Frequently Asked Questions

Release 10.7

Geant4 Collaboration

Rev5.0 - December 4th, 2020

CONTENTS:

1 Installation **1**

2 Run Time Problems **3**

3 Geometry **5**
3.1 Conversion Global to Local Co-ordinates 6

4 Tracks and steps **7**
4.1 Access Track Info 7

5 Physics and cuts **11**

6 Visualization **13**

7 User Support Policy **15**

CHAPTER

ONE

INSTALLATION

Q: When I download the source from the web, and unpack the tar file, some files unpack into the top level directory.

A: The problem you describe usually is the result of using "UNIX" tar to unpack the gtar ("GNU-tar") file, or vice versa, or using zip on either the gtar or tar file. Please make certain that you download the correct file for your system, and that you use the correct unpacking tool. Note that for Linux you must download the gtar.gz file.

Q: I cannot find CLHEP files or library and I have it installed in my system.

A: If the standard CLHEP installation procedure has been adopted, the variable `CLHEP_BASE_DIR` should point to the area where `include/` and `lib/` directories for CLHEP headers & library are installed in your system. In case the library file name is different than the one expected (`libCLHEP.a`), you should either create a symbolic link with the expected name, or define the variable `CLHEP_LIB` in your environment which explicitly sets the name of the CLHEP library. If a non-standard CLHEP installation has been adopted, define variables `CLHEP_INCLUDE_DIR`, `CLHEP_LIB_DIR` (and `CLHEP_LIB`) to refer explicitly to the place where headers, library (and library-name) respectively are placed in your system. On Windows systems, the full library file name (with extension) should be specified as `CLHEP_LIB`, while for UNIX-like systems, just the name is required (i.e. `CLHEP` for `libCLHEP.a`).

Q: While installing the Geant4 libraries I get the following message printed:

```
gmake[1]: cernlib: Command not found
```

Has Geant4 been installed properly ? What to do to solve this error ?

A: The message:

```
gmake[1]: cernlib: Command not found
```

shows that you don't have the 'cernlib' command installed in your system; 'cernlib' is a command from the CERN program library (cernlib) returning a list of libraries needed to link a cernlib application. This command is only used in the 'g3tog4' module, however, if you do not make use of the 'g3tog4' tool, it's harmless. The cernlib script (and the needed cernlib libraries) are available from http://cern.ch/cernlib.

Q: Trying building the Geant4 libraries I see several of these errors appearing and my installation fails:

```
.....G4Exception.d:1: ***  missing separator. Stop.
...../G4DalitzDecayChannel.d:1: *** missing separator. Stop.
  :
  :
```

Has Geant4 been installed properly ? What to do to solve this error ?

A: It looks like some file dependencies (.d) are corrupted, possibly due to previous build attempts which failed for some reason. You need to remove each of them. A quick recipe for doing this is to:

- Configure the environment with the installation to be repaired
- Unset the `G4WORKDIR` environment variable (in case it is eventually set)
- Type:

```
gmake clean dependencies=''
```

from the affected module (i.e. for this case, from `$G4INSTALL/source/global/management` and `$G4INSTALL/source/particles/management`) and rebuild. Alternatively, you may use:

```
gmake clean dependencies=''
```

from `$G4INSTALL/source` and rebuild.

CHAPTER

TWO

RUN TIME PROBLEMS

Q: On Linux, I get a segmentation fault as soon as I run one of the official examples.

A: Check that the CLHEP library has been installed and compiled coherently with the same compiler you use for installing Geant4 and for the same version of Linux distribution. For example, a binary object produced with Red-Hat 7.X is not fully compatible with binaries running on RH 9.X or higher, due to different libc used in the two configurations.

Q: I installed Geant4 libraries and built my application, when I try to run it I get:

```
error in loading shared libraries:
libCLHEP.so: cannot open shared object file:
No such file or directory.
```

A: Your installation of CLHEP includes shared libraries. You need to specify the path where libCLHEP.so is installed through your environment variable `LD_LIBRARY_PATH`. For example, in tcsh UNIX shell:

```
setenv LD\_LIBRARY\_PATH ${LD\_LIBRARY\_PATH}:$CLHEP\_BASE\_DIR/lib
```

Q: On my system I get a Floating Point Exception (FPE) since some physics processes sometimes return `DBL_MAX` as interaction length and this number is afterwards multiplied by a number greater than 1.

A: Geant4 coding conventions and installation setup explicitly follow the ANSI/IEEE-754 Standard for the initialization of floating-point arithmetic hardware and portability. The Standard foresees floating-point arithmetic to be nonstop and underflows to be gradual. On DEC platforms, for example, the ANSI/IEEE-754 Standard compliance needs to be explicitly set (since deactivated by default); in this case we use infact the option "-ieee" on the DEC/cxx native C++ compiler to achieve this. You should check if your compiler provides compilation options for activating Standard initialization of FP arithmetic (it may be platform specific).

CHAPTER

THREE

GEOMETRY

Q: I have a generic point and I would like to know in which physical volume I'm located in my detector geometry.

A: The best way of doing this is by invoking the G4Navigator. First get a pointer of the navigator through the G4TransportationManager, and then locate the point. i.e.

```
#include "G4TransportationManager.hh"
#include "G4Navigator.hh"
G4ThreeVector myPoint = ....;
G4Navigator* theNavigator = G4TransportationManager::GetTransportationManager()
                          ->GetNavigatorForTracking();
G4VPhysicalVolume* myVolume = theNavigator->LocateGlobalPointAndSetup(myPoint);
```

Note: by using the navigator for tracking as shown above, the actual particle gets also -relocated- in the specified position. Therefore, if this information is needed during tracking time, in order to avoid affecting tracking, you should either use an alternative G4Navigator object (which you then assign to your world-volume), or you access the information through the track or touchable as specified in the FAQ for tracking and steps.

Q: How can I access the daughter volumes of a specific physical volume?

A: Through the associated logical volume.

```
G4VPhysicalVolume* myPVolume = ....;
G4LogicalVolume* myLVolume = myPVolume->GetLogicalVolume();
for (G4int i=0; iGetNoDaughters(); i++) myPVolume = myLVolume->GetDaughter(i);
```

Q: How can I identify the exact copy-number of a specific physical volume in my mass geometry? I tried with GetCopyNo() from my physical volume pointer, but it doesn't seem to work!

A: The correct way to identify -uniquely- a physical volume in your mass geometry is by using the touchables (see also chapter 4.1.5 of the User's Guide for Application Developers), as follows:

```
G4Step* aStep = ..;
G4StepPoint* preStepPoint = aStep->GetPreStepPoint();
G4TouchableHandle theTouchable = preStepPoint->GetTouchableHandle();
G4int copyNo = theTouchable->GetCopyNumber();
G4int motherCopyNo = theTouchable->GetCopyNumber(1);
```

where `Copy` here stays for any duplicated instance of a physical volume, either if it is a `G4PVPlacement` (multiple placements of the same logical volume) or a `G4PVReplica/G4PVParameterised`. The method `GetCopyNo()` is meant to return only the serial number of placements not duplicated in the geometry tree.

3.1 Conversion Global to Local Co-ordinates

Q: How can I determine the exact position in global coordinates in my mass geometry during tracking and how can I convert it to coordinates local to the current volume ?

A: You need again to do it through the touchables (see also chapter 4.1.5 of the User's Guide for Application Developers), as follows:

```
G4Step* aStep = ..;
G4StepPoint* preStepPoint = aStep->GetPreStepPoint();
G4TouchableHandle theTouchable = preStepPoint->GetTouchableHandle();
G4ThreeVector worldPosition = preStepPoint->GetPosition();
G4ThreeVector localPosition = theTouchable->GetHistory()->GetTopTransform().
TransformPoint(worldPosition);
```

where `worldPosition` here stays for the position related to the world volume, while `localPosition` refers to the coordinates local to the volume where the particle is currently placed.

CHAPTER

FOUR

TRACKS AND STEPS

4.1 Access Track Info

Q: How can I access the track information through the step object and what information am I allowed to access ?

A: A `G4Step` object consists of two points:

```
const G4StepPoint* preStepPoint = step->GetPreStepPoint();
const G4StepPoint* postStepPoint = step->GetPostStepPoint();
```

Note that we are encouraging the use of `const`. This is good practice. It introduces safety - you cannot accidentally change a `const`.

To get their positions in the global coordinate system:

```
const G4ThreeVector& preStepPosition = preStepPoint->GetPosition();
const G4ThreeVector& postStepPosition = postStepPoint->GetPosition();
```

Also here, note, we encourage `const <type>&`, i.e., a `const` reference, which avoids copying the quantity if possible, which saves CPU time. (We simply do `const` for built-in types such as `G4int`, which is really an `int`, because copying is no slower than creating a reference, but for composite types, such as `G4ThreeVector`, there is an advantage.)

Hereafter we call current volume the volume where the step has just gone through. Geometrical informations are available from `preStepPoint`. `G4VTouchable` and its derivates keep these geometrical informations. We retrieve a touchable by creating a handle for it:

```
const G4TouchableHandle& preStepTouch = preStepPoint->GetTouchableHandle();
```

To get the current volume:

```
const G4VPhysicalVolume* volume = preStepTouch->GetVolume();
```

To get its name:

```
const G4String& name = volume->GetName();
```

To get the physical volume copy number:

```
const G4int copyNumber = preStepTouch->GetCopyNumber();
```

To get logical volume:

```
const G4LogicalVolume* lVolume = volume->GetLogicalVolume();
```

To get the associated material: the following statements are equivalent:

```
const G4Material* material = preStepPoint ->GetMaterial();
const G4Material* material = lVolume ->GetMaterial();
```

To get the geometrical region:

```
const G4Region* region = lVolume->GetRegion();
```

To get its mother volume:

```
const G4VPhysicalVolume* mother = preStepTouch->GetVolume(depth=1);
```

grandMother: depth=2 . . . etc. . .

To get the copy number of the mother volume:

```
G4int depth;
const G4int copyNumber = preStepTouch->GetCopyNumber(depth=1);
```

grandMother: depth=2 . . . etc. . .

To get the process which has limited the current step:

```
const G4VProcess* aProcess = postStepPoint->GetProcessDefinedStep();
```

To check that the particle has just entered in the current volume (i.e. it is at the first step in the volume; the `preStepPoint` is at the boundary):

```
if (preStepPoint->GetStepStatus() == fGeomBoundary)
```

To check that the particle is leaving the current volume (i.e. it is at the last step in the volume; the `postStepPoint` is at the boundary):

```
if (postStepPoint->GetStepStatus() == fGeomBoundary)
```

In the above situation, to get touchable of the next volume: G4TouchableHandle touch2 = postStepPoint->GetTouchableHandle(); From `touch2`, all informations on the next volume can be retrieved as above.

Physics quantities are available from the step (`G4Step`) or from the track (`G4Track`).

To get the energy deposition, step length, displacement and time of flight spent by the current step:

```
const G4double eDeposit = step->GetTotalEnergyDeposit();
const G4double sLength = step->GetStepLength();
const G4ThreeVector& displace = step->GetDeltaPosition();
const G4double tof = step->GetDeltaTime();
```

To get momentum, kinetic energy and global time (time since the beginning of the event) of the track after the completion of the current step:

```
const G4Track* track = step->GetTrack();
const G4ThreeVector& momentum = track->GetMomentum();
const G4double kinEnergy = track->GetKineticEnergy();
const G4double globalTime = track->GetGlobalTime();
...etc...
```

Note: To transform a position from the global coordinate system to the local system of the current volume, use the `preStepPoint` transformation, as described in the *Conversion Global to Local Co-ordinates* above.

Q: How can I get and store (or plot) informations at tracking time from a given volume ?

A: To get the information at tracking time in a given volume A, one can adopt either one or a combination of the following strategies:

1. If the geometry is simple enough, and wish to score some commonly used physics quantities (e.g. energy deposition, dose, flux, etc.), just activate `G4ScoringManager` in your main program, and use the scorer-based UI commands to transform volume A into a scorer.
   See Option 6 below, and the example `examples/extended/runAndEvent/RE03`.
2. Through the `SteppingAction`, check that the particle is inside volume A and do whatever needed. Hints can be found in *Access Track Info* of this FAQ document.
   Usually, the hits containers and histograms are attributes of a Track, Event or Run and can be managed through either a `TrackingAction`, `EventAction` and/or `RunAction` and eventually messaging their pointer to the `SteppingAction`.
   A similar approach is illustrated in `examples/basic/B2`, `B4`, `extended/electromagnetic`, `optical`, and many others...
3. In `DetectorConstruction`, by declaring volume A as a `SensitiveDetector`. At stepping time, the Geant4 kernel will automatically check that a particle is inside volume A and will handle the control to a specific function `G4VSensitiveDetector::ProcessHits()`. It is just necessary to instanciate a class inherited from `G4VSensitiveDetector`, say `VolumeA_SD`, and do whatever needed by implementing the function `VolumeA_SD::ProcessHits()`, as described in Option 2 above.
4. In addition to Option 3 above, should create a `HitsCollection` to store the information. A `HitsCollection` can be created in `VolumeA_SD::Initialize()`. A `Hit` can be created or filled in `VolumeA_SD::ProcessHits()`. Additional operations on `HitsCollection` can be performed in `VolumeA_SD::EndOfEvent()`.
   This approach is illustrated in `examples/basic/B2`, `B4` and `extended/analysis`, `extended/runAndEvent/RE01`, etc...
5. In `DetectorConstruction`, volume A can be declared as `SensitiveDetector`, and one or several pre-defined scorers can be attached to volume A. In this case, neither a `SteppingAction` nor a spcific `VolumeA_SD` sensitive detector is needed any longer. It is just necessary to create a dedicated scorer, e.g. `MyRunScorer`, inherited from `G4Run`, and handle the `HitsCollections` within `MyRunScorer::RecordEvent()`. `MyRunScorer` itself can be instanciated from `RunAction::GenerateRun()`.
   This approach is illustrated in `examples/novice/N07`, `extended/runAndEvent/RE02`.
6. A set of build-in scorer-based UI commands allows to perform most possible operations described through the previous Option 5 directly from run-time macros.
   See example `extended/runAndEvent/RE03`.

CHAPTER

FIVE

PHYSICS AND CUTS

Q: How do production cuts (in range) work in Geant4 ? Are they also used in tracking ? If a particle has an energy lower than the converted cut in energy for the given material and the distance to the next boundary is smaller than the cut in range, is the particle killed ?

A: Geant4 does **NOT** have a "tracking cut". The toolkit's default behaviour is to track particles down to zero range (i.e. zero energy). Of course, it is possible for the user to create and register a process that kills particles below a certain energy or range; this is however **NOT** provided by default in Geant4. So there's **NO** "tracking cut". For example, suppose a particle that is nearing zero energy will at some point be proposed by its Ionisation process to undergo one final step, from its current energy down to zero energy. This is still only a proposal. If during this step the particle crosses a boundary, then the transportation will limit the step at a length smaller than the Ionisation – so the particle will still see and cross the relevant boundary, and another step will occur on the other side of that boundary. In summary the "production threshold" range and its equivalent in energy are not utilised as a "tracking cut". A particle is not abandoned by Geant4 below a certain range/energy unless the user registers a process to do this by him/her-self.

CHAPTER

SIX

VISUALIZATION

Q: While visualizing my geometry setup I often see the following error message printed out: BooleanProcessor: boolean operation failed .

A: There is a known limitation for the visualization of Boolean solids in the so-called BooleanProcessor which is used to make polyhedra for visualisation. It does not affect the tracking through such solids. So the error message you see does not affect the simulation in any way.

One workaround is to move one of the affected solids by a small distance in order to avoid shared surfaces.

This problem does not affect the ray tracer, because it uses Geant4's tracking algorithms rather than polyhedral representations. So:

```
/vis/open RayTracer # or RayTracerX
/vis/drawVolume
```

If you want to use the view parameters of a previous view, say "viewer-0":

```
/vis/viewer/copyFrom viewer-0
```

Then:

```
/vis/viewer/refresh
```

If you want to study a particular subsystem:

```
/vis/drawVolume <sub-system-physical-volume-name>
```

There are many vis commands. Not all can be described here. Use the help facility - "help" or "ls" on the command line or the help facility in Qt or Section "Built-in commands" in the Application Developers Guide.

CHAPTER

SEVEN

USER SUPPORT POLICY

Q: How do I contribute an FAQ to this page?

A: This is done manually at the moment. Please send your FAQ with solution to the Documentation Working Group.

Q: If I need to discuss technical matters specific to my simulation application or ask for first-aid help, who can I contact?

A: Every institute and experiment participating in Geant4 has a G4 Technical Steering Board (TSB) representative who may be contacted for help with problems relating to simulations. Please contact the TSB representative closest to your project or to your laboratory. To find out who your TSB representative is go to G4 Technical Board. You may also post your question in the Geant4 Discourse Forum.

Q: If I find a bug or other problem with the code, who should be informed?

A: The Geant4 problem tracking system will forward the bug report to the person responsible for the affected Geant4 domain. The Geant4 web makes available a database of open incident reports, tagging the ones already fixed and showing their status. An acknowledgement of the bug report will be sent.

Q: If I propose a fix, who is responsible for approving it?

A: The responsible person is the working group coordinator of the domain in which the fix is to be applied. This person is usually also a TSB member. If the fix affects more than one domain, the matter will be addressed by the TSB.

Q: To whom should I send a proposal for an improvement in Geant4 functionality?

A: Any new requirement should be submitted via the automatic web system. It will be discussed at the Geant4 TSB. You may also ask your TSB representative to forward your requirement to the TSB. A new requirement will trigger a cycle of analysis, design, implementation, testing and documentation, which may involve different working groups. Any new software or enhancement which will become a part of Geant4 must be agreed upon by the TSB, which is charged with ensuring the consistency of the entire toolkit.

Q: Is there a regular user meeting which I should attend?

A: There is one Geant4 Collaboration Workshop per year, and one or two Geant4 Technical Forum meetings. These will be announced in the *Events* section of the Geant4 Web. However, many experiments and institutes in the Geant4 collaboration organize their own regular and/or special Geant4 user workshops.

Q: Where can I find solutions to particular problems as well as general user support?

A: Solutions and tips for solving practical problems can be found on the current FAQ page. General and specific user support information is available at the User Support page.

Q: What is the proper way to cite GEANT4 in my journal or conference paper?

A: The two main reference papers for Geant4 are published in *Nuclear Instruments and Methods in Physics Research* A 506 (2003) 250-303, and *IEEE Transactions on Nuclear Science* 53 No. 1 (2006) 270-278.

Status of this Document

Frequently Asked Questions (FAQ) for the Geant4 Toolkit.

- Rev 1.0: First sphinx version implemented for Geant4 Release 10.4, 8th Dec 2017
- Rev 2.0: Updates and fixes in documentation for GEANT4 Release 10.4, 15th May 2018
- Rev 3.0: GEANT4 Release 10.5, 11th December 2018
- Rev 3.1: GEANT4 Updates and fixes - especially to search functionality, 5th March 2019
- Rev 4.0: GEANT4 Release 10.6, 6th December 2019
- Rev 5.0: GEANT4 Release 10.7, 4th December 2020