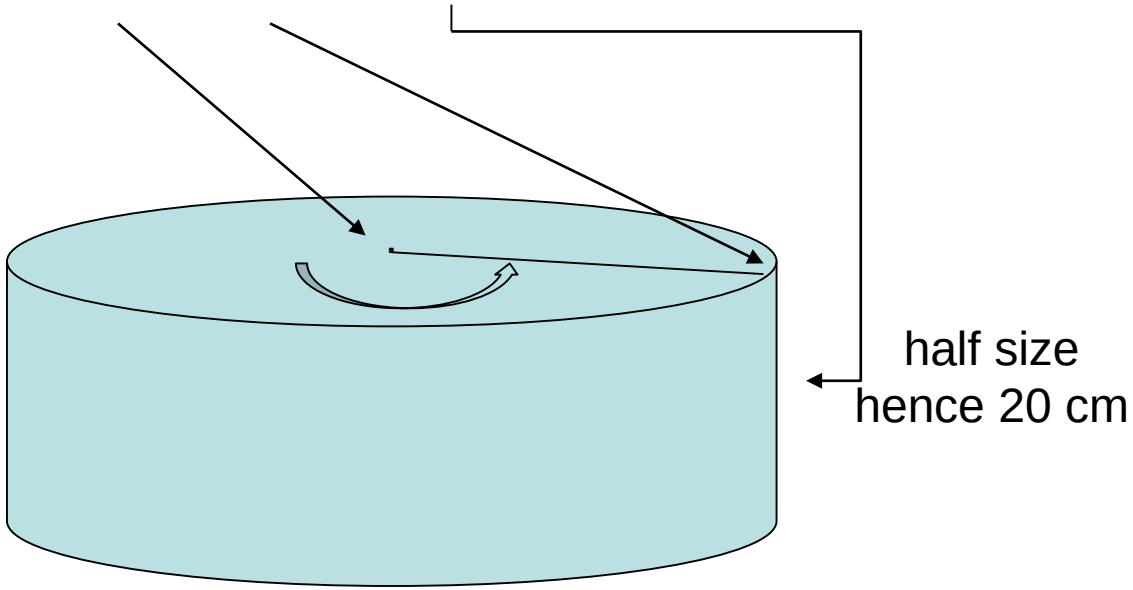
# Geometry, example

```
G4Box *solidWorld = new G4Box("World",1.*m,2.*m,3.*m);
G4LogicalVolume *logicWorld =
                          new G4LogicalVolume(solidWorld,air,"World");
G4PVPlacement *physicWorld =
  new G4PVPlacement(0,G4ThreeVector(),"World",logicWorld,0,false,0);
```



# Geometry, example

```
G4Box *solidWorld = new G4Box("World",1.*m,2.*m,3.*m);
G4LogicalVolume *logicWorld =
                          new G4LogicalVolume(solidWorld,air,"World");
G4PVPlacement *physicWorld =
  new G4PVPlacement(0,G4ThreeVector(),"World",logicWorld,0,false,0);
G4Tubs *solidCyl =
       new G4Tubs("mycyl",0.*cm,20.*cm,10.*cm,0.*deg,360*deg);
```



# Geometry, example

```
G4Box *solidWorld = new G4Box("World",1.*m,2.*m,3.*m);
G4LogicalVolume *logicWorld =
                          new G4LogicalVolume(solidWorld,air,"World");
G4PVPlacement *physicWorld =
  new G4PVPlacement(0,G4ThreeVector(),"World",logicWorld,0,false,0);
G4Tubs *solidCyl =
       new G4Tubs("mycyl",0.*cm,20.*cm,10.*cm,0.*deg,360*deg);
G4LogicalVolume *logicCyl =
                         new G4LogicalVolume(solidCyl,teflon,"Cyl");
```

# Geometry, example

```
G4Box *solidWorld = new G4Box("World",1.*m,2.*m,3.*m);
G4LogicalVolume *logicWorld =
                          new G4LogicalVolume(solidWorld,air,"World");
G4PVPlacement *physicWorld =
  new G4PVPlacement(0,G4ThreeVector(),"World",logicWorld,0,false,0);
G4Tubs *solidCyl =
       new G4Tubs("mycyl",0.*cm,20.*cm,10.*cm,0.*deg,360*deg);
G4LogicalVolume *logicCyl =
                         new G4LogicalVolume(solidCyl,teflon,"Cyl");
G4PVPlacement *physicCyl =
           new G4PVPlacement(0,G4ThreeVector(-1.*cm,-1.*cm,0.),
                                 "Cyl",logicCyl,physicWorld,false,0);
```

# GEANT4 example code

Download from moodle the file G4example-v0.tar.gz and unpack it:

```
|Emi@w13 g4>tar zxvf G4example-v0.tar.gz
G4example-v0/
G4example-v0/include/
G4example-v0/include/SteppingAction.h
G4example-v0/include/EventAction.h
G4example-v0/include/RunAction.h
G4example-v0/include/Run.h
G4example-v0/include/ActionInitialization.h
G4example-v0/include/DetectorConstruction.h
G4example-v0/include/PrimaryGeneratorAction.h
G4example-v0/run1.mac
G4example-v0/src/
G4example-v0/src/SteppingAction.cpp
G4example-v0/src/EventAction.cpp
G4example-v0/src/PrimaryGeneratorAction.cpp
G4example-v0/src/ActionInitialization.cpp
G4example-v0/src/DetectorConstruction.cpp
G4example-v0/src/RunAction.cpp
G4example-v0/src/Run.cpp
G4example-v0/G4example.in
G4example-v0/vis.mac
G4example-v0/init_vis.mac
G4example-v0/CMakeLists.txt
G4example-v0/run2.mac
G4example-v0/G4example.cpp
G4example-v0/GNUmakefile
|Emi@w13 g4>
```

# GEANT4 example: main

```
// taken from exampleB1.cc Emiliano Mocchiutti 11/05/2017
//
/// \file G4example.cpp
/// \brief Main program of the  example

// our code headers
#include "DetectorConstruction.h"
#include "ActionInitialization.h"

// GEANT4 headers
#include "G4RunManager.hh"
#include "G4UImanager.hh"
#include "QGSP_BERT.hh"
#include "G4VisExecutive.hh"
#include "G4UIExecutive.hh"
#include "Randomize.hh"

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

int main(int argc,char** argv){
  // Detect interactive mode (if no arguments) and define UI session
  //
  G4UIExecutive* ui = 0;
  if ( argc == 1 ) {
    ui = new G4UIExecutive(argc, argv);
```

# GEANT4 example: main

```
  ui = new G4UIExecutive(argc, argv);
}

// Choose the Random engine
G4Random::setTheEngine(new CLHEP::RanecuEngine);

// Construct the default run manager
//
G4RunManager* runManager = new G4RunManager();

// Set mandatory initialization classes
//
// Detector construction
DetectorConstruction *mydet = new DetectorConstruction(); // our code
runManager->SetUserInitialization(mydet);

// Physics list
G4VModularPhysicsList* physicsList = new QGSP_BERT();
physicsList->SetVerboseLevel(1);
runManager->SetUserInitialization(physicsList);

// User action initialization
ActionInitialization *myaction = new ActionInitialization(); // our code
runManager->SetUserInitialization(myaction);
```

--- G4example.cpp 43% L72 (C++/l Abbrev)----------

# GEANT4 example: main

```
  // User action initialization
  ActionInitialization *myaction = new ActionInitialization(); // our code
  runManager->SetUserInitialization(myaction);

  // Initialize visualization
  //
  G4VisManager* visManager = new G4VisExecutive();
  visManager->Initialize();

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

  // Process macro or start UI session
  //
  if ( ! ui ) {
    // batch mode
    G4String command = "/control/execute ";
    G4String fileName = argv[1];
    UImanager->ApplyCommand(command+fileName);
  }
  else {
    // interactive mode
    UImanager->ApplyCommand("/control/execute init_vis.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;
}
```

# GEANT4 example: construction header

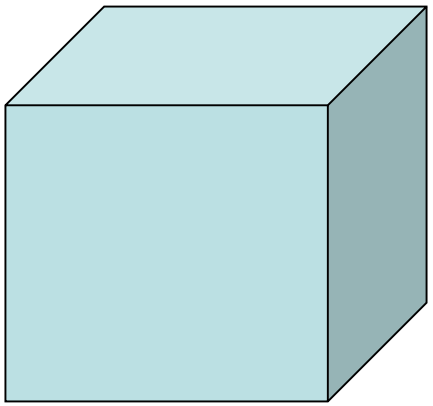
```
//
/// \file DetectorConstruction.hh
/// \brief Definition of the DetectorConstruction class

#ifndef DetectorConstruction_h
#define DetectorConstruction_h 1

#include "G4VUserDetectorConstruction.hh"
#include "globals.hh"

class G4VPhysicalVolume;
class G4LogicalVolume;

/// Detector construction class to define materials and geometry.

class DetectorConstruction : public G4VUserDetectorConstruction
{
  public:
    DetectorConstruction();
    virtual ~DetectorConstruction();

    virtual G4VPhysicalVolume* Construct();

    G4LogicalVolume* GetScoringVolume() const { return fScoringVolume; }

  protected:
    G4LogicalVolume*  fScoringVolume;
};

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

#endif
```

# GEANT4 example: construction implementation

```
//
/// \file DetectorConstruction.cpp
/// \brief Implementation of the DetectorConstruction class

#include "DetectorConstruction.h"

#include "G4RunManager.hh"
#include "G4NistManager.hh"
#include "G4Box.hh"
#include "G4Tubs.hh"
#include "G4Cons.hh"
#include "G4Orb.hh"
#include "G4Sphere.hh"
#include "G4Trd.hh"
#include "G4LogicalVolume.hh"
#include "G4PVPlacement.hh"
#include "G4SystemOfUnits.hh"
#include "G4VisAttributes.hh"

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

DetectorConstruction::DetectorConstruction()
    : G4VUserDetectorConstruction(),
      fScoringVolume(0)
{ }

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

DetectorConstruction::~DetectorConstruction()
{ }

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

G4VPhysicalVolume* DetectorConstruction::Construct()
{
  // Get nist material manager
  G4NistManager* nist = G4NistManager::Instance();

  // Option to switch on/off checking of volumes overlaps
  //
  G4bool checkOverlaps = true;
```

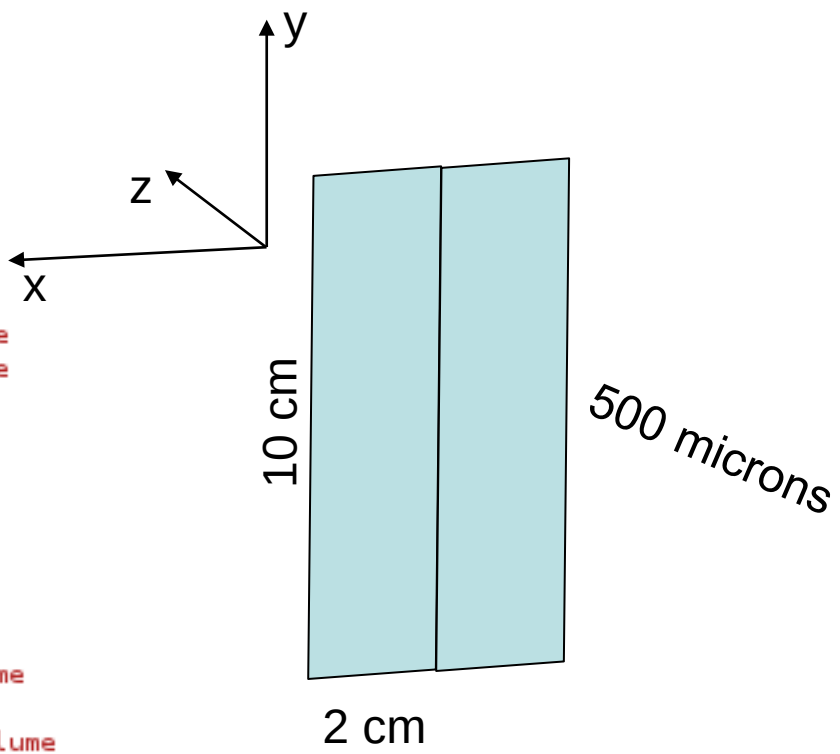
# GEANT4 example: construction implementation

```
// World
//
G4double world_sizeXY = 100*cm;
G4double world_sizeZ  = 100*cm;
G4Material* world_mat = nist->FindOrBuildMaterial("G4_AIR");

G4Box* solidWorld =
    new G4Box("World",                          //its name
              0.5*world_sizeXY, 0.5*world_sizeXY, 0.5*world_sizeZ);     //its size

G4LogicalVolume* logicWorld =
    new G4LogicalVolume(solidWorld,             //its solid
                        world_mat,              //its material
                        "World");               //its name

G4VPhysicalVolume* physWorld =
    new G4PVPlacement(0,                        //no rotation
                      G4ThreeVector(),          //at (0,0,0)
                      logicWorld,               //its logical volume
                      "World",                  //its name
                      0,                        //its mother  volume
                      false,                    //no boolean operation
                      0,                        //copy number
                      checkOverlaps);           //overlaps checking
```



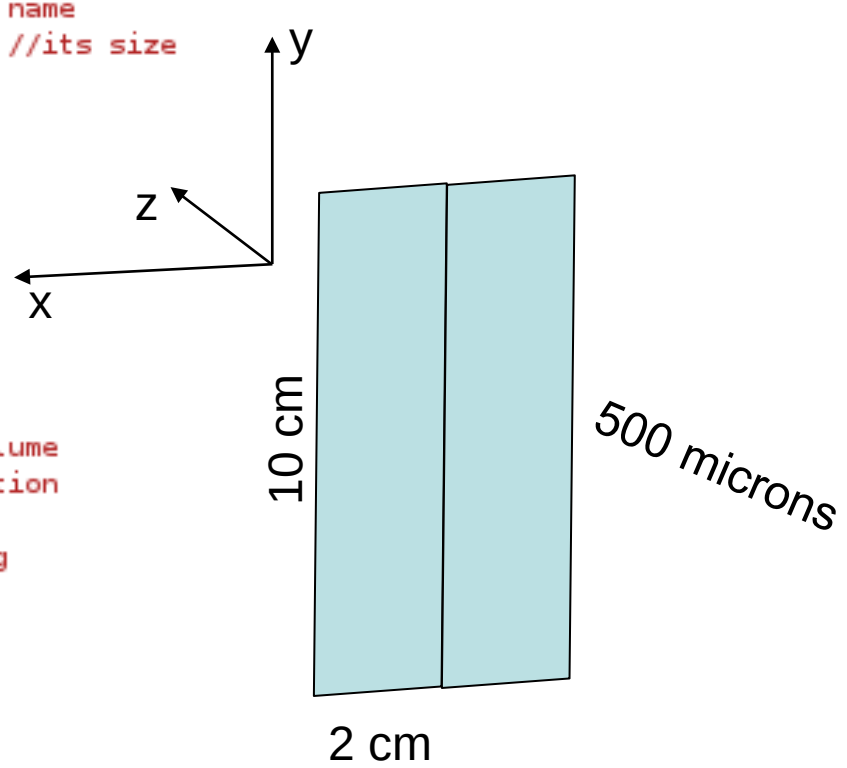
# GEANT4 example: construction implementation

```
//
// Silicon plane X
//
G4Material* Silicon = nist->FindOrBuildMaterial("G4_Si");

G4ThreeVector pos1X = G4ThreeVector(-1.*cm, 0*cm, -7*cm);
G4ThreeVector pos2X = G4ThreeVector(+1.*cm, 0*cm, -7*cm);

// strip 1
G4Box* solidStripX =
    new G4Box("stripX",                                  //its name
              0.5*2*cm, 0.5*10*cm, 0.5*0.5*mm);          //its size

G4LogicalVolume* logicStripX =
    new G4LogicalVolume(solidStripX,          //its solid
                        Silicon,              //its material
                        "StripX");            //its name

new G4PVPlacement(0,                         //no rotation
                  pos1X,                      //at position
                  logicStripX,               //its logical volume
                  "Strip1X",                  //its name
                  logicWorld,                  //its mother  volume
                  false,                     //no boolean operation
                  0,                         //copy number
                  checkOverlaps);            //overlaps checking

new G4PVPlacement(0,                         //no rotation
                  pos2X,                      //at position
                  logicStripX,               //its logical volume
                  "Strip2X",                  //its name
                  logicWorld,                  //its mother  volume
                  false,                     //no boolean operation
                  1,                         //copy number
                  checkOverlaps);            //overlaps checking
```



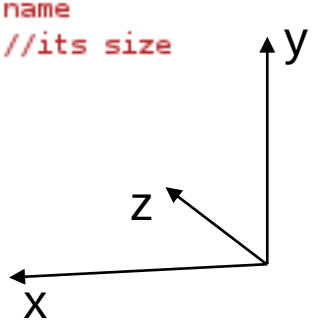
# GEANT4 example: construction implementation

```
//
// Lead disk
//
G4Material* Lead = nist->FindOrBuildMaterial("G4_Pb");

G4ThreeVector posT = G4ThreeVector(0*cm, 0.*cm, -7*cm+ 0.5*mm +1*cm);

// disk
G4Tubs* solidDisk =
    new G4Tubs("solidDisk",                                    //its name
               0.5*cm, 6*cm, 0.5*2*mm,  0.*deg,360.*deg);      //its size

G4LogicalVolume* logicDisk =
    new G4LogicalVolume(solidDisk,          //its solid
                        Lead,               //its material
                        "Disk");            //its name

new G4PVPlacement(0,                        //no rotation
                  posT,                     //at position
                  logicDisk,              //its logical volume
                  "Disk",                 //its name
                  logicWorld,                 //its mother  volume
                  false,                    //no boolean operation
                  0,                        //copy number
                  checkOverlaps);           //overlaps checking
```



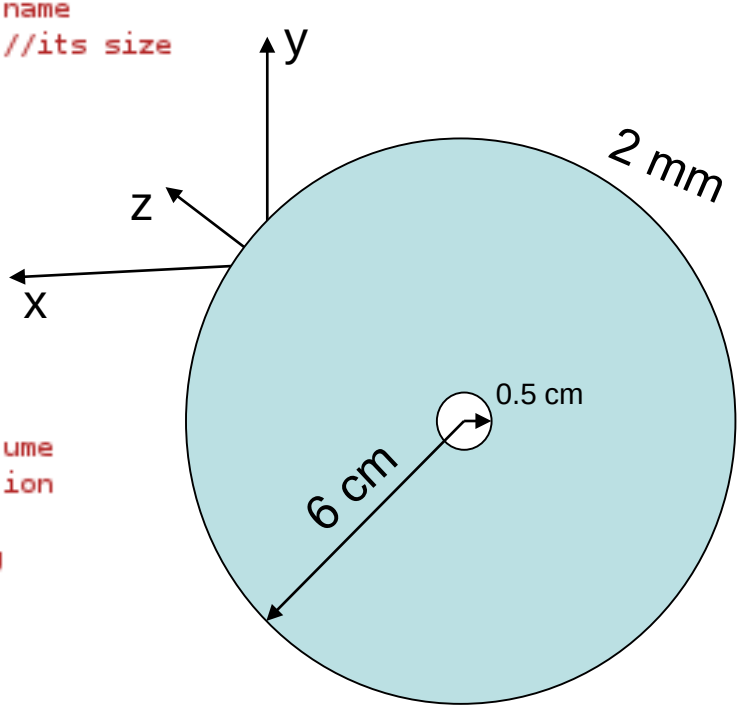
# GEANT4 example: construction implementation

```
//
// Lead disk
//
G4Material* Lead = nist->FindOrBuildMaterial("G4_Pb");

G4ThreeVector posT = G4ThreeVector(0*cm, 0.*cm, -7*cm+ 0.5*mm +1*cm);

// disk
G4Tubs* solidDisk =
    new G4Tubs("solidDisk",                                  //its name
               0.5*cm, 6*cm, 0.5*2*mm,  0.*deg,360.*deg);     //its size

G4LogicalVolume* logicDisk =
    new G4LogicalVolume(solidDisk,          //its solid
                        Lead,              //its material
                        "Disk");           //its name

new G4PVPlacement(0,                       //no rotation
                  posT,                    //at position
                  logicDisk,              //its logical volume
                  "Disk",                 //its name
                  logicWorld,                 //its mother  volume
                  false,                   //no boolean operation
                  0,                       //copy number
                  checkOverlaps);          //overlaps checking
```



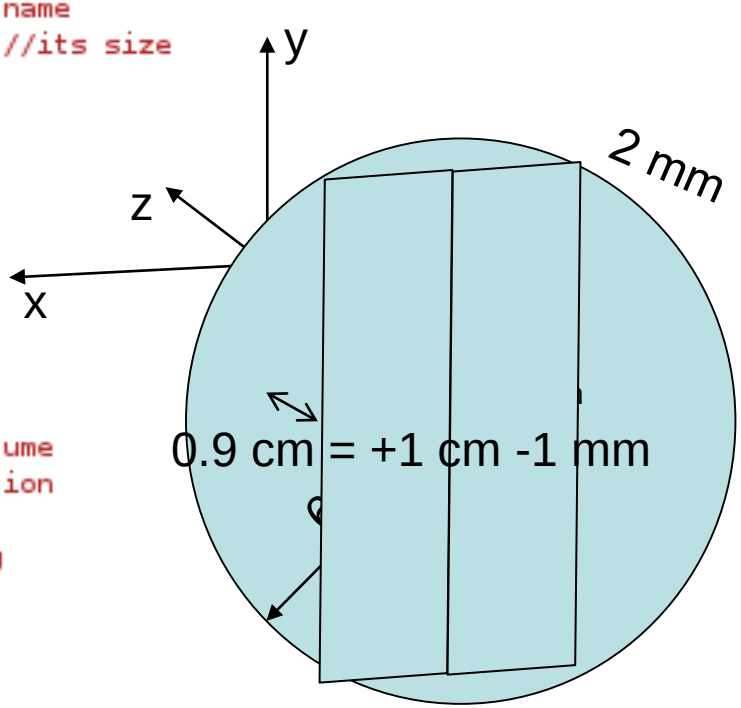
# GEANT4 example: construction implementation

```
//
// Lead disk
//
G4Material* Lead = nist->FindOrBuildMaterial("G4_Pb");

G4ThreeVector posT = G4ThreeVector(0*cm, 0.*cm, -7*cm+ 0.5*mm +1*cm);

// disk
G4Tubs* solidDisk =
    new G4Tubs("solidDisk",                                    //its name
               0.5*cm, 6*cm, 0.5*2*mm,  0.*deg,360.*deg);      //its size

G4LogicalVolume* logicDisk =
    new G4LogicalVolume(solidDisk,          //its solid
                        Lead,               //its material
                        "Disk");            //its name

new G4PVPlacement(0,                        //no rotation
                  posT,                     //at position
                  logicDisk,              //its logical volume
                  "Disk",                 //its name
                  logicWorld,                 //its mother  volume
                  false,                    //no boolean operation
                  0,                        //copy number
                  checkOverlaps);           //overlaps checking
```



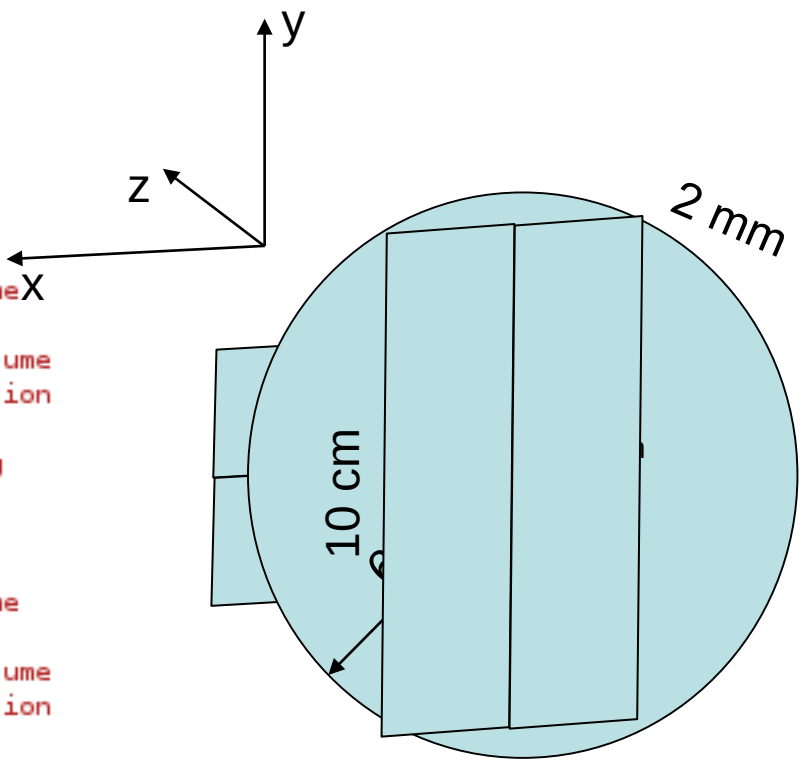
# GEANT4 example: construction implementation

```
//
// Lead disk
//
G4Material* Lead = nist->FindOrBuildMaterial("G4_Pb");

G4ThreeVector posT = G4ThreeVector(0*cm, 0.*cm, -7*cm+ 0.5*mm +1*cm);

// disk
G4Tubs* solidDisk =
    new G4Tubs("solidDisk",                                    //its name
               0.5*cm, 6*cm, 0.5*2*mm,  0.*deg,360.*deg);      //its size

G4LogicalVolume* logicDisk =
    new G4LogicalVolume(solidDisk,          //its solid
                        Lead,               //its material
                        "Disk");            //its name

new G4PVPlacement(0,                        //no rotation
                  posT,                     //at position
                  logicDisk,              //its logical volume
                  "Disk",                 //its name
                  logicWorld,                 //its mother  volume
                  false,                    //no boolean operation
                  0,                        //copy number
                  checkOverlaps);           //overlaps checking
```



# GEANT4 example: construction implementation

```
// Silicon plane Y
//
G4ThreeVector pos1Y = G4ThreeVector(0*cm, -1.*cm, -7*cm+ 0.5*mm +1*cm+2*mm+1*cm);
G4ThreeVector pos2Y = G4ThreeVector(0*cm, +1.*cm, -7*cm+ 0.5*mm +1*cm+2*mm+1*cm);

// strip 1
G4Box* solidStripY =
    new G4Box("stripY",                                  //its name
              0.5*10*cm, 0.5*2*cm, 0.5*0.5*mm);      //its size

G4LogicalVolume* logicStripY =
    new G4LogicalVolume(solidStripY,          //its solid
                        Silicon,              //its material
                        "StripY");            //its name

new G4PVPlacement(0,                         //no rotation
                  pos1Y,                      //at position
                  logicStripY,               //its logical volume
                  "Strip1Y",                  //its name
                  logicWorld,                  //its mother  volume
                  false,                     //no boolean operation
                  0,                         //copy number
                  checkOverlaps);            //overlaps checking

new G4PVPlacement(0,                         //no rotation
                  pos2Y,                      //at position
                  logicStripY,               //its logical volume
                  "Strip2Y",                  //its name
                  logicWorld,                  //its mother  volume
                  false,                     //no boolean operation
                  1,                         //copy number
                  checkOverlaps);            //overlaps checking
```



# GEANT4 example: construction implementation

```
  // Set StripY as scoring volume
  //
  fScoringVolume = logicStripY;

  // visualization attributes
  //
  G4VisAttributes* VisAtt1 = new G4VisAttributes(G4Colour(0.3, 0.8, 0.1));
  VisAtt1->SetVisibility(true);
  VisAtt1->SetForceSolid(TRUE);

  G4VisAttributes* VisAtt2 = new G4VisAttributes(G4Colour(0., 1., 1.));
  VisAtt2->SetVisibility(true);
  VisAtt2->SetForceSolid(TRUE);

  G4VisAttributes* VisAttGold = new G4VisAttributes(G4Colour(1., 1., 0.));
  VisAttGold->SetVisibility(true);
  VisAttGold->SetForceSolid(TRUE);

  logicStripX->SetVisAttributes(VisAtt1);
  logicStripY->SetVisAttributes(VisAtt2);
  logicDisk->SetVisAttributes(VisAttGold);


  //
  //always return the physical World
  //
  return physWorld;
}

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

# GEANT4 code: compilation

**1) setup GEANT4 environment:**

```
source /usr/local/geant4.10.01.p01-install/bin/geant4.sh
```

```
|Emi@w13 G4example-v0>ll
total 40K
drwxr-x--- 2 mocchiut users 4.0K May 11  2017 src/
-rw-r----- 1 mocchiut users 3.9K May 11 14:18 G4example.cpp
-rw-r----- 1 mocchiut users  338 May 11 12:50 init_vis.mac
-rw-r----- 1 mocchiut users 1.8K May 11 12:50 vis.mac
drwxr-x--- 2 mocchiut users 4.0K May 11 10:46 include/
-rw-r----- 1 mocchiut users 2.4K May 11 10:14 CMakeLists.txt
-rw-r----- 1 mocchiut users  226 Apr  1  2015 G4example.in
-rw-r----- 1 mocchiut users  475 Apr  1  2015 GNUmakefile
-rw-r----- 1 mocchiut users  553 Apr  1  2015 run1.mac
-rw-r----- 1 mocchiut users  448 Apr  1  2015 run2.mac
|Emi@w13 G4example-v0>cd ..
|Emi@w13 g4>ll
total 20K
-rw-r----- 1 mocchiut users 9.2K May 11 14:25 G4example-v0.tar.gz
drwxr-x--- 4 mocchiut users 4.0K May 11 14:25 G4example-v0/
drwxr-x--- 3 mocchiut users 4.0K May 11 14:20 old/
|Emi@w13 g4>mkdir build
|Emi@w13 g4>cd build/
/home/mocchiut/g4/build
|Emi@w13 build>cmake /home/mocchiut/g4/G4example-v0
```

**2) create an empty directory (anywhere)**

# GEANT4 code: compilation

3) cmake /path/to/source/directory

```
|Emi@w13 build>
|Emi@w13 build>cmake /home/mocchiut/g4/G4example-v0
-- 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/mocchiut/g4/build
|Emi@w13 build>
```

# GEANT4 code: compilation

4) make

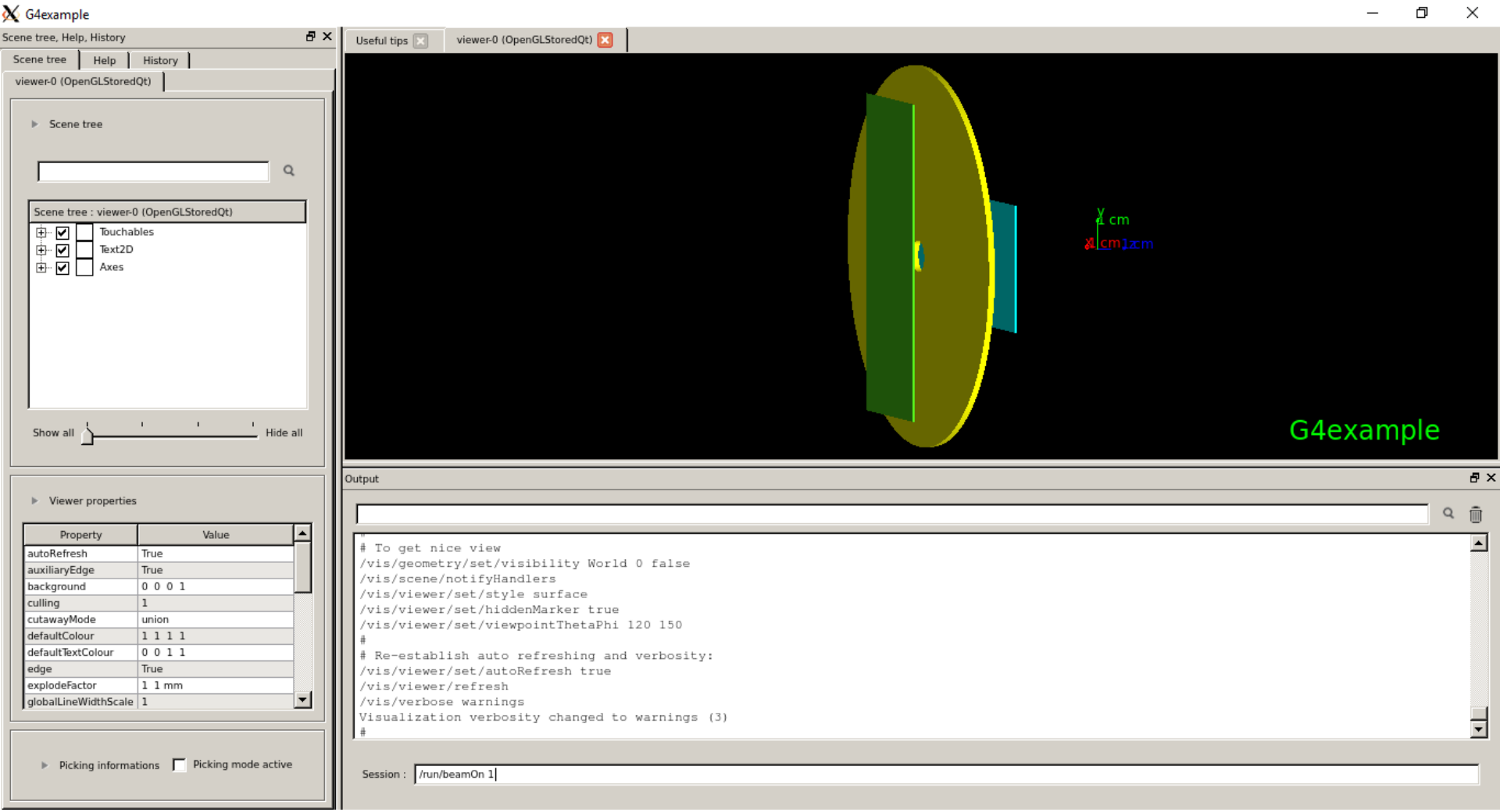
```
|Emi@w13 build>make
make: Warning: File `Makefile' has modification time 0.81 s in the future
make[1]: Warning: File `CMakeFiles/Makefile2' has modification time 0.79 s in the future
make[2]: Warning: File `CMakeFiles/G4example.dir/flags.make' has modification time 0.78 s in the future
Scanning dependencies of target G4example
make[2]: warning:  Clock skew detected.  Your build may be incomplete.
make[2]: Warning: File `CMakeFiles/G4example.dir/flags.make' has modification time 0.4 s in the future
[ 12%] Building CXX object CMakeFiles/G4example.dir/G4example.cpp.o
[ 25%] Building CXX object CMakeFiles/G4example.dir/src/SteppingAction.cpp.o
[ 37%] Building CXX object CMakeFiles/G4example.dir/src/EventAction.cpp.o
[ 50%] Building CXX object CMakeFiles/G4example.dir/src/PrimaryGeneratorAction.cpp.o
[ 62%] Building CXX object CMakeFiles/G4example.dir/src/ActionInitialization.cpp.o
[ 75%] Building CXX object CMakeFiles/G4example.dir/src/DetectorConstruction.cpp.o
[ 87%] Building CXX object CMakeFiles/G4example.dir/src/RunAction.cpp.o
[100%] Building CXX object CMakeFiles/G4example.dir/src/Run.cpp.o
Linking CXX executable G4example
make[2]: warning:  Clock skew detected.  Your build may be incomplete.
[100%] Built target G4example
make[1]: warning:  Clock skew detected.  Your build may be incomplete.
make: warning:  Clock skew detected.  Your build may be incomplete.
|Emi@w13 build>
```

ignore this warnings about time skew...

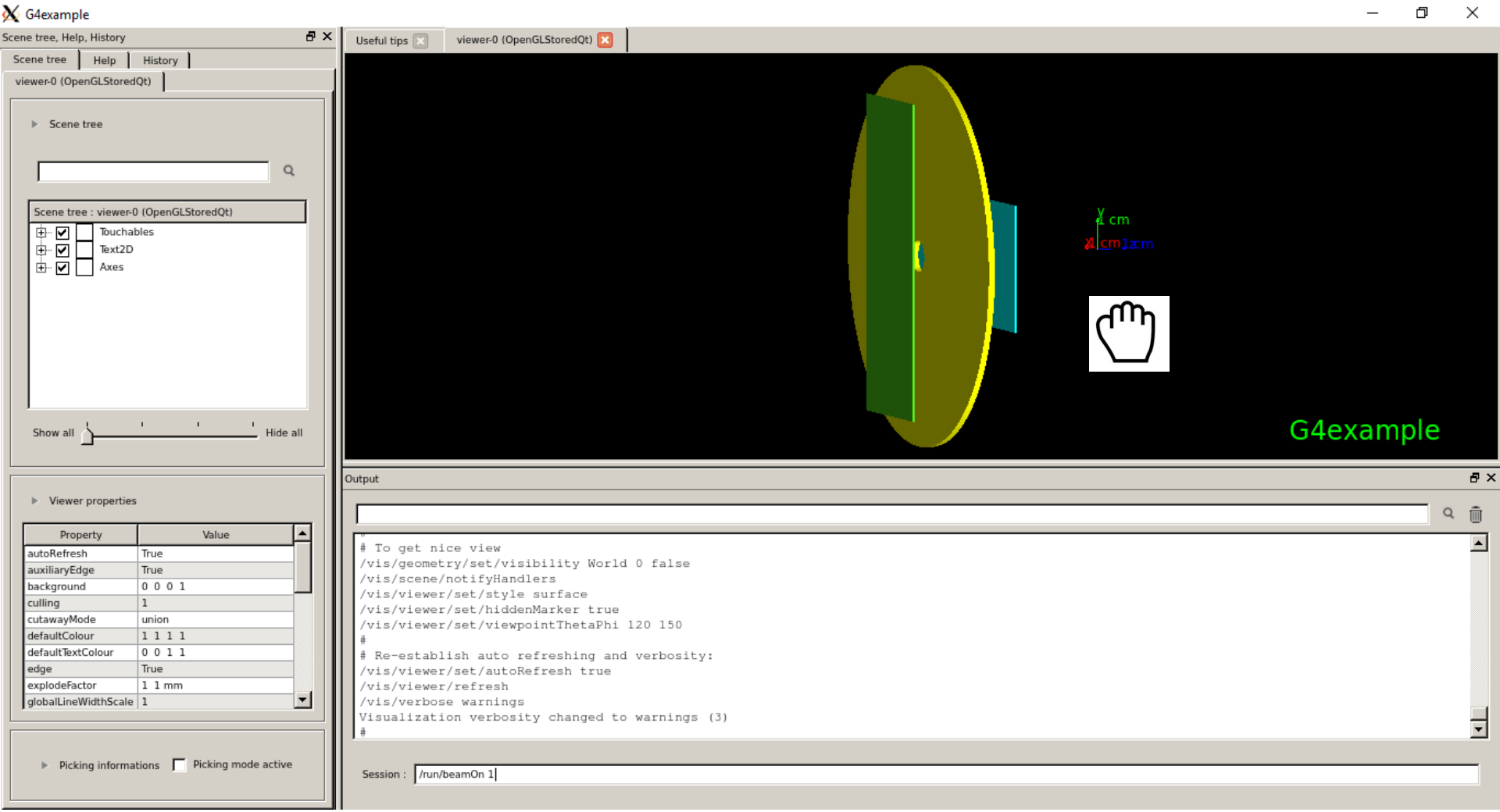
# GEANT4 code: running

```
|Emi@w13 build>make
make: Warning: File `Makefile' has modification time 0.81 s in the future
make[1]: Warning: File `CMakeFiles/Makefile2' has modification time 0.79 s in the future
make[2]: Warning: File `CMakeFiles/G4example.dir/flags.make' has modification time 0.78 s in the future
Scanning dependencies of target G4example
make[2]: warning:  Clock skew detected.  Your build may be incomplete.
make[2]: Warning: File `CMakeFiles/G4example.dir/flags.make' has modification time 0.4 s in the future
[ 12%] Building CXX object CMakeFiles/G4example.dir/G4example.cpp.o
[ 25%] Building CXX object CMakeFiles/G4example.dir/src/SteppingAction.cpp.o
[ 37%] Building CXX object CMakeFiles/G4example.dir/src/EventAction.cpp.o
[ 50%] Building CXX object CMakeFiles/G4example.dir/src/PrimaryGeneratorAction.cpp.o
[ 62%] Building CXX object CMakeFiles/G4example.dir/src/ActionInitialization.cpp.o
[ 75%] Building CXX object CMakeFiles/G4example.dir/src/DetectorConstruction.cpp.o
[ 87%] Building CXX object CMakeFiles/G4example.dir/src/RunAction.cpp.o
[100%] Building CXX object CMakeFiles/G4example.dir/src/Run.cpp.o
Linking CXX executable G4example
make[2]: warning:  Clock skew detected.  Your build may be incomplete.
[100%] Built target G4example
make[1]: warning:  Clock skew detected.  Your build may be incomplete.
make: warning:  Clock skew detected.  Your build may be incomplete.
|Emi@w13 build>./G4example
Available UI session types: [ Qt, GAG, tcsh, csh ]
```

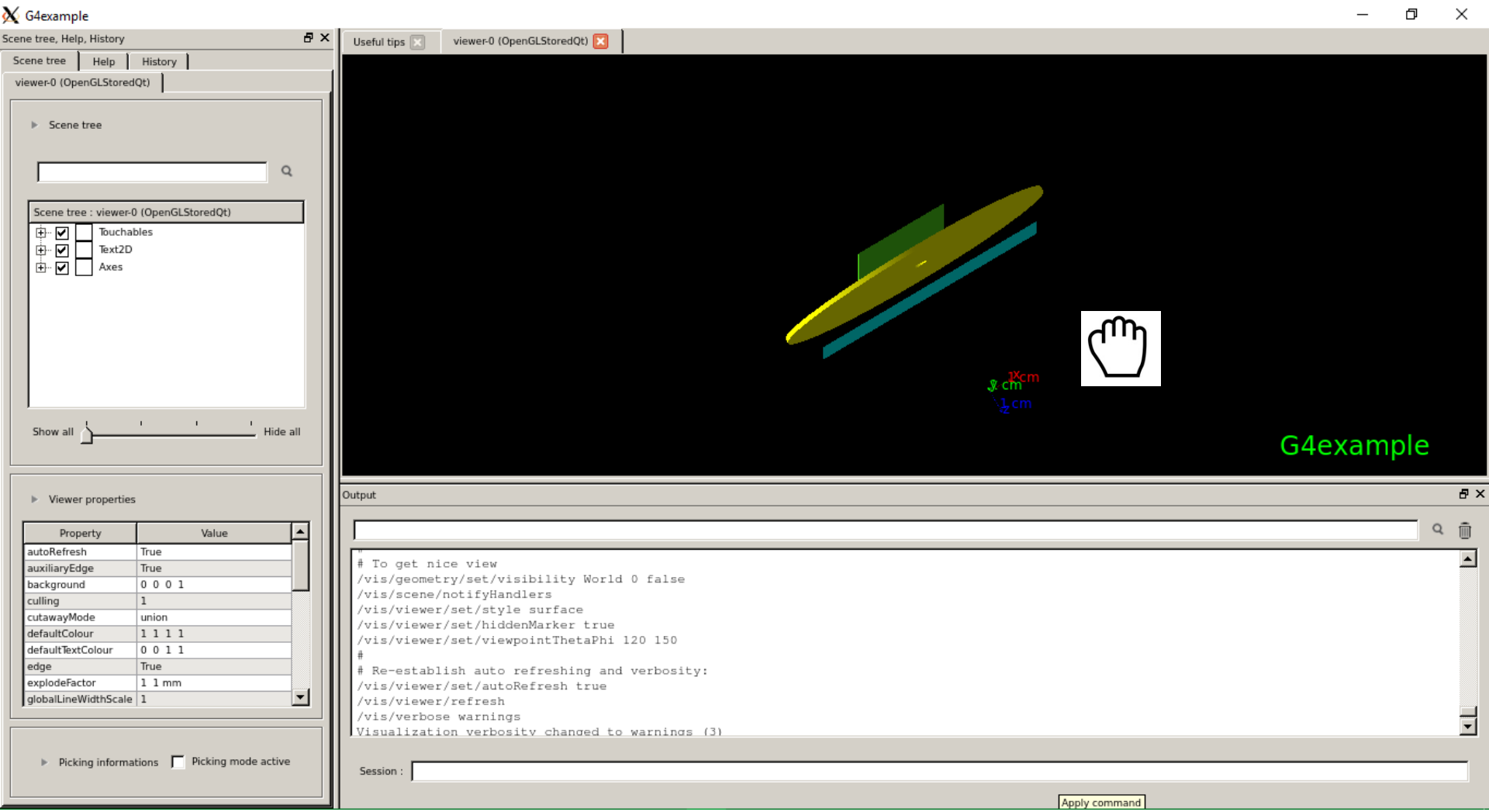
# GEANT4 code: running



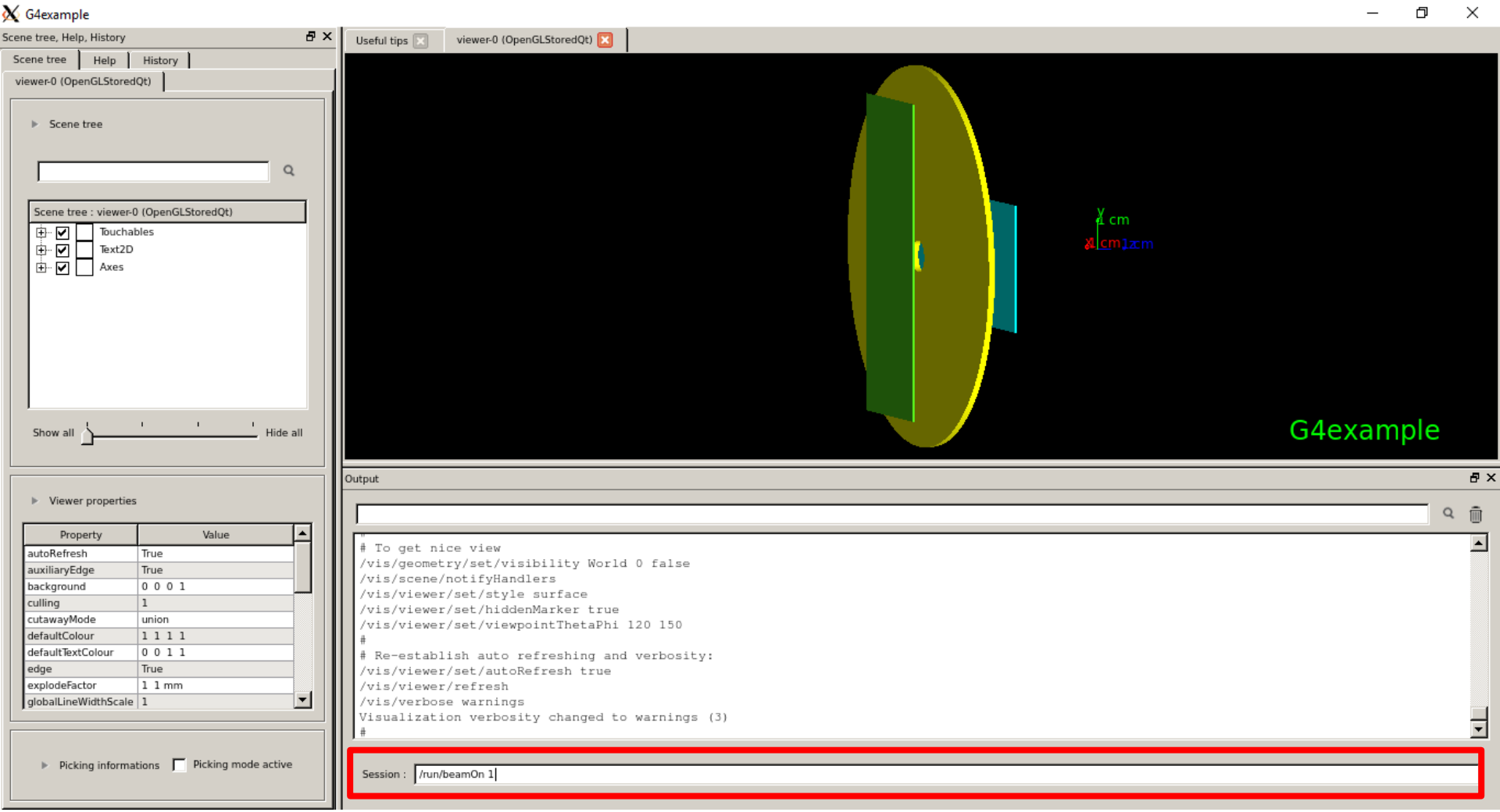
# GEANT4 code: running



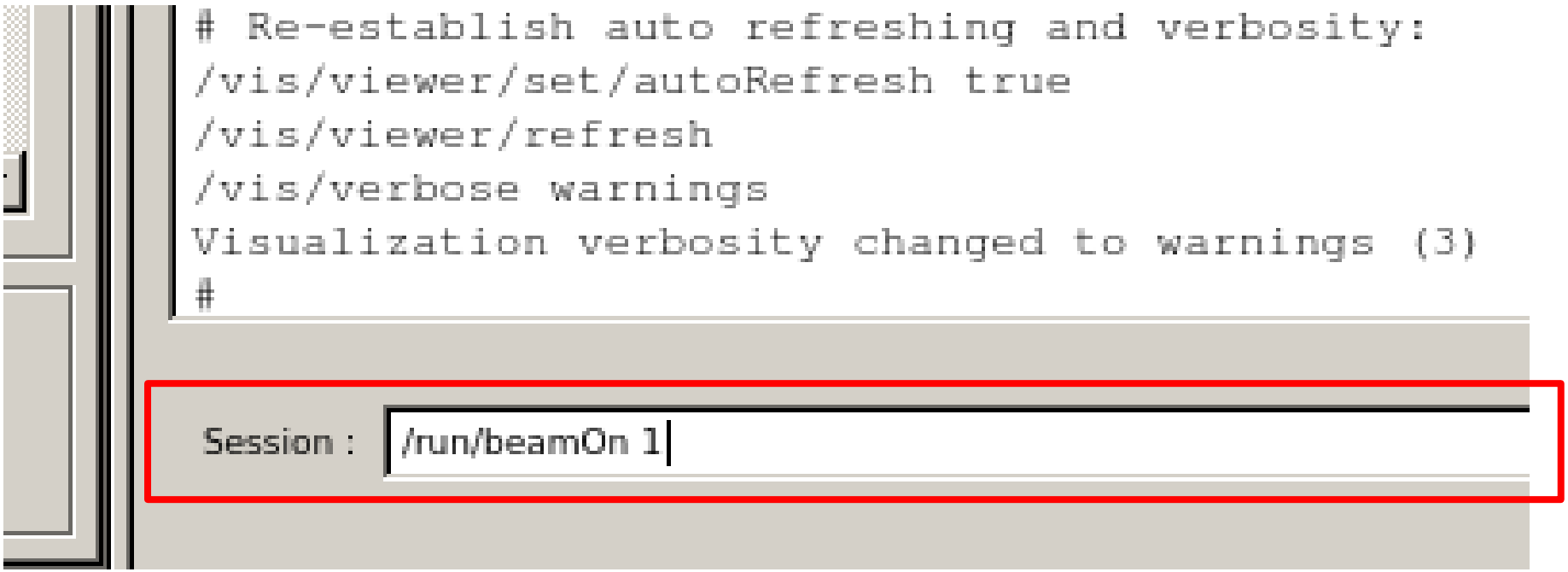
# GEANT4 code: running



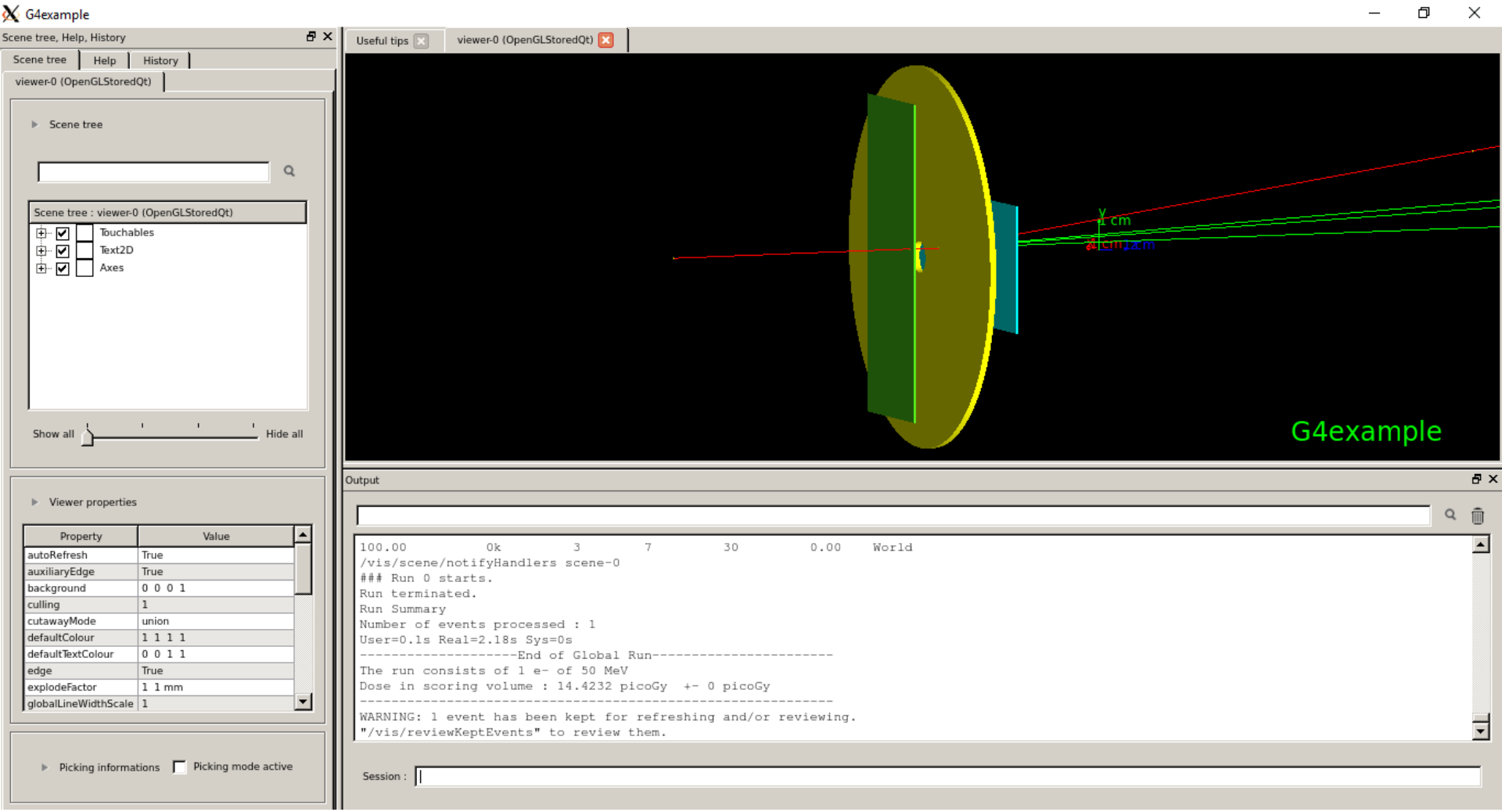
# GEANT4 code: running



# GEANT4 code: running



# GEANT4 code: running



# 12/05/2017 take home message

- GEANT4 is a giant toolkit, quite complex
- "main" program must be written by the user
- three inputs are needed: geometry, physics, and generation of primary particles
- learn how to build materials and construct objects geometry

# Exercises

1) download G4example-v0.tar.gz from moodle, compile and run it
2) add a square (box: 100x100x1.5 mm3) tungsten (G4_W) plane after "StripY"
3) add a new silicon plane (x strips, i.e. use logical volume for x strips) made by 4 strips