GATE Documentation

OpenGATE Collaboration

Jun 07, 2021

Contents

CHAPTER 1

Getting started

1.1 Introduction

Table of Contents

GATE a Monte-Carlo simulation toolkit for medical physics applications

1.1.1 Authors

- OpenGATE collaboration : http://www.opengatecollaboration.org
- OpenGATE spokesperson: L. Maigne (LPC UMR 6533 CNRS/IN2P3, Clermont-Ferrand, France)
- OpenGATE technical coordinator: D. Sarrut (CREATIS UMR CNRS 5220, Lyon, France)
- Gate authors: authors
- Members of the OpenGATE Collaboration: members
- Special Thanks: Geant4 Collaboration and LOW energy WG

1.1.2 Forewords

Monte Carlo simulation is an essential tool in emission tomography to assist in the design of new medical imaging devices, assess new implementations of image reconstruction algorithms and/or scatter correction techniques, and optimise scan protocols. Although dedicated Monte Carlo codes have been developed for Positron Emission Tomography (PET) and for Single Photon Emission Computerized Tomography (SPECT), these tools suffer from a variety of drawbacks and limitations in terms of validation, accuracy, and/or support (Buvat). On the other hand, accurate and versatile simulation codes such as GEANT3 (G3), EGS4, MCNP, and GEANT4 have been written for high energy physics. They all include well-validated physics models, geometry modeling tools, and efficient visualization utilities. However these packages are quite complex and necessitate a steep learning curve.

GATE, the *GEANT4 Application for Emission Tomography* (MIC02, Siena02, ITBS02, GATE, encapsulates the GEANT4 libraries in order to achieve a modular, versatile, scripted simulation toolkit adapted to the field of nuclear medicine. In particular, GATE provides the capability for modeling time-dependent phenomena such as detector movements or source decay kinetics, thus allowing the simulation of time curves under realistic acquisition conditions.

GATE was developed within the OpenGATE Collaboration with the objective to provide the academic community with a free software, general-purpose, GEANT4-based simulation platform for emission tomography. The collaboration currently includes 21 laboratories fully dedicated to the task of improving, documenting, and testing GATE thoroughly against most of the imaging systems commercially available in PET and SPECT (Staelens, Lazaro).

Particular attention was paid to provide meaningful documentation with the simulation software package, including installation and user's guides, and a list of FAQs. This will hopefully make possible the long term support and continuity of GATE, which we intend to propose as a new standard for Monte Carlo simulation in nuclear medicine.

In name of the OpenGATE Collaboration

Christian MOREL CPPM CNRS/IN2P3, Marseille, 2004

1.1.3 Overview

GATE combines the advantages of the GEANT4 simulation toolkit well-validated physics models, sophisticated geometry description, and powerful visualization and 3D rendering tools with original features specific to emission tomography. It consists of several hundred C++ classes. Mechanisms used to manage time, geometry, and radioactive sources form a core layer of C++ classes close to the GEANT4 kernel Fig. 1.1. An application layer allows for the

implementation of user classes derived from the core layer classes, e.g. building specific geometrical volume shapes and/or specifying operations on these volumes like rotations or translations. Since the application layer implements all appropriate features, the use of GATE does not require C++ programming: a dedicated scripting mechanism - hereafter referred to as the macro language - that extends the native command interpreter of GEANT4 makes it possible to perform and to control Monte Carlo simulations of realistic setups.

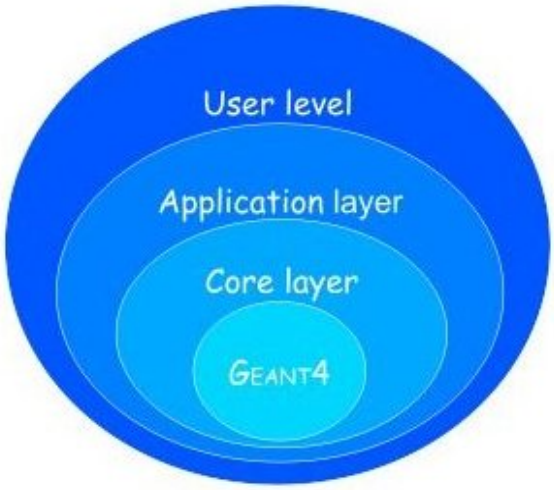


Fig. 1.1: Structure of GATE

One of the most innovative features of GATE is its capability to synchronize all time-dependent components in order to allow a coherent description of the acquisition process. As for the geometry definition, the elements of the geometry can be set into movement via scripting. All movements of the geometrical elements are kept synchronized with the evolution of the source activities. For this purpose, the acquisition is subdivided into a number of time-steps during which the elements of the geometry are considered to be at rest. Decay times are generated within these time-steps so that the number of events decreases exponentially from time-step to time-step, and decreases also inside each time-step according to the decay kinetics of each radioisotope. This allows for the modeling of time-dependent processes such as count rates, random coincidences, or detector dead-time on an event-by-event basis. Moreover, the GEANT4 interaction histories can be used to mimic realistic detector output. In GATE, detector electronic response is modeled as a linear processing chain designed by the user to reproduce e.g. the detector cross-talk, its energy resolution, or its trigger efficiency.

The first users guide was organized as follow: chapter 1 of this document guides you to get started with GATE. The macro language is detailed in Chapter 2. Visualisation tools are described in Chapter 3. Then, Chapter 4 illustrates how to define a geometry by using the macro language, Chapter 5 how to define a system, Chapter 6 how to attach sensitive detectors, and Chapter 7 how to set up the physics used for the simulation. Chapter 8 discusses the different radioactive source definitions. Chapter 9 introduces the digitizer which allows you to tune your simulation to the very experimental parameters of your setup. Chapter 10 draws the architecture of a simulation. Data output are described in Chapter 11. Finally, Chapter 12 gives the principal material definitions available in GATE. Chapter 13 illustrates the interactive, bathc, or cluster modes of running GATE.

1.2 Installation Guide V9.0

Table of Contents

1.2.1 General Information about GATE

The GATE mailing list

You are encouraged to participate in the dialog and post your suggestion or even implementation on the Gate-users mailing list, the GATE mailing list for users. You can subscribe to the Gate-users mailing list, by signing up to the gate-users mailing list.

If you have a question, it is possible that it has been asked and answered before, and stored in the archives. These archives are public and are indexed by the usual search engines. By starting your Google search string with *site:lists.opengatecollaboration.org* you'll get list of all matches of your search on the gate-users mailing list, e.g. site:lists.opengatecollaboration.org pencilbeam.

The GATE project on GitHub

GATE project is now publicly available on GitHub. You can use this to:

- Check out the bleeding edge development version
- Report bugs by creating a new issue. (If you are not entirely sure that what you are reporting is indeed a bug in Gate, then please first check the gate-users mailing list.)
- Contribute to Gate by changing the source code to fix bugs or implement new features:
    - Get a (free) account on GitHub, if you do not have one already.
    - Install Git on the computer where you do your development, if it has not yet been installed already. And make sure to configure git with your name and with your email address.
    - Start by making a fork of the GATE public repository (click the "Fork" button in the upper right corner on the Gate main page on GitHub.
    - Note that we use the *develop* branch to collect all the bleeding edge developments and the *master* to track the releases. In the future we may merge these two, and use only *master*, like it's done in most other projects on GitHub. Releases are defined using "tags".
    - Then clone your own fork: *git clone https://github.com/YOUR_USERNAME/Gate.git* to get the code on the computer that you will use to develop and compile Gate.
    - Make a new branch, dedicated to the bugfix or new feature that want to implement in Gate. You can either create the branch first on GitHub and then *git pull* it to your clone, or create it directly in your clone and

git push it later. Make sure that your branch is based on the *develop* branch. Note that after creating your branch you also need to check it out.

- With *git branch -l* you can check which branches are available in your clone and which one is currently checked out. With *git checkout <branchname>* you can change between branches. Be careful not to do this when you still have uncommitted changes (unless you deliberately want to undo those changes).
- Now: implement your bugfix or your new feature and *commit* your changes to your new branch. It's usually better to make many small commits than a single big one (though it is of course also desirable that every commit leaves the code in a compilable state). Please provide concise but informative commit messages ! Use *git push* to upload your commits to (your fork on) GitHub. This facilitates developing on multiple machines and also avoids loss of time and effort in the unfortunate event of a hardware failure.
- If you are working for a longer time on your fix or new feature, like a few days, weeks or even months, then it is important to make sure to keep your fork in sync with the upstream repository.
- Once you are convinced that your code is OK, make sure it's all pushed to your fork on GitHub. Then:
  1) Create a pull-request from the branch on your Gate repository to the official Gate repository
  2) Provide an example that tests your new feature
  3) If you implemented a new feature, have the associated documentation ready
  4) Inform these three people from the collaboration (Sebastien Jan, David Sarrut and David Boersma) who will then get in touch with you to integrate your changes in the official repository.
- For your next bugfix or new feature you do not need to make a new fork, you can use the existing one. But before doing any new work you should make sure to synchronize the *develop* branch in your fork with the "upstream" (main) *develop* branch:
  1) Check your "remote repositories" with *git remote -v*
  2) The "origin" repository should be your own fork on GitHub, *https://github.com/YOUR_USERNAME/Gate*.
  3) The "upstream" repository should be the main Gate one, that is *https://github.com/OpenGATE/Gate*.
  4) If your clone does not yet have an "upstream", then add it with *git remote add upstream https://github.com/OpenGATE/Gate*.
  5) Run *git status* to make sure that you checked out the *develop* branch, and *git pull* to make sure that it is in sync with your fork on GitHub and that there no uncommitted edits.
  6) Then run *git fetch upstream*, followed by *git merge upstream/develop*.
  7) Now you are ready to create new branches for new bugfixes and features.

- For more detailed references, recipes, and tutorials on git: please check the web. When copypasting commands, remember that in Gate the "develop" branch currently plays the role of the "master" branch. Our "master" branch is used to track the releases. You will not find the latest bleeding edge code on it. We may change this policy in the near future, to be more conforming to the predominant conventions.

Installing GATE on Linux

This section describes the installation procedure of GATE. This includes three steps:

- Install Geant4
- Install ROOT
- Install GATE

This section starts with a brief overview of the recommended configurations, followed by a short introduction to the installation of Geant4, and then explains the installation of GATE itself on Linux.

It should be highlighted that features depending on external components (libraries or packages) may only be activated if the corresponding component is installed. It is the user's responsibility to check that these components are installed before activating a feature. Except for Geant4, which is closely related to GATE, the user should refer to the Installation Guide of the external components.

In addition, you should also install any Geant4 helper you wish to use, especially *OpenGL* if required, before installing Geant4 itself. You can either download the source codes and compile the libraries or download precompiled packages which are available for a number of platform-compiler. If you choose to or have to compile the packages, you will need:

- a C++ compiler (new enough to compile code with the C++11 standard)
- the GNU version of *make*
- CMAKE tool (3.3 or newer)

The ROOT data analysis package may also be needed for post-processing or for using the GATE online plotter (enabling the visualization of various simulation parameters and results in real time). ROOT is available for many platforms and a variety of precompiled packages can be found on the ROOT homepage. If your gcc compiler is version 6 or newer, then you should use a recent ROOT 6 release.

The LMF and ecat7 packages are also provided on the GATE website. They offer the possibility to have different output formats for your simulations. Note that this code is very old and not supported by the Gate collaboration, only provided "as is". With newer compilers you may have to do some minor hacking (for ECAT you may need to add compiler flags to select the C90 standard, for instance).

Package Requirements

Compiling software usually requires certain system libraries and compilation tools. Furthermore, GATE and Geant4 have various package requirements which have to be met BEFORE installing or compiling. Currently lists have been created for Ubuntu 14.04 (and newer) and SuSE Leap 42.3. Visit the *Package Requirements* page for detailed package lists.

Installing with MacPorts on OS X

GATE can be installed on Mac OS X by following the previous installation instruction on Linux. An alternative way is to install Gate via MacPorts (http://www.macports.org/) with:

```
sudo port install gate
```

Apart from the *Gate* command this also installs a standalone app:

```
/Applications/MacPorts/Gate.app
```

(Thanks Mojca Miklavec for this contribution).

1.2.2 GATE compilation and installation

Recommended configuration

For the 9.0 release, the recommended configuration is the following:

- Geant4 10.6 (available in http://geant4.web.cern.ch/geant4/support/download.shtml), but remains backward compatible with 10.5 also.
- The GateRTion 1.0 release, which is very similar to Gate 8.1, can *only* be built with Geant4 10.03.p03.
- CMake minimal version: 3.3 (with SSL support)

Compilation instructions

Compiling GATE (V9.0)

1.2.3 Validating Installation

If you are able to run Gate after installation by typing:

```
Gate
```

it is an indication that your installation was successful.

However, before you do any research, it is highly recommended that you validate your installation.

See *Validating Installation* for benchmarks and further information.

1.2.4 Other Web Sites

- G4 Agostinelli S et al 2003 GEANT4 - a simulation toolkit Institute Nucl. Instr. Meth. A506 250-303 GEANT4 website: http://geant4.web.cern.ch/geant4/
- CLHEP - A Class Library for High Energy Physics: http://proj-clhep.web.cern.ch
- OGL OpenGL Homepage: http://www.opengl.org
- DAWN release: http://geant4.kek.jp/
- ROOT Brun R, Rademakers F 1997 ROOT - An object oriented data analysis framework Institute Nucl. Instr. Meth. A389 81-86 ROOT website: http://root.cern.ch
- libxml website: http://www.libxml.org

1.3 Package Requirements

Table of Contents

Note: some of this information may be out of date for the 8.1 release.

Compiling software usually requires certain system libraries and compilation tools. Furthermore, GATE and Geant4 have various package requirements which have to be met BEFORE installing or compiling. Visit the Package Requirements page for detailed package lists. This list may change frequently, last update: Dec. 31 2015

1.3.1 Ubuntu 18.04.2 LTS (for GATE v8.2 w/ Geant4 10.5 p01)

In Terminal, **type**

```
sudo apt-get update
sudo apt-get install <package_1 here> <package_2 here> ... <package_N here>
```

to install the packages. Replace <package_X here> with the correct packages. For example:

```
sudo apt-get install cmake cmake-curses-gui build-essential libqt4-opengl-dev qt4-
↪qmake libqt4-dev libx11-dev libxmu-dev libxpm-dev libxft-dev
```

The following packages are required for GATE v8.2 with minimal options turned ON (Qt5, OpenGL turned on)

```
cmake-curses-gui libqt5-default libxmu-dev
```

Note: In the event a package does not exist in the Ubuntu repository, you can search for a potential replacement by typing

```
sudo apt-cache search <search_term>
```

1.3.2 Ubuntu 16.04.2 LTS (for GATE v8.0 w/ Geant4 10.3 p01) and Ubuntu 16.04 LTS (for GATE v7.2 w/ Geant4 10.2 p01)

In Terminal, **type**

```
sudo apt-get update
sudo apt-get install <package_1 here> <package_2 here> ... <package_N here>
```

to install the packages. Replace <package_X here> with the correct packages. For example:

```
sudo apt-get install cmake cmake-curses-gui build-essential libqt4-opengl-dev qt4-
↪qmake libqt4-dev libx11-dev libxmu-dev libxpm-dev libxft-dev
```

The following packages are required for GATE v7.2/8.0 with minimal options turned ON (For example, GATE_USE_GPU = OFF)

```
cmake cmake-curses-gui build-essential libqt4-opengl-dev qt4-qmake libqt4-dev libx11-
↪dev libxmu-dev libxpm-dev libxft-dev
```

The following package is required for GATE v8.0 with GATE_USE_OPTICAL turned ON

```
libxml2-dev
```

Note: In the event a package does not exist in the Ubuntu repository, you can search for a potential replacement by typing

```
sudo apt-cache search <search_term>
```

1.3.3 Ubuntu 14.04.3 LTS (for GATE v7.1 w/ Geant4 10.1 p02) and Ubuntu 12.04.5 LTS (for GATE v7.0 w/ Geant4 9.6 p04)

In Terminal, type

```
sudo apt-get update
sudo apt-get install <package_1 here> <package_2 here> ... <package_N here>
```

to install the packages. Replace <package_X here> with the correct packages. For example:

```
sudo apt-get install cmake cmake-curses-gui build-essential libqt4-opengl libqt4-
↪opengl-dev libqt4-core qt4-qmake libqt4-dev libX11-dev libxmu-dev
```

The following packages are required for GATE v7.0/7.1 with minimal options turned ON (For example, GATE_USE_GPU = OFF)

```
cmake cmake-curses-gui build-essential libqt4-opengl libqt4-opengl-dev libqt4-core␣
↪qt4-qmake libqt4-dev libX11-dev libxmu-dev
```

Note: In the event a package does not exist in the Ubuntu repository, you can search for a potential replacement by typing:

```
sudo apt-cache search <search_term>
```

WARNING: For GATE validation, please refer to http://wiki.opengatecollaboration.org/index.php/Validating_Installation

GATE benchmark results here http://www.opengatecollaboration.org/PETBenchmark **appears to be outdated.**

1.3.4 Ubuntu 11.x

Use:

```
sudo apt-get update
sudo apt-get install <packages here>
```

to install the packages. Replace <packages here> with the correct packages.

The following packages are required:

```
build-essential autoconf automake tcl tk g++ libglu1-mesa-dev libxt-dev libxmu-dev␣
↪gfortran libxaw7-dev
libX11-dev libxft-dev libxpm-dev libxt-dev freeglut3 freeglut3-dev x11proto-print-dev␣
↪libmudflap0 po-debconf
libusb-dev libboost-dev libtool libc6-dev graphviz graphviz-dev libxext-dev libpcre3-
↪dev libglew1.5-dev libfftw3-dev
libftgl-dev graphviz-dev libgsl0-dev libkrb5-dev libssl-dev libxml2-dev libldap2-dev␣
↪libavahi-compat-libdnssd-dev
libncurses5-dev libglu1-mesa-dev libcfitsio3-dev libmotif4 libmotif-dev libxml2␣
↪libxml2-dev libqt4-opengl
```

(continues on next page)

(continued from previous page)

```
libqt4-opengl-dev libgl1-mesa-dev libglw1-mesa-dev libxpm4 libxerces-c3-dev libqt4-
↪core qt4-qmake libqt4-dev
libgtkgl2.0-dev libgtkglarea-cil-dev liblablgl-ocaml-dev liblablgl-ocaml libxerces-c-
↪dev libxerces-c3.1
libxmltooling-dev happycoders-libsocket-dev happycoders-libsocket libvtk5.6 libvtk5-
↪dev libglui-dev libfftw3-3 libxt-dev
libfftw3-dev libfftw3-doc
```

1.3.5 Ubuntu 10.04

Use:

```
sudo apt-get update
sudo apt-get install <packages here>
```

to install the packages. Replace <packages here> with the correct packages.

The following packages are required:

```
build-essential autoconf automake tcl tk g++ libglu1-mesa-dev libxt-dev libxmu-dev␣
↪gfortran libxaw7-dev
libX11-dev libxft-dev libxpm-dev libxt-dev freeglut3 freeglut3-dev libglut3 libglut3-
↪dev  x11proto-print-dev
libmudflap0 po-debconf libusb-dev libboost-dev libtool libc6-dev graphviz graphviz-
↪dev libxext-dev libpcre3-dev
libglew1.5-dev libfftw3-dev libftgl-dev graphviz-dev libgsl0-dev libkrb5-dev libssl-
↪dev libxml2-dev libldap2-dev
libavahi-compat-libdnssd-dev libncurses5-dev libglu1-mesa-dev libcfitsio3-dev␣
↪libmotif-dev libxml2 libxml2-dev
libqt4-opengl libqt4-opengl-dev libgl1-mesa-dev libglw1-mesa-dev libxpm4 libxerces-c3-
↪dev libqt4-core qt4-qmake
libqt4-dev libgtkgl2.0-dev libgtkglarea-cil-dev liblablgl-ocaml-dev liblablgl-ocaml␣
↪libxerces-c-dev libxerces-c3.1
libxmltooling-dev libvtk5.2 libvtk5-dev libmotif3 happycoders-libsocket-dev␣
↪happycoders-libsocket libfftw3-3  libxt-dev
libfftw3-dev libfftw3-doc
```

1.3.6 Fedora

Use:

```
sudo yum install <packages here>
```

to install the packages. Replace <packages here> with the correct packages.

The following packages are required:

```
freeglut freeglut-devel gtkglext-devel gtkglext-libs gcc gcc-gfortran gcc-c++ compat-
↪libgfortran-41
libgfortran glibc-kernheaders glibc-headers glibc-devel glibc glibc-static openmotif␣
↪openmotif-devel
libXaw-devel libXaw libXpm-devel libXpm libxml2-devel libxml2 xerces-c-devel qt qt-
↪devel qt-x11 binutils
```

(continues on next page)

(continued from previous page)

```
libX11-devel libXft-devel libXext-devel ncurses-devel pcre-devel mesa-libGL-devel␣
↪mesa-libGL gtkglarea2
gtkglarea2-devel InventorXt InventorXt-devel lesstif lesstif-devel libfftw3-3␣
↪libfftw3-dev libfftw3-doc
```

1.3.7 Scientific Linux 6

Use:

```
sudo yum install <packages here>
```

to install the packages. Replace <packages here> with the correct packages.

The following packages are required:

```
freeglut freeglut-devel gtkglext-devel gtkglext-libs gcc gcc-gfortran gcc-c++ compat-
↪libgfortran-41
libgfortran glibc-kernheaders compat-glibc-headers glibc-headers glibc-devel glibc␣
↪glibc-static
compat-libstdc++ compat-glibc openmotif openmotif-devel libXaw-devel libXaw libXpm-
↪devel libXpm libxml2-devel
libxml2 xerces-c-devel qt qt-devel qt-x11 binutils libX11-devel libXft-devel libXext-
↪devel ncurses-devel
pcre-devel mesa-libGL-devel mesa-libGL libxml2-dev libxml2 gtkglarea2 gtkglarea2-
↪devel fftw-devel fftw2-devel
fftw fftw2
```

1.4 Compiling GATE (V9.0)

Table of Contents

IMPORTANT Before continuing, make sure all required packages are properly installed on your system. See *Package Requirements*

1.4.1 Required dependencies

For compiling GATE V9.0, the required dependencies are

```
Geant4 10.06  # including the embedded CLHEP
ROOT (ROOT 6.xx) # still required, but it may become optional in the future
```

Optional packages

```
CLHEP 2.3.4.3  # by default the one provided in Geant4 is used
ITK (version 5.xx or later)
ECAT
LMF
libTorch # see section below
```

1.4.2 CLHEP

Since the GATE V7.0 release, users can use the CLHEP embedded within each Geant4 distribution. This case, the CMAKE flag GEANT4_USE_SYSTEM_CLHEP, which is OFF by default, must stay OFF.

However, users can use an external CLHEP version (2.3.4.3) by turning flag GEANT4_USE_SYSTEM_CLHEP ON and following the CLHEP installation procedures: http://proj-clhep.web.cern.ch/proj-clhep

1.4.3 Geant4

First, download the Geant sources at this address: http://geant4.web.cern.ch/

During the cmake: QT and OPENGL are optional. We recommand to set GEANT4_INSTALL_DATA.

Finally, update your environment variables file with the following command lines:

- bash or zsh:

  source /PATH_TO/geant4.10.06-install/bin/geant4.sh

- [t]csh:

  source /PATH_TO/geant4.10.06-install/bin/geant4.csh

For details, read the official GEANT4 installation procedure.

1.4.4 ROOT

You can download ROOT at the following address: https://root.cern.ch/downloading-root

It is recommended to install directly the ROOT binaries. Users have to compile the ROOT sources only if necessary.

Depending on your environment, you need to source ROOT as follow (this final step is summarized in the item 5 of this documentation):

- bash or zsh:

```
source /PATH_TO/root_v6.xx/bin/thisroot.sh
```

- [t]csh:

```
source /PATH_TO/root_v6.xx/bin/thisroot.csh
```

1.4.5 libtorch (optional)

The goal is here to make Gate use the torch library, an open source machine learning framework : https://pytorch.org

Pytorch is usually used via a Python module, but here we need an additional library named 'libtorch' that will be used by Gate during compilation.

To download 'libtorch', go to https://pytorch.org at the section QUICK START LOCALLY, and select PyTorch Build stable, Your OS, Package libtorch, Language C++, your CUDA version if you have CUDA installed on your computer or None if you want to use only your CPU (Note: GATE is currently using only CPU with libtorch). Then download the zip archive. For Linux platform you can choose between Pre-cxx11 ABI or cxx11 ABI versions according to your gcc version. If you cannot compile (or link) Gate with the former version, try the latter version. Finally unziped somewhere on your disk. No compilation required here.

Then, during the installation of Gate (next section) use the following option to set the path to libtorch

```
GATE_USE_TORCH      ON
Torch_DIR           /home/YOURNAME/libtorch-1.3.0/share/cmake/Torch
```

In some configuration, the following path should also be set

```
CUDNN_INCLUDE_DIR   /home/YOURNAME/cuda/include
CUDNN_LIBRARY       /home/YOURNAME/cuda/lib64/libcudnn.so
```

We recommend you to use libtorch version 1.4.0 but if you want to use a version greater than 1.7.0, check https://github.com/OpenGATE/Gate/pull/424

1.4.6 GATE V9.0

First, download the GATE sources at this address: https://github.com/OpenGATE/Gate/archive/v9.0.zip Unzip the downloaded file:

```
unzip Gate-9.0.zip
```

Alternatively, if you are familiar with git, then instead of downloading and extracting the tar file, you can also clone the sources from github and check out the *v9.0* release tag.

git clone https://github.com/OpenGATE/Gate.git Gate cd Gate git checkout v9.0

Create two directories to build and install GATE:

```
mkdir gate_v9.0-build
mkdir gate_v9.0-install
```

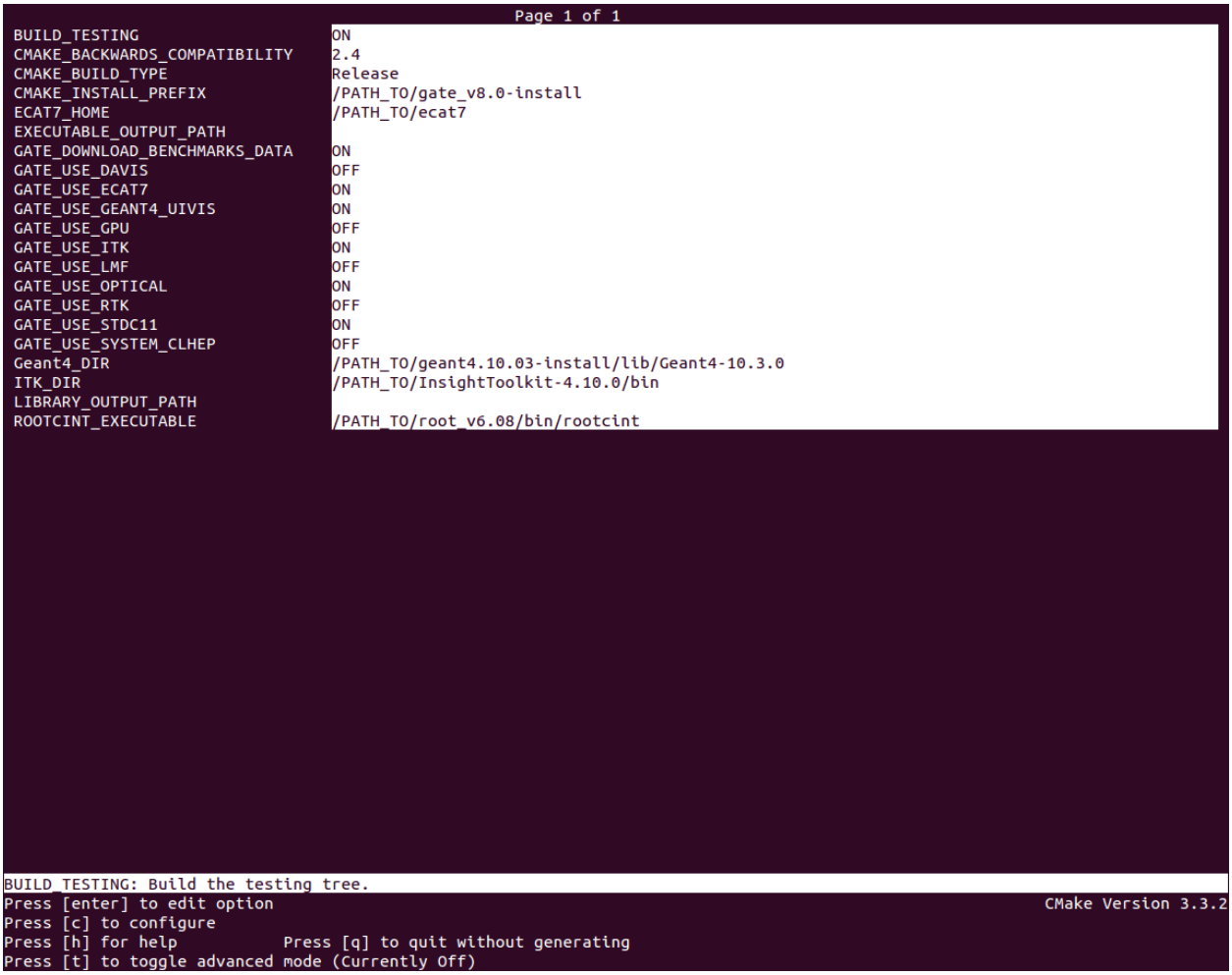
Move into the GATE build directory:

```
cd gate_v9.0-build
```

Run ccmake as follows:

```
ccmake ../Gate-9.0
```

You need to change the *CMAKE_INSTALL_PREFIX*, it should be set to the install directory (defined above). The default given by CMake is */usr/local*; if you have root/sudo permissions on your machine then it's possible to install Gate there, but it's not recommended, especially if you need to work with more than one version of Gate (for instance, if you want to do development on Gate, or if you want to verify that a new release is compatible with the old release that you have been using). You should get something like this (the screen shot is taken from the 8.0 release, the only difference is the version number):



Warning Information about following environment variables:

```
BUILD_TESTING                       OFF: by default, set to ON if you want to perform␣
↪build testing
GATE_DOWNLOAD_BENCHMARKS_DATA       OFF: by default, set to ON if you want to download␣
↪the benchmark data to run validation tests (with the command *make test*)
GATE_USE_ECAT7                      OFF: by default, set to ON if you want to use this␣
↪library
GATE_USE_GPU                        OFF: by default, set to ON if you want to use GPU␣
↪modules
GATE_USE_ITK                        OFF: by default, set to ON if you want to access␣
↪DICOM reader and thermal therapy capabilities
GATE_USE_LMF                        OFF: by default, set to ON if you want to use this␣
↪library
GATE_USE_OPTICAL                    OFF: by default, set to ON if you want to perform␣
↪simulation for optical imaging applications
```

(continues on next page)

(continued from previous page)

```
GATE_USE_RTK                        OFF: by default, set to ON if you want to use this␣
↪toolkit
GATE_USE_STDC11                     ON : by default, set to OFF if you want to use␣
↪another standard for the C programming language (advanced users)
GATE_USE_DAVIS                      OFF: by default, set to ON if you want to use the␣
↪Davis LUT model
GEANT4_USE_SYSTEM_CLHEP             OFF: by default, set to ON if you want to use an␣
↪external CLHEP version
```

As it was the case for Geant4, press 'c' to configure (you may need to do this multiple times) and then 'g' to generate the compilation environment.

Finally:

```
make -jN (N is the number of processor(s) in your PC)
make install
```

Finally, update your environment variables file with the following command lines: (this part is summarized in the item 5 of this document)

- bash or zsh:

  export PATH=/PATH_TO/gate_v9.0-install/bin:$PATH

- [t]csh

  setenv PATH /PATH_TO/gate_v9.0-install/bin:${PATH}

1.4.7 Environment configuration and starting GATE

We highly recommended to create a *gate_env.sh* (or *gate_env.csh* if you are a [t]csh user) file to set up all environment variables which are mandatory to perform a full GATE simulation, and save this file in the bin directory of your Gate installation. (In future releases of Gate we hope to provide such an environment setup file automatically.)

This file should be defined as follows:

- bash or zsh:

```
source /PATH_TO/root_v6.XX/bin/thisroot.sh
source /PATH_TO/geant4.10.06-install/bin/geant4.sh
export PATH=$PATH:/PATH_TO/gate_v9.0-install/bin
# the following lines only if you are using an external CLHEP library (and␣
↪similar for ITK, if you enabled it):
export PATH=$PATH:/PATH_TO/2.3.4.3/CLHEP/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/PATH_TO/2.3.4.3/CLHEP/lib
```

- csh or tcsh:

```
source /PATH_TO/root_v6.XX/bin/thisroot.csh
source /PATH_TO/geant4.10.06-install/bin/geant4.csh
setenv PATH ${PATH}:$/PATH_TO/gate_v9.0-install/bin
# the following lines only if you are using an external CLHEP library (and␣
↪similar for ITK, if you enabled it):
setenv PATH ${PATH}:/PATH_TO/2.3.4.3/CLHEP/bin
setenv LD_LIBRARY_PATH ${LD_LIBRARY_PATH}:/PATH_TO/2.3.4.3/CLHEP/lib
```

Save this file in */PATH_TO/gate_v8.2-install/bin*. Finally, before to start a GATE session:

```
source /PATH_TO/gate_v9.0-install/bin/gate_env.sh
```

In order to save typing, you may want to define an alias for that: include the following line in your *$HOME/.bashrc* or *$HOME/.bash_aliases* file:

```
alias gate90='source /PATH_TO/gate_v9.0-install/bin/gate_env.sh'
```

(For csh and tcsh the syntax is different but the idea is the same.)

With your shell environment properly set up, you should be able to run Gate. To try it out, just start it without any arguments:

```
Gate
```

! If you are using the Qt interface on non-English locales then you must force Qt to use a locale with a dot for the decimal separator:

```
LC_NUMERIC=C Gate --qt
```

1.4.8 ITK

See: https://itk.org and follow the instructions.

Here are some additional cmake options:

```
ccmake -DITK_USE_REVIEW=ON ..
```

You will obtain the following screen and you need to configure the different options as follows:

```
BUILD_EXAMPLES                  OFF
BUILD_TESTING                   OFF
ITKV3_COMPATIBILITY             OFF
ITK_BUILD_DEFAULT_MODULES       ON
ITK_WRAP_PYTHON                 OFF
```

1.4.9 ECAT7

First, create and enter an ecat7 sub-directory:

```
mkdir /PATH_TO/ecat7
cd /PATH_TO/ecat7
```

Download the ECAT library sources at this address:

http://www.opengatecollaboration.org/ECAT

Unzip and untar the downloaded file:

```
tar -xzf ecat.tar.gz
```

WARNING: if you want to use ECAT7 output module, don't forget to set CMake option GATE_USE_ECAT7 to ON and to provide the path to ECAT7 source directory (i.e /PATH_TO/ecat7)

Copy the right Makefile.<os> to Makefile. If Makefile exists this step is not necessary:

```
cp Makefile.unix Makefile
```

Compile:

```
make
```

This will build the library

Go to the utils directory Copy the right Makefile.<os> to Makefile if Makefile exists this step is not necessary:

```
cp Makefile.unix Makefile
```

Compile (do not use make -j4 !!!):

```
make
```

This will create some utility programs

After compilation, create the following folder: include/:

```
mkdir /PATH_TO/ecat7/include
```

In this folder copy all *.h files:

```
cp *.h /PATH_TO/ecat7/include
```

Check that the file libecat.a is in lib/. If it isn't copy it there:

```
mkdir lib
cp libecat.a lib/
```

1.4.10 LMF 3

(Disclaimer: the LMF code and build instructions are provided "as is", we do not give an warranty of it's correctness or usefulness for any purpose, and do not officially support LMF.)

Enter the source directory:

```
/PATH_TO/lmf_3_0
```

Configure lmf

./configure

Make sure that you have ROOT in your environment. If this is not the case yet, then run *source /PATH/TO/ROOT/bin/thisroot.sh* (change the "path to root" according to your local ROOT installation). Then edit the *makefile* to inform the modern compiler on your machine that the code is antique:

```
obj/%.o : src/%.c
        gcc $(CFLAGS) -std=c99 -c -o $@ $<

obj/outputRootMgr.o : src/outputRootMgr.cc
        gcc $(CFLAGS) $(ROOTCFLAGS) -std=c++98 -c -o $@ $<

obj/%.o : src/%.cc
        gcc $(CFLAGS) -std=c++98 -c -o $@ $<
```

(And be careful, it's important that the whitespace in front of each *gcc* is a TAB; if you use normal spaces then it won't work!)

Compile (do not use make -j4 !!!):

```
make clean
make
```

If it does not exist, after compilation create the following folder: includes:

```
mkdir /PATH_TO/lmf_3_0/includes
```

In this folder copy all *.h files, if they aren't in there already:

```
cp *.h /PATH_TO/lmf_3_0/includes
```

Check that the file libLMF.a is in lib/ If it isn't copy it there

1.4.11 Installation of cluster tools

jobsplitter

Go to /PATH_TO/gate_v9.0/cluster_tools/jobsplitter:

```
cd /PATH_TO/gate_v9.0/cluster_tools/jobsplitter
```

Make sure ROOT and Geant4 environment variables are set:

```
source /PATH_TO/root_v6.XX/bin/thisroot.sh
source /PATH_TO/geant4.10.06-install/bin/geant4.sh
```

Compile:

```
make
```

Copy the gjs executable file to the correct place:

```
cp /PATH_TO/gate_v9.0/cluster_tools/jobsplitter/gjs /PATH_TO/gate_v9.0-install/bin
```

filemerger

Go to /PATH_TO/gate_v9.0/cluster_tools/filemerger Make sure ROOT and Geant4 environment variables are set:

```
source /PATH_TO/root_v6.XX/bin/thisroot.sh
source /PATH_TO/geant4.10.06-install/bin/geant4.sh
```

Compile:

```
make
```

Copy the gjs executable file to the correct place:

```
cp /PATH_TO/gate_v9.0/cluster_tools/filemerger/gjm /PATH_TO/gate_v9.0-install/bin
```

1.5 Validating Installation

Table of Contents

1.5.1 Benchmarks tests

Warning: benchmark tests are only provided for Gate compiled with Geant 10.3 distribution

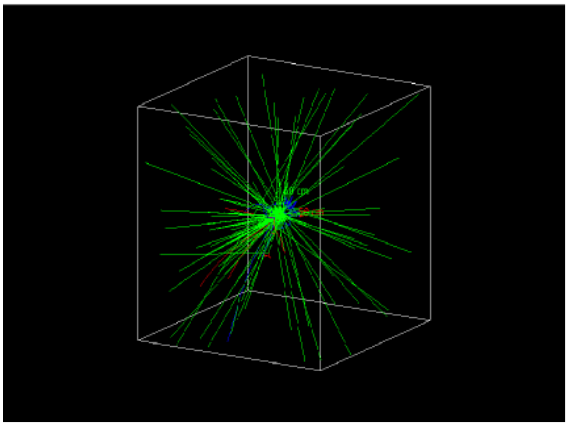
In order to ensure regular updates of the revised Gate software modules to be developed and ensure a faster public release, we have included in Gate V8.0 a new benchmarking system. It provides 3 benchmark tests dedicated to radiation therapy applications and 4 to imaging applications.

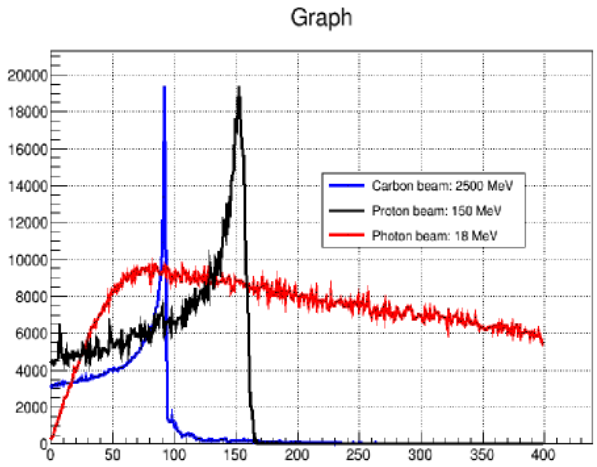
Radiation therapy applications benchmark tests

All these tests are located in benchmarks/benchRT folder.

They all consists of a water box through which a gamma (*gamma.mac*), a proton (*proton.mac*), and a carbon (*carbon.mac*) beam is launched.

They all have a less than 10-seconde duration.



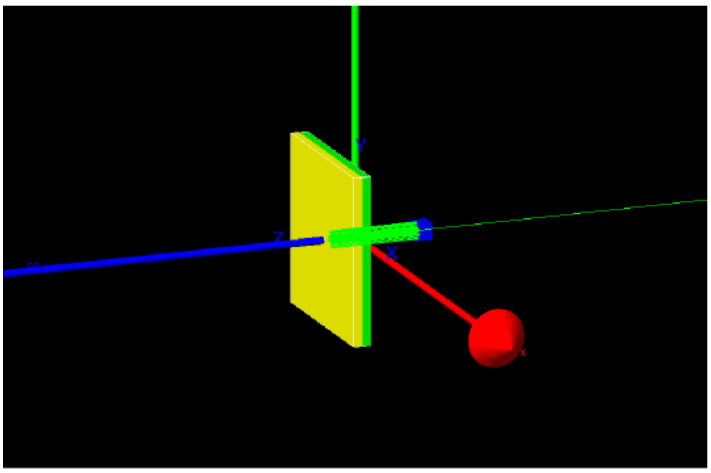


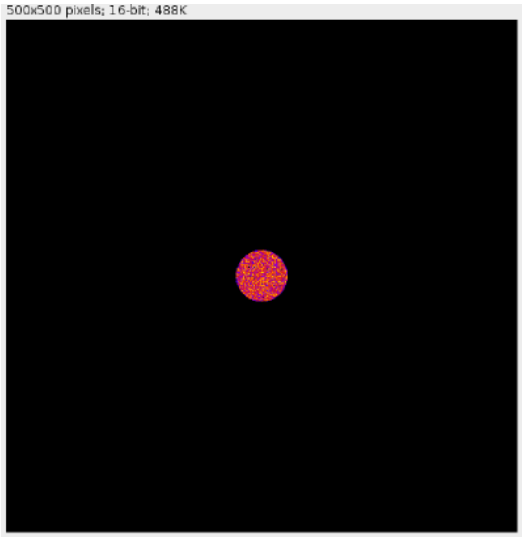
Imaging applications benchmark tests

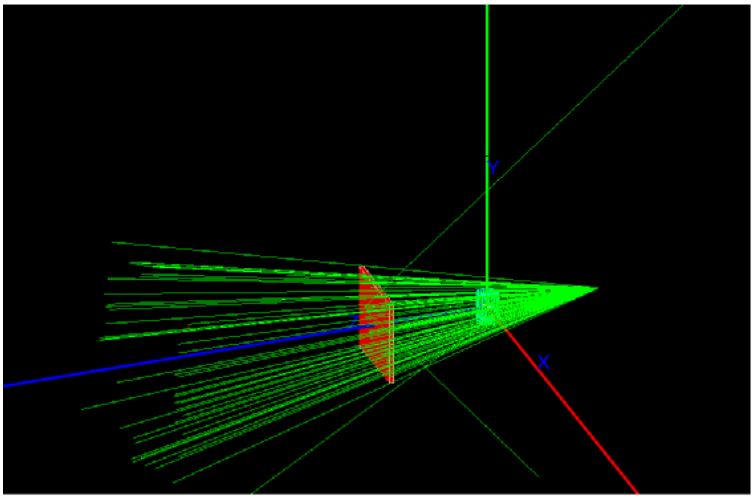
All these tests are located in benchmarks/benchImaging folder.

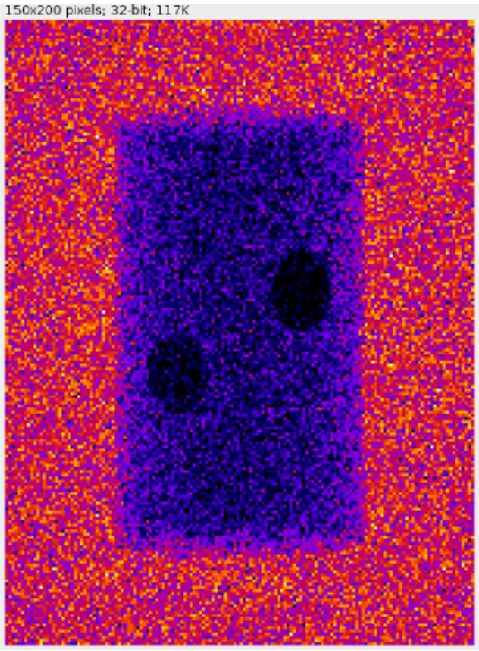
The **optical** test mimics a bioluminescence experiment using optical photon as source distribution (*optical.mac*)

The **CT** test consists of a CTScanner system together with an X-ray source distribution defined using a histogram (*ct.mac*)

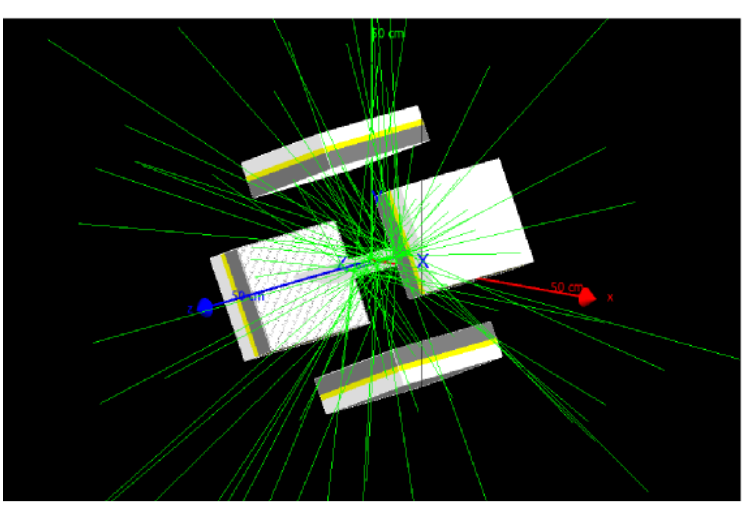


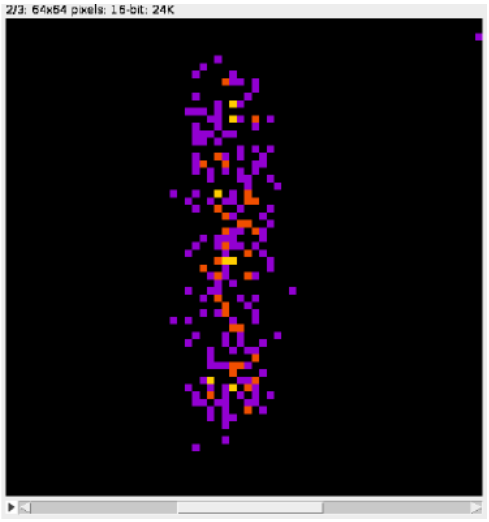

500x500 pixels; 16-bit; 488K



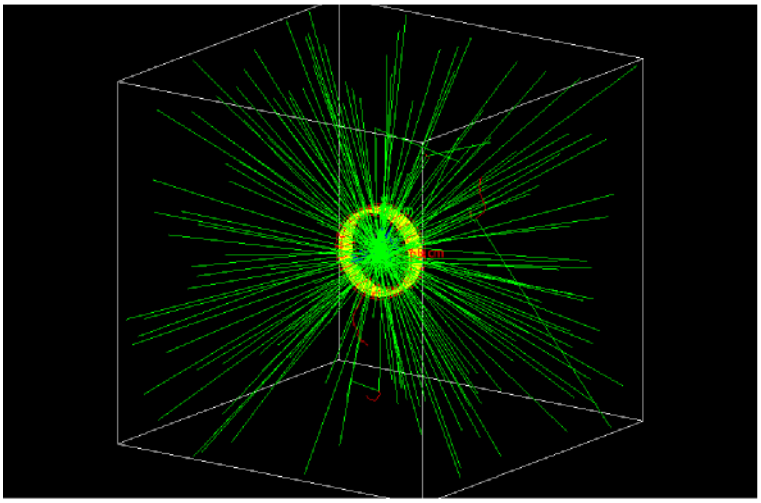

150x200 pixels; 32-bit; 117K

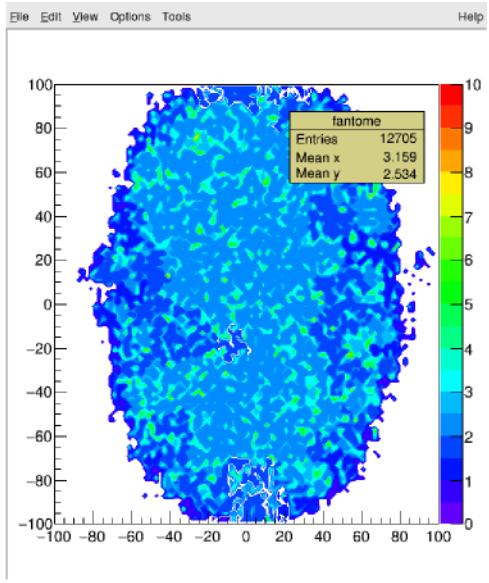
The **SPECT** test consists of a SPECTHead system in orbiting movement together with a 140 keV-gamma ray source distribution (*spect.mac*)





The **PET** test consists of a ECAT system together with a voxelized phantom of human brain and a source distribution of 18-F (*pet.mac*)





They all have a less than 1-minute duration.

How to run tests

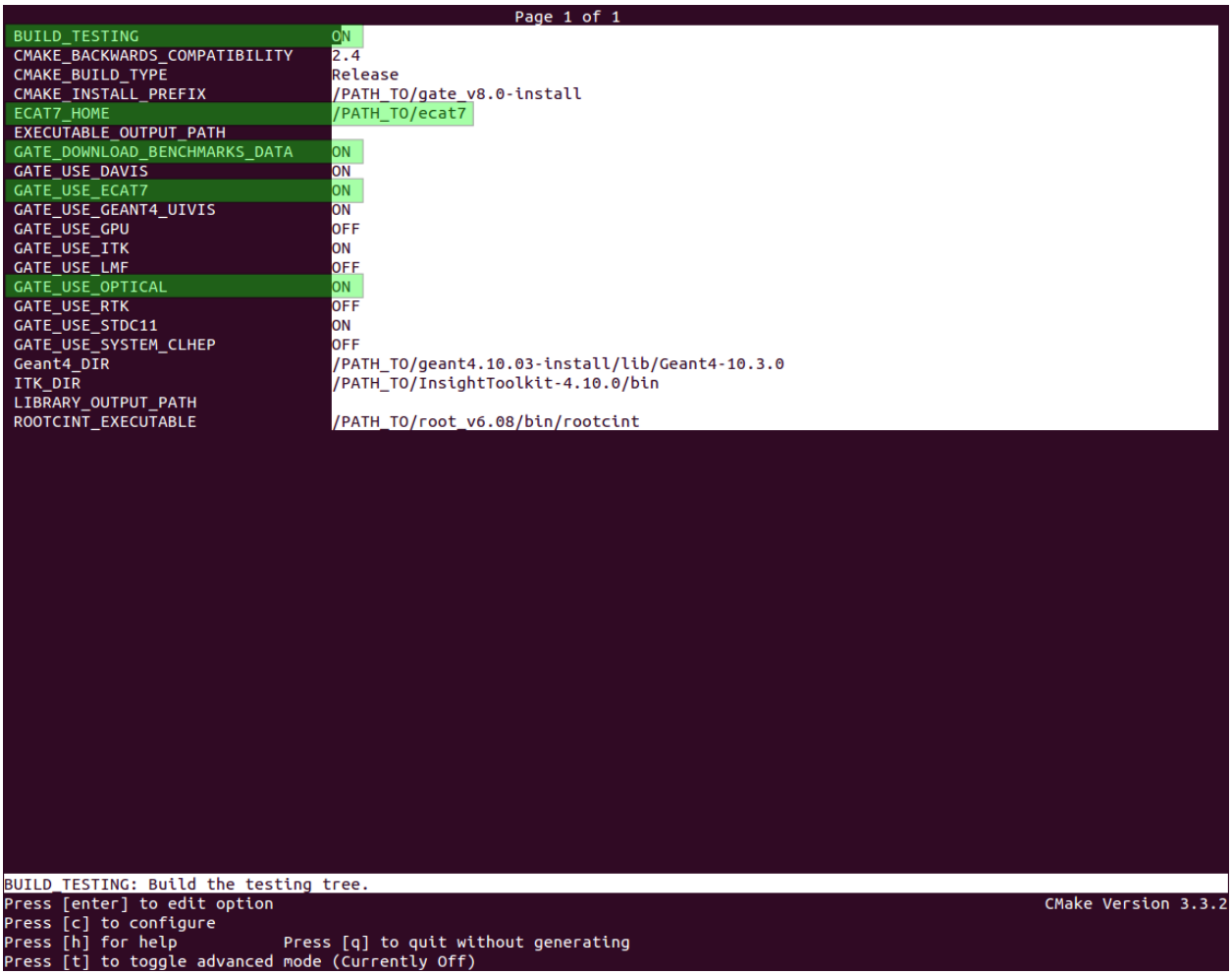
All benchmark tests can be run with 3 different methods:

1) **Automatically**, every night in the dashboard See: http://my.cdash.org/index.php?project=GATE

2) **Manually**, by executing *make test* after completing the build process

Be careful, it requires some prior adjustments in CMake configuration

After *make* and *make install*, you just have to type *make test* to automatically run all the tests


Page 1 of 1
BUILD_TESTING ON
CMAKE_BACKWARDS_COMPATIBILITY 2.4
CMAKE_BUILD_TYPE Release
CMAKE_INSTALL_PREFIX /PATH_TO/gate_v8.0-install
ECAT7_HOME /PATH_TO/ecat7
EXECUTABLE_OUTPUT_PATH
GATE_DOWNLOAD_BENCHMARKS_DATA ON
GATE_USE_DAVIS ON
GATE_USE_ECAT7 ON
GATE_USE_GEANT4_UIVIS ON
GATE_USE_GPU OFF
GATE_USE_ITK ON
GATE_USE_LMF OFF
GATE_USE_OPTICAL ON
GATE_USE_RTK OFF
GATE_USE_STDC11 ON
GATE_USE_SYSTEM_CLHEP OFF
Geant4_DIR /PATH_TO/geant4.10.03-install/lib/Geant4-10.3.0
ITK_DIR /PATH_TO/InsightToolkit-4.10.0/bin
LIBRARY_OUTPUT_PATH
ROOTCINT_EXECUTABLE /PATH_TO/root_v6.08/bin/rootcint
BUILD_TESTING: Build the testing tree.
Press [enter] to edit option
CMake Version 3.3.2
Press [c] to configure
Press [h] for help
Press [q] to quit without generating
Press [t] to toggle advanced mode (Currently Off)

At the end of make test process, you should see:

```
Running tests...
Test project /PATH_TO/gate_v8.0-build
    Start 1: benchRT_gamma
1/7 Test #1: benchRT_gamma .....................   Passed   15.04 sec
    Start 2: benchRT_proton
2/7 Test #2: benchRT_proton ....................   Passed    1.55 sec
    Start 3: benchRT_carbon
3/7 Test #3: benchRT_carbon ....................   Passed    6.11 sec
    Start 4: benchImaging_ct
4/7 Test #4: benchImaging_ct ...................   Passed   37.71 sec
    Start 5: benchImaging_optical
5/7 Test #5: benchImaging_optical ..............   Passed    4.09 sec
    Start 6: benchImaging_spect
6/7 Test #6: benchImaging_spect ................   Passed   27.70 sec
    Start 7: benchImaging_pet
7/7 Test #7: benchImaging_pet ..................   Passed   14.18 sec
100% tests passed, 0 tests failed out of 7
Total Test time (real) = 106.38 sec
```

3) **Manually**, by running the same shell script for each test In /PATH_TO/benchmarks folder, type:

```
./gate_run_test.sh <folder>  <testname>
```

For instance:

```
./gate_run_test.sh benchRT gamma
```

This script will do the following:

- Sanity checks
- Execute Gate, generate a log
- Get reference results (downloaded by CMake)
- Perform some diff commands
- Return result (success/failure)

This script needs data and macro files (in /data and /mac folders, respectively) + parameters : *testname.txt*

- Contains the list of files to be compared
- Tools to perfom the comparisons: *diff*, *diff_stat*, *diff_dose* ...

If everything went well, you should see at the end of each test:

```
-----------------------
exit_status_final is:
0
-----------------------
(Gate output in gate_simulation_log.txt)
```

1.6 GateRT

Some examples could be found here : https://davidsarrut.pages.in2p3.fr/gate-exercices-site/

Disclaimer: *the tools dedicated to radiation therapy simulations provided in this GATE release are provided "as is" and on an "as available" basis without any representation or endorsement made and without warranty of any kind.*

Location: The examples are located in the source code into to the folder : *Gate/examples/example_Radiotherapy/*.

Proton therapy examples:

- Example 4 : Beam optics simulation in vacuum for a pencil beam + depth-dose profile in water. A root macro is provided to analysis the produced phase space files (PhS-Analysis.C).
- Example 5 : Treatment plan simulation of proton active scanning beam delivery (TPSPencilBeam source). A root macro is provided to analysis the produced phase space files (PhS-Analysis.C).
- Example 6 : Example of proton pencil beam in heterogeneous phantom (water, bones, Lung) with Pencil Beam Scanning source: comparison between dose to water and dose to dose to medium.

Carbon ion therapy examples:

- Example 1 : Example of Carbon beam in water tank or in patient CT image. Output is a 3D dose distribution map (with associated statistical uncertainty) and map of produced C11.

Photon/electron therapy examples:

- Example 2 : Example of photon beam in patient CT image. Output is a 3D dose distribution map (with associated uncertainty). Two different navigators are tested NestedParameterized and Regionalized, with two number of materials). (This example is similar to the example1 presented on the wiki web site of Gate V6.1)
- Example 3 : Example of photon beam in patient CT image with IMRT irradiation. 100 slices with different MLC positions. (This example is similar to the example2 presented on the wiki web site of Gate V6.1)
- Example 7 : Example to use repeater/mover and both at the same time. (This example is similar to the example5 presented on the wiki web site of Gate V6.1)
- Example 8 : Photon beam from a Linac into a box with water/alu/lung. See Figure4 from [Jan et al PMB 2011].
- Example 9 : Electron beam from a Linac into a box with water/alu/lung. See Figure5 from [Jan et al PMB 2011].

Radiography examples:

- Example 10 : Radiography of a thorax phantom. Outputs are 3D dose distribution maps computed with the classical method and the accelerated (TLE) method.

1.7 Enabling LUT Davis Model

Table of Contents

1.7.1 Manually Modifying Geant4

(Necessary before Geant4 release Summer 2017)

In order to use the Davis LUT model Geant4 has to be extended. This documentation provides detailed step-by-step compiling instructions.

1) Follow compiling instructions here: *libtorch (optional)* stop before running ccmake, proceed with step 2.

2) Get the modified code and Look-up-Tables: 5 Geant4 classes to be replaced in Geant4 10.2 or 10.3, respectively and 18 LUTs provided as a tar.gz archive

3) **Replace header files in your local Geant4 directory**

    - **Go to /PATH_TO_yourGEANT4Directory/source/materials/include**
        - Replace the "G4SurfaceProperty.hh" file with the provided file
        - Replace the "G4OpticalSurface.hh" file with the provided file
    - **Go to /PATH_TO_yourGEANT4Directory/source/materials/src**
        - Replace the "G4OpticalSurface.cc" file with the provided file
    - **Go to /PATH_TO_yourGEANT4Directory/source/processes/optical/include**
        - Replace the "G4OpBoundaryProcess.hh" file with the provided file
    - **Go to /PATH_TO_yourGEANT4Directory/source/processes/optical/src**
        - Replace the "G4OpBoundaryProcess.cc" file with the provided file

4) **Open "G4OpticalSurface.hh" in /PATH_TO_yourGEANT4Directory/source/materials/include**

    - In line 270 set the path of the Davis_LUTs-folder that was downloaded in step 2
    - *G4String PathToLUT="/home/. . . /Davis_LUTs";*

5) Go on with ccmake process in of step 1. and finish as documented.

6) To enable modifications of Geant4 in GATE: Follow compiling instructions: *libtorch (optional)*

7) In the ccmake process set CMake options GATE_USE_OPTICAL and GATE_USE_DAVIS to ON. (Run provided example to validate installation for LUT Davis model and compare to provided output file.)

CHAPTER 2

General concept

2.1 Getting Started

Table of Contents

This section is an overview of the main steps one must go through to perform a simulation using Gate. It is presented in the form of a simple example that the user is encouraged to try out, while reading this section. A more detailed description of the different steps is given in the following sections of this user's guide. The use of Gate does not require any C++ programming, thanks to a dedicated scripting mechanism that extends the native command interpreter of

Geant4. This interface allows the user to run Gate programs using command scripts only. The goal of this first section is to give a brief description of the user interface and to provide understanding of the basic principles of Gate by going through the different steps of a simulation.

2.1.1 Simulation architecture for imaging applications

In each simulation, the user has to:

1. define the scanner geometry
2. define the phantom geometry
3. set up the physics processes
4. initialize the simulation:

```
/gate/run/initialize
```

5. set up the detector model
6. define the source(s)
7. specify the data output format
8. start the acquisition

Steps 1) to 4) concern the initialization of the simulation. Following the initialization, the geometry can no longer be changed.

2.1.2 Simulation architecture for dosimetry and radiotherapy applications

In each simulation, the user has to:

1. define the beam geometry
2. define the phantom geometry
3. specify the output (actor concept for dose map etc. . .)
4. set up the physics processes
5. initialize the simulation:

```
/gate/run/initialize
```

6. define the source(s)
7. start the simulation with the following command lines:

```
/gate/application/setTotalNumberOfPrimaries [particle_number]
/gate/application/start
```

2.1.3 The user interface: a macro language

Gate, just as GEANT4, is a program in which the user interface is based on scripts. To perform actions, the user must either enter commands in interactive mode, or build up macro files containing an ordered collection of commands.

Each command performs a particular function, and may require one or more parameters. The Gate commands are organized following a tree structure, with respect to the function they represent. For example, all geometry-control commands start with *geometry*, and they will all be found under the *geometry* branch of the tree structure.

When Gate is run, the **Idle>** prompt appears. At this stage the command interpreter is active; i.e. all the Gate commands entered will be interpreted and processed on-line. All functions in Gate can be accessed to using command lines. The geometry of the system, the description of the radioactive source(s), the physical interactions considered, etc., can be parameterized using command lines, which are translated to the Gate kernel by the command interpreter. In this way, the simulation is defined one step at a time, and the actual construction of the geometry and definition of the simulation can be seen on-line. If the effect is not as expected, the user can decide to re-adjust the desired parameter by re-entering the appropriate command on-line. Although entering commands step by step can be useful when the user is experimenting with the software or when he/she is not sure how to construct the geometry, there remains a need for storing the set of commands that led to a successful simulation.

Macros are ASCII files (with '.mac' extension) in which each line contains a command or a comment. Commands are GEANT4 or Gate scripted commands; comments start with the character '#'. Macros can be executed from within the command interpreter in Gate, or by passing it as a command-line parameter to Gate, or by calling it from another macro. A macro or set of macros must include all commands describing the different components of a simulation in the right order. Usually these components are visualization, definitions of volumes (geometry), systems, digitizer, physics, initialization, source, output and start. These steps are described in the next sections. A single simulation may be split into several macros, for instance one for the geometry, one for the physics, etc. Usually, there is a master macro which calls the more specific macros. Splitting macros allows the user to re-use one or more of these macros in several other simulations, and/or to organize the set of all commands. Examples of complete macros can be found on the web site referenced above. To execute a macro (mymacro.mac in this example) from the Linux prompt, just type:

```
Gate mymacro.mac
```

To execute a macro from inside the Gate environment, type after the "Idle>" prompt:

```
Idle>/control/execute mymacro.mac
```

And finally, to execute a macro from inside another macro, simply write in the master macro:

```
/control/execute mymacro.mac
```

In the following sections, the main steps to perform a simulation for imaging applications using Gate are presented in details. To try out this example, the user can run Gate and execute all the proposed commands, line by line.

2.1.4 Step 1: Defining a scanner geometry

The user needs to define the geometry of the simulation based on volumes. All volumes are linked together following a tree structure where each branch represents a volume. Each volume is characterized by shape, size, position, and material composition. The default material assigned to a new volume is Air. The list of available materials is defined in the GateMaterials.db file. (See *Materials*). The location of the material database needs to be specified with the following command:

```
/gate/geometry/setMaterialDatabase MyMaterialDatabase.db
```

The base of the tree is represented by the world volume (Fig. 2.1) which sets the experimental framework of the simulation. All Gate commands related to the construction of the geometry are described in detail in *Defining a geometry*. The world volume is a box centered at the origin. It can be of any size and has to be large enough to include the entire simulation geometry. The tracking of any particle stops when it escapes from the world volume. The example given here simulates a system that fits into a box of 40 x 40 x 40 cm3. Thus, the world volume may be defined as follows:

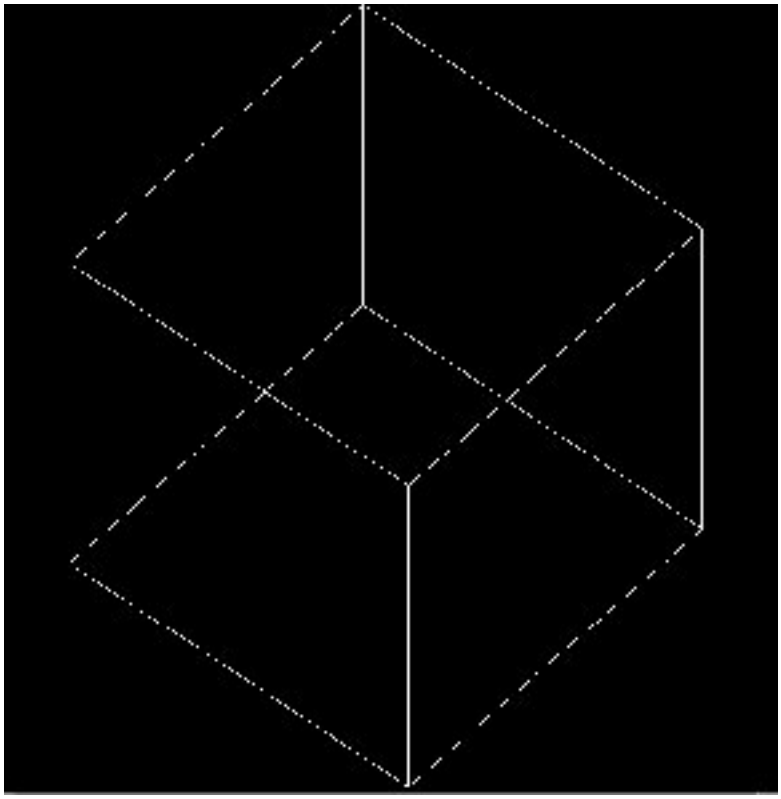


Fig. 2.1: World volume.

```
# W O R L D
/gate/world/geometry/setXLength 40. cm
/gate/world/geometry/setYLength 40. cm
/gate/world/geometry/setZLength 40. cm
```

The world contains one or more sub volumes referred to as daughter volumes:

```
/gate/world/daughters/name vol_name
```

The name vol_name of the first daughter of the world has a specific meaning and name. It specifies the type of scanner to be simulated. *Defining a system* gives the specifics of each type of scanner, also called system. In the current example, the system is a CylindricalPET system. This system assumes that the scanner is based on a cylindrical configuration (Fig. 2.2) of blocks, each block containing a set of crystals:

```
#  S Y S T E M
/gate/world/daughters/name cylindricalPET
/gate/world/daughters/insert cylinder
/gate/cylindricalPET/setMaterial Water
/gate/cylindricalPET/geometry/setRmax 100 mm
/gate/cylindricalPET/geometry/setRmin 86 mm
/gate/cylindricalPET/geometry/setHeight 18 mm
/gate/cylindricalPET/vis/forceWireframe
/vis/viewer/zoom 3
```

These seven command lines describe the global geometry of the scanner. The shape of the scanner is a cylinder filled with water with an external radius of 100 mm and an internal radius of 86 mm. The length of the cylinder is 18 mm. The last command line sets the visualization as wireframe.

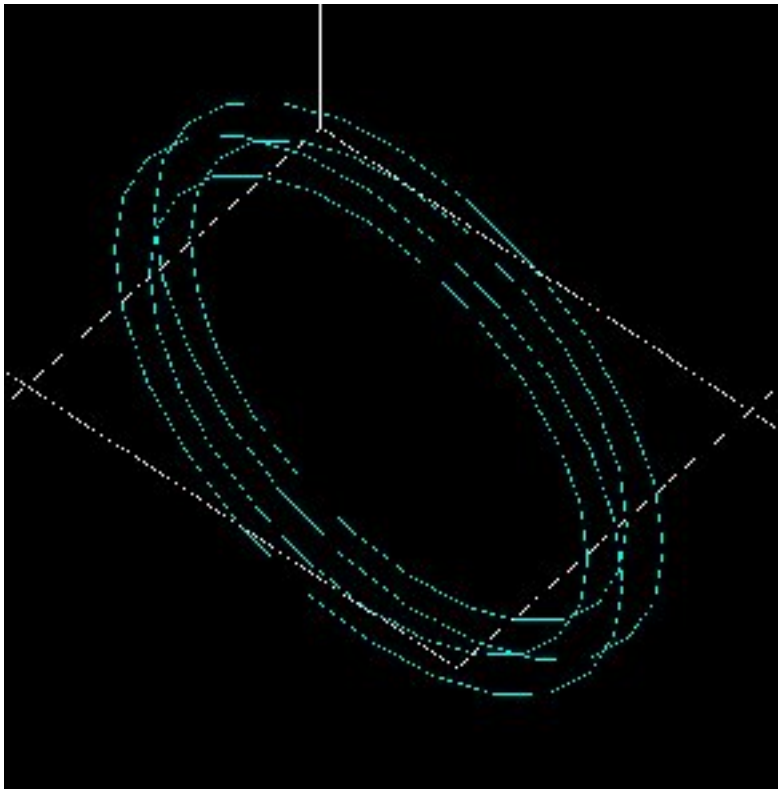


Fig. 2.2: Cylindrical scanner

You may see the following message when creating the geometry:

```
G4PhysicalVolumeModel::Validate() called.
Volume of the same name and copy number ("world_phys", copy 0) still exists and is␣
↪being used.
WARNING: This does not necessarily guarantee it's the same
volume you originally specified in /vis/scene/add/.
```

This message is normal and you can safely ignore it.

At any time, the user can list all the possible commands. For example, the command line for listing the visualization commands is:

```
Idle> ls /gate/cylindricalPET/vis/
```

Let's assume that the scanner is made of 30 blocks (box1), each block containing 8 times 8 LSO crystals (box2).

The following command lines describe this scanner (see *Defining a geometry* to find a detailed explanation of these commands). First, the geometry of each block needs to be defined as the daughter of the system (here cylindricalPET system):

```
# FIRST LEVEL OF THE SYSTEM
/gate/cylindricalPET/daughters/name box1
/gate/cylindricalPET/daughters/insert box
/gate/box1/placement/setTranslation 91. 0 0 mm
/gate/box1/geometry/setXLength 10. mm
/gate/box1/geometry/setYLength 17.75 mm
/gate/box1/geometry/setZLength 17.75 mm
```

(continues on next page)

(continued from previous page)

```
/gate/box1/setMaterial Water
/gate/box1/vis/setColor yellow
/gate/box1/vis/forceWireframe
```

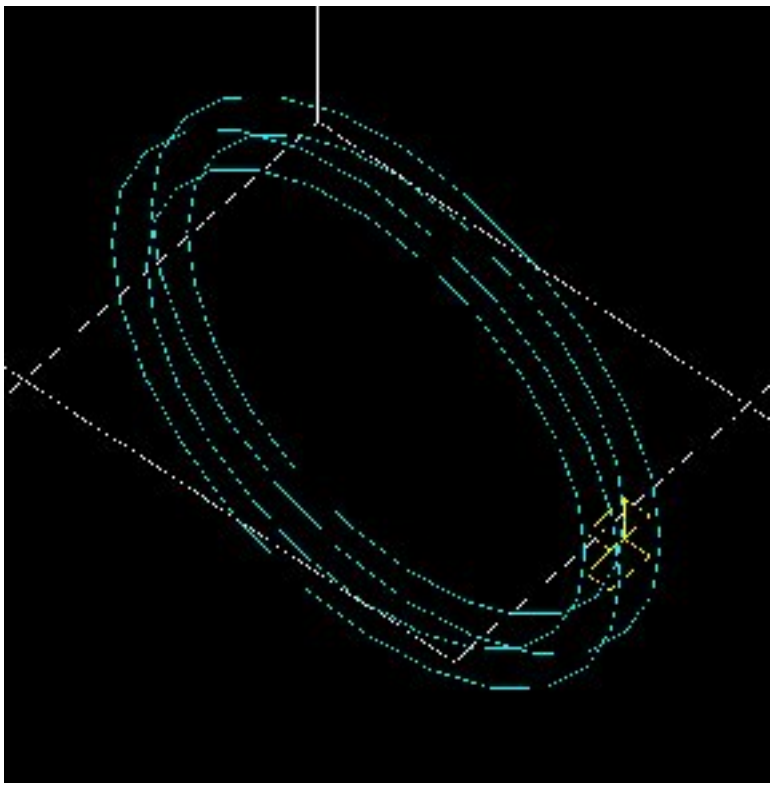


Fig. 2.3: First level of the scanner

Once the block is created (Fig. 2.3), the crystal can be defined as a daughter of the block (Fig. 2.4)

The zoom command line in the script allows the user to zoom the geometry and the panTo command translates the viewer window in 60 mm in horizontal and 40 mm in vertical directions (the default is the origin of the world (0,0,0)).

To obtain the complete matrix of crystals, the volume box2 needs to be repeated in the Y and Z directions (Fig. 2.5). To obtain the complete ring detector, the original block is repeated 30 times (Fig. 2.6):

```
# C R Y S T A L
/gate/box1/daughters/name box2
/gate/box1/daughters/insert box
/gate/box2/geometry/setXLength 10. mm
/gate/box2/geometry/setYLength 2. mm
/gate/box2/geometry/setZLength 2. mm
/gate/box2/setMaterial LSO
/gate/box2/vis/setColor red
/gate/box2/vis/forceWireframe

# Z O O M
/vis/viewer/zoom 4
/vis/viewer/panTo 60 -40 mm

# R E P E A T    C R Y S T A L
```

(continues on next page)

(continued from previous page)

```
/gate/box2/repeaters/insert cubicArray
/gate/box2/cubicArray/setRepeatNumberX 1
/gate/box2/cubicArray/setRepeatNumberY 8
/gate/box2/cubicArray/setRepeatNumberZ 8
/gate/box2/cubicArray/setRepeatVector 0. 2.25 2.25 mm
```

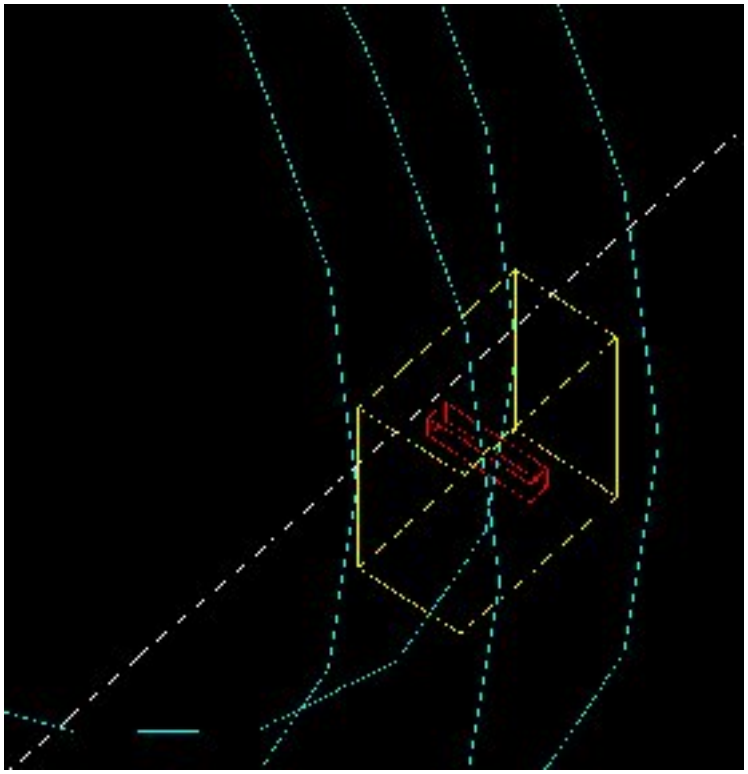


Fig. 2.4: Crystal, daughter of the block

The geometry of this simple PET scanner has now been specified. The next step is to connect this geometry to the system in order to store data from particle interactions (called hits) within the volumes which represent detectors (sensitive detector or physical volume). Gate only stores hits for those volumes attached to a sensitive detector. Hits regarding interactions occurring in non-sensitive volumes are lost. A volume must belong to a system before it can be attached to a sensitive detector. Hits, occurring in a volume, cannot be scored in an output file if this volume is not connected to a system because this volume can not be attached to a sensitive detector. The concepts of system and sensitive detector are discussed in more detail in *Defining a system* and *Attaching the sensitive detectors* respectively.

The following commands are used to connect the volumes to the system:

```
# R E P E A T   R S E C T O R
/gate/box1/repeaters/insert ring
/gate/box1/ring/setRepeatNumber 30
# Z O O M
/vis/viewer/zoom 0.25
/vis/viewer/panTo 0 0 mm
# A T T A C H   V O L U M E S   T O   A   S Y S T E M
/gate/systems/cylindricalPET/rsector/attach box1
/gate/systems/cylindricalPET/module/attach box2
```

The names rsector and module are dedicated names and correspond to the first and the second levels of the Cylindri-

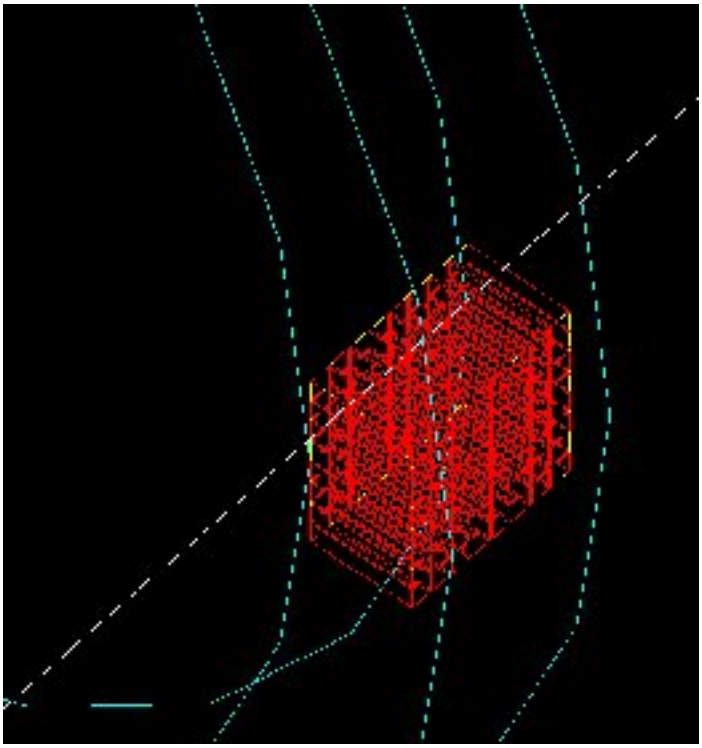


Fig. 2.5: Matrix of crystals

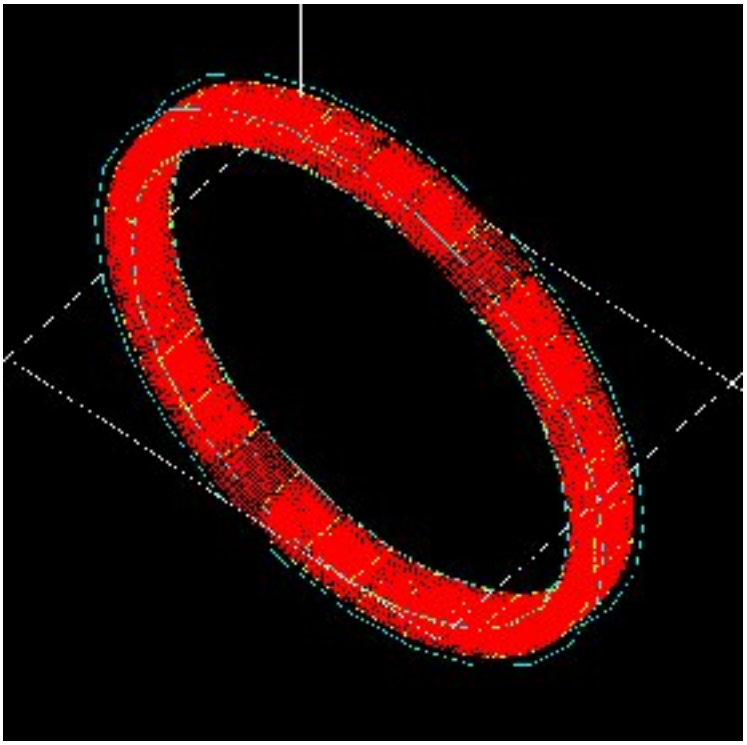


Fig. 2.6: Complete ring of 30 block detectors

calPET system (see *Defining a system*).

In order to save the hits (see *Digitizer and readout parameters*) in the volumes corresponding to the crystals the appropriate command, in this example, is:

```
# D E F I N E   A   S E N S I T I V E   D E T E C T O R
/gate/box2/attachCrystalSD vglue 1cm
```

At this level of the macro file, the user can implement detector movement. One of the most distinctive features of Gate is the management of time-dependent phenomena, such as detector movements and source decay leading to a coherent description of the acquisition process. For simplicity, the simulation described in this tutorial does not take into account the motion of the detector or the phantom. *Defining a geometry* describes the movement of volumes in detail.

2.1.5 Second step: Defining a phantom geometry

The volume to be scanned is built according to the same principle used to build the scanner. The external envelope of the phantom is a daughter of the *world*. The following command lines describe a cylinder with a radius of 10 mm and a length of 30 mm. The cylinder is filled with water and will be displayed in gray. This object represents the attenuation medium of the phantom:

```
# P H A N T O M
/gate/world/daughters/name my_phantom
/gate/world/daughters/insert cylinder
/gate/my_phantom/setMaterial Water
/gate/my_phantom/vis/setColor grey
/gate/my_phantom/geometry/setRmax   10. mm
/gate/my_phantom/geometry/setHeight 30. mm
```

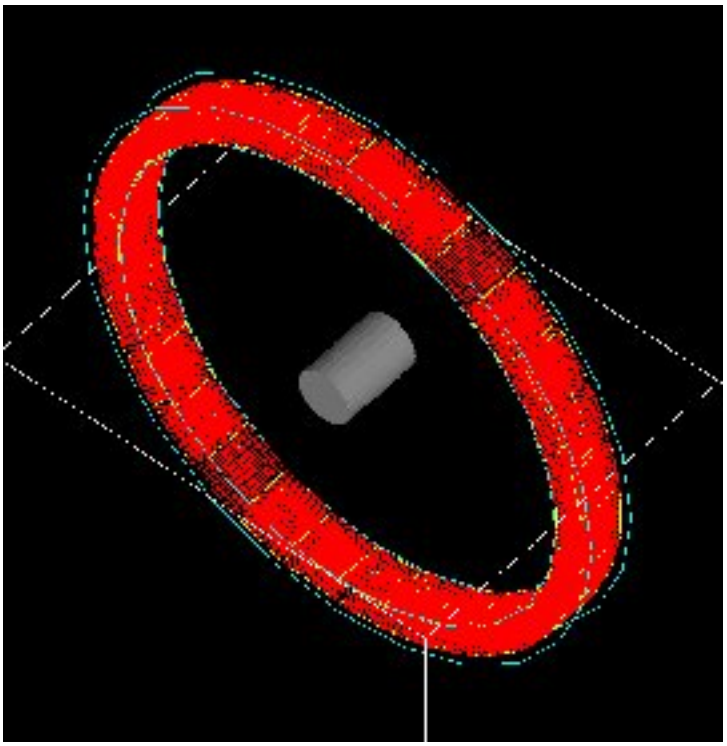


Fig. 2.7: Cylindrical phantom

To retrieve information about the Compton and the Rayleigh interactions within the phantom, a sensitive detector (*phantomSD*) is associated with the volume using the following command line:

```
# P H A N T O M  D E F I N E D  A S  S E N S I T I V E
/gate/my_phantom/attachPhantomSD
```

Two types of information will now be recorded for each hit in the hit collection:

- The number of scattering interactions generated in all physical volumes attached to the *phantomSD*.
- The name of the physical volume attached to the *phantomSD* in which the last interaction occurred.

These concepts are further discussed in *Attaching the sensitive detectors*.

2.1.6 Third step: Setting-up the physics processes

Once the volumes and corresponding sensitive detectors are described, the interaction processes of interest in the simulation have to be specified. Gate uses the GEANT4 models for physical processes. The user has to choose among these processes for each particle. Then, user can customize the simulation by setting the production thresholds, the cuts, the electromagnetic options...

Some typical physics lists are available in the directory *examples/PhysicsLists*:

- egammaStandardPhys.mac (physics list for photons, e- and e+ with standard processes and recommended Geant4 "option3")
- egammaLowEPhys.mac (physics list for photons, e- and e+ with low energy processes)
- egammaStandardPhysWithSplitting.mac (alternative egammaStandardPhys.mac with selective bremsstrahlung splitting)
- hadrontherapyStandardPhys.mac (physics list for hadrontherapy with standard processes and recommended Geant4 "option3")
- hadrontherapyLowEPhys.mac (physics list for hadrontherapy with low energy processes)

The details of the interactions processes, cuts and options available in Gate are described in *Setting up the physics*.

2.1.7 Fourth step: Initialization

When the 3 steps described before are completed, corresponding to the pre-initialization mode of GEANT4, the simulation should be initialized using:

```
# I N I T I A L I Z E
/gate/run/initialize
```

This initialization actually triggers the calculation of the cross section tables. After this step, the physics list cannot be modified any more and new volumes cannot be inserted into the geometry.

2.1.8 Fifth step: Setting-up the digitizer

The basic output of Gate is a *hit* collection in which data such as the position, the time and the energy of each hit are stored. The history of a particle is thus registered through all the *hits* generated along its track. The goal of the *digitizer* is to build physical observables from the *hits* and to model readout schemes and trigger logics. Several functions are grouped under the Gate *digitizer* object, which is composed of different modules that may be inserted into a linear signal processing sequence. As an example, the following command line inserts an *adder* to sum the hits generated per elementary volume (a single crystal defined as box2 in our example):

```
/gate/digitizer/Singles/insert adder
```

Another module can describe the readout scheme of the simulation. Except when one crystal is read out by one photo-detector, the readout segmentation can be different from the elementary geometrical structure of the detector. The readout geometry is an artificial geometry which is usually associated with a group of sensitive detectors. In this example, this group is box1:

```
/gate/digitizer/Singles/insert readout
/gate/digitizer/Singles/readout/setDepth 1
```

In this example, the readout module sums the energy deposited in all crystals within the block and determines the position of the crystal with the highest energy deposited ("winner takes all"). The setDepth command specifies at which geometry level (called "depth") the readout function is performed. In the current example:

- base level (CylindricalPET) = depth 0
- 1srt daughter (box1) of the system = depth 1
- next daughter (box2) of the system = depth 2
- and so on ....

In order to take into account the energy resolution of the detector and to collect singles within a pre-defined energy window only, other modules can be used:

```
# E N E R G Y   B L U R R I N G
/gate/digitizer/Singles/insert blurring
/gate/digitizer/Singles/blurring/setResolution 0.19
/gate/digitizer/Singles/blurring/setEnergyOfReference 511. keV
# E N E R G Y   W I N D O W
/gate/digitizer/Singles/insert thresholder
/gate/digitizer/Singles/thresholder/setThreshold 350. keV
/gate/digitizer/Singles/insert upholder
/gate/digitizer/Singles/upholder/setUphold 650. keV
```

Here, an energy resolution of 19% at 551 KeV is considered.

Furthermore, the energy window is set from 350 keV to 600 keV.

For PET simulations, the coincidence sorter is also implemented at the *digitizer* level:

```
# C O I N C I D E N C E   S O R T E R
/gate/digitizer/Coincidences/setWindow 10. ns
```

Other *digitizer* modules are available in Gate and are described in *Digitizer and readout parameters*.

2.1.9 Sixth step: Setting-up the source

In Gate, a source is represented by a volume in which the particles (positron, gamma, ion, proton, ...) are emitted. The user can define the geometry of the source and its characteristics such as the direction of emission, the energy distribution, and the activity. The lifetime of unstable sources (radioactive ions) is usually obtained from the GEANT4 database, but it can also be set by the user.

A voxelized phantom or a patient dataset can also be used to define the source, in order to simulate realistic acquisitions. For a complete description of all functions to define the sources, see *Voxelized source and phantom*.

In the current example, the source is a 1 MBq line source. The line source is defined as a cylinder with a radius of 0.5 mm and a length of 50 mm. The source generates pairs of 511 keV gamma particles emitted 'back-to-back' (for

a more realistic source model, the range of the positron and the non collinearity of the two gammas can also be taken into account):

```
# S O U R C E
/gate/source/addSource twogamma
/gate/source/twogamma/setActivity 100000. becquerel
/gate/source/twogamma/setType backtoback
# POSITION
/gate/source/twogamma/gps/centre 0. 0. 0. cm
# PARTICLE
/gate/source/twogamma/gps/particle gamma
/gate/source/twogamma/gps/energytype Mono
/gate/source/twogamma/gps/monoenergy 0.511 MeV
# TYPE = Volume or Surface
/gate/source/twogamma/gps/type Volume
# SHAPE = Sphere or Cylinder
/gate/source/twogamma/gps/shape Cylinder
/gate/source/twogamma/gps/radius 0.5 mm
/gate/source/twogamma/gps/halfz 25 mm

# SET THE ANGULAR DISTRIBUTION OF EMISSION
/gate/source/twogamma/gps/angtype iso
# SET MIN AND MAX EMISSION ANGLES
/gate/source/twogamma/gps/mintheta 0. deg
/gate/source/twogamma/gps/maxtheta 180. deg
/gate/source/twogamma/gps/minphi    0. deg
/gate/source/twogamma/gps/maxphi 360. deg
/gate/source/list
```

2.1.10 Seventh step: Defining data output

By default, the data output formats for all systems used by Gate are ASCII and ROOT as described in the following command lines:

```
# ASCII OUTPUT FORMAT
/gate/output/ascii/enable
/gate/output/ascii/setFileName test
/gate/output/ascii/setOutFileHitsFlag 0
/gate/output/ascii/setOutFileSinglesFlag 1
/gate/output/ascii/setOutFileCoincidencesFlag 1
# ROOT OUTPUT FORMAT
/gate/output/root/enable
/gate/output/root/setFileName test
/gate/output/root/setRootSinglesFlag 1
/gate/output/root/setRootCoincidencesFlag 1
```

Given this script, several ASCII files (.dat extension) and A ROOT file (test.root) will be created. *Data output* explains how to read the resulting files.

For some scanner configurations, the events may be stored in a sinogram format or in List Mode Format (LMF). The sinogram output module stores the coincident events from a cylindrical scanner system in a set of 2D sinograms according to the parameters set by the user (number of radial bins and angular positions). One 2D sinogram is created for each pair of crystal-rings. The sinograms are stored either in raw format or ecat7 format. The List Mode Format is the format developed by the Crystal Clear Collaboration (LGPL licence). A library has been incorporated in Gate to read, write, and analyze the LMF format. A complete description of all available outputs is given in *Data output* .

2.1.11 Eighth step: Starting an acquisition

In the next and final step the acquisition is defined. The beginning and the end of the acquisition are defined as in a real life experiment. In addition, Gate needs a time slice parameter which defines time period during which the simulated system is assumed to be static. At the beginning of each time-slice, the geometry is updated according to the requested movements. During each time-slice, the geometry is kept static and the simulation of particle transport and data acquisition proceeds. Each slice corresponds to a Geant4 run.

If the sources involved in the simulation are not radioactive or if activity is not defined, user can fix the total number of events. In this case, the number of particles is splitted between slices in function of the time of each slice:

```
/gate/application/setTotalNumberOfPrimaries [N]
```

User can also fix the same number of events per slice. In this case, each event is weighted by the ratio between the time slice and the total simulation time:

```
/gate/application/setNumberOfPrimariesPerRun [N]
```

It also can be useful to set a different number of primaries for each run. This can be done using a file containing the number of primaries and with the command:

```
/gate/application/readNumberOfPrimariesInAFile [path/to/filename]
```

An example of use can be found in GateContrib

Regular time slice approach

This is the standard Gate approach for imaging applications (PET, SPECT and CT). User has to define the beginning and the end of the acquisition using the commands setTimeStart and setTimeStop. Each slice has the same duration. User has to define the slice duration (setTimeSlice):

```
/gate/application/setTimeSlice     1.  s
/gate/application/setTimeStart     0.  s
/gate/application/setTimeStop      1.  s
```

The choice of the generator seed is also extremely important. There are 3 ways to make that choice:

- The 'default' option. In this case the default CLHEP internal seed is taken. This seed is always the same.
- The 'auto' option. In this case, a new seed is automatically generated each time GATE is run.

To randomly generate the seed, the time in millisecond since January 1, 1970 and the process ID of the GATE instance (i.e. the system ID of the running GATE process) are used. So each time GATE is run, a new seed is used.

- The 'manual' option. In this case, the user can manually set the seed. The seed is an unsigned integer value and it is recommended to be included in the interval [0,900000000].

The commands associated to the choice of the seed with the 3 different options are the following:

```
/gate/random/setEngineSeed default
/gate/random/setEngineSeed auto
/gate/random/setEngineSeed 123456789
```

It is also possible to control directly the initialization of the engine by selecting the file containing the seeds with the command:

```
/gate/random/resetEngineFrom fileName

# S T A R T  the A C Q U I S I T I O N
/gate/application/startDAQ
```

The number of projections or runs of the simulation is thus defined by:

$Nrun = \frac{setTimeStop - setTimeStart}{setTimeSlice}$

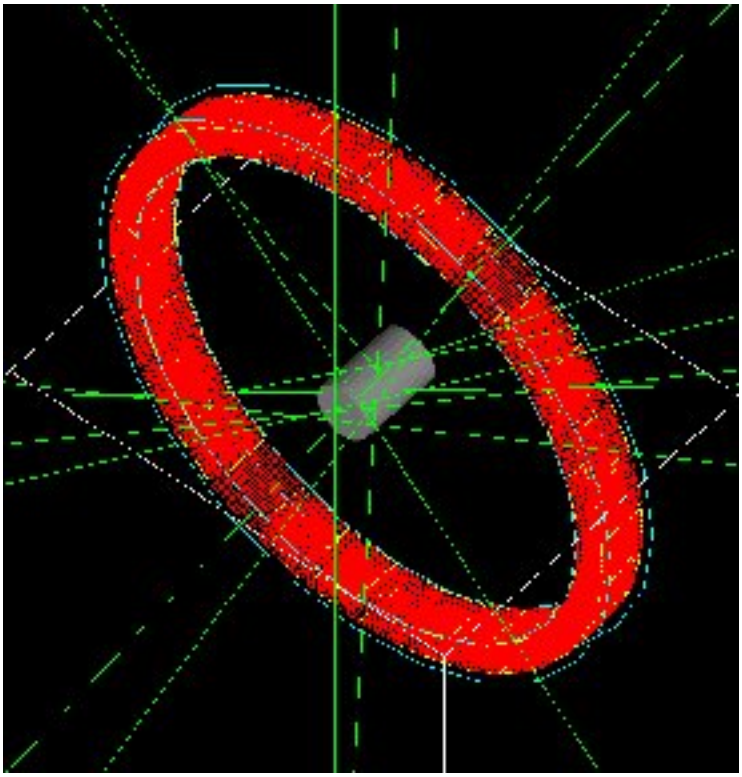


Fig. 2.8: Simulation is started

In the current example, there is no motion, the acquisition time equals 1 second and the number of projections equals one.

If you want to exit from the Gate program when the simulation time exceed the time duration, the last line of your program has to be **exit**.

As a Monte Carlo tool, GATE needs a random generator. The CLHEP libraries provide various ones. Three different random engines are currently available in GATE, the Ranlux64, the James Random and the Mersenne Twister. The default one is the Mersenne Twister, but this can be changed easily using:

```
/gate/random/setEngineName aName    (where aName can be: Ranlux64, JamesRandom, or␣
↪MersenneTwister)
```

NB Several users have reported artifacts in PET data when using the Ranlux64 generator. These users have said that the artifacts are not present in data generated with the Mersenne Twister generator.

Slices with variable time

In this approach, each slice has a specific duration. User has to define the time of each slice. The first method is to use a file of time slices:

```
/gate/application/readTimeSlicesIn [File Name]
```

the second method is to add each slice with the command:

```
/gate/application/addSlice [value] [unit]
```

User has to define the beginning of the acquisition using the command setTimeStart. The end of acquisition is calculated by summing each time slice. The simulation is started with the commands:

```
/gate/application/start
```

or:

```
/gate/application/startDAQ
```

2.1.12 Verbosity

The level of verbosity of the random engine can be chosen. It consists into printing the random engine status, depending on the type of generator used. The command associated to the verbosity is:

```
/gate/random/verbose 1
```

Values from 0 to 2 are allowed, higher values will be interpreted as 2. A value of 0 means no printing at all, a value of 1 results in one printing at the beginning of the acquisition, and a value of 2 results in one printing at each beginning of run.

2.2 Defining a geometry

Table of Contents

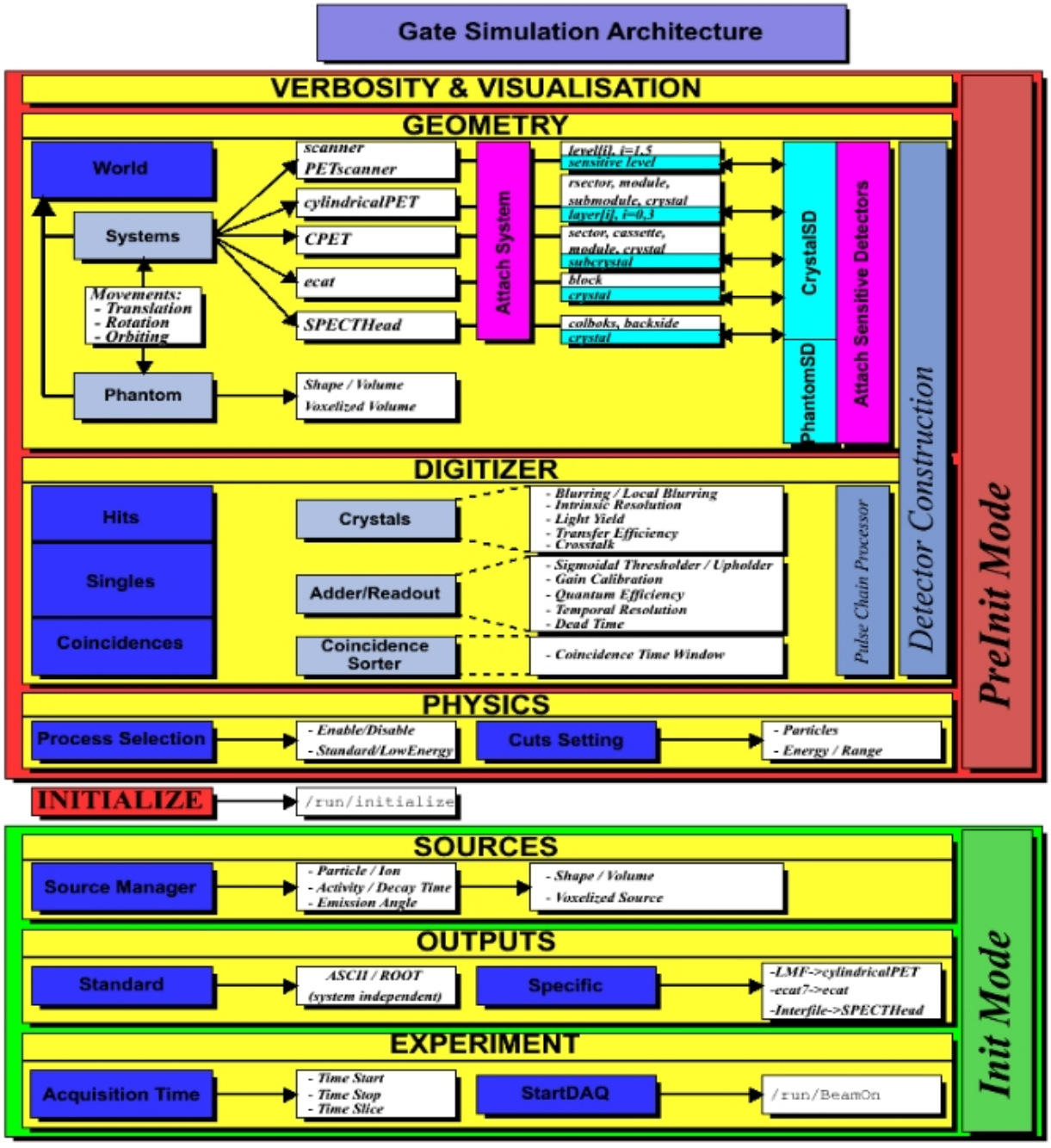


Fig. 2.9: GATE simulation architecture

The definition of a geometry is a key step in designing a simulation because it is through the geometry definition that the imaging device and object to be scanned are described. Particles are then tracked through the components of the geometry. This section explains how to define the different components of the geometry.

2.2.1 The world

Definition

The *world* is the only volume already defined in GATE when starting a macro. All volumes are defined as daughters or grand-daughters of the *world*. The *world* volume is a typical example of a GATE volume and has predefined properties. The *world* volume is a box centered at the origin. For any particle, tracking stops when it escapes from the *world* volume. The *world* volume can be of any size and has to be large enough to include all volumes involved in the simulation.

Use

The first volume that can be created must be the daughter of the *world* volume. Any volume must be included in the *world* volume. The geometry is built from the *world* volume.

Description and modification

The *world* volume has default parameters: shape, dimensions, material, visibility attributes and number of children. These parameters can be edited using the following GATE command:

```
/gate/world/describe
```

The output of this command is shown in Fig. 2.10. The parameters associated with the *world* volume can be modified to be adapted to a specific simulation configuration. Only the shape of the *world* volume, which is a box, cannot be changed. For instance, the X length can be changed from 50 cm to 2 m using:

```
/gate/world/geometry/setXLength 2. m
```

```
Idle> /gate/world/describe
----------------------------------------------

Nb of moves:       1

        GATE object:         'world/placement'
        Is enabled?          Yes
        Move type:      placement
        Translation:         0 0 0 fm
        Rotation axis:       (0,0,0)
        Rotation angle:      0 deg

Nb of repeaters:       0
----------------------------------------------
```

Fig. 2.10: Description of the default parameters associated with the world

The other commands needed to modify the *world* volume attributes will be given in the next sections.

2.2.2 Creating a volume

Generality - Tree creation

When a volume is created with GATE, it automatically appears in the GATE tree. All commands applicable to the new volume are then available from this GATE tree. For instance, if the name of the created volume is *Volume_Name*, all commands applicable to this volume start with:

```
/gate/Volume_Name/
```

The tree includes the following commands:

- setMaterial: To assign a material to the volume
- attachCrystalSD: To attach a crystal-SensitiveDetector to the volume
- attachPhantomSD: To attach a phantom-SensitiveDetector to the volume
- enable: To enable the volume
- disable: To disable the volume
- describe: To describe the volume

The tree includes sub-trees that relate to different attributes of the volume *Volume_Name*. The available sub-trees are:

- daughters: To insert a new 'daughter' in the volume
- geometry: To control the geometry of the volume
- vis: To control the display attributes of the volume
- repeaters: To apply a new 'repeater' to the volume
- moves: To 'move' the volume
- placement: To control the placement of the volume

The commands available in each sub-tree will be described in *Building a volume*, *Repeating a volume*, *Placing a volume*, *Moving a volume*.

Units

Different units are predefined in GATE (see Table 2.1) and shall be referred to using the corresponding abbreviation. Inside the GATE environment, the list of units available in GATE can be edited using:

```
/units/list
```

Axes

Any position in the *world* is defined with respect to a three-axis system: X, Y and Z. These three axes can be seen in the display window using:

```
/vis/scene/add/axes
```

Table 2.1: List of units available in GATE and corresponding abbreviations

LENGTH	SURFACE	VOLUME	ANGLE
parsec pc			radian rad
kilometer km	kilometer2 km2	kilometer3 km3	milliradian mrad
meter m	meter2 m2	meter33 m3	steradian sr
centimeter cm	centimeter2 cm2	centimeter3 cm3	degre deg
micrometer mum	millimeter2 mm2	millimeter3 mm3	
nanometer nm			
angstrom Ang			

Continued

Table 2.1 – continued from previous page

LENGTH	SURFACE	VOLUME	ANGLE
TIME	**SPEED**	**ANGULAR SPEED**	**ENERGY**
second s	meter/s m/s	radian/s rad/s	electronvolt eV
millisecond ms	centimeter/s cm/s	degree/s deg/s	kiloelectronvolt KeV
microsecond mus	millimeter/s mm/s	rotation/s rot/s	megaelectronvolt MeV
nanosecond ns	meter/min m/min	radian/min rad/min	gigaelectronvolt GeV
picosecond ps	centimeter/min cm/min	degree/min deg/min	teraelectronvolt TeV
	millimeter/min m/min	rotation/min rot/min	petaelectronvolt PeV
	meter/h m/h	radian/h rad/h	joule j
	centimer/h cm/h	degree/h deg/h	
	millimeter/h mm/h	rotation/h rot/h	
ACTIVITY DOSE	**AMOUNT OF SUBSTANCE**	**MASS**	**VOLUMIC MASS**
becquerel Bq	mole mol	milligram mg	g/cm3 g/cm3
curie Ci			mg/cm3 mg/cm3
gray Gy		kilogram kg	kg/m3 kg/m3
ELECTRIC CHARGE	**ELECTRIC CURRENT**	**ELECTRIC POTENTIAL**	**MAGNETIC FLUX - MAGNETIC FL**
eplus e+	ampere A	volt V	weber Wb
coulomb C	milliamper mA	kilovolt kV	tesla T
microampere muA		megavolt MV	gauss G
nanoampere nA		kilogauss kG	
TEMPERATURE	**FORCE - PRESSURE**	**POWER**	**FREQUENCY**
kelvin K	newton N	watt W	hertz Hz
	pascal Pa		kilohertz kHz
	bar bar		megaherz MHz
	atmosphere atm		

Building a volume

Any new volume must be created as the daughter of another volume (i.e., *World* volume or another volume previously created).

Three rules must be respected when creating a new volume:

- A volume which is located inside another must be its daughter
- A daughter must be fully included in its mother
- Volumes must not overlap

Errors in building the geometry yield wrong particle transportation, hence misleading results!

Creating a new volume

To create a new volume, the first step is to give it a name and a mother using:

```
/gate/mother_Volume_Name/daughters/name Volume_Name
```

This command prepares the creation of a new volume named *Volume_Name* which is the daughter of *mother_Volume_Name.*

Some names should not be used as they have precise meanings in gate. These names are the names of the GATE systems (see *Defining a system*) currently defined in GATE: *scanner*, *PETscanner*, *cylindricalPET*, *SPECTHead*, *ecat*, *CPET*, *OPET* and *OpticalSystem*.

The creation of a new volume is completed only when assigning a shape to the new volume. The tree

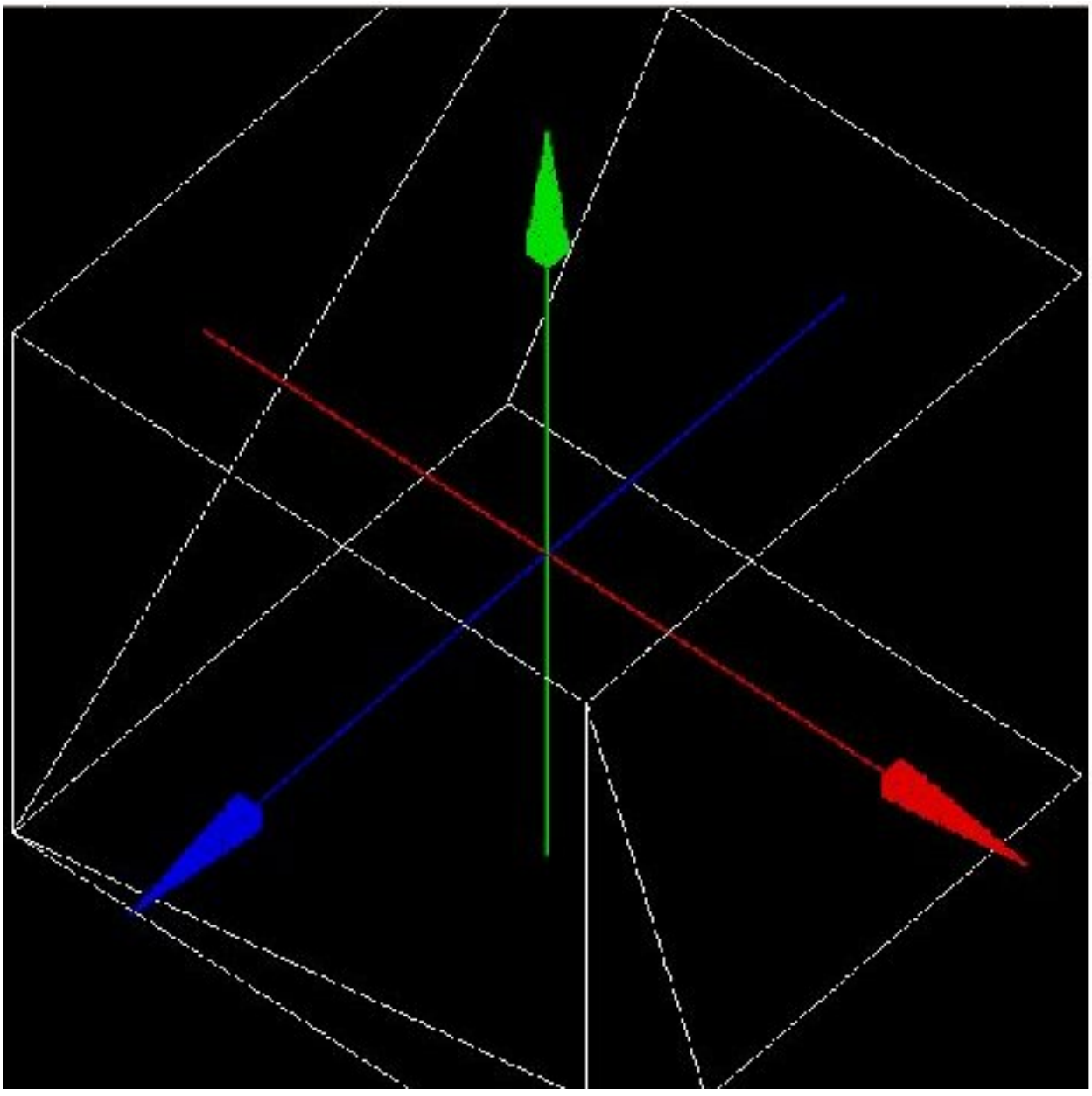


Fig. 2.11: Three-axis system defined in GATE. The red, green and blue axes are the X, Y and Z axes respectively

```
/gate/Volume_Name/
```

is then generated and all commands in the tree and the sub-trees are available for the new volume.

Different volume shapes are available, namely: **box, sphere, cylinder, cone, hexagon, general or extruded trapezoid, wedge, elliptical tube, tessellated and TetMeshBox.**

The command line for listing the available shapes is:

```
/gate/world/daughters/info
```

The command line for assigning a shape to a volume is:

```
/gate/mother_Volume_Name/daughters/insert Volume_shape
```

where *Volume_shape* is the shape of the new volume.

Volume_shape must necessarily be one of the available names:

box for a box - **sphere** for a sphere - **cylinder** for a cylinder - **ellipsoid** for an ellipsoid - **cone** for a cone - **eltub** for a tube with an elliptical base - **hexagone** for an hexagon - **polycone** for a polygon - **trap** for a general trapezoid - **trpd** for an extruded trapezoid - **wedge** for a wedge - **tessellated** for a tessellated volume and **TetMeshBox** for a box which contains a tetrahedral mesh.

The command line assigns the shape to the last volume that has been named.

The following command lists the daughters of a volume:

```
/gate/Volume_Name/daughters/list
```

- Example:

```
/gate/world/daughters/name Phantom
/gate/world/daughters/insert box
```

The new volume *Phantom* with a box shape is inserted in the *World* volume.

Defining a size

After creating a volume with a shape, its dimensions are the default dimensions associated with that shape. These default dimensions can be modified using the sub-tree /geometry/

The commands available in the sub-tree depend on the shape. The different commands for each type of shape are listed in Table 2.2

These commands can be found in the directory:

```
/gate/Volume_Name/geometry   (Some volumes visualisation are available here: http://
↪gphysics.net/geant4/geant4-gdml-format.html )
```

Table 2.2: Commands of the sub-tree geometry for different shapes

BOX	TRPD
setXLength: Set the length of the box along the X axis	setX1Length: Set half length along X of the plane at -dz p
setYLength: Set the length of the box along the Y axis	setY1Length: Set half length along Y of the plane at -dz p
setZLength: Set the length of the box along the Z axis	setX2Length: Set half length along X of the plane at +dz p
SPHERE	setY2Length: Set half length along Y of the plane at +dz p
setRmin: Set the internal radius of the sphere (0 for full sphere)	setZLength: Set half length along Z of the trapezoid
setRmax: Set the external radius of the sphere	setXBoxLength: Set half length along X of the extruded b

Table 2.2 – continued from previous page

BOX	TRPD
setPhiStart: Set the start phi angle	setYBoxLength: Set half length along Y of the extruded b
setDeltaPhi: Set the phi angular span (2PI for full sphere)	setZBoxLength: Set half length along Z of the extruded bo
setThetaStart: Set the start theta angle	setXBoxPos: Set center position X of the box
setDeltaTheta: Set the theta angular span (2PI for full sphere)	setYBoxPos: Set center position Y of the box
CYLINDER	setZBoxPos: Set center position Z of the box
setRmin: Set the internal radius of the cylinder (0 for full cylinder)	**PARALLELEPIPED (... not yet implemented...)**
setRmax: Set the external radius of the cylinder	setDx: Set Dx dimension of the parallelepiped
setHeight: Set the height of the cylinder	setDy: Set Dy dimension of the parallelepiped
setPhiStart: Set the start phi angle	setDz: Set Dz dimension of the parallelepiped
setDeltaPhi: Set the phi angular span (2PI for full cylinder)	setAlpha: Set Alpha angle
CONE	setTheta: Set Theta angle
setRmin1: Set the internal radius of one side of the cone (0 for full cone)	setPhi: Set Phi angle
setRmax1: Set the external radius of one side of the cone	**POLYCONE (... not yet implemented...)**
setRmin2: Set the internal radius of one side of the cone (0 for full cone)	setProfile: Set vectors of z, rInner, rOuter positions
setRmax2: Set the external radius of one side of the cone	setPhiStart: Set the start phi angle
setHeight: Set the height of the cone	setDeltaPhi: Set the phi angular span (2PI for full cone)
setPhiStart: Set the start phi angle	**HEXAGONE**
setDeltaPhi: Set the phi angular span (2PI for full cone)	setRadius: Set the radius of the hexagon
ELLIPSOID	setHeight: Set the height of the hexagon
setXLength: Set the half axis length in the X direction	**WEDGE**
setYLength: Set the half axis length in the Y direction	NarrowerXLength: Set the length of the shorter side of the
setZLength: Set the half axis length in the Z direction	XLength: Set the length of the wedge in the X direction
setZBottomCut: To cut the ellipsoide along the Z axis	YLength: Set the length of the wedge in the Y direction
setZTopCut: To cut the ellipsoide along the Z axis	ZLength: Set the length of the wedge in the Z direction
ELLIPTICAL TUBE	**TET-MESH BOX**
setLong: Set the length of the semimajor axis	reader/setPathToELEFile: Set path to '.ele' input file, whi
setShort: Set the length of the semiminor axis	reader/setUnitOfLength: Set unit of length for the values i
setHeight: Set the height of the tube	setPathToAttributeMap: Set path to txt-file which defines
TESSELLATED	
setPathToVerticesFile: Set the path to vertices text file	

For a box volume called *Phantom* , the X, Y and Z dimensions can be defined by:

```
/gate/Phantom/geometry/setXLength 20. cm
/gate/Phantom/geometry/setYLength 10. cm
/gate/Phantom/geometry/setZLength 5. cm
```

The dimensions of the *Phantom* volume are then 20 cm, 10 cm and 5 cm along the X, Y and Z axes respectively.

Defining a material

A material must be associated with each volume. The default material assigned to a new volume is Vacuum. The list of available materials is defined in the GateMaterials.db file. (see *Materials*).

The following command fills the volume *Volume_Name* with a material called *Material*:

```
/gate/Volume_Name/setMaterial Material
```

- Example:

```
/gate/Phantom/setMaterial Water
```

The *Phantom* volume is filled with Water.

Defining a color or an appearance

To make the geometry easy to visualize, some display options can be set using the sub-tree /vis/

The commands available in this sub-tree are: setColor, setVisible, setDaughtersInvisible, setLineStyle, setLineWidth, forceSolid and forceWireframe (see Table 2.3)

Table 2.3: List of commands of the GATE sub-tree geometry

Command	Action	Argument
setColor	Selects the color for the current volume	white, gray, black, red, green, blue, cyan, magenta and yellow
setVisible	Shows or hides the current volume	
setDaughtersInvisible	Shows or hides the current volume daughters	
setLineStyle	Sets the current volume line-style	dashed, dotted and unbroken
setLineWidth	Sets the current volume line-width	
forceSolid	Forces solid display for the current volume	
forceWireframe	Forces wireframe display for the current volume	

These commands can be found in the tree /gate/Volume_Name/vis.

- Example:

```
/gate/Phantom/vis/setColor blue
/gate/Phantom/vis/forceWireframe
```

The *Phantom* volume will be displayed in blue and will be transparent.

Enabling or disabling a volume

A volume cannot be destroyed. The only possible action is to disable it: this makes the volume disappear from the display window but not from the geometry.

Only the *world* volume cannot be disabled.

To disable a volume *Volume_Name*, the command is:

```
/gate/Volume_Name/disable
```

The volume *Volume_Name* can be enabled again using:

```
/gate/Volume_Name/enable
```

- Example:

```
/gate/Phantom/disable
```

The *Phantom* volume is disabled.

Describing a volume

The parameters associated with a volume *Volume_name* can be listed using:

```
/gate/Volume_Name/describe
```

- Example:

```
/gate/Phantom/describe
```

The parameters associated with the *Phantom* volume are listed.

Examples

How to build a NaI crystal

A volume named crystal is created as the daughter of a volume whose shape is defined as a box:

```
/gate/mother_Volume_Name/daughters/name     crystal
/gate/mother_Volume_Name/daughters/insert   box
```

The X, Y and Z dimensions of the volume crystal are set to 1 cm, 40 cm, and 54 cm respectively:

```
/gate/crystal/geometry/setXLength           1. cm
/gate/crystal/geometry/setYLength           40. cm
/gate/crystal/geometry/setZLength           54. cm
```

The new volume crystal is filled with NaI:

```
/gate/crystal/setMaterial                   NaI
```

The new volume crystal is colored in yellow:

```
/gate/crystal/vis/setColor                  yellow
```

The next command lists the parameters associated with the crystal volume:

```
/gate/crystal/describe
```

The crystal volume is disabled:

```
/gate/crystal/disable
```

How to build a "trpd" volume

An alternative way of describing complicated geometries is to use a so-called "boolean" volume in order to describe one piece using a single volume instead of using a mother-children couple. This can make the description easier and more synthetic. The example below describes how the shape shown in Fig. 2.12 can be defined using a trpd shape, based on a "boolean" volume consisting of a trapezoid "minus" a box:

```
# V I S U A L I S A T I O N
/vis/open OGLSX /vis/viewer/reset
/vis/viewer/viewpointThetaPhi 60 60
/vis/viewer/zoom 1
/vis/viewer/set/style surface
/vis/drawVolume /tracking/storeTrajectory 1
/vis/scene/endOfEventAction accumulate
/vis/viewer/update
/vis/verbose 2
/gate/geometry/enableAutoUpdate
/gate/world/daughters/name                Volume_Name
```

(continues on next page)

(continued from previous page)

```
/gate/world/daughters/insert                box
/gate/Volume_Name/geometry/setXLength       40 cm
/gate/Volume_Name/geometry/setYLength       40 cm
/gate/Volume_Name/geometry/setZLength       40 cm
/gate/Volume_Name/vis/forceWireframe
/gate/Volume_Name/daughters/name            trapeze_name
/gate/Volume_Name/daughters/insert          trpd
/gate/trapeze_name/geometry/setX1Length     23.3 mm
/gate/trapeze_name/geometry/setY1Length     21.4 mm
/gate/trapeze_name/geometry/setX2Length     23.3 mm
/gate/trapeze_name/geometry/setY2Length     23.3 mm
/gate/trapeze_name/geometry/setZLength      6. mm
/gate/trapeze_name/geometry/setXBoxPos      0. mm
/gate/trapeze_name/geometry/setYBoxPos      0. m
/gate/trapeze_name/geometry/setZBoxPos      0.7501 mm
/gate/trapeze_name/geometry/setXBoxLength 20.3 mm
/gate/trapeze_name/geometry/setYBoxLength 20.3 mm
/gate/trapeze_name/geometry/setZBoxLength 4.501 mm
```

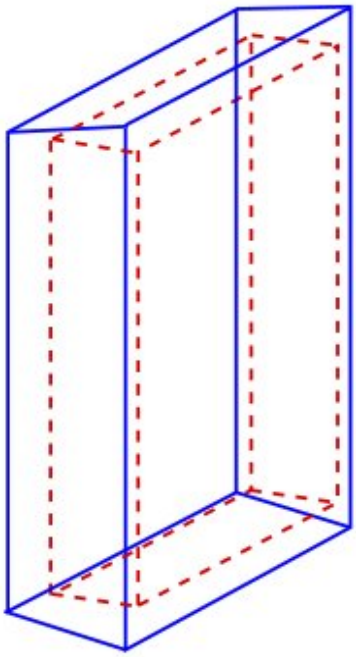


Fig. 2.12: Side view of an extruded trapezoid based on a boolean solid. The contours in blue and dashed red represent the contours of the trapezoid and the box respectively

The new volume called *trapeze_name*, which is the daughter of the *Volume_Name* volume, is described with 5+6 parameters. The first 5 parameters relate to the trapezoid, whereas the last 6 parameters describe the extruded volume using a box shape.

How to build a "wedge" volume

Gate provides the class **GateTrapCreator** to create and insert trapezoidal volumes into the geometry. To create a trapezoid, the user needs to specify eleven parameters (besides its name and material), which does not make it easy to use.

To model "slanted" crystals, a new class called **GateWedgeCreator** (derived from **G4Trap**) builds right angular wedges. As shown in Fig. 2.13, a wedge is defined by only three parameters that are easily understood:

1. XLength: is the length of the wedge in the X direction.
2. NarrowerXLength: is the length of the shorter side of the wedge in the X direction.
3. YLength: is the length in the Y direction.
4. ZLength: is the length in the Z direction.

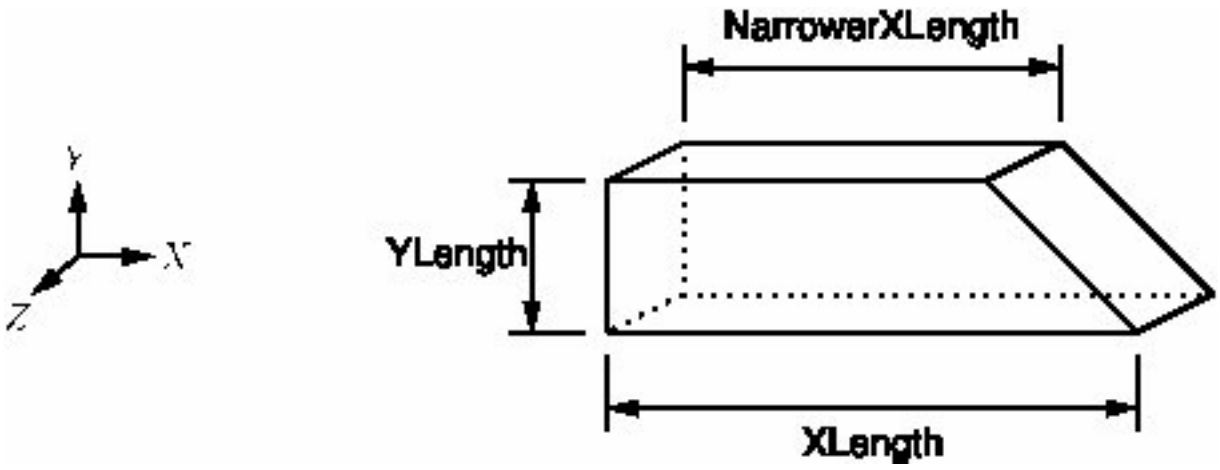


Fig. 2.13: When a wedge is inserted, it is oriented as shown in this figure

For instance, the following macro lines insert a wedge crystal as a daughter of a module:

```
/gate/module/daughters/name                wedge0
/gate/module/daughters/insert              wedge
/gate/wedge0/geometry/setXLength           10 mm
/gate/wedge0/geometry/setNarrowerXLength   8.921 mm
/gate/wedge0/geometry/setYLength           2.1620 mm
/gate/wedge0/geometry/setZLength           2.1620 mm
/gate/wedge0/setMaterial                   LSO
/gate/wedge0/vis/setColor                  yellow
```

How to build a "tessellated" volume

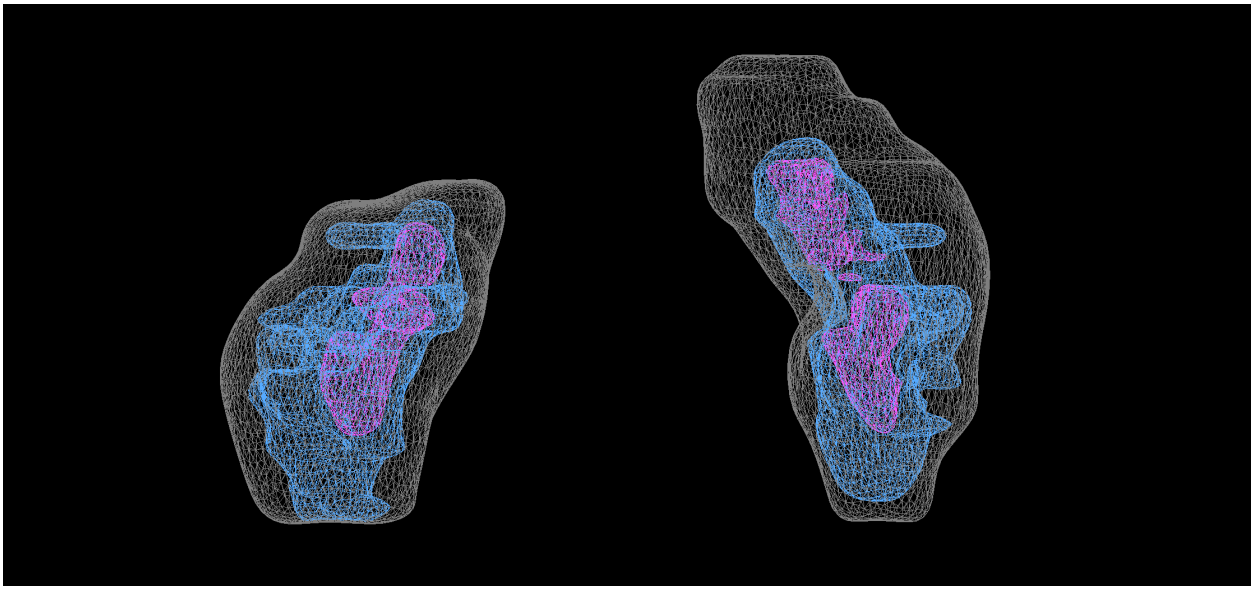
In GATE, you have the possibility to create a tessellated volume from an STL file. STL is a common file format that uses triangular facets to define the surface of a three-dimensional object. This allows to simulate a complex geometry imported from a CAD software. The surface described in the STL file is used to create a volume in GATE using the Geant4 G4TessellatedSolid class. It's important to note that only one material is associated to a tessellated volume. You can use either ASCII or binary STL files.

Here is an example to create a tessellated volume from an STL file in a GATE macro:

```
/gate/world/daughters/name                                  kidneyLeft
/gate/world/daughters/insert                                tessellated
/gate/kidneyLeft/placement/setTranslation                   -265.3625 -121.5875␣
↪-842.16 mm
/gate/kidneyLeft/geometry/setPathToSTLFile                  data/Label89.stl
/gate/kidneyLeft/setMaterial                                Kidney
```

Label89.stl being the STL file containing the triangular facets.

Declaring other tessellated volumes (including daughters), one can create a complex geometry (for example kidneys) for accurate dosimetry:



The complete code used to generate this figure can be found in the GateContrib GitHub repository under misc/geometry_STL/kidneys.

How to build a "TetMeshBox" volume

The **TetMeshBox** volume is a box volume which contains a tetrahedral mesh. The tetrahedral mesh can be loaded from an '.ele/.node' file pair, which can be generated by TetGen, an open-source tetrahedral mesh generator. Please refer to the TetGen manual for a comprehensive explanation of the structure of '.ele' and '.node' files. An example usage of the TetMeshBox would look like this:

```
/gate/world/daughters/name                  meshPhantom
/gate/world/daughters/insert                TetMeshBox
/gate/meshPhantom/setMaterial               Air
/gate/meshPhantom/reader/setPathToELEFile   data/BodyHasHeart.ele
/gate/meshPhantom/reader/setUnitOfLength    1.0 mm
/gate/meshPhantom/setPathToAttributeMap     data/RegionAttributeTable.dat
```

Here, GATE would implicitly assume that two files exist, namely 'data/BodyHasHeart.node' and 'data/BodyHasHeart.ele'. The numerical values defined in those files are interpreted according to the 'setUnitOfLength' command. GATE assumes that the '.ele' input file defines a region attribute for each tetrahedron – an integer attribute, which logically groups tetrahedra that form a sub-structure of the mesh. The user has to provide an 'attribute map', which defines material and colour for each region within the tetrahedral mesh. An attribute map is a txt-file and looks as follows:

```
# [first region,    last region]    material    visible   r       g       b       alpha
# ------------------------------------------------------------------------------------
1                   1               Heart       true      1.00    0.0     0.0     1.0
2                   3               Adipose     true      1.00    0.89    0.77    1.0
```

The first two columns refer to the region attributes defined in the '.ele' file.

The size of the bounding box will adapt to the extent of the tetrahedral mesh and the material of the bounding box can be set via the 'setMaterial'. Here, a visual example of the TetMeshBox volume:

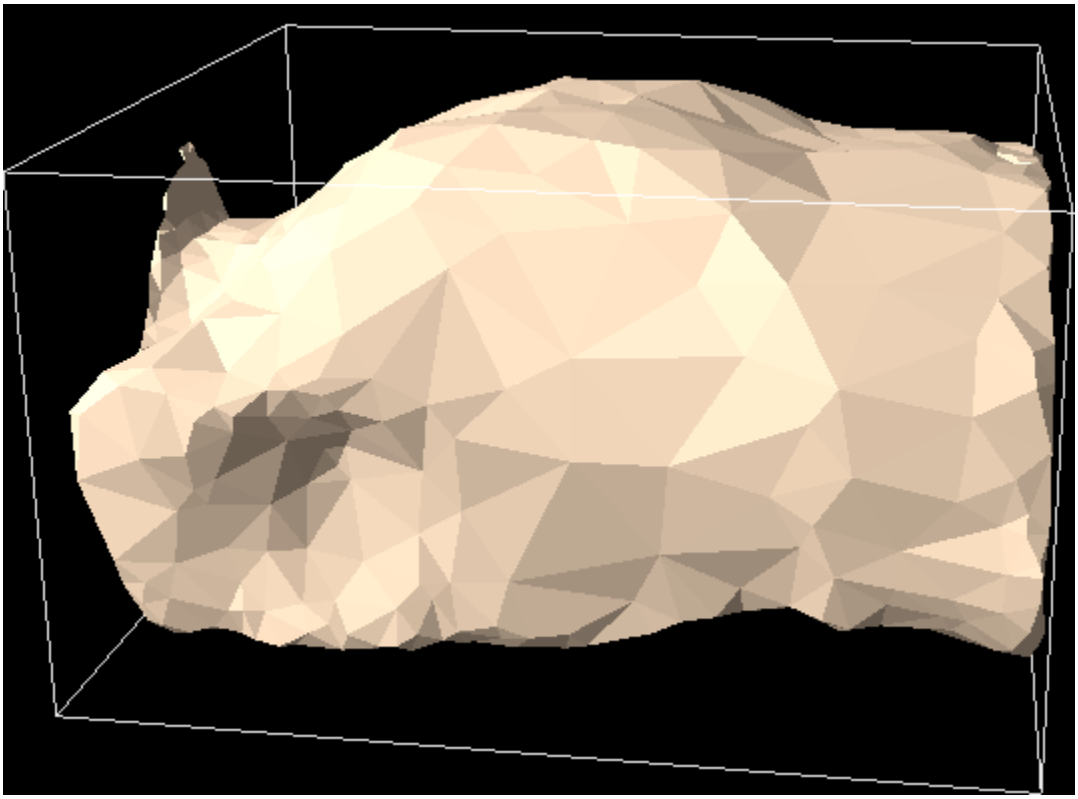


Fig. 2.14: tet_mesh_box.png

The complete code used to generate this figure can be found in the GateContrib repository on Github under misc/TetrahedralMeshGeometry.

2.2.3 Repeating a volume

To create X identical volumes, there is no need to create X different volumes. Only one volume must be created and then repeated. There are four different ways to repeat a volume: the linear repeater, the ring repeater, the cubic array repeater and the quadrant repeater.

To list the repeaters defined for the volume *Name_Volume*, use:

```
/gate/Name_Volume/repeaters/info
```

Linear repeater

The linear repeater is appropriate to repeat a volume along a direction (X, Y or Z axis). To use the linear repeater, first select this type of repeater using:

```
/gate/Name_Volume/repeaters/insert linear
```

Then define the number of times N the volume *Name_Volume* has to be repeated using:

```
/gate/Name_Volume/linear/setRepeatNumber N
```

Finally, define the step and direction of the repetition using:

```
/gate/Name_Volume/linear/setRepeatVector 0. 0. dZ. mm
```

A step of dZ mm along the Z direction is defined.

The "autoCenter" command allows the user to set the position of the repeated volumes:

```
/gate/Name_Volume/linear/autoCenter true or false
```

The "true" option centers the group of repeated volumes around the position of the initial volume that has been repeated.

The "false" option centers the first copy around the position of the initial volume that has been repeated. The other copies are created by offset. The default option is true.

- Example:

```
/gate/hole/repeaters/insert          linear
/gate/hole/linear/setRepeatNumber    12
/gate/hole/linear/setRepeatVector    0. 4. 0. cm
```

The *hole* volume is repeated 12 times every 4 cm along the Y axis. The application of this linear repeater is illustrated in Fig. 2.15.

Ring repeater

The ring repeater makes it possible to repeat a volume along a ring. It is useful to build a ring of detectors in PET.

To select the ring repeater, use:

```
/gate/Name_Volume/repeaters/insert ring
```

To define the number of times *N* the volume *Name_Volume* has to be repeated, use:

```
/gate/Name_Volume/ring/setRepeatNumber N
```

Finally, the axis around which the volume *Name_Volume* will be repeated must be defined by specifying two points using:

```
/gate/Name_Volume/ring/setPoint1 0. 1. 0. mm
/gate/Name_Volume/ring/setPoint2 0. 0. 0. mm
```

The default rotation axis is the Z axis. Note that the default ring repetition goes counter clockwise.

These three commands are enough to repeat a volume along a ring over 360°. However, the repeat action can be further customized using one or more of the following commands. To set the rotation angle for the first copy, use:

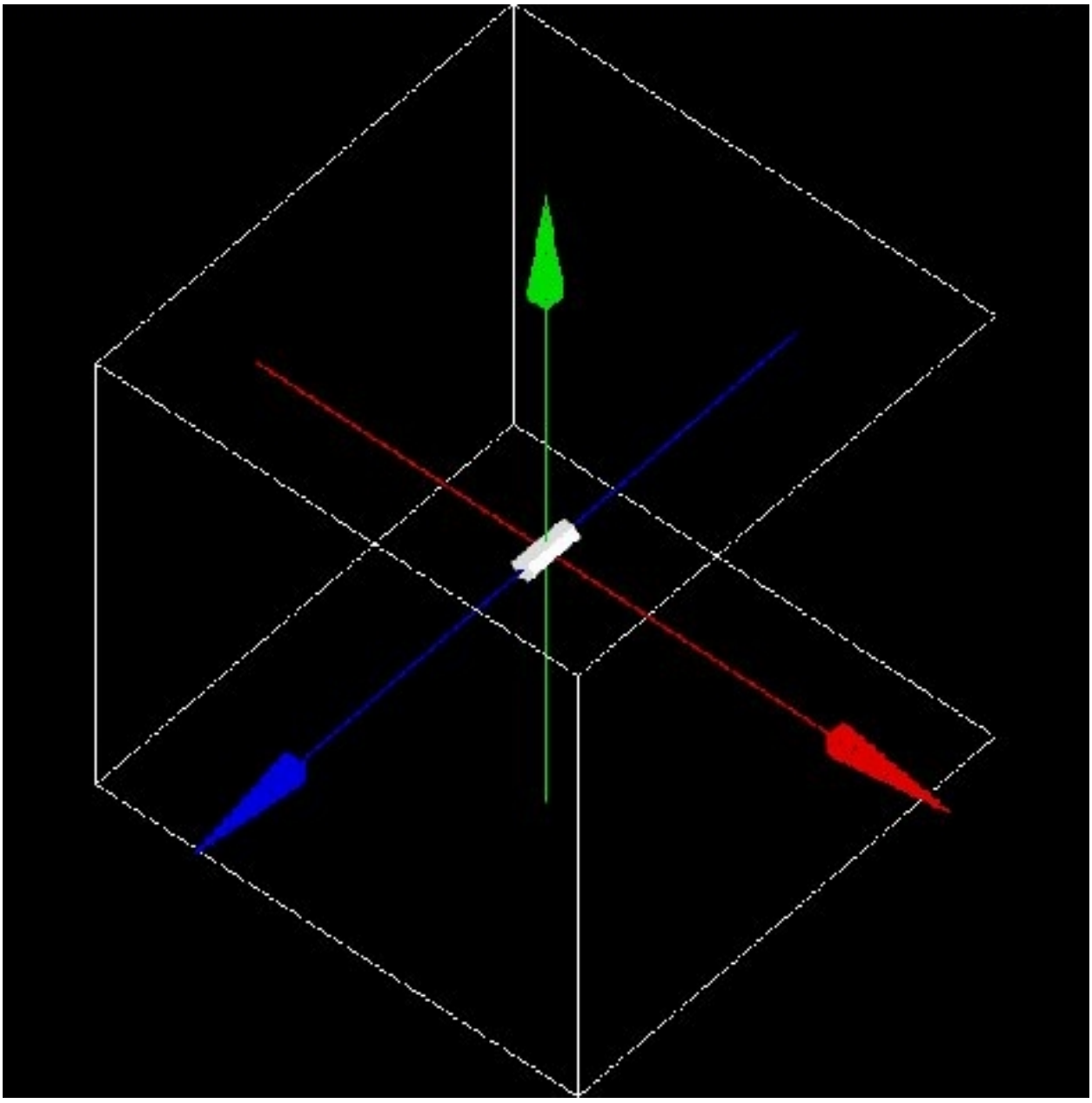


Fig. 2.15: Illustration of the application of the linear repeater

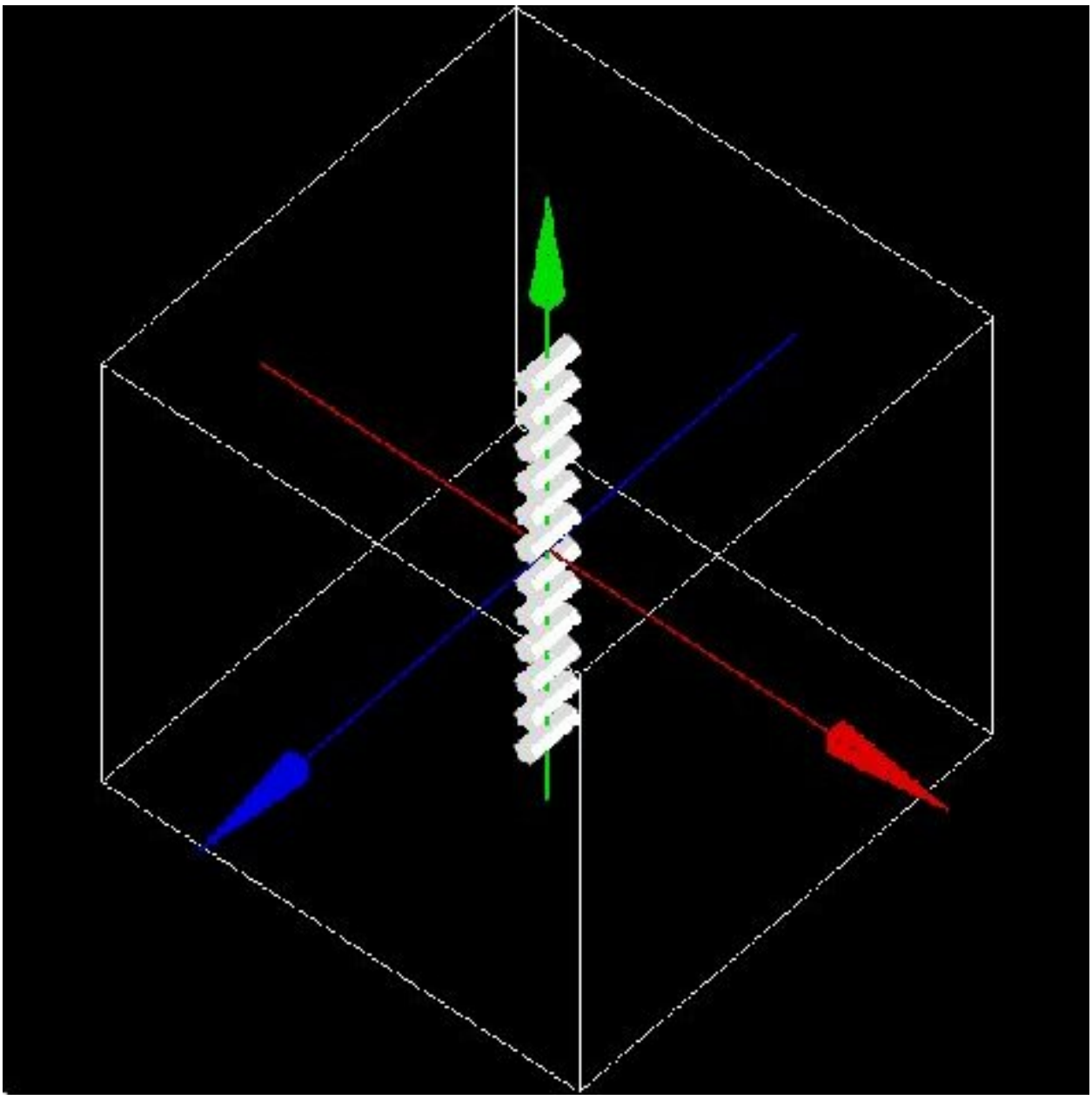


Fig. 2.16: Illustration of the application of the linear repeater

```
/gate/Name_Volume/ring/setFirstAngle x deg
```

The default angle is 0 deg.

To set the rotation angle difference between the first and the last copy, use:

```
/gate/Name_Volume/ring/setAngularSpan x deg
```

The default angle is 360 deg.

The AngularSpan, the FirstAngle and the RepeatNumber allow one to define the rotation angle difference between two adjacent copies (AngularPitch).

$\frac{AngularSpan-FirstAngle}{RepeatNumber-1} = AngularPitch$

To set the number of objects in the periodic structure, hence the periodicity, use:

```
/gate/Name_Volume/ring/setModuloNumber M
```

When the volume auto-rotation option is enabled, the volume itself is rotated so that its axis remains tangential to the ring (see Fig. 2.17). If this option is disabled, all repeated volumes keep the same orientation (see Fig. 2.18). The commands for enabling or disabling the auto-rotation option are:

```
/gate/Name_Volume/ring/enableAutoRotation
/gate/Name_Volume/ring/disableAutoRotation
```

A volume can also be shifted along Z periodically. Each element of a sequence is shifted according to its position *inside* the sequence, defined as "j" below. In a sequence composed of $M_{ModuloNumber}$ elements, the shift values are defined as $Zshift_i \equiv Zshift_j$ where :

- i is the position in the full ring
- j =(i % $M_{ModuloNumber}$)+1 is the position in a sequence, starting at 1.

To set a shift and the value of this shift, use:

```
/gate/Name_Volume/ring/setModuloNumber 1
/gate/Name_Volume/ring/setZShift1 Z mm
```

Up to 8 shifts and different shift values can be defined (setZShift1 to setZShift8).

Remark: This geometry description conforms to the document "List Mode Format Implementation: Scanner geometry description Version 4.1 M.Krieguer et al " and is fully described in the LMF output, in particular in the ASCII header file entry:

z shift sector j mod $M_{ModuloNumber}$: Zshift_j units

Here j (j starting here at 0) stands for the $n^{th.}$ object being shifted each $M_{ModuloNumber}$ object. Each shift value introduced in the command line below corresponds to a new line in the .cch file.

The LMF version 22.10.03 supports a geometry with a cylindrical symmetry. As an example, a repeater starting at 0 degree and finishing at 90 degree (a quarter of ring) will not be supported by the LMF output.

- Example 1:

```
/gate/hole/repeaters/insert          ring
/gate/hole/ring/setRepeatNumber      10
/gate/hole/ring/setPoint1            0. 1. 0. mm
/gate/hole/ring/setPoint2            0. 0. 0. mm
```

The *hole* volume is repeated 10 times around the Y axis. The application of this ring repeater is illustrated in Fig. 2.19.

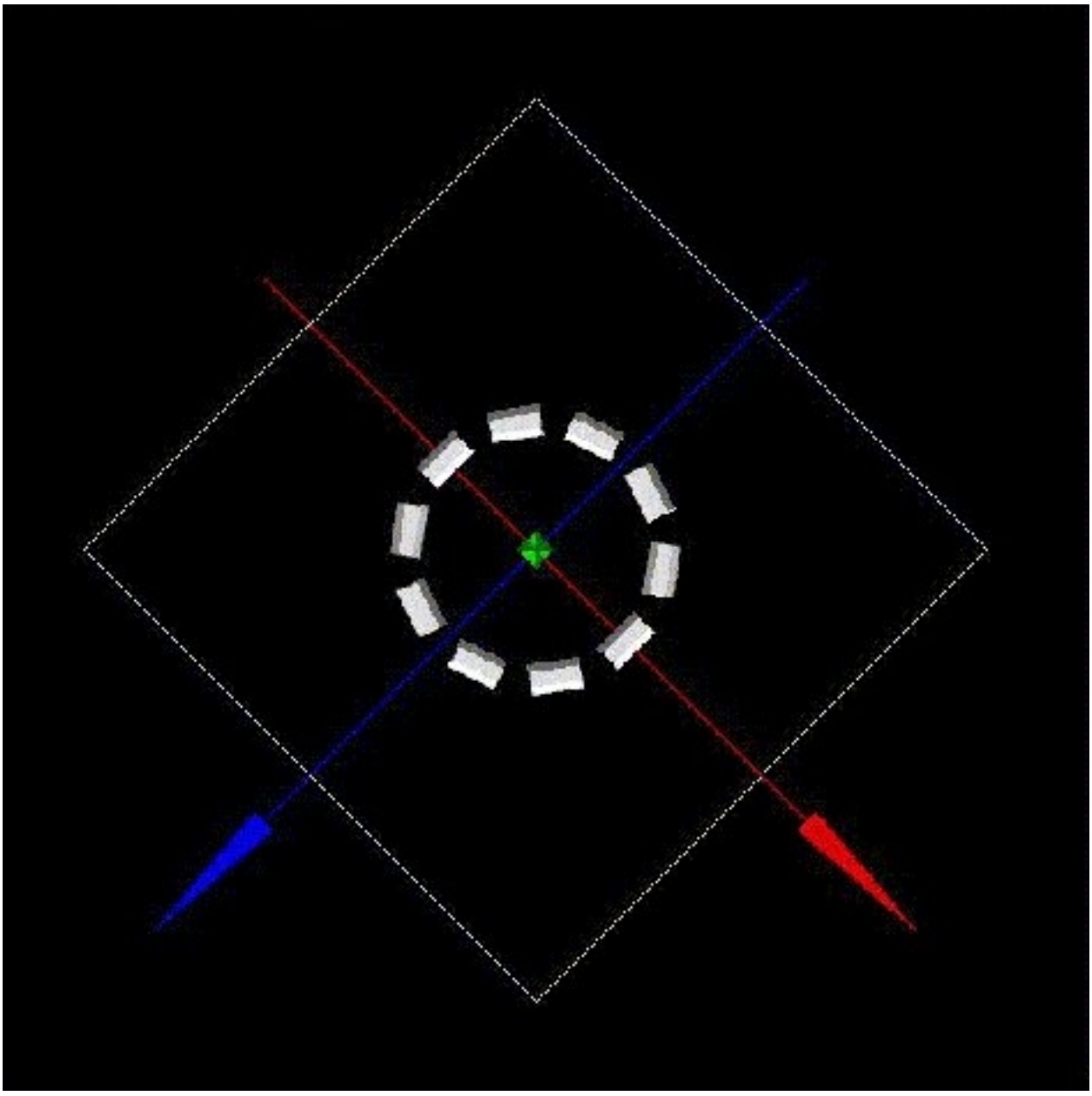


Fig. 2.17: Illustration of the application of the auto-rotation option

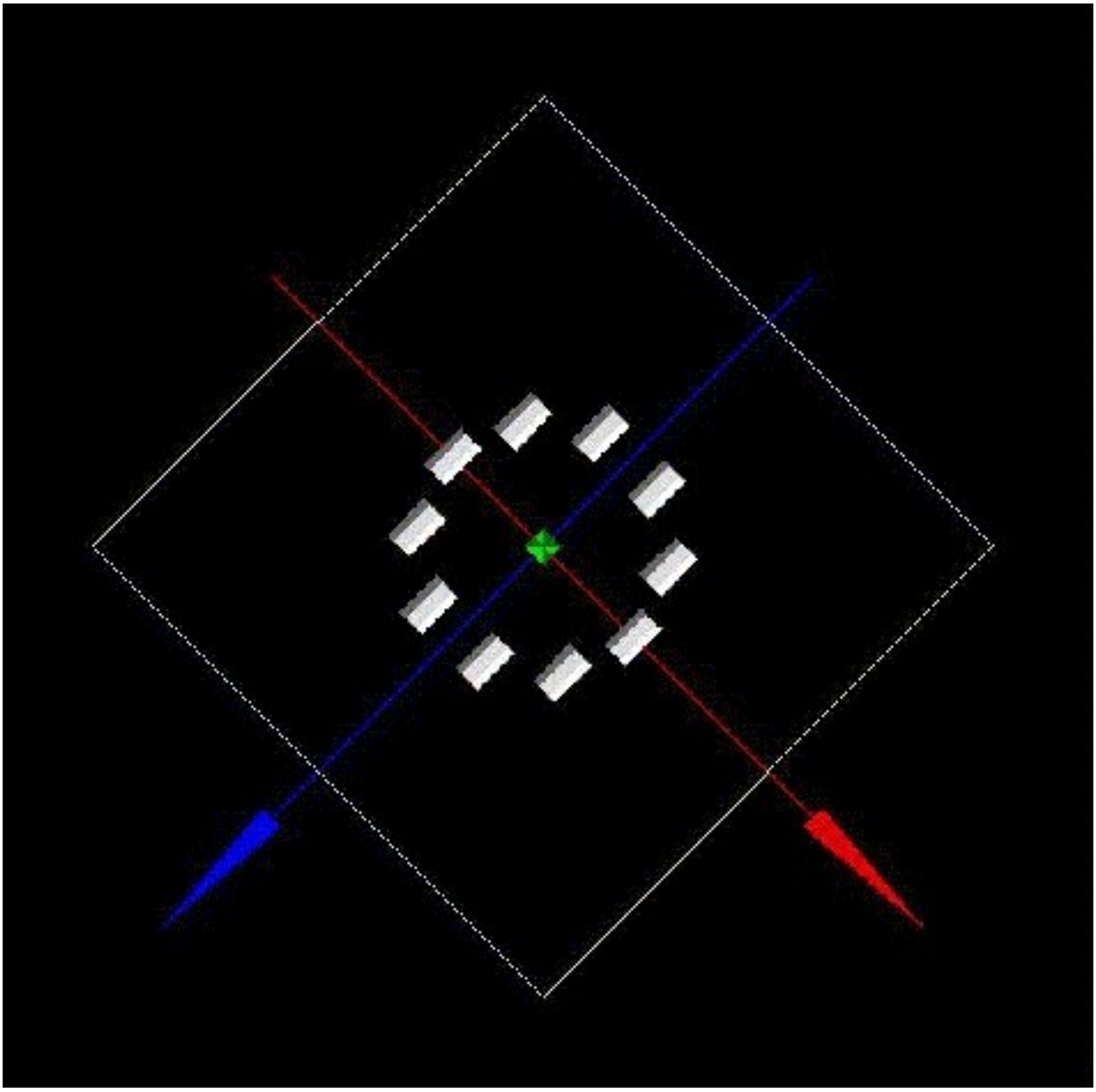


Fig. 2.18: Illustration of the application of the ring-repeater when the auto-rotation option is disabled

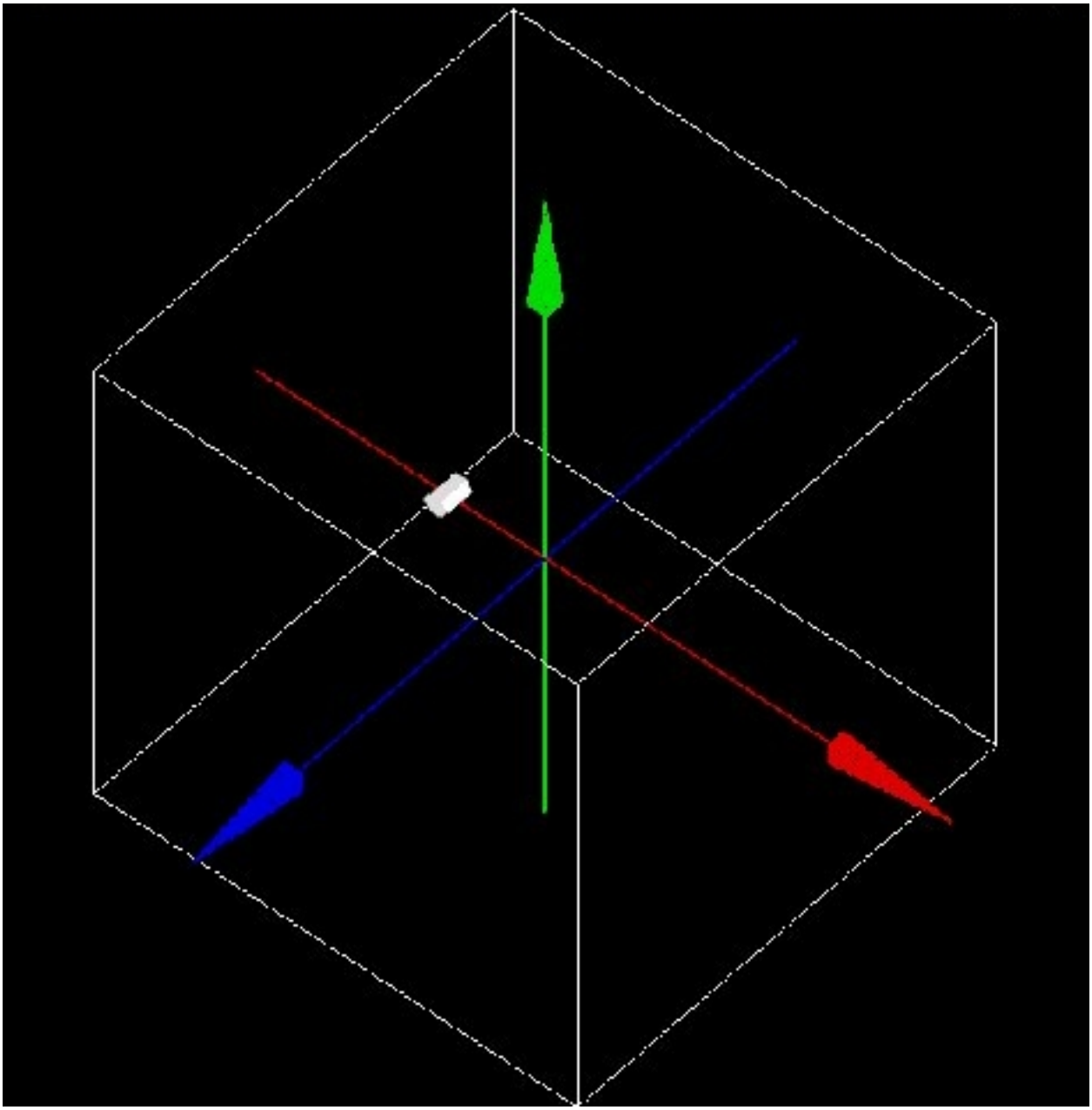


Fig. 2.19: Illustration of the application of the ring repeater

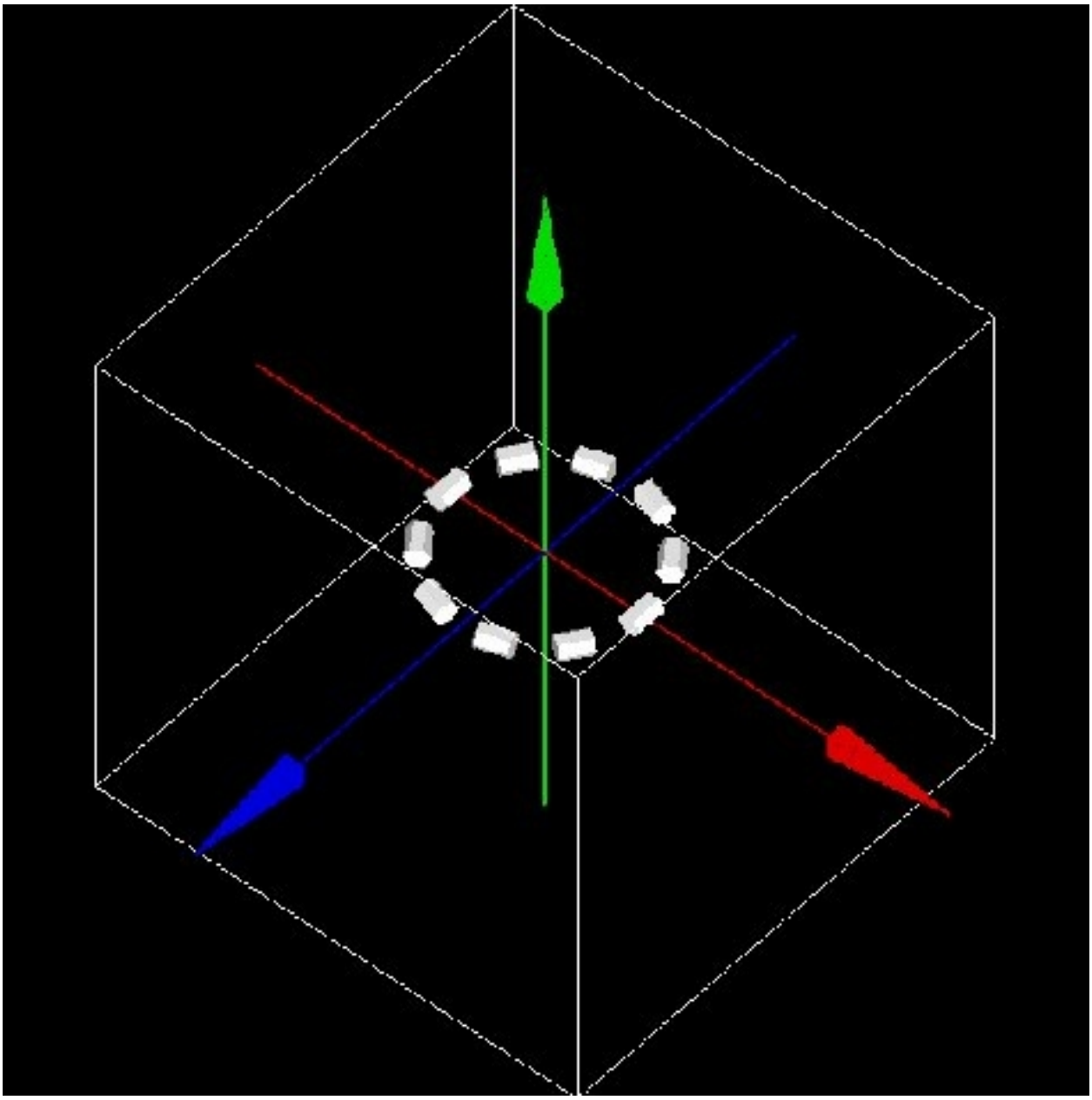


Fig. 2.20: Illustration of the application of the ring repeater

- Example 2:

```
/gate/rsector/repeaters/insert       ring
/gate/rsector/ring/setRepeatNumber   20
/gate/rsector/ring/setModuloNumber   2
/gate/rsector/ring/setZShift1        -3500 mum
/gate/rsector/ring/setZShift2        +3500 mum
/gate/rsector/ring/enableAutoRotation
```

The *rsector* volume is repeated 20 times along a ring. The sequence length is 2, with the first and the second volume shifted by -3500 µ m and 3500 µ m respectively. The *rsector* volume could also include several volumes itself, each of them being duplicated, which is illustrated in Fig. 2.21.

Cubic array repeater

The cubic array repeater is appropriate to repeat a volume along one, two or three axes. It is useful to build a collimator for SPECT simulations.

To select the cubic array repeater, use:

```
/gate/Name_Volume/repeaters/insert cubicArray
```

To define the number of times *Nx, Ny and Nz* the volume *Name_Volume* has to be repeated along the X, Y and Z axes respectively, use:

```
/gate/hole/cubicArray/setRepeatNumberX Nx
/gate/hole/cubicArray/setRepeatNumberY Ny
/gate/hole/cubicArray/setRepeatNumberZ Nz
```

To define the step of the repetition *X mm*, *Y mm* and *Z mm* along the X, Y and Z axes respectively, use:

```
/gate/hole/cubicArray/setRepeatVector X Y Z mm
```

The autocentering options are available for the cubic array repeater. If a volume is initially at a position P, the set of volumes after the repeater has been applied is centered on P if autoCenter is true (default). If autoCenter is false, the first copy of the group is centered on P.

- Example:

```
/gate/hole/repeaters/insert              cubicArray
/gate/hole/cubicArray/setRepeatNumberX   1
/gate/hole/cubicArray/setRepeatNumberY   5
/gate/hole/cubicArray/setRepeatNumberZ   2
/gate/hole/cubicArray/setRepeatVector    0. 5. 15. cm
```

The *hole* volume is repeated 5 times each 5 cm along the Y axis and twice each 15 cm along the Z axis. The application of this cubic array repeater is illustrated in figure Fig. 2.22.

Quadrant repeater

The quadrant repeater is appropriate to repeat a volume in a triangle-like pattern similar to that of a Derenzo resolution phantom.

To select the quadrant repeater, use:

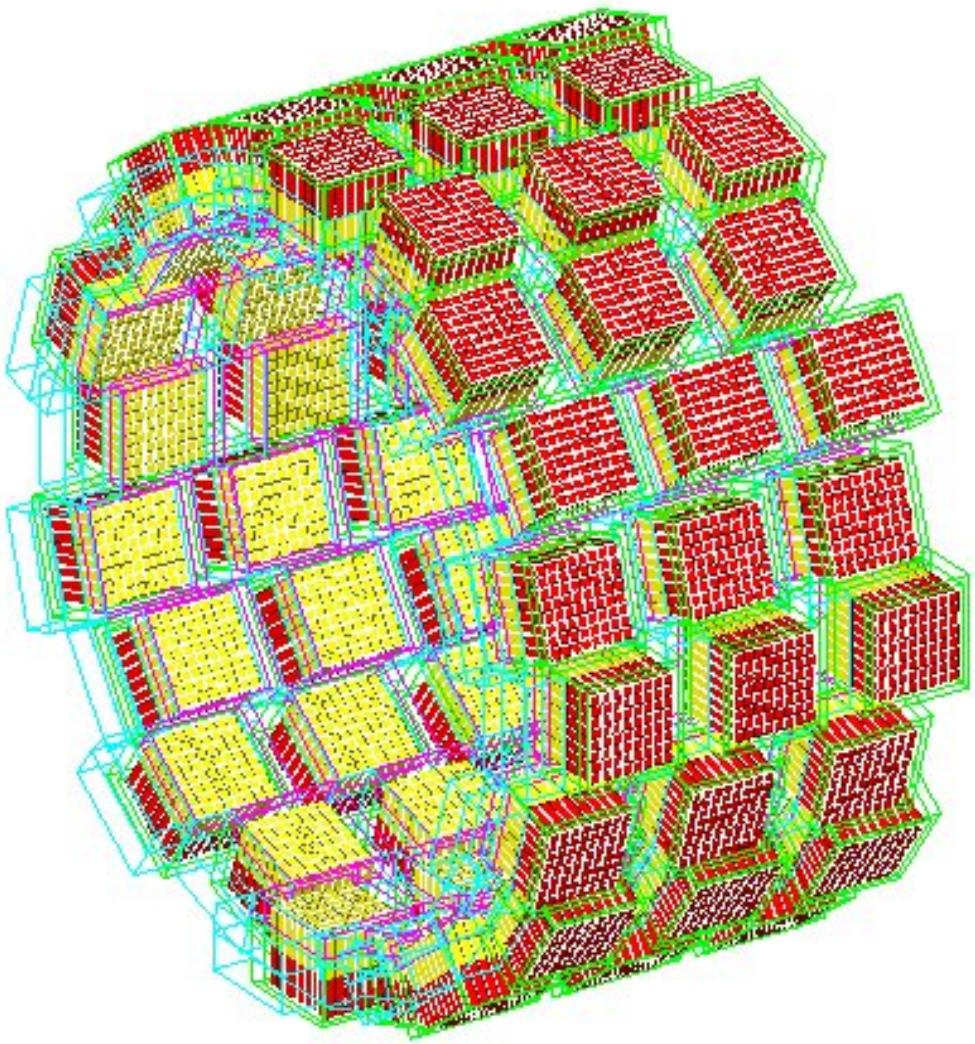


Fig. 2.21: Example of a ring repeater with a shift. An array of 3 crystal matrices has been repeated 20 times with a modulo N=2 shift

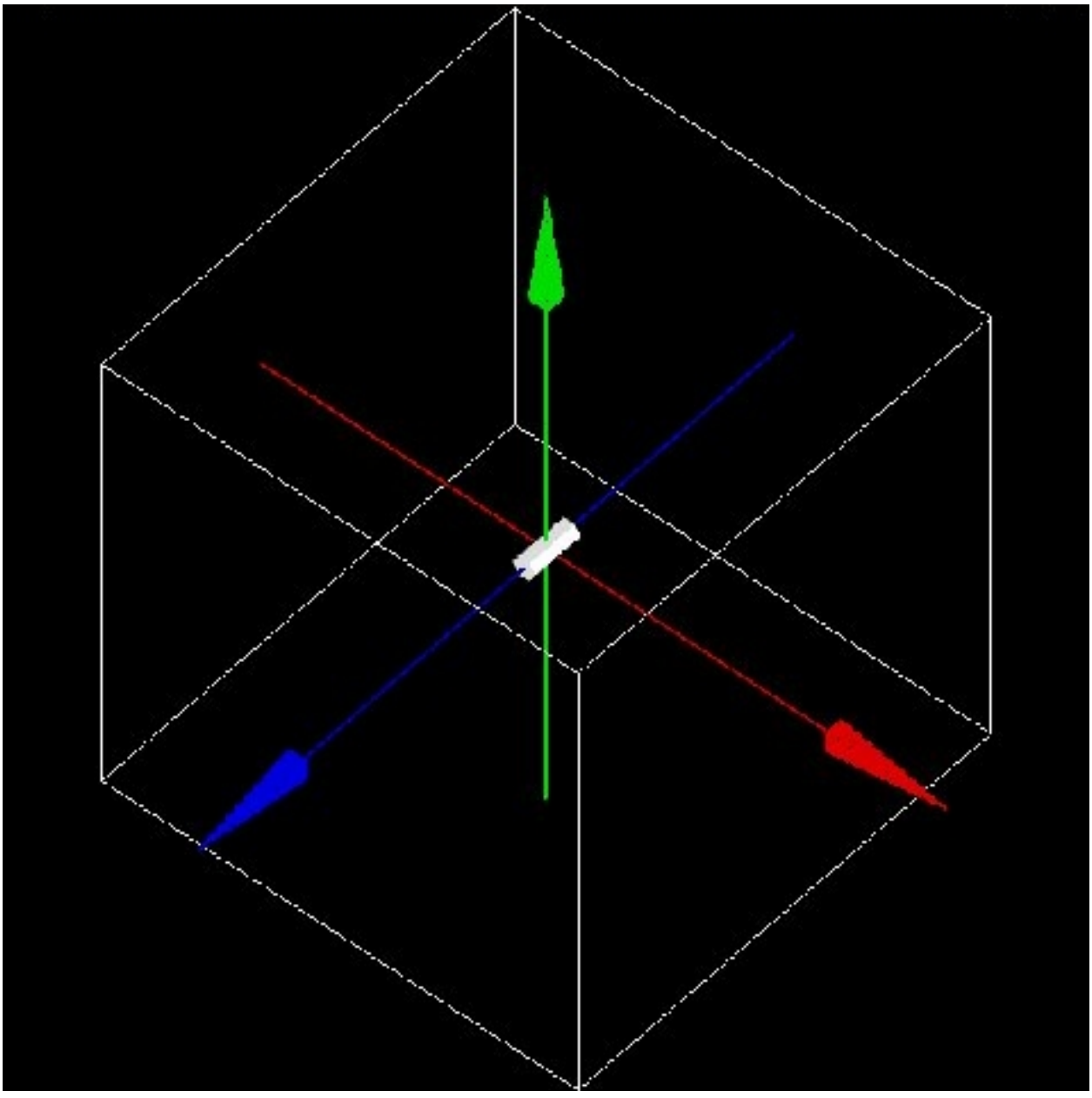


Fig. 2.22: Illustration of the application of the cubic array repeater

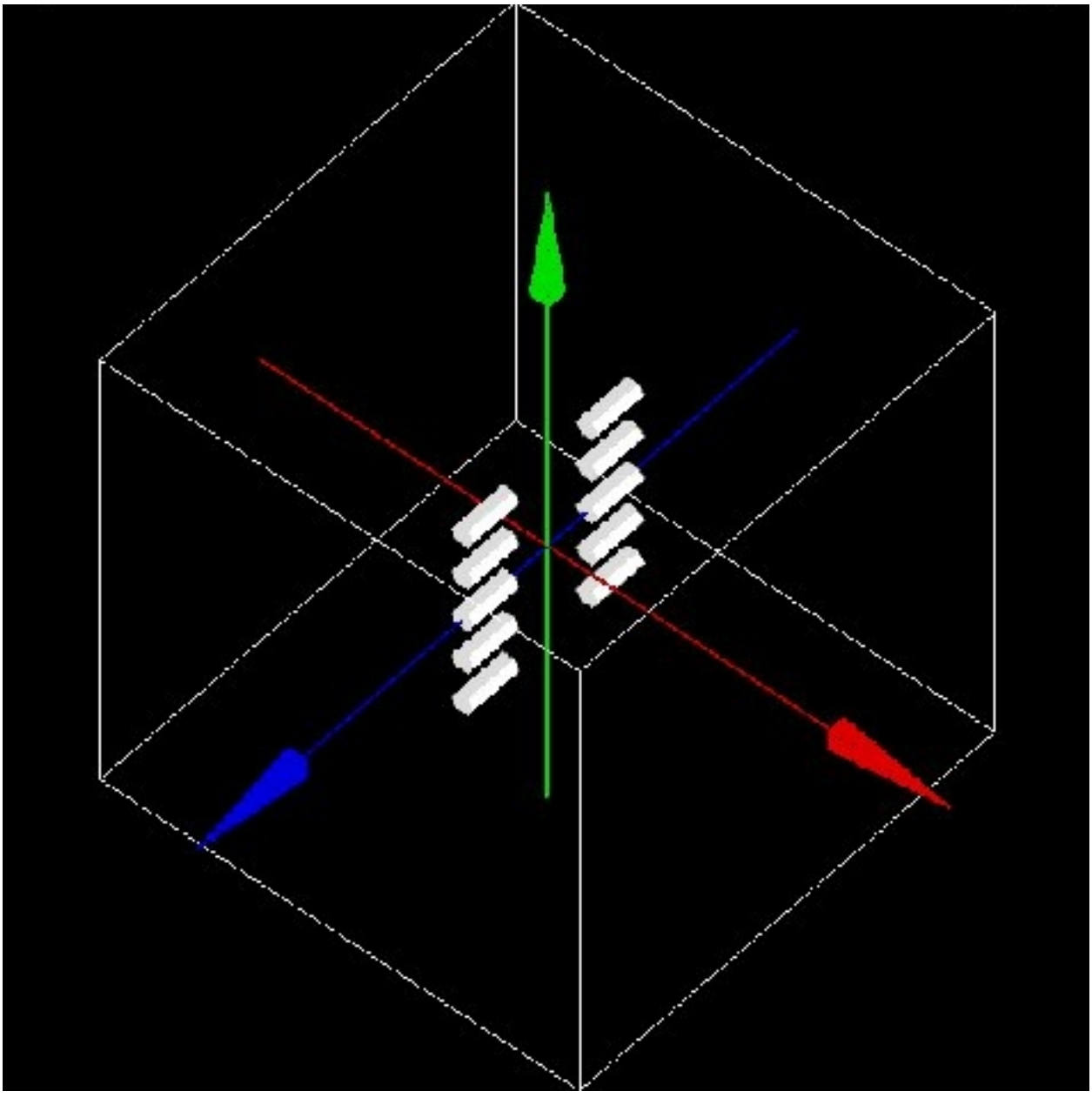


Fig. 2.23: Illustration of the application of the cubic array repeater (after)

```
/gate/Name_Volume/repeaters/insert quadrant
```

To define the number of repetition lines, use:

```
/gate/hole/quadrant/setLineNumber X
```

To define the orientation of the quadrant (the direction of line repetition), use:

```
/gate/hole/quadrant/setOrientation N deg
```

To define the distance between adjacent copies, use:

```
/gate/hole/quadrant/setCopySpacing xx cm
```

To define the maximum range of the repeater which is the maximum distance between a copy and the original volume, use:

```
/gate/hole/quadrant/setMaxRange xx cm
```

This command can be used to remove corner-copies that would fall outside your phantom

- Example:

```
/gate/hole/repeaters/insert           quadrant
/gate/hole/quadrant/setLineNumber     5
/gate/hole/quadrant/setOrientation    90 deg
/gate/hole/quadrant/setCopySpacing    6 cm
/gate/hole/quadrant/setMaxRange       30 cm
```

The *hole* volume is repeated in a triangle-like pattern. The application of this quadrant repeater is illustrated in Fig. 2.15.

Remark: The repeaters that are applied to the *Name_Volume* volume can be listed using:

```
/gate/Name_Volume/repeaters/list
```

Sphere repeater

The sphere repeater makes it possible to repeat a volume along a spherical ring. It is useful to build rings of detectors for PET scanners having gantry of spherical shape (e.g. SIEMENS Ecat Accel, Hi-Rez, ...)

To select the sphere repeater, use:

```
/gate/Name_Volume/repeaters/insert sphere
```

Then, the radius R of the sphere can be set using:

```
/gate/Name_Volume /sphere/setRadius X cm
```

To define the number of times N1 and N2 the volume *Name_Volume* has to repeated in the transaxial plane and the axial plane respectively, use:

```
/gate/Name_Volume/sphere/setRepeatNumberWithTheta N1
/gate/Name_Volume/sphere/setRepeatNumberWithPhi N2
```

To set the rotation angle difference between two adjacent copies in the transaxial direction, use:

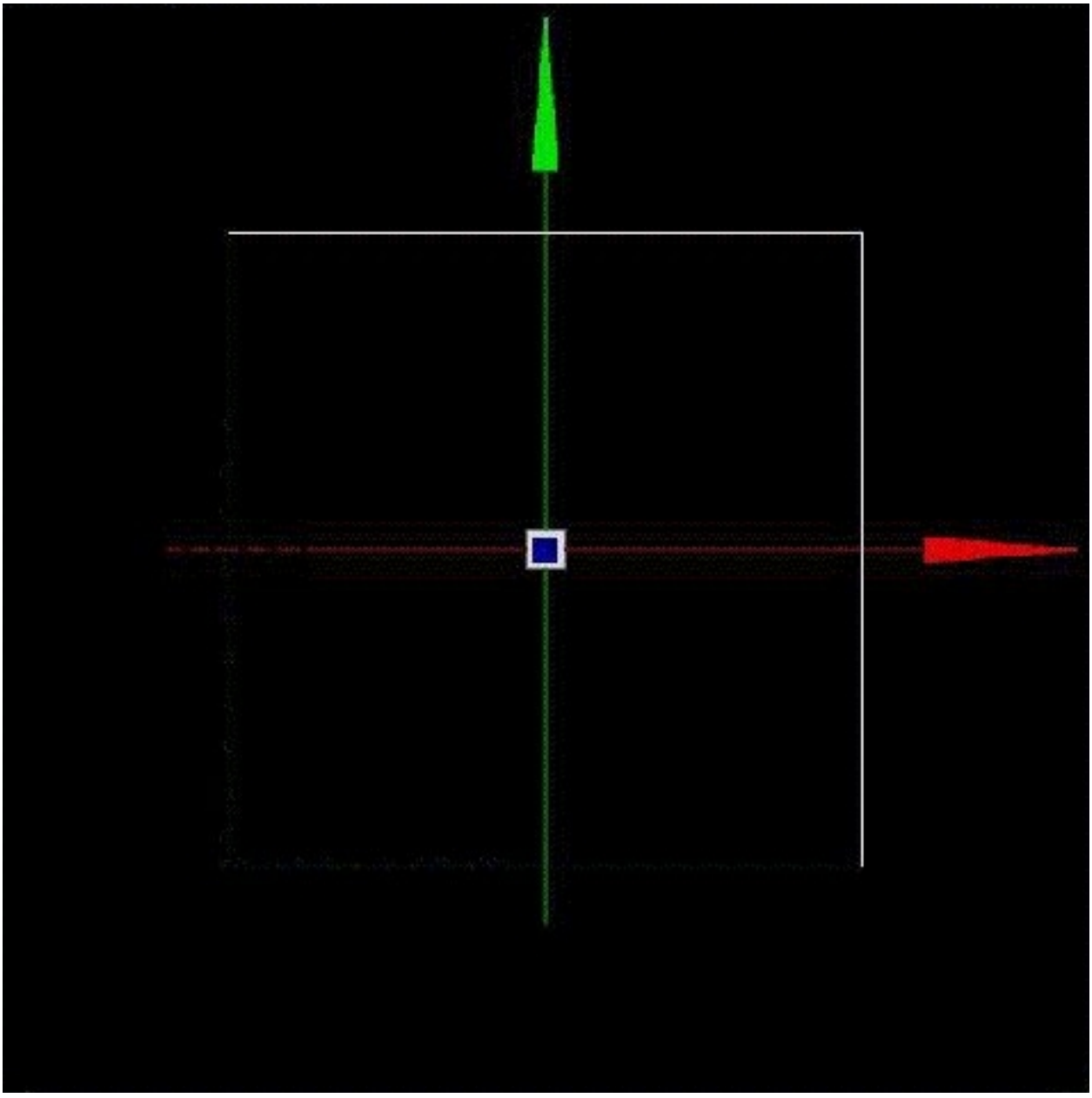


Fig. 2.24: Illustration of the application of the cubic array repeater

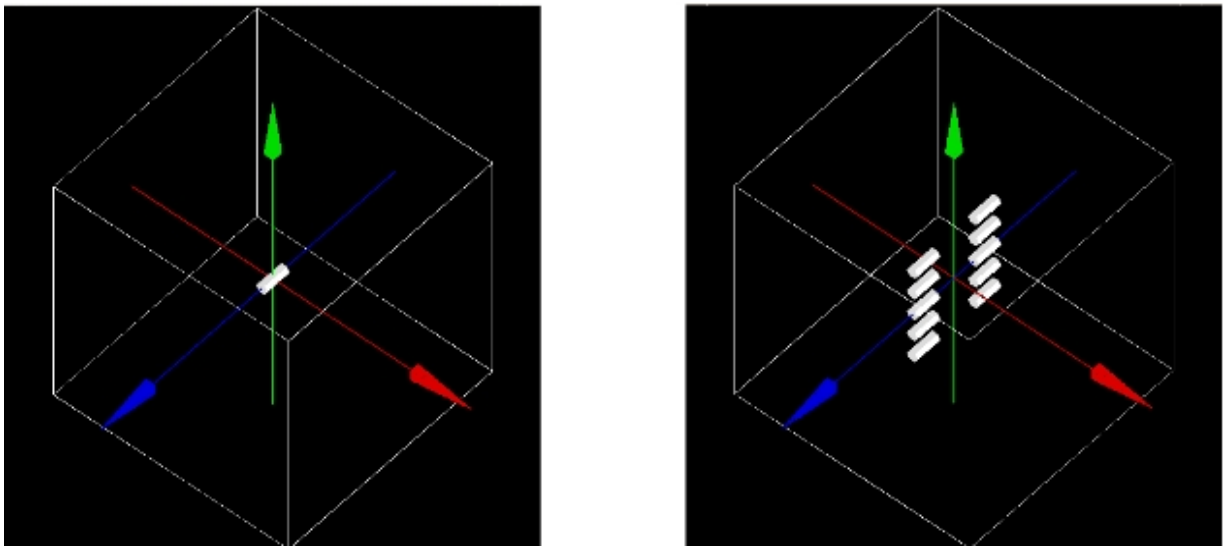


Fig. 2.25: Illustration of the application of the cubic array repeater (after)

```
/gate/Name_Volume/sphere/setThetaAngle x deg
```

To set the rotation angle difference between two adjacent copies in the axial direction, use:

```
/gate/Name_Volume/sphere/setPhiAngle y deg
```

The replicates of the volume *Name_Volume* will be placed so that its axis remains tangential to the ring.

Example Fig. 2.26:

```
/gate/block/repeaters/insert                    sphere
/gate/block/sphere/setRadius                    25. cm
/gate/block/sphere/setRepeatNumberWithTheta     10
/gate/block/sphere/setRepeatNumberWithPhi       3
/gate/block/setThetaAngle                       36 deg
/gate/block/setThetaAngle                       20 deg
```

The block volume is repeated 10 times along the transaxial plane, with a rotation angle between two neighbouring blocks of 36 deg, and is repeated 3 times in the axial direction with a rotation angle between two neighbouring blocks of 20 deg. The sphere defined here has a 25 cm radius.

Generic repeater

It is also possible to repeat a volume according to a list of transformations (rotation and translation). The time column is ignored by the generic repeater.

The following macros read the transformations into a simple text file:

```
/gate/myvolume/repeaters/insert                          genericRepeater
/gate/myvolume/genericRepeater/setPlacementsFilename     data/myvolume.placements
/gate/myvolume/genericRepeater/useRelativeTranslation    1
```

The text file “myvolume.placements” is composed as follows:

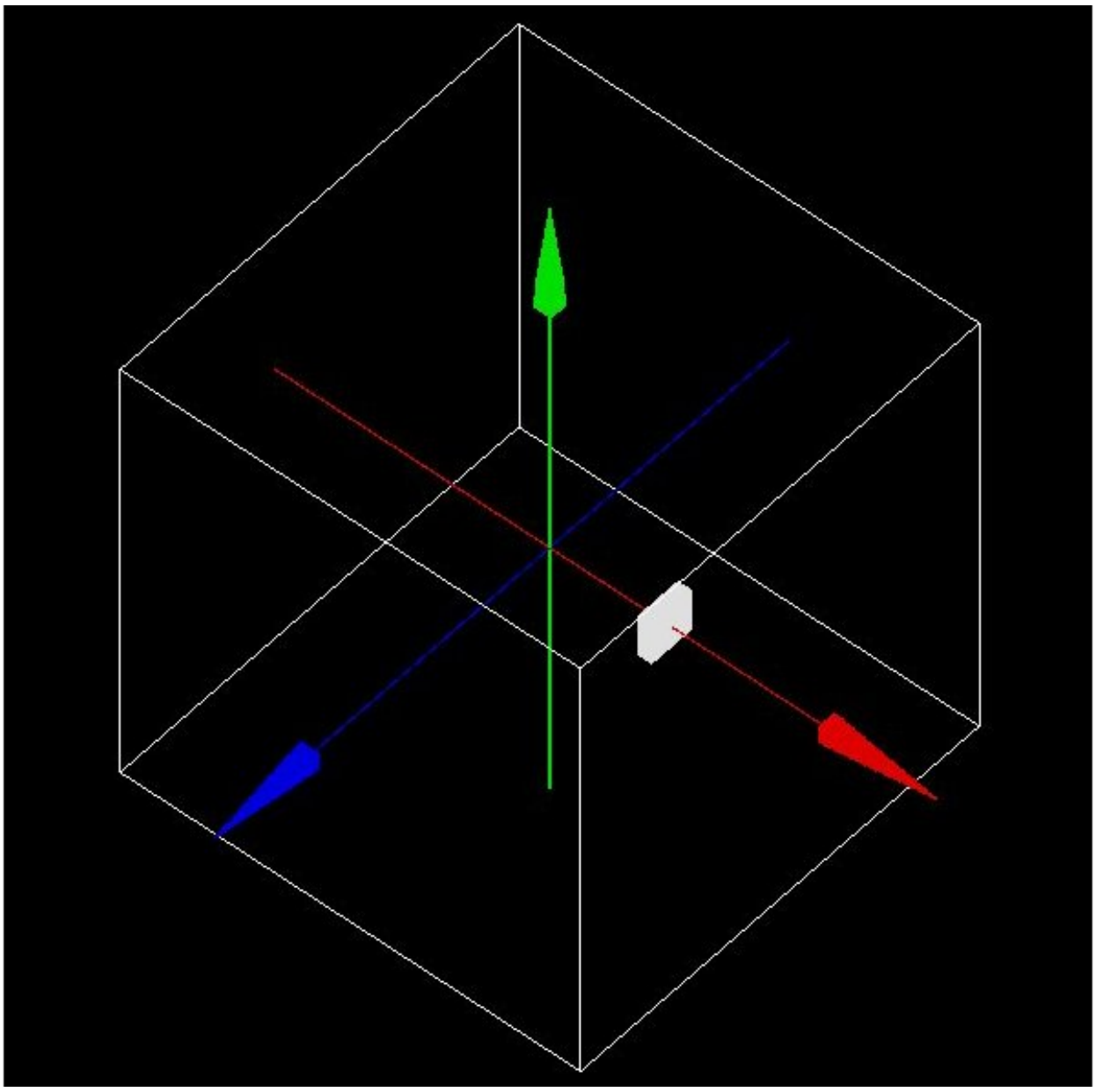


Fig. 2.26: Illustration of the application of the sphere repeater

```
###### List of placement (translation and rotation)
###### Column 1      is time in seconds
###### Column 2      is rotationAngle in degree
###### Columns 3,4,5 are rotation axis
###### Columns 6,7,8 are translation in mm
Time     s
Rotation deg
Translation mm
0      0       0 1 0       0 0 10
0      10      0 1 0       0 0 10
0      15      0 1 0       0 0 20
```

- line with # are ignored
- first word must be Time followed with the units
- next words must be Rotation then Translation followed with the units (deg and mm here)
- Rotation are described with 4 columns, the first for the angle, three others for the rotation axis
- Translation are described with X Y Z.
- using "useRelativeTranslation 1" (default) allows to compose the transformation according to the initial volume translation. If set to 0, the transformation is set as is (in the coordinate system of the mother volume).

See example *GateRT*

2.2.4 Placing a volume

The position of the volume in the geometry is defined using the sub-tree:

```
/placement/
```

Three types of placement are available: translation, rotation and alignment.

Translation

To translate the *Name_Volume* volume along the X direction by x cm, the command is:

```
/gate/Name_Volume/placement/setTranslation x. 0. 0. cm
```

The position is always given with respect to the center of the mother volume.

To set the Phi angle (in XY plane) of the translation vector, use:

```
/gate/Name_Volume/placement/setPhiOfTranslation N deg
```

To set the Theta angle (with regard to the Z axis) of the translation vector, use:

```
/gate/Name_Volume/placement/setThetaOfTranslation N deg
```

To set the magnitude of the translation vector, use:

```
/gate/Name_Volume/placement/setMagOfTranslation xx cm
```

- Example:

```
/gate/Phantom/placement/setTranslation          1. 0. 0. cm
/gate/Phantom/placement/setMagOfTranslation     10. cm
```

The *Phantom* volume is placed at 10 cm, 0 cm and 0 cm from the center of the mother volume (here the *world* volume). The application of this translation placement is illustrated in Fig. 2.27.

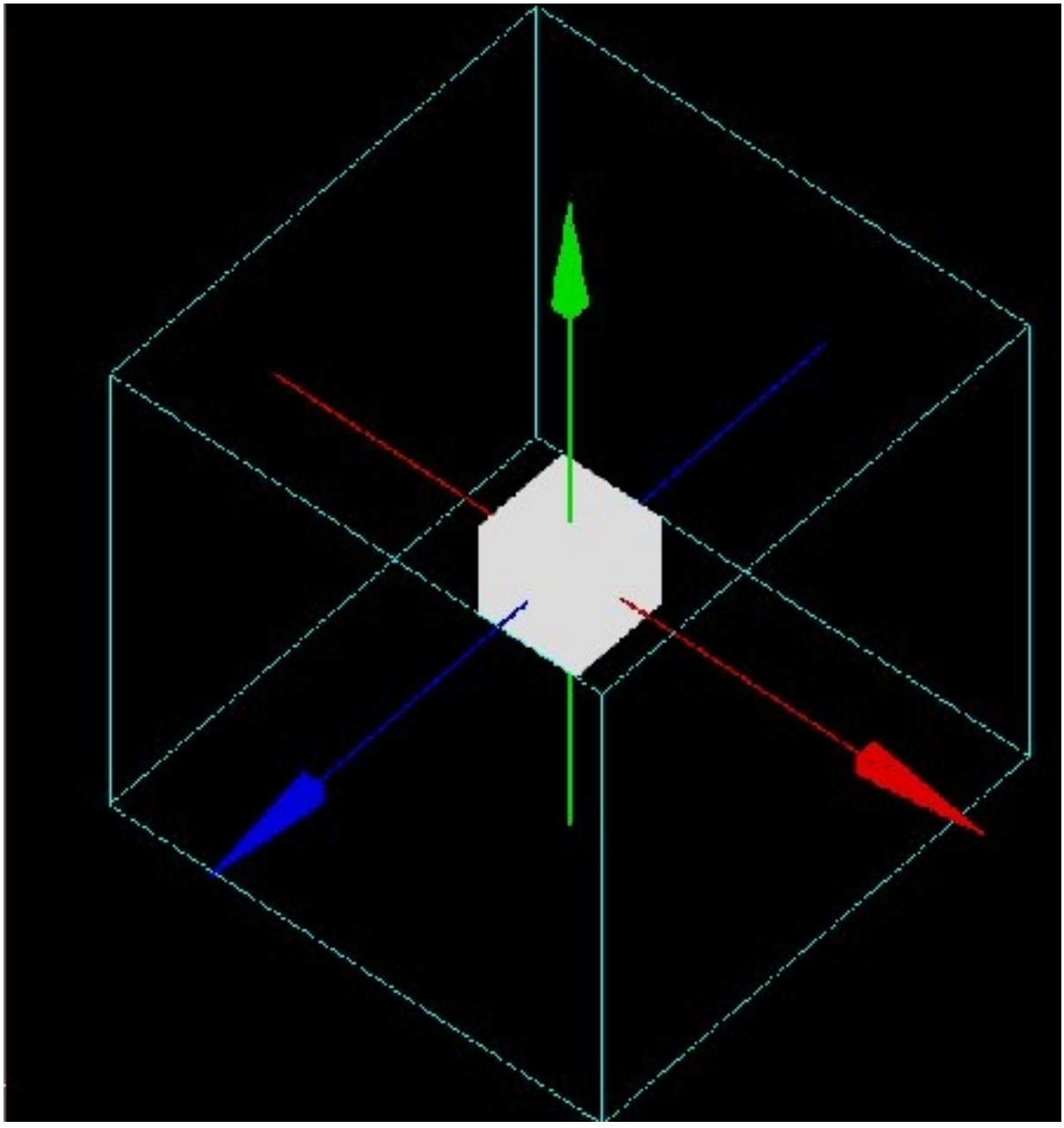


Fig. 2.27: Illustration of the translation placement

Rotation

To rotate the *Name_Volume* volume by *N* degrees around the *X* axis, the commands are:

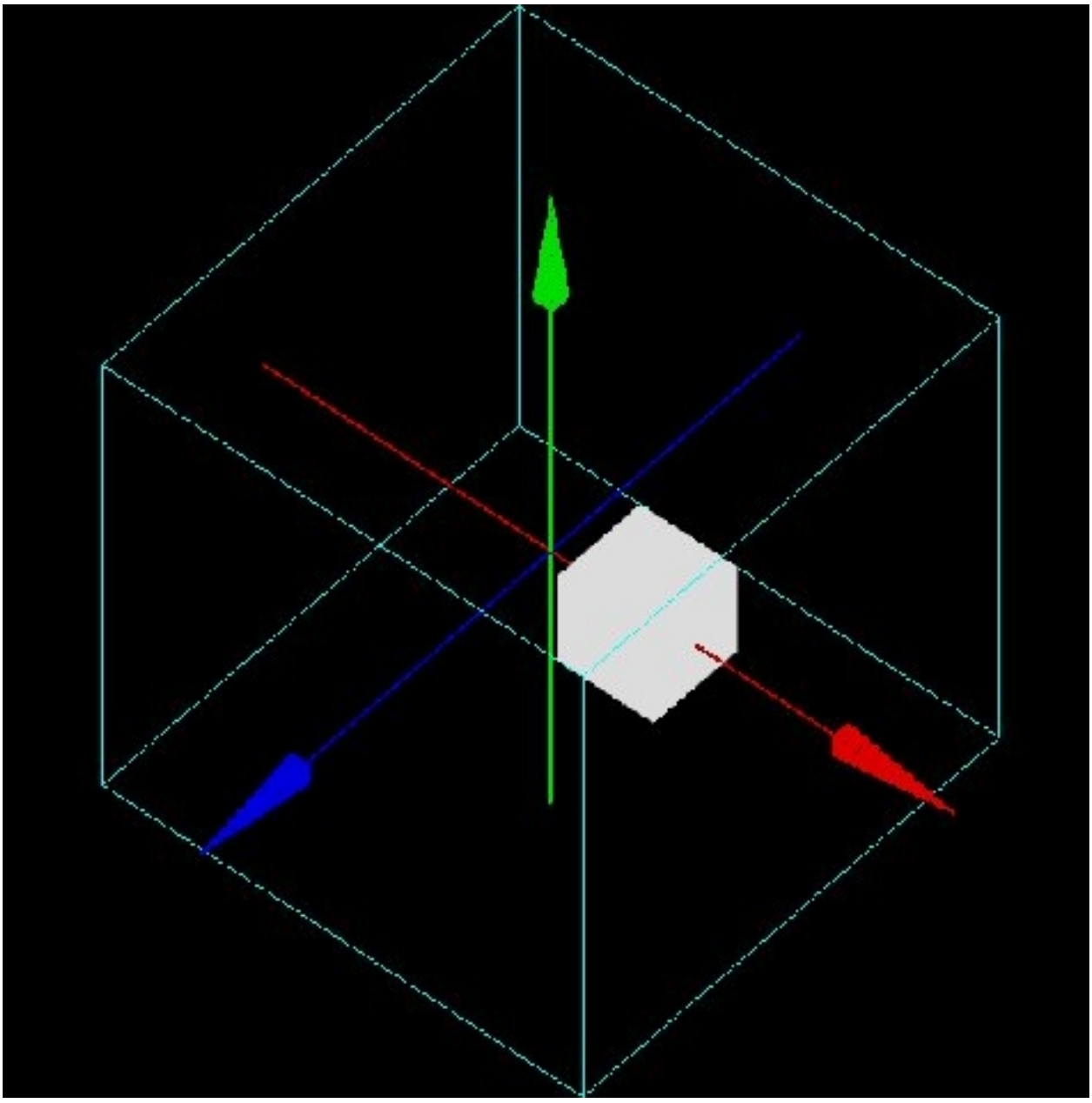


Fig. 2.28: Illustration of the translation placement

```
/gate/Name_Volume/placement/setRotationAxis    X 0 0
/gate/Name_Volume/placement/setRotationAngle   N deg
/gate/Name_Volume/placement/setAxis            0 1 0
```

The default rotation axis is the Z axis.

- Example:

```
/gate/Phantom/placement/setRotationAxis        0 1 0
/gate/Phantom/placement/setRotationAngle       90 deg
```

The *Phantom* volume is rotated by 90 degrees around the Y axis. The application of this rotation placement is illustrated in Fig. 2.29.

Alignment

Using the alignment command, a volume having an axis of symmetry (cylinder, ellipso, cone and hexagone) can be aligned parallel to one of the three axes of the axis system.

To align the *Name_Volume* volume along the X axis, use:

```
/gate/Name_Volume/placement/alignToX
```

The rotation parameters of the *Name_Volume* volume are then set to +90 degree around the Y axis.

To align the *Name_Volume* volume along the Y axis, use:

```
/gate/Name_Volume/placement/alignToY
```

The rotation parameters of the *Name_Volume* volume are then set to -90 degree around the X axis.

To align the *Name_Volume* volume along the Z axis (default axis of rotation) use:

```
/gate/Name_Volume/placement/alignToZ
```

The rotation parameters of the *Name_Volume* volume are then set to 0 degree.

Special example: Wedge volume and OPET scanner

The wedge is always created as shown in Fig. 2.13, that is with the slanted plane oriented towards the positive X direction. If one needs to have it oriented differently, one could, for instance, rotate it:

```
/gate/wedge0/placement/setRotationAxis 0 1 0
/gate/wedge0/placement/setRotationAngle 180 deg
```

The center of a wedge in the Y and Z directions are simply

$\frac{setYLength}{2}, \frac{setZLength}{2}$

respectively. For the X direction, the center is located such that

$2\Delta = \frac{setXLength+setNarrowerXLength}{2}$

where Delta is the length of the wedge across the middle of the Y direction, as shown in Fig. 2.31.

Wedge crystals are used to build the OPET scanner, in which the scanner ring geometry approximates a true circular ring.

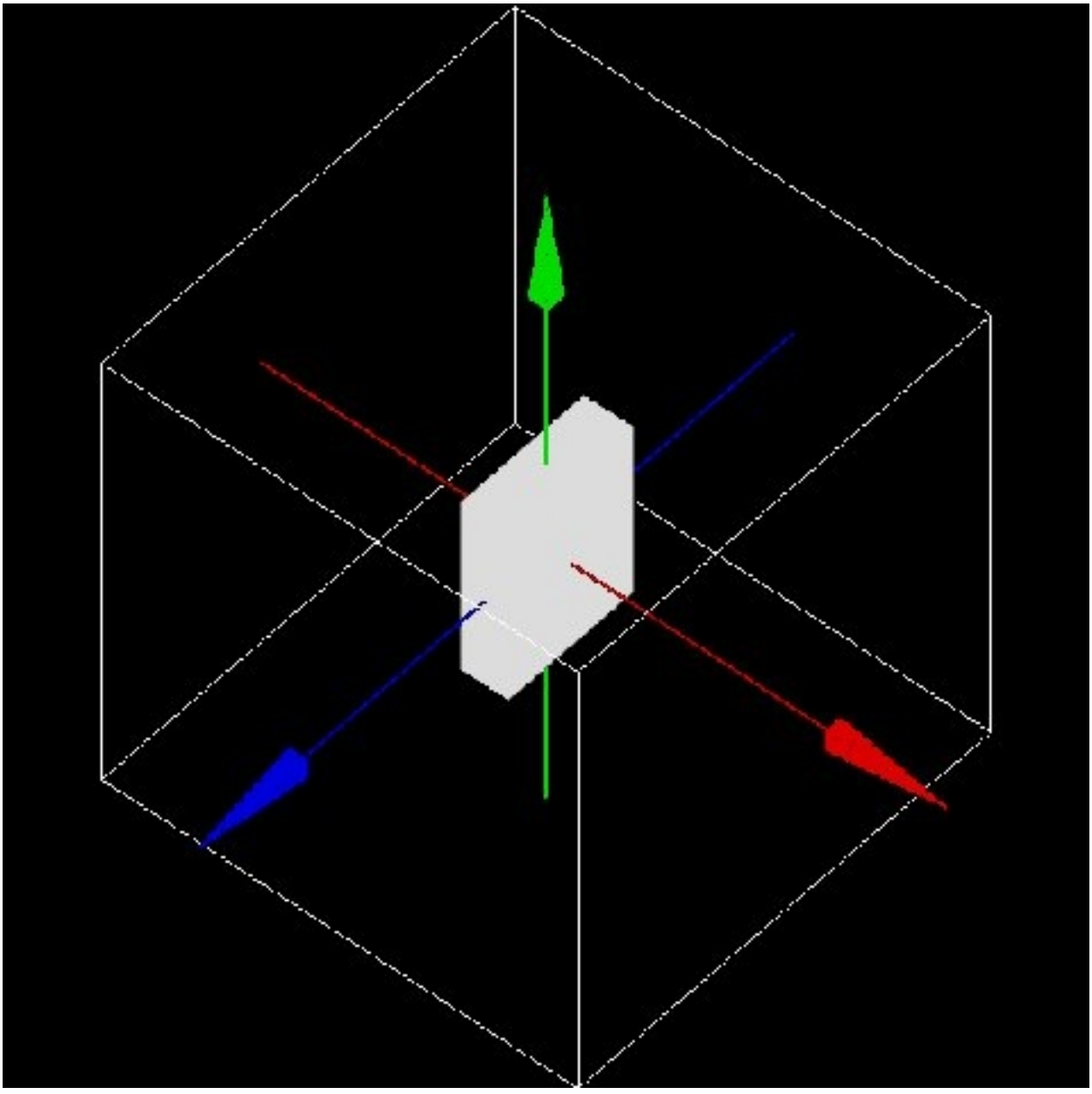


Fig. 2.29: Illustration of the rotation placement

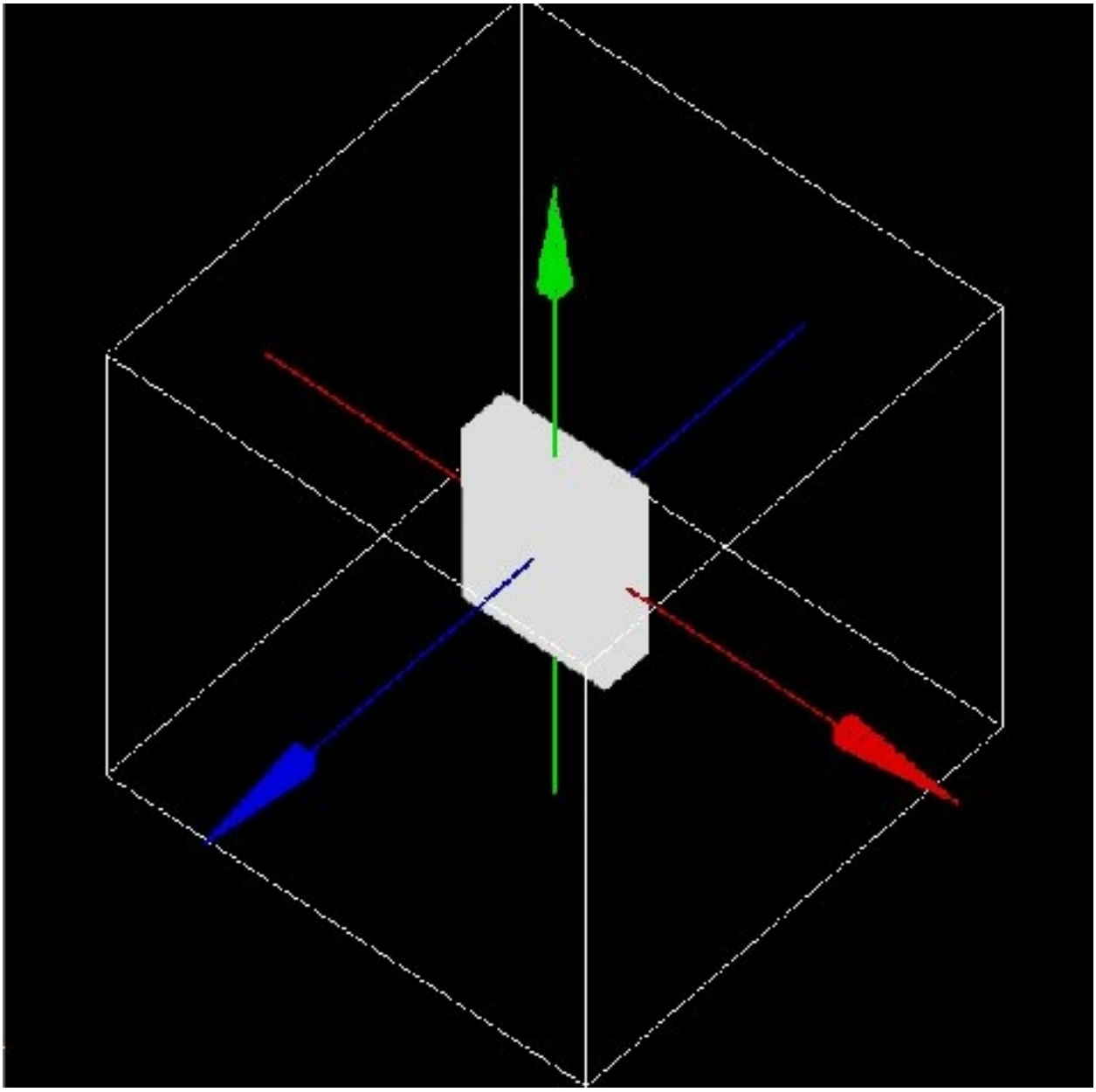


Fig. 2.30: Illustration of the rotation placement

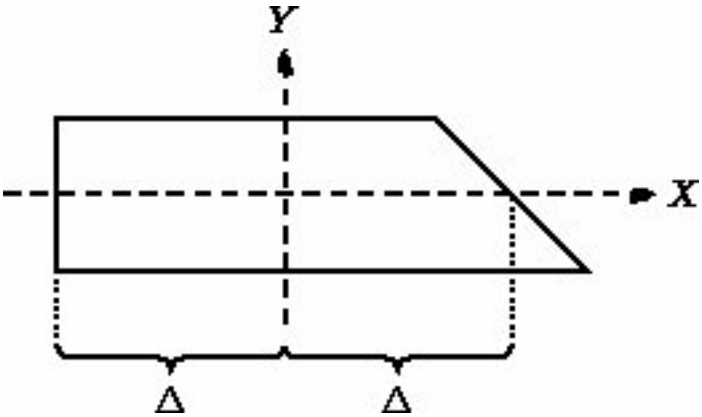


Fig. 2.31: Center of wedge

By knowing the radius gantry R and the length of the longest crystal, it is possible to arrange a series of 8 crystals with varying the lengths as shown in Fig. 2.32.

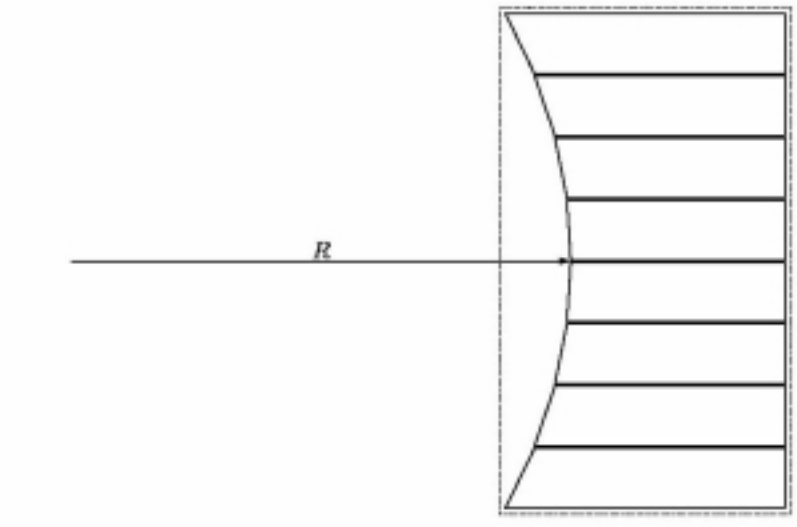


Fig. 2.32: A block approximating a true circular geometry

It is first necessary to create by-hand the first row of crystals. This is accomplished by first creating a module just big enough to contain one row of wedge crystals:

```
/gate/rsector/daughters/name          module
/gate/rsector/daughters/insert        box
/gate/module/geometry/setXLength      10 mm
/gate/module/geometry/setYLength      17.765 mm
/gate/module/geometry/setZLength      2.162 mm
/gate/module/setMaterial              Air
```

Then, a box that will contain the first wedge crystal is located inside the module:

```
/gate/module/daughters/name                   crystal0
/gate/module/daughters/insert                 box
/gate/crystal0/geometry/setXLength            10 mm
```

(continues on next page)

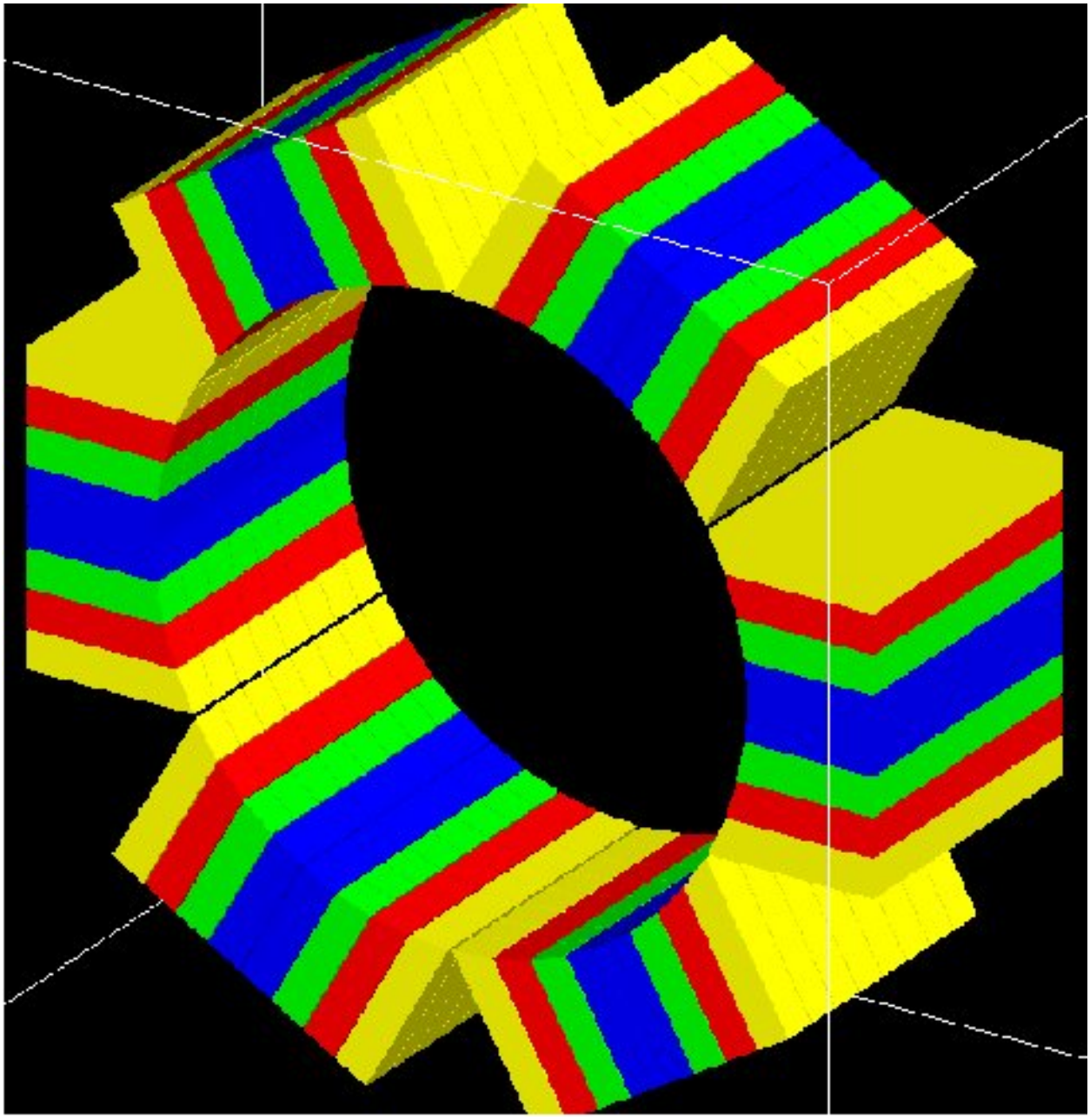


Fig. 2.33: The OPET scanner

(continued from previous page)

```
/gate/crystal0/geometry/setYLength           2.1620 mm
/gate/crystal0/geometry/setZLength           2.1620 mm
/gate/crystal0/placement/setTranslation      0. -7.8015 0. mm
/gate/crystal0/setMaterial                   Air
/gate/crystal0/vis/setColor                  black
/gate/crystal0/vis/setVisible                false
```

Finally, the actual crystal is placed inside its box:

```
/gate/crystal0/daughters/name                LSO0
/gate/crystal0/daughters/insert              wedge
/gate/LSO0/geometry/setXLength               10 mm
/gate/LSO0/geometry/setNarrowerXLength       8.921 mm
/gate/LSO0/geometry/setYLength               2.1620 mm
/gate/LSO0/geometry/setZLength               2.1620 mm
/gate/LSO0/placement/setRotationAxis         0 1 0
/gate/LSO0/placement/setRotationAngle        180 deg
/gate/LSO0/placement/setTranslation          0.2698 0. 0. mm
/gate/LSO0/setMaterial                       BGO
```

It is necessary to locate each crystal in separate "layers".

The last two steps are repeated for each crystal inside the module. Then the module is repeated along the Z axis and the block is repeated 6 times around the center of the scanner.

Fig. 2.33 shows the final OPET scanner.

2.2.5 Moving a volume

The GEANT geometry architecture requires the geometry to be static during a simulation. However, the typical duration of a single event (*e.g.* ps for the particle transport, µs for scintillation, or ms for the response of the electronics) is very short when compared to most of the geometrical changes to be modeled (*e.g.* movements of the phantom or of the detector or bio-kinetics). Therefore, the elements of the geometry are considered to be at rest during each time-step. Between every time-step, the position and the orientation of a subset of daughter volumes can be changed to mimic a movement such as a rotation or a translation. These displacements are parametrized by their velocity. Hence, the amplitude of the volume displacement is deduced from the duration of the time-step multiplied by the velocity of the displacement.

Given the speed of the components of the geometry, it is the responsibility of the user to set the time step duration short enough in order to produce smooth changes.

A volume can be moved during a simulation using five types of motion: rotation, translation, orbiting, wobbling and eccentric rotation, as explained below.

Translation

To translate a *Name_Volume* volume during the simulation, the commands are:

```
/gate/Name_Volume/moves/insert translation
/gate/Name_Volume/translation/setSpeed x 0 0 cm/s
```

where x is the speed of translation and the translation is performed along the X axis. These commands can be useful to simulate table motion during a scan for instance.

- Example:

```
/gate/Table/moves/insert           translation
/gate/Table/translation/setSpeed   0 0 1 cm/s
```

The *Table* volume is translated along the Z axis with a speed of 1 cm per second.

Rotation

To rotate a *Name_Volume* volume around an axis during the simulation, with a speed of N degrees per second, the commands are:

```
/gate/Name_Volume/moves/insert rotation
/gate/Name_Volume/rotation/setSpeed N deg/s
/gate/Name_Volume/rotation/setAxis 0 y 0
```

- Example:

```
/gate/Phantom/moves/insert         rotation
/gate/Phantom/rotation/setSpeed    1 deg/s
/gate/Phantom/rotation/setAxis     0 1 0
```

The *Phantom* volume rotates around the Y axis with a speed of 1 degree per second.

Orbiting

Rotating a volume around any axis during a simulation is possible using the orbiting motion. This motion is needed to model the camera head rotation in SPECT. To rotate the *Name_Volume* volume around the X axis with a speed of N degrees per second, the commands are:

```
/gate/SPECThead/moves/insert orbiting
/gate/SPECThead/orbiting/setSpeed N. deg/s
/gate/SPECThead/orbiting/setPoint1 0 0 0 cm
/gate/SPECThead/orbiting/setPoint2 1 0 0 cm
```

The last two commands define the rotation axis.

It is possible to enable or disable the volume auto-rotation option using:

```
/gate/Name_Volume/orbiting/enableAutoRotation
/gate/Name_Volume/orbiting/disableAutoRotation
```

Example:

```
/gate/camera_head/moves/insert            orbiting
/gate/camera_head/orbiting/setSpeed       1. deg/s
/gate/camera_head/orbiting/setPoint1      0 0 0 cm
/gate/camera_head/orbiting/setPoint2      0 0 1 cm
```

The *camera_head* volume is rotated around the Z axis during the simulation with a speed of 1 degree per second.

Wobbling

The wobbling motion enables an oscillating translation movement to the volume.

This motion is needed to mimic the behavior of certain PET scanners that wobble to increase the spatial sampling of the data during the acquisition.

The movement that is modeled is defined by $dM(t) = A.sin(2.PI.f.t + phi)$ where dM(t) is the translation vector at time t, A is the maximum displacement vector, f is the movement frequency, phi is the phase at t=0, and t is the time.

To set the parameters of that equation, use:

```
/gate/Name_Volume/moves/insert osc-trans
```

To set the amplitude vector of the oscillating translation:

```
/gate/Name_Volume/osc-trans/setAmplitude x. 0. 0. cm
```

To set the frequency of the oscillating translation:

```
/gate/Name_Volume/osc-trans/setFrequency N Hz
```

To set the period of the oscillating translation:

```
/gate/Name_Volume/osc-trans/setPeriod N s
```

To set the phase at t=0 of the oscillating translation:

```
/gate/Name_Volume/osc-trans/setPhase N deg
```

- Example:

```
/gate/crystal/moves/insert               osc-trans
/gate/crystal/osc-trans/setAmplitude     10. 0. 0. cm
/gate/crystal/osc-trans/setFrequency     50 Hz
/gate/crystal/osc-trans/setPeriod        1 s
/gate/crystal/osc-trans/setPhase         90 deg
```

In this example, the movement that is modeled is defined by $dM(t) = 10.sin(100.PI.t + 90)$

Eccentric rotation

The eccentric rotation motion enables an eccentric rotation movement of the volume. It is a particular case of the orbiting movement. To set the object in eccentric position (X-Y-Z) and rotate it around the OZ lab frame axis, use:

```
/gate/Name_Volume/moves/insert eccent-rot
```

To set the shifts in the X-Y-Z directions:

```
/gate/Name_Volume/eccent-rot/setShiftXYZ x y z cm
```

To set the orbiting angular speed:

```
/gate/Name_Volume/eccent-rot/setSpeed N deg/s
```

Remark: This particular move is closely related to the LMF definition since the move parameters (shifts in all 3 directions and angular speed) are propagated in the .cch header.

- Example:

```
/gate/crystal/moves/insert                 eccent-rot
/gate/crystal/eccent-rot/setShiftXYZ       5. 0. 0. cm
/gate/crystal/eccent-rot/setSpeed          10 deg/s
```

The *crystal* volume is placed at 10 cm, 0 cm and 0 cm from the center of its mother volume and will rotate around the Z axis during the simulation with a speed of 10 degrees per second.

Generic move

A volume can be move at given time value thanks to the following macros:

```
/gate/myvolume/moves/insert                          genericMove
/gate/myvolume/genericMove/setPlacementsFilename  data/myvolume.placements
```

In the same idea than *Generic repeater*, the placements file contains the transformations (rotation, translation) and the time value where this transformations is applied:

```
###### List of placement (translation and rotation) according to time
###### Column 1      is Time in s (second)
###### Column 2      is rotationAngle in degree
###### Columns 3,4,5 are rotation axis
###### Columns 6,7,8 are translation in mm
Time s
Rotation deg
Translation mm
0           0      0 1 0      0  0 100
250.7       3      0 1 0      0 10 100
492.9       4      0 1 0      0 20 100
742.9       8      0 1 0      30 0 100
```

WARNING. The time values given here do not necessarily correspond to simulation's *run*. The real runs are defined with the time slices (see *Eighth step: Starting an acquisition* for example). At each new run, GATE looks into the time-placements list and chooses the one that corresponds to the starting time of the run. It leads that some placements can be not applied (if one run start before the placement time and the next run start after the next placement time). If run time is after the last placements time in the list, the last placements is applied.

See example *GateRT*

Generic repeater move

You can combine generic repeater and generic move to allow different repeated configurations according to time. This is for example useful to describe multi-leaf collimator from a single leaf which is repeated at different positions, and which move according to each beam:

```
/gate/myvolume/moves/insert                                  genericRepeaterMove
/gate/myvolume/genericRepeaterMove/setPlacementsFilename  data/myvolume.placements
/gate/myvolume/genericRepeaterMove/useRelativeTranslation 1

###### List of placement (translation and rotation)
###### Column 1      is rotationAngle in degree
###### Columns 2,3,4 are rotation axis
###### Columns 5,6,7 are translation in mm
Time s
NumberOfPlacements 3
Rotation deg
Translation mm
#Time  # Placement 1            # Placement 2           # Placement 3
0      10  0 1 0  20  0  0      10 0 1 0   80  0  0     10 0 1 0    -60  0  0
1      20  0 1 0  20 10  0      20 0 1 0   80 10  0     20 0 1 0    -60 10  0
```

(continues on next page)

(continued from previous page)

```
2       30  1 1 0  20  0  0       30 1 1 0   80  0  0     30 1 1 0    -60  0  0
4       40  0 1 1  20  0 40       40 0 1 1   80  0 40     40 0 1 1    -60  0 40
```

The 'NumberOfPlacements' is needed to indicate how many different repetition are performed at each motion.

2.2.6 Updating the geometry

Updating the geometry is needed to take into account any change in the geometry. It also refreshes the display window. The geometry can be updated with the following command:

```
/gate/geometry/rebuild
```

2.3 Materials

Table of Contents

2.3.1 The Gate material database

The primary method for defining the properties of the materials used in Gate is by a materials database. This file holds all the information required for Gate to assign the nuclear properties from the Geant4 data sets, and is easily modified by the user. The OpenGate collaboration supplies a fairly extensive listing of materials in this file as part of Gate. This chapter describes the details of how to modify this database.

As alluded to in the previous paragraph, there exists an alternate method for materials definitions. As discussed in previous chapters, Gate scripts are developed from Geant4 C++ data classes in order to simplify and standardize input for Geant4. As a result, materials definitions can be written and compiled in C++ directly using the Geant4 tools. Specifying materials in this manner is beyond the scope of this document. For those interested in direct access to Geant4's materials should refer to the *Geant4 User's Guide: For Application Developers* and the *Geant4 User's Guide: For Toolkit Developers* for more detailed information.

The material database contains two Geant4 structures called elements and materials that are used to define the physical properties of the atoms, molecules, and compounds. In contrast with Geant4, Gate does not use isotopic abundances.

This omission has little bearing on Gate applications because isotopic abundances are unimportant in low to mid energy photon and charged particle interactions. In fact, this distinction is only important for enriched or depleted materials interacting with neutrons or, high energy (>5 MeV) photons or charged particles.

It is possible to use several material database. If a material is defined in several database, Gate keeps the material in last database called (with a warning message). To call a database:

```
/gate/geometry/setMaterialDatabase MyMaterialDatabase.db
```

Elements

Elements are the building blocks of all the materials used in Gate simulations. Elements in Gate are defined as in a periodic table. Gate stores the elements name, symbol, atomic number, and molar mass. As stated above, isotopic abundances are not referenced or used. The supplied file *GateMaterials.db* contains the most commonly used elements and their molar masses as they are found in nature.

Some elements, particularly those that have an isotope with a large cross section for neutron absorption, have isotopic abundances and thus molar masses that vary depending upon their source. One element that exhibits this behavior is boron. In practice this behavior is not important for Gate applications.

Materials

In Gate, materials are defined as combinations of elements, and are an important parameter that Gate uses for all of the particle interactions that take place during a simulation. These combinations of elements require defining four additional parameters. These are the material's name, density, constituent element(s), and their individual abundances.

The composition of elements within a material can be defined in two different ways. If the material is a chemical compound then its relative amounts of elements are specified by the number of atoms in the chemical formula of the compound. For example, methane CH_{4} would be defined as having one carbon atom and four hydrogen atoms. If the material is better described as a mixture, such as 304-stainless steel, then the relative combinations of the elements are given by mass fraction. In the case of 304-stainless steel, the various mass fractions are given as 0.695 Iron, 0.190 Chromium, 0.095 Nickel, and 0.020 Manganese. Note that the mass fractions from the elements must all sum to one.

Densities of materials often vary greatly between different sources and must be carefully selected for the specific application in mind. Units of density must also be defined. These are typically given in g/cm3 but can be given in more convenient units for extreme cases. For example, a vacuum's density may be expressed in units of mg/cm3.

2.3.2 Modifying the Gate material database

New element

Defining a new element is a simple and straightforward process. Simply open the *GateMaterials.db* file with the text editor of your choice. At the top of the file is the header named **[Elements]** and somewhere in the middle of the file is another header named **[Materials]**. All element definitions required by the application must be included between these two headers. The format for entering an element is given by the elements name, symbol, atomic number, and molar mass. Below is an example:

```
Element Example GateMaterials.db:

[Elements]
Hydrogen:   S= H   ; Z=  1. ; A=   1.01  g/mole
Helium:     S= He  ; Z=  2. ; A=   4.003 g/mole
Lithium:    S= Li  ; Z=  3. ; A=   6.941 g/mole
```

(continues on next page)

(continued from previous page)

```
Beryllium:  S= Be  ; Z=  4. ; A=   9.012 g/mole
Boron:      S= B   ; Z=  5. ; A=  10.811 g/mole
Carbon:     S= C   ; Z=  6. ; A=  12.01  g/mole
```

In this example the name of the element is given first and is followed by a colon. Next, the standard symbol for the element is given by S=symbolic name followed by a semi-colon. The atomic number and molar mass follow the symbolic name given by *Z=atomic number* with a semi-colon and by *A=molar mass units* for the molar mass and its units.

New material

Materials are defined in a similar manner to elements but contain some additional parameters to account for their density and composition. Defining density is straightforward and performed the same way for all materials. However, material compositions require different definitions depending upon their form. These compositional forms are pure substances, chemical compounds, and mixtures of elements.

To add or modify a material in the material database, the *GateMaterials.db* file should be open using a text editor. The new entry should be inserted below the header named *Materials*. All material definitions required by the application must be included below this second header. Material definitions require several lines. The first line specifies their name, density, number of components, and an optional parameter describing the materials state (solid, liquid, or gas). The second and following lines specify the individual components and their relative abundances that make up this material.

The compositional forms of materials that Gate uses are pure substances, chemical compounds, mixtures of elements, and mixtures of materials. Gate defines each of these cases slightly differently and each will be dealt with separately below. In every case, the elements being used in a material definition must be previously defined as elements.

Elements as materials

Substances made of a pure element are the easiest materials to define. On the first line, enter the name of the material (the name of the material can be the same as that of the element), its density, its number of constituents (which is one in this case), and optionally its state (solid, liquid, or gas). The default state is gaseous. On the second line enter the element that it is composed of and the number of atoms of that element (in the case of an element as a material this number is 1). For example:

```
Elements as materials example GateMaterials.db:

[Materials]
Vacuum: d=0.000001 mg/cm3 ; n=1
       +el: name=Hydrogen ; n=1
Aluminium: d=1.350 g/cm3 ; n=1 ; state=solid
       +el: name=auto ; n=1
Uranium: d=18.90 g/cm3 ; n=1 ; state=solid
       +el: name=auto ; n=1
```

On the first line the density (with units) is defined by *d=material density units* and is separated by a semi-colon from the number of constituents in the material defined by *n=number of elements*. If the optional material form parameter is used it is also separated by a semi-colon. The available forms are gas, liquid, and solid. On the second line the individual elements and their abundances are defined by *+el: name=name of the element* ; *n=number of atoms*. If the name of the element and the material are the same, the element name can be defined by *+el: name=auto* command.

Compounds as materials

Chemical compounds are defined based upon the elements they are made of and their chemical formula. The first line is identical to the first line of a pure substance except that the number of constituent elements is now greater than one. On the second and subsequent lines, the individual elements and their abundances are defined by *+el: name=name of the element*;*n=number of atoms*.

For example:

```
Compounds as materials example GateMaterials.db:

[Materials]
NaI: d=3.67 g/cm3; n=2;  state=solid
       +el: name=Sodium ; n=1
       +el: name=Iodine ; n=1

PWO: d=8.28 g/cm3; n=3 ; state=Solid
       +el: name=Lead; n=1
       +el: name=Tungsten; n=1
       +el: name=Oxygen; n=4
```

Mixtures as materials

Mixture of elements are defined by indicating the mass fraction of the elements that make up the mixture. The first line of this definition is identical to the first line of the definition of a chemical compound. On the second and subsequent lines, the individual elements and their mass fractions are defined by *+el: name=name of element*;*f=mass fraction*.

In the case of material mixtures, the sum of the mass fractions should be one. For example:

```
Mixtures as materials example GateMaterials.db:

[Materials]
Lung:  d=0.26 g/cm3 ; n=9
       +el: name=Hydrogen  ; f=0.103
       +el: name=Carbon    ; f=0.105
       +el: name=Nitrogen  ; f=0.031
       +el: name=Oxygen    ; f=0.749
       +el: name=Sodium    ; f=0.002
       +el: name=Phosphor  ; f=0.002
       +el: name=Sulfur    ; f=0.003
       +el: name=Chlorine  ; f=0.003
       +el: name=Potassium ; f=0.002

SS304:  d=7.92 g/cm3 ; n=4 ; state=solid
       +el: name=Iron      ; f=0.695
       +el: name=Chromium  ; f=0.190
       +el: name=Nickel    ; f=0.095
       +el: name=Manganese ; f=0.020
```

Mixtures of materials as materials

Another way material can be defined is as mixtures of other materials and elements. As an example:

```
Mixtures of mixtures as materials example GateMaterials.db:

[Materials]
Aerogel:  d=0.200 g/cm3 ; n=3
        +mat: name=SiO2     ; f=0.625
        +mat: name=Water    ; f=0.374
        +el:  name=Carbon   ; f=0.001
```

In this example, the material, Aerogel, is defined to be made up of two materials, silicon dioxide and water, and one element, carbon. Mass fractions of the silicon dioxide, water, and carbon are given to specify the atom densities of the material when related to the density of the Aerogel. When specifying materials rather than elements the *+mat: name=identifier* must be used.

2.3.3 Ionization potential

The ionization potential is the energy required to remove an electron to an atom or a molecule. By default, the ionization potential is calculated thanks to the Bragg's additivity rule. It is possible to define the ionization potential of each material defined in gate. For example:

```
/gate/geometry/setIonisationPotential Water 75 eV
```

2.4 Setting up the physics

Table of Contents

2.4.1 New physics list mechanism

WARNING : big change from Gate V7.0

Up to now, physic lists were set in GATE according to macro files containing list of command to add physical processes. From GATE Version 7.0, we highly recommend to switch to the "physic list builder" mechanism. The physic-list builders are provided by the Geant4 community. To use a builder, use the "addPhyslicsList" command as follow:

```
/gate/physics/addPhysicsList QGSP_BERT_EMV
```

The name of the builder is one of the following for the electromagnetic physic-lists:

- emstandard
- emstandard_opt1
- emstandard_opt2
- emstandard_opt3
- emlivermore
- emlivermore_polar
- empenelope

And for the hadronic physics list builders:

- QGSP
- QGSP_EMV
- QGSC
- QGSC_EMV
- QGSP_EFLOW
- QGSP_BERT
- QGSP_BERT_EMV
- QGSP_BERT_HP
- QGSP_BERT_TRV
- QGSP_BIC
- QGSP_BIC_HP
- QGSP_NEQ
- QGSP_EMV_NQE
- QGSP_BERT_NQE
- QGSP_INCLXX
- FTFP_BERT
- FTFP
- FTFP_EMV
- ...

See more details in the Geant4 physics reference manual. Another (short, old) guide can be found here.

The production cuts special cuts can still be used as exemplified below:

```
/gate/physics/Gamma/SetCutInRegion      world 10 mm
/gate/physics/Electron/SetCutInRegion   world 10 mm
/gate/physics/Positron/SetCutInRegion   world 10 mm
/gate/physics/Proton/SetCutInRegion     world 10 mm

/gate/physics/Gamma/SetCutInRegion      patient 1 mm
/gate/physics/Electron/SetCutInRegion   patient 1 mm
/gate/physics/Positron/SetCutInRegion   patient 1 mm
/gate/physics/Proton/SetCutInRegion     patient 1 mm

/gate/physics/SetMaxStepSizeInRegion patient 1 mm
/gate/physics/ActivateStepLimiter proton
```

2.4.2 Physics list selection

Processes have three possible states: "Available", "Enabled" and "Initialized". In a simulation, GATE uses initialized processes. By default, all processes are only "Available". To initialize processes you want to use, you have to choose them in the list of available processes and put them in the list of "Enabled" processes with the *addProcess/removeProcess* commands. Then, you have to define models, data sets or energy range if it is necessary. Enabled processes are initialized when the command */gate/run/initialize* is called. After initialization, physics list cannot be modified. Processes must be initialized before the source definition.

To obtain the list of processes available, enabled or initialized:

```
/gate/physics/processList [State] [Particle]

Parameter : State
Parameter type  : s
Omittable       : True
Default value   : Available
Candidates      : Available Enabled Initialized

Parameter : Particle
Parameter type  : s
Omittable       : True
Default value   : All
```

User can print the list of initialized processes for each particles and the list of enabled processes with all informations (particles, models, data sets and energy range):

```
/gate/physics/print [File name]

Parameter : File Name
Parameter type  : s
Omittable       : False
```

Particles

To designate a particle in a macro command for physics, you have to use the Geant4 name of the particle. The command ‘’/particle/list” gives the list of particles available in GATE. Note that there are 5 particles defined for nuclei: “deuteron”, “triton”, “alpha”, “He3” and a generic particle for all ions called “GenericIon”. For more information about particles in Geant4, see Geant4 user support.

User can also use a particle group name. For example, “Charged” designates all charged particles or “EM” designates gamma, e+ and e-. Here, the list of particle groups available in GATE:

- Default : defaults particles defined for a process (see the “processList” command before for more informations)
- EM : e+, e-, gamma
- Charged : all charged particles

< ! > Particle name in non-physics macro command could be slightly different in particular for ions!

Adding a process

To add a process to the list of “Enabled” processes for a particle or a group of particles:

```
/gate/physics/addProcess [Process] [Particle]

Parameter : Process
Parameter type  : s
Omittable       : False

Parameter : Particle
Parameter type  : s
Omittable       : True
Default value   : Default
```

Removing a process

To remove a process of the "Enabled" processes list:

```
/gate/physics/removeProcess [Process] [Particle]

Parameter : Process
Parameter type  : s
Omittable       : False

Parameter : Particle
Parameter type  : s
Omittable       : True
Default value   : Default
```

Models and Data sets

Some processes have several models or several data sets available. To known if you can choose a model or a data set for a process, use the commands:

```
/gate/physics/processes/[Process Name]/modelList [Particle]
/gate/physics/processes/[Process Name]/dataSetList [Particle]

Parameter : Particle
Parameter type  : s
Omittable       : True
Default value   : Default
```

To select or unselect a model or a data set, use the command:

```
/gate/physics/processes/[Process Name]/setModel   [Model Name] [Particle]
/gate/physics/processes/[Process Name]/unSetModel [Model Name] [Particle]
/gate/physics/processes/[Process Name]/setDataSet   [DataSet Name] [Particle]
/gate/physics/processes/[Process Name]/unSetDataSet [DataSet Name] [Particle]

Parameter : Model/DataSet Name
Parameter type  : s
Omittable       : False
Candidates      : ............

Parameter : Particle
Parameter type  : s
Omittable       : True
Default value   : Default
```

Models can be selected for an energy range or for specific materials or elements. To do this, use commands:

```
/gate/physics/processes/[Process Name]/[Model Name]/setEmax [Value] [Unit] [Particle]␣
↪[Option]
/gate/physics/processes/[Process Name]/[Model Name]/setEmin [Value] [Unit] [Particle]␣
↪[Option]

Parameter : Value
Parameter type  : d
Omittable       : False
Default value   : 0.0
```

(continues on next page)

(continued from previous page)

```
Parameter : Unit
Parameter type  : s
Omittable       : False
Default value   : MeV
Candidates      : eV keV MeV GeV

Parameter : Particle
Parameter type  : s
Omittable       : True
Default value   : Default

Parameter : Option
Parameter type  : s
Omittable       : True
Default value   : NoOption
```

The parameter "Option" allow to define material or element for this model (see chapter on detector construction for more informations on materials and elements). For example:

```
/gate/physics/processes/IonInelastic/G4BinaryLightIonReaction/setEmin 100 keV
/gate/physics/processes/IonInelastic/G4BinaryLightIonReaction/setEmin 200 keV␣
↪GenericIon Water
```

A command allows to clear energy ranges defined for a model:

```
/gate/physics/[Process Name]/[Model Name]/clearEnergyRange [Particle]

Parameter : Particle
Parameter type  : s
Omittable       : True
Default value   : All
```

2.4.3 Physics processes

Electromagnetic processes

Electromagnetic processes are used to simulate the electromagnetic interaction of particles with matter. The mean free path of a process, λ, also called the interaction length, can be given in terms of the total cross section:

$\lambda(E) = (\sum_i [n_i \cdot \sigma(Z_i, E)])^{-1}$

where $\sigma(Z_i, E)$ is the cross section of the process for atom i composing the material. Cross-sections per atom and mean free path values are tabulated during initialization.

In Geant4, three models are available for electromagnetic processes:

- Standard processes are effective between 1 keV and 100 TeV
- Low energy processes are effective between 250 eV and 100 GeV (there is also the LivermorePolarizedModel for polarized gamma)
- Penelope processes are effective between 250 eV and 1 GeV

Models and cross-sections are based on the theoretical calculations and on exploitation of evaluated data. For the standard processes based on data, models and cross-sections rely on parameterizations of these data. Because atomic shell structure is more important in most cases at low energies, the low energy processes make direct use shell cross

section data. The data used for the determination of cross-sections and for sampling of the final state are extracted from a set of freely distributed evaluated data libraries:

- EPDL97 (Evaluated Photons Data Library)
- EEDL (Evaluated Electrons Data Library)
- EADL (Evaluated Atomic Data Library)
- stopping power data
- binding energy values based on data of Scofield .

The Penelope models have been specifically developed for Monte Carlo simulation and great care was given to the low energy description (i.e. atomic effects, etc.). These processes are the Geant4 implementation of the physics models developed for the PENELOPE code (PENetration and Energy LOss of Positrons and Electrons), version 2001.

***< ! > For the low energy processes, the download of G4EMLOW data files is required.**

< ! > Positron have no low energy process.'

< ! > Since Geant4 9.3, users can mixed electromagnetic models.

Photoelectric effect

The photoelectric effect is the absorption of a photon by an atomic electon with the ejection of this electron from the atom. The energy of the outgoing electron is:

$E = h\nu - L$

where L is the binding energy of the electron. Since a free electron cannot absorb a photon and also conserve momentum, the photoelectric effect always occurs on bound electrons while the nucleus absorbs the recoil momentum. The cross-section calculation is complex due to the combinaison of the electron Dirac wave functions. It is simulated by using a parameterized photon absorption cross section to determine the mean free path, atomic shell data to determine the energy of the ejected electron, and the K-shell angular distribution to sample the direction of the electron.

The cross-section depends on the atomic number Z of the material. The photoelectric process is favored by high Z materials. In the current implementation the relaxation of the atom is not simulated, but instead is counted as a local energy deposit. For low energy process, the deexcitation of the atom is simulated.

To add photoelectric effect in GATE:

```
/gate/physics/addProcess PhotoElectric gamma
```

Note that the two following macro are equivalent because the default particle for the photoelectric process is the gamma only:

> /gate/physics/addProcess PhotoElectric /gate/physics/addProcess PhotoElectric Default /gate/physics/addProcess PhotoElectric gamma

Then, choose the model:

```
/gate/physics/processes/PhotoElectric/setModel StandardModel
```

or:

```
/gate/physics/processes/PhotoElectric/setModel LivermoreModel
```

or:

```
/gate/physics/processes/PhotoElectric/setModel LivermorePolarizedModel
```

or:

```
/gate/physics/processes/PhotoElectric/setModel PenelopeModel
```

Compton scattering

Compton process describes the photon scattering by free electrons. Although electrons are bound in matter, the electron can be considered as essentially free for photons of energy much greater than the binding energy of the electron.

In the simulation, an empirical cross section formula is used, which reproduces the cross section data down to 10 keV. The final state is generated following the Klein-Nishina formula. For low energy incident photons, the simulation of the Compton scattering process is performed according to the same procedure used for the standard Compton scattering simulation, with the addition that Hubbel's atomic form factor (or scattering function) is taken into account. The angular and energy distribution of the incoherently scattered photon is then given by the product of the Klein-Nishina formula and the scattering function.

To add Compton scattering in GATE:

```
/gate/physics/addProcess Compton gamma
```

Note that the two following macro are equivalent because the defaut particle for the Compton process is only the gamma:

```
/gate/physics/addProcess Compton
/gate/physics/addProcess Compton Default
/gate/physics/addProcess Compton gamma
```

Then, choose the model:

```
/gate/physics/processes/Compton/setModel StandardModel
```

or:

```
/gate/physics/processes/Compton/setModel LivermoreModel
```

or:

```
/gate/physics/processes/Compton/setModel LivermorePolarizedModel
```

or:

```
/gate/physics/processes/Compton/setModel PenelopeModel
```

Rayleigh scattering

Thomson and Rayleigh scattering are linked to Compton scattering. Thomson scattering is similer to Compton scattering in the classical limit *i.e.* the scattering of photons by free electrons. For Rayleigh scattering, all the electrons of the atom contribute in a coherent way (this process is also called coherent scattering).

For these processes, no energy is transferred to the target. The direction of the photon is the only modified parameter. Atoms are not excited or ionized. At high energies, the cross-sections of Thomson and Rayleigh scattering are very small and are neglected. For these reasons, the Rayleigh process is defined only for low energy and Penelope models:

```
/gate/physics/addProcess RayleighScattering gamma
/gate/physics/processes/RayleighScattering/setModel LivermoreModel
/gate/physics/processes/RayleighScattering/setModel LivermorePolarizedModel
/gate/physics/processes/RayleighScattering/setModel PenelopeModel
```

Pair production

The process of pair production describes the transformation of a photon into an electron-positron pair. In order to conserve momentum, this can only occur in the presence of a third body, usually a nucleus. Moreover, the minimum energy required to create a pair is equal to the sum of the electron mass and positron mass (1.022 MeV).

To add pair production process in GATE:

```
/gate/physics/addProcess GammaConversion
```

Then, choose the model:

```
/gate/physics/processes/GammaConversion/setModel StandardModel
```

or:

```
/gate/physics/processes/GammaConversion/setModel LivermoreModel
```

or:

```
/gate/physics/processes/GammaConversion/setModel LivermorePolarizedModel
```

or:

```
/gate/physics/processes/GammaConversion/setModel PenelopeModel
```

Ionization

A charged particle passing through matter loses energy due to inelastic collision with atomic electrons of the material. Lost energy is transferred to the atom causing ionization or excitation. The ionization energy loss is calculated using the Bethe-Bloch formula. The particle energy loss E is divided into continuous energy loss and production of secondary electrons. The production threshold is defined as the minimum energy E_{cut} above which secondary particles will be produced and tracked. When $E < E_{cut}$, E is included into the continuous energy loss and when $E > E_{cut}$, secondary electrons are produced. Energy loss due to excitation is included into continuous energy loss.

The mean excitation potential I is the main parameter of the Bethe-Bloch formula. This quantity can be defined by user for each material (see chapter about geometry and materials).

There are three processes to handle ionization:

- for electron and positron:

```
/gate/physics/addProcess ElectronIonisation e+
/gate/physics/addProcess ElectronIonisation e-
```

- for hadrons and ions:

```
/gate/physics/addProcess HadronIonisation [Particle Name]
```

- for ions:

```
/gate/physics/addProcess IonIonisation [Particle Name]
```

The electron ionization process has three models available:

```
/gate/physics/processes/ElectronIonisation/setModel StandardModel e+
/gate/physics/processes/ElectronIonisation/setModel StandardModel e-
/gate/physics/processes/ElectronIonisation/setModel LivermoreModel e-
/gate/physics/processes/ElectronIonisation/setModel PenelopeModel e+
/gate/physics/processes/ElectronIonisation/setModel PenelopeModel e-
```

The new Geant4 model selection has not yet implemented for hadron ionization process in GATE V6.2. So users have to the old process/model selection.

The low energy model have a specific process for ionization of hadrons, ions, muons and taus:

```
/gate/physics/addProcess LowEnergyHadronIonisation
```

For the energy range between 1 keV and 2 MeV, this process has several models for the parametrization of electronic and nuclear stopping power:

```
/gate/physics/processes/LowEnergyHadronIonisation/setModel Elec_Ziegler1977p proton
```

Electronic stopping power models:

- Elec_ICRU_R49p (default)
- Elec_ICRU_R49He
- Elec_Ziegler1977p
- Elec_Ziegler1977He
- Elec_Ziegler1985p
- Elec_SRIM2000p

Nuclear stopping power models:

- Nuclear_ICRU_R49 (default)
- Nuclear_Ziegler1977
- Nuclear_Ziegler1985

It is possible to enable/disable nuclear stopping power for all hadrons and ions ionization processes:

```
/gate/physics/processes/[Process name]/setNuclearStoppingOn
/gate/physics/processes/[Process name]/setNuclearStoppingOff
```

Example:

```
/gate/physics/processes/HadronIonisation/setNuclearStoppingOff
```

Bremsstrahlung

Bremsstrahlung process is the production of an electromagnetic radiation by a charged particle accelerated in the field of another charged particle, such as a nucleus. The cross-section of bremsstrahlung is inversely proportional to the mass squared. Thus this process is more important for electron and positron than other charge particles. As for ionization process, above a given threshold energy the energy loss is simulated by the explicit production of photons.

Below the threshold, emission of soft photons is treated as a continuous energy loss. The bremsstrahlung energy spectrum is continuous:

```
/gate/physics/addProcess Bremsstrahlung e+
/gate/physics/addProcess Bremsstrahlung e-
```

The electron ionization process has three models available:

```
/gate/physics/processes/Bremsstrahlung/setModel StandardModel e+
/gate/physics/processes/Bremsstrahlung/setModel StandardModel e-
/gate/physics/processes/Bremsstrahlung/setModel LivermoreModel e-
/gate/physics/processes/Bremsstrahlung/setModel PenelopeModel e+
/gate/physics/processes/Bremsstrahlung/setModel PenelopeModel e-
```

Positron and electron annihilation

In Geant4, the process which simulated the in-flight annihilation of a positron with an atomic electron is attached to positron. As is usually done in shower programs, it is assumed here that the atomic electron is initially free and at rest. Also, annihilation processes producing one, or three or more, photons are ignored because these processes are negligible compared to the annihilation into two photons:

```
/gate/physics/addProcess G4PositronAnnihilation e+
```

Then, choose the model:

```
/gate/physics/processes/G4PositronAnnihilation/setModel StandardModel
```

or:

```
/gate/physics/processes/G4PositronAnnihilation/setModel PenelopeModel
```

An improvement of the positron-electron annihilation has been developed to take into account the $\gamma\gamma$ non-colinearity . The mean value of the angle distribution is $\simeq 0.5^0$ (do not need model selection):

```
/gate/physics/addProcess PositronAnnihilation  e+
```

Single and multiple Scattering

In addition to inelastic collisions with atomic electrons, charged particles passing through matter also suffer repeated elastic Coulomb scatterings from nuclei. Elastic cross section is huge when particle energy decreases, so multiple scattering approach (MSC) should be introduced in order to have acceptable CPU performance of the simulation. The MSC model used in GEANT4 belongs to the class of condensed algorithm in which the global effects of the collisions are simulated at the end of a track segment (step). The global effects generally computed in these codes are the net displacement, energy loss, and change of direction of the charged particle. The model is based on Lewis' MSC theory and uses model functions to determine the angular and spatial distributions after a step. The functions have been chosen in such a way as to give the same moments of the (angular and spatial) distributions as the Lewis theory. Two processes are available for multiple scattering. These processes are similar but they allow to define some sets of options for group of particles (electron/positron and hadron). The new Geant4 model selection has not yet implemented for multiple scattering process in GATE V6.2. So users have to the old process/model selection. The old MultipleScattering is now deprecated:

```
/gate/physics/addProcess eMultipleScattering e-
/gate/physics/addProcess eMultipleScattering e+

/gate/physics/addProcess hMultipleScattering proton
/gate/physics/addProcess hMultipleScattering alpha
/gate/physics/addProcess hMultipleScattering GenericIon
...
```

Single elastic scattering process is an alternative to the multiple scattering process. The advantage of the single scattering process is in possibility of usage of theory based cross sections, in contrary to the Geant4 multiple scattering model, which uses a number of phenomenological approximations on top of Lewis theory. Because each of elastic collisions are simulated the simulation CPU time of charged particles significantly increasing in comparison with the multiple scattering approach:

```
/gate/physics/addProcess SingleScattering ...
```

Muon electromagnetic processes

Muons have their own electromagnetic processes:

- Ionization:

```
/gate/physics/addProcess MuIonisation mu+
/gate/physics/addProcess MuIonisation mu-
```

- Bremsstrahlung:

```
/gate/physics/addProcess MuBremsstrahlung mu+
/gate/physics/addProcess MuBremsstrahlung mu-
```

- Direct production of (e+, e-) pairs by mu+ and mu-:

```
/gate/physics/addProcess MuPairProduction mu+
/gate/physics/addProcess MuPairProduction mu-
```

For ionization, the low energy model can handle muons.

Electromagnetic options

< ! > This part is recommanded for advanced users only!

< ! > Valid only from Geant4 version 9.2 - Gate V6.0 & V6.0.p01

< ! > In Geant4 version 9.2, options are available only for standard processes - Gate V6.0 & V6.0.p01

< ! > In case of using Gate V6.2, users must compile with Geant4 version 9.5 patch 01 - This Gate release is NOT compatible with previous Geant4 version

Options are available for steering the standard electromagnetic processes. Some options modify all electromagnetic processes initialized in the simulation (Global options). Some options have to be defined for each processes.

Global options:

The following options manage the DEDX, mean free path and cross sections tables. User can defined the table range (default 0.1 keV - 100 TeV):

```
/gate/physics/setEMin 0.1 keV
/gate/physics/setEMax 10 GeV
```

the number of bins of the DEDX table (default = 84):

```
/gate/physics/setDEDXBinning 500
```

and the number of bins of the mean free path table (default = 84):

```
/gate/physics/setLambdaBinning 500
```

Using cubic spline interpolation of DEDX and cross section tables, better interpolation was found to increase stability when varying transport parameters, such as cuts, of energy deposition of hadrons. Cubic spline interpolation is enabled by default. If the option was disable, the old linear interpolation is used:

```
/gate/physics/setSplineFlag true
/gate/physics/setSplineFlag false
```

Step function

Continuous energy loss imposes a limit on the step size because of the energy dependence of the cross sections. It is generally assumed in MC programs that the particle cross sections are approximately constant along a step, i.e. the step size should be small enough that the change in cross section, from the beginning of the step to the end, is also small. In principle one must use very small steps in order to insure an accurate simulation, however the computing time increases as the step size decreases. A good compromise is to limit the step size by not allowing the stopping range of the particle to decrease by more than a value *[Ratio]* during the step ([Ratio] = Δ range/range). This condition works well for particles with kinetic energies > 1 MeV, but for lower energies it gives very short step sizes.

To cure this problem a lower limit on the step size was introduced (*[Final range]*). The step size limit varies smoothly with decreasing energy from the value *[Ratio]* to the lowest possible value range cut *[Final range]*. By default for electron, [Ratio] = 0.2 and [Final range] = 0.1 mm:

```
/gate/physics/processes/[Process name]/SetStepFunction [Particle] [Ratio] [Final␣
↪range] [Unit]

Parameter : [Particle]
Parameter type  : s
Omittable       : False

Parameter : [Ratio] (step/range)
Parameter type  : d
Omittable       : False

Parameter : [Final range]
Parameter type  : d
Omittable       : False

Parameter : [Unit]
Parameter type  : s
Omittable       : False
```

Example:

```
/gate/physics/processes/ElectronIonisation/setStepFunction e+ 0.01 1 mm
```

Linear loss limit

This cut is an other approach to limit the step size. In a step, the energy loss by a particle with a kinetic energy E cannot exceed a value E_{cut} such as $E_{cut}/E <$ *[Ratio]*. By default for the ionization of an electron, the limit is 0.01:

```
/gate/physics/processes/IonIonisation/setLinearLossLimit [Particle] [Ratio]

Parameter : Particle or Group of particles
Parameter type  : s
Omittable       : False

Parameter : Limit
Parameter type  : d
Omittable       : False
```

Example:

```
/gate/physics/processes/HadronIonisation/setLinearLossLimit proton 0.0001
```

Geometrical step limit type

This option allow to choose the transport algorithm for the multiple scattering process near to boundary. The two options proposed are *safety* and *distanceToBoundary* For instance, the *distanceToBoundary* algorithm limit the step size near to geometrical boundaries: only single scattering is applied very close to the boundaries:

```
/gate/physics/processes/MultipleScattering/setGeometricalStepLimiterType [Particle]␣
↪[Limit type]

Parameter : Particle or Group of particles
Parameter type  : s
Omittable       : False

Parameter : Limit type
Parameter type  : s
Omittable       : False
```

Example:

```
/gate/physics/processes/MultipleScattering/setGeometricalStepLimiterType proton␣
↪distanceToBoundary
```

Hadronic processes

Hadronic processes described the interactions between incident hadrons/ions and the target nuclei. We also include decays of hadron and nuclei in this part. There is no strict frontier between nucleon-nucleon collision processes but we can distinguish four main process types in function of the energy and the impact parameter. At low energies, collisions lead to incomplete fusion in central collisions and to elastic scattering or inelastic scattering in peripherical collisions. At higher energies, there is fragmentation into several lighter fragments or nucleons for central collisions and into participant and the spectator regions for peripherical collisions. In Geant4, fusion, inelastic scattering and fragmentation processed are included in the inelastic process type. Inelastic processes had three main steps:

- cascade: the incident particle interacts strongly with the target and produces secondary particles
- preequilibrium (thermalization process): the excited target nucleus switches into equilibrated state by emitting excitons and light nuclei.
- de-excitation: the equilibrated nuclear residues evaporates into nucleons/light nuclei or breaks up into several fragments.

Each hadronic process may have one or more data sets associated with it. The term "data set" is meant, in a broad sense, to be an object that encapsulates methods and data for calculating total cross sections for a given process. The methods and data may take many forms, from a simple equation using a few hard-wired numbers to a sophisticated parameterization using large data tables. For the evaluation of cross sections, the list has a LIFO (Last In First Out) priority, meaning that data sets added later to the list will have priority over those added earlier to the list.

The final state is produced using models coupled to processes. In Geant4, any model can be run together with any other model and the ranges of applicability for the different models can be steered. This way, highly specialized models (valid only for one material and particle, and applicable only in a very restricted energy range) can be used in the same application, together with more general code, in a coherent fashion. Each model has an intrinsic range of applicability, and the model chosen for a simulation depends very much on the use-case. Three types of hadronic models have been implemented: parametrization driven models, data driven models, and theory driven models.

Most of hadronic processes need an explicit choice of models while a lots of processes have a default data set.

Main Geant4 models

LHEP

LHEP is the model used by default. It is based on code GHEISHA developed since 1978 by H. Fesefeldt to simulate hadron-nucleus interactions. The low energy part is valid from few hundred MeV to 20 GeV. The model is based on the principle of the intranuclear cascade and only the first hadron-nucleus interaction is simulated in detail. Other interactions in the nucleus are simulated by generating additional hadrons, simply treated as secondary particles can themselves generate their own intranuclear cascade. LHEP is a fully parameterized model but the physical meaning of the large number of adjustable parameters is sometimes unclear. Its main assets are the broad energy range covered, good reproduction of average values of distributions and computation times.

Example: G4LCapture, G4LENeutronInelastic, G4LFission, G4LCapture, G4LEProtonInelastic, G4LEPionMinusInelastic, G4LEPionPlusInelastic...

Bertini cascade

The intranuclear cascade model of Bertini has been developed by H W. Bertini in 1963. This code includes the intranuclear cascade model of Bertini, a pre-equilibrium model, a simple model of nucleus explosion, a model of fission and evaporation model.

Binary cascade

The Geant4 Binary Cascade is an intranuclear cascade propagating primary and secondary particles in a nucleus. The energy range and type of projectile covered are the same as Bertini model. From a theoretical point of view, this model is much more evolved taking into account a large number of resonances and fully modeled in three dimensions. Interactions are between a primary or secondary particle and an individual nucleon of the nucleus, leading to the name Binary Cascade. Cross section data are used to select collisions. Where available, experimental cross sections are used by the simulation. Propagating of particles is the nuclear field is done by numerically solving the equation of motion. The cascade terminates when the average and maximum energy of secondaries is below threshold. The remaining fragments are treated by precompound and de-excitation models.

QMD

Pre compound

Elastic scattering

In elastic scattering, the projectile and the target particles do not changed during the collision and no other particles are produced. Two processes are available for hadrons and ions. The first one is the old elastic process:

```
/gate/physics/addProcess HadronElastic Default
/gate/physics/processes/HadronElastic/setModel G4LElastic Default
```

or:

```
/gate/physics/addProcess HadronElastic
/gate/physics/processes/HadronElastic/setModel G4LElastic
```

or:

```
/gate/physics/addProcess HadronElastic proton
/gate/physics/processes/HadronElastic/setModel G4LElastic proton
/gate/physics/addProcess HadronElastic alpha
/gate/physics/processes/HadronElastic/setModel G4LElastic alpha
.........
```

This process has a default dataset (G4HadronElasticDataSet) and 6 models

- G4LElastic
- G4NeutronHPElastic
- G4NeutronHPorLElastic
- G4ElasticHadrNucleusHE
- G4LEpp
- G4LEnp

The alternative process is an improvement of the first process. It is supposed to be a good mix between the models of the fist process:

```
/gate/physics/addProcess HadronElastic
/gate/physics/processes/HadronElastic/setModel G4HadronElastic
/gate/physics/processes/HadronElastic/setDataSet G4HadronElasticDataSet
```

For this process, there is only one model and a main dataset (G4HadronElasticDataSet). An other dataset for low energy neutrons is also available (G4NeutronHPElasticData).

Inelastic process for proton

This process manage inelastic interaction of proton with matter. For example, the selection of two models with energy range for proton inelastic process (the only default particle is the proton):

```
/gate/physics/addProcess ProtonInelastic
/gate/physics/processes/ProtonInelastic/setModel G4BinaryCascade
/gate/physics/processes/ProtonInelastic/G4BinaryCascade/setEmin 170 MeV
/gate/physics/processes/ProtonInelastic/G4BinaryCascade/setEmax 500 GeV
/gate/physics/processes/ProtonInelastic/setModel PreCompound
/gate/physics/processes/ProtonInelastic/PreCompound/setEmin 0 MeV
/gate/physics/processes/ProtonInelastic/PreCompound/setEmax 170 MeV
```

This process has a default dataset (G4HadronInelasticDataSet) and an alternative dataset (G4ProtonInelasticCrossSection). There are 6 models available:

- G4LEProtonInelastic
- G4BertiniCascade

- G4BinaryCascade
- PreCompound
- G4QMDReaction

Inelastic process for ion

This process manage inelastic interaction of ions with matter. This process is valid for GenericIon, alpha, deuteron and triton. For example, a complete selection of models and data set for ions:

```
/gate/physics/addProcess IonInelastic Default
/gate/physics/processes/IonInelastic/setModel G4BinaryLightIonReaction Default
/gate/physics/processes/IonInelastic/setModel G4LEDeuteronInelastic deuteron
/gate/physics/processes/IonInelastic/setModel G4LETritonInelastic triton
/gate/physics/processes/IonInelastic/setModel G4LEAlphaInelastic alpha
/gate/physics/processes/IonInelastic/G4BinaryLightIonReaction/setEmin 80 MeV deuteron
/gate/physics/processes/IonInelastic/G4BinaryLightIonReaction/setEmax 20 GeV deuteron
/gate/physics/processes/IonInelastic/G4BinaryLightIonReaction/setEmin 80 MeV triton
/gate/physics/processes/IonInelastic/G4BinaryLightIonReaction/setEmax 20 GeV triton
/gate/physics/processes/IonInelastic/G4BinaryLightIonReaction/setEmin 80 MeV alpha
/gate/physics/processes/IonInelastic/G4BinaryLightIonReaction/setEmax 20 GeV alpha
/gate/physics/processes/IonInelastic/G4LEDeuteronInelastic/setEmin 0 MeV deuteron
/gate/physics/processes/IonInelastic/G4LEDeuteronInelastic/setEmax 80 MeV deuteron
/gate/physics/processes/IonInelastic/G4LETritonInelastic/setEmin 0 MeV triton
/gate/physics/processes/IonInelastic/G4LETritonInelastic/setEmax 80 MeV triton
/gate/physics/processes/IonInelastic/G4LEAlphaInelastic/setEmin 0 MeV alpha
/gate/physics/processes/IonInelastic/G4LEAlphaInelastic/setEmax 80 MeV alpha
/gate/physics/processes/IonInelastic/setDataSet G4IonsShenCrossSection GenericIon
/gate/physics/processes/IonInelastic/setDataSet G4TripathiLightCrossSection deuteron
/gate/physics/processes/IonInelastic/setDataSet G4TripathiLightCrossSection triton
/gate/physics/processes/IonInelastic/setDataSet G4TripathiLightCrossSection alpha
```

The IonInelastic process includes the G4IonInelasticProcess for GenericIon, the G4DeuteronInelasticProcess for deuteron, the G4TritonInelasticProcess for triton and the G4AlphaInelasticProcess for alpha. The G4QMDReaction model and the G4BinaryLightIonReaction model are available for all ions. For GenericIon, one additional model (G4BinaryLightIonReaction) and 5 datasets are available :

- G4TripathiCrossSection
- G4IonsKoxCrossSection
- G4IonsShenCrossSection
- G4IonsSihverCrossSection
- G4TripathiLightCrossSection

Alpha, deuteron and triton have a default data set (G4HadronInelasticDataSet) and a alternative dataset (G4TripathiLightCrossSection). There are also specific models for each particle: G4LEDeuteronInelastic, G4LETritonInelastic, G4LEAlphaInelastic.

Pions

The inelastic interaction of pi+ and pi- with matter is handled by PionPlusInelastic and PionMinusInelastic processes. These processes have two specific models (G4LEPionMinusInelastic - G4LEPionPlusInelastic) and three common models:

- Bertini Cascade
- Binary Cascade
- Leading Particle Bias

The default dataset is G4HadronInelasticDataSet. There is an alternative dataset: G4PiNuclearCrossSection:

```
/gate/physics/addProcess PionPlusInelastic pi+
/gate/physics/processes/PionPlusInelastic/setModel G4LEPionPlusInelastic pi+
/gate/physics/addProcess PionMinusInelastic pi-
/gate/physics/processes/PionMinusInelastic/setModel G4LEPionMinusInelastic pi-
```

Neutrons

The interactions of neutrons at low energies are split into four parts. We consider radiative capture, elastic scattering, fission, and inelastic scattering as separate processes. Each processes have standard models and datasets like others particles. In additions, some "high precision" models and datasets are provided for low energy interactions. The high precision neutron models depend on an evaluated neutron data library (G4NDL) for cross sections, angular distributions and final state information. G4NDL data comes largely from the ENDF/B-VI library.

< ! > For the low energy processes, the download of G4NDL data files is required.

Radiative Capture

The G4LCapture model generates the final state for neutron capture. The G4NeutronHPCapture model generates the final state for neutron capture using the high precision neutron model. The G4NeutronHPorLCapture model generates the final state for neutron capture using the high precision neutron model when sufficient high precision data is available for the selected element or isotope. When there is insufficient data, the neutron is captured using the less precise Low Energy Parameterized model.

The G4HadronCaptureDataSet is the default dataset for this process. The alternative high precision dataset is G4NeutronHPCaptureData:

```
/gate/physics/addProcess NeutronCapture
/gate/physics/processes/NeutronCapture/setModel G4LCapture
```

Inelastic scattering

The G4NeutronInelasticProcess is similar than proton inelastic and ion inelastic processes. In addtion to the standard models (G4LENeutronInelastic, G4BertiniCascade, G4BinaryCascade, PreCompound, LeadingParticleBias), two models using the high precision data are available. The G4NeutronHPInelastic model generates the final state for inelastic neutron scattering. The G4NeutronHPorLEInelastic model generates the final state for inelastic neutron scattering using the high precision neutron model when sufficient high precision data is available for the selected element or isotope. When there is insufficient data, the neutron is scattered inelastically using the less precise Low Energy Parameterized model (G4LENeutronInelastic).

The G4HadronInelasticDataSet is the default dataset for this process. An alternative dataset is the G4NeutronInelasticCrossSection. The high precision dataset is G4NeutronHPInelasticData:

```
/gate/physics/addProcess NeutronInelastic
/gate/physics/processes/NeutronInelastic/setModel PreCompound
```

Fission

The G4LFission model generates the final state for fission. The G4NeutronHPFission model generates the final state for neutron-induced fission using the high precision neutron model. The G4NeutronHPorLFission model generates the final state for neutron-induced fission using the high precision neutron model when sufficient high precision data

is available for the selected element or isotope. When there is insufficient data, neutron-induced fission is performed using the less precise Low Energy Parameterized model.

The G4HadronFissionDataSet is the default dataset for this process. The alternative high precision dataset is G4NeutronHPFissionData:

```
/gate/physics/addProcess Fission
/gate/physics/processes/Fission/setModel G4LFission
```

Particle decay

Particle decay is the spontaneous process of one elementary particle transforming into other elementary particles. If the particles created are not stable, the decay process can continue. The majority of decays in Geant4 are implemented using the G4PhaseSpaceDecayChannel class. It simulates phase space decays with isotropic angular distributions in the center-of-mass system:

```
/gate/physics/addProcess Decay
```

Radioactive decay

Radioactive decay is the process in which an unstable atomic nucleus spontaneously loses energy by emitting ionizing particles and radiation. In Geant4, the decay of radioactive nuclei by α, $\beta+$, and β emission and by electron capture are taken into account. The simulation model is empirical and data-driven, and uses the Evaluated Nuclear Structure Data File (ENSDF).

< ! > The download of radioactive decay data files is required.:

/gate/physics/addProcess RadioactiveDecay

Optical physics processes

For detailed information, see *Generating and tracking optical photons* .

Bulk Absorption

This process kills the optical photon. It requires the Material properties filled by the user with the Absorption length (average distance traveled by a photon before being absorbed by the medium):

```
/gate/physics/addProcess OpticalAbsorption
```

Rayleigh Scattering

This process depends on the particle's polarization. A photon which is not assigned a polarization at production may not be Rayleigh scattered. The photon is scattered in a new direction that is required to be perpendicular to the photon's new polarization in such a way that the final direction, initial and final polarizations are all in one plane. The process requires the Material properties filled with Rayleigh scattering length (average distance traveled by a photon before it is Rayleigh scattered in the medium).

N.B: For Water ONLY, when scattering lengths are not specified but the user, the Geant4 code calculates them following the Einstein-Smoluchowski formula:

```
/gate/physics/addProcess OpticalRayleigh
```

Mie Scattering

Mie Scattering is an analytical solution of Maxwell's equations for scattering of optical photons by spherical particles. It is significant only when the radius of the scattering object is of order of the wave length.The analytical expressions for Mie Scattering are very complicated since they are a series sum of Bessel functions.One common approximation made is call Henyey-Greenstein (HG). The implementation in GATE (Geant4) follows the HG approximation and the treatment of polarization and momentum are similar to that of Rayleigh scattering.

The process requires Material properties to be filled by the user with Mie scattering length data (MIEHG). In practice, the user not only needs to provide the attenuation length of Mie scattering, but also needs to provide the constant parameters of the approximation: MIEHG_FORWARD, MIEHG_BACKWARD, and MIEHG_FORWARD_RATIO:

```
/gate/physics/addProcess OpticalMie
```

Processes at Boundaries

The optical boundary process design relies on the concept of surfaces: physical properties of the surface itself (stored in Materials.xml) and characteristics of the surface specifying the two ordered pairs of physical volumes touching at the surface (Surface.xml).

When the surface concept is not needed, and a perfectly smooth surface exists between two dielectric materials, the only relevant property is the index of refraction, a quantity stored with the material:

```
/gate/physics/addProcess OpticalBoundary
```

Wavelength Shifting or Fluorescence

Fluorescence is the result of a three-stage process that occurs in certain molecules called fluorophores or fluorescent dyes. A fluorescent probe is a fluorophore designed to respond to a specific stimulus or to localize within a specific region of a biological specimen. The process responsible for the fluorescence involves the creation of an excited electronic singlet state by optical absorption and subsequent emission of an optical photon of lower energy than the excitation photon:

```
/gate/physics/addProcess OpticalWLS
```

X-ray at boundary

Provided that you compile Gate with GATE_USE_XRAYLIB ON , *i.e.*, that you activate the dependency to the xraylib, you can account for x-ray refraction by using:

```
/gate/physics/addProcess XrayBoundary
```

This will only work with shapes described analytically because the orthogonal to a boundary is not well defined in voxelized volume.

The process is computed following what is described in Wang et al, NSS/MIC, 2009.

An example is available at GateContrib: XRayRefraction.

2.4.4 Magnetic field

A magnetic field can be defined. It will be attached and thus active in the whole world volume. It is currently not possible to confine the field to another volume. The following command can be used to activate and define the magnetic field properties:

```
/gate/geometry/setMagField  Bx  By  Bz  Unit
```

Here is the help for the command:

```
Command /gate/geometry/setMagField
Guidance :
Define magnetic field.

Parameter : Bx
 Parameter type  : d
 Omittable       : False

Parameter : By
 Parameter type  : d
 Omittable       : False

Parameter : Bz
 Parameter type  : d
 Omittable       : False

Parameter : Unit
 Parameter type  : s
 Omittable       : True
 Default value   : tesla
 Candidates      : T kG G tesla kilogauss gauss
```

2.4.5 Electromagnetic field

A custom electromagnetic field can be generated from an external look-up table (e.g., text file) containing the spatial positions (x, y, z) in centimeter and their associated electric and magnetic field strengths (Ex, Ey, Ez) in volt per meter and (Bx, By, Bz) in tesla. The 3 dimensional field grid is read in and interpolated to the entire simulation geometry determined by the minimum and maximum value of the grid using a linear interpolation method. The structure of the input file is shown in Fig. 2.34.

The coordinates specified in the input file are assumed to be absolute cartesian coordinates. The first line of the file must be the number of values per coordinate, e.g., for a field reaching from 5cm x,y,z 5cm and a grid size of 1cm, the number of values per coordinate is 11. The permittivity and permeability of various materials are assumed to be already taken into account in the field strength. The method can also be used for pure magnetic or electric fields, by setting the corresponding field values to zero.

The following command can be used to activate and define the electromagnetic field:

```
/gate/geometry/setElectMagTabulateField3D PATH_TO_TEXT_FILE
```

2.5 Cut and Variance Reduction Technics

```
#x y z Ex Ey Ez Bx By Bz
11 11 11
-5.00 -5.00 -5.00 0 0 0 0 3 0
-4.00 -5.00 -5.00 0 0 0 0 3 0
-3.00 -5.00 -5.00 0 0 0 0 3 0
-2.00 -5.00 -5.00 0 0 0 0 3 0
-1.00 -5.00 -5.00 0 0 0 0 3 0
0.00 -5.00 -5.00 0 0 0 0 3 0
1.00 -5.00 -5.00 0 0 0 0 3 0
2.00 -5.00 -5.00 0 0 0 0 3 0
3.00 -5.00 -5.00 0 0 0 0 3 0
4.00 -5.00 -5.00 0 0 0 0 3 0
5.00 -5.00 -5.00 0 0 0 0 3 0
-5.00 -4.00 -5.00 0 0 0 0 3 0
-4.00 -4.00 -5.00 0 0 0 0 3 0
-3.00 -4.00 -5.00 0 0 0 0 3 0
-2.00 -4.00 -5.00 0 0 0 0 3 0
-1.00 -4.00 -5.00 0 0 0 0 3 0
0.00 -4.00 -5.00 0 0 0 0 3 0
1.00 -4.00 -5.00 0 0 0 0 3 0
2.00 -4.00 -5.00 0 0 0 0 3 0
3.00 -4.00 -5.00 0 0 0 0 3 0
4.00 -4.00 -5.00 0 0 0 0 3 0
5.00 -4.00 -5.00 0 0 0 0 3 0
.
.
.
```

Fig. 2.34: Input file for electromagnetic vector field maps.

Table of Contents

2.5.1 Production threshold, step limiter and special cuts

Production threshold

To avoid infrared divergence, charged particles processes (ionization and bremsstrahlung) require a threshold below which no secondary particles will be generated. Because of this requirement, gammas, electrons and positrons require production thresholds which the user should define. This threshold should be defined as a distance, or range cut-off, which is internally converted to an energy for individual materials. Production thresholds are defined for a geometrical region. In GATE, each volume is considered as a geometrical region. If no cut is defined, the region inherited the threshold of the parent volume. The default cut value of the world is set to 1.0 mm:

```
/gate/physics/Gamma/SetCutInRegion      [Volume Name]  [Cut value] [Unit]
/gate/physics/Electron/SetCutInRegion   [Volume Name]  [Cut value] [Unit]
/gate/physics/Positron/SetCutInRegion   [Volume Name]  [Cut value] [Unit]
```

For example:

```
/gate/physics/Gamma/SetCutInRegion     world  1.0 mm
```

The list of production threshold in range and in energy for each volume can be display with the command:

```
/gate/physics/displayCuts
```

User should use this command after the initilization.

X-Ray cuts and auger electrons in photo-electric process

Secondary electron and photon production in low energy photo-electric process (livermore and penelope models) could be customized.

The auger electron could be activated in livermore and penelope models:

```
/gate/physics/processes/PhotoElectric/setAugerElectron true
```

In livermore model, an energy threshold could be defined from which secondary particles are produced. The threshold is defined for the whole world of the simulation:

```
/gate/physics/processes/PhotoElectric/setDeltaRayCut [value] [unit]
/gate/physics/processes/PhotoElectric/setXRayCut [value] [unit]
```

Step limiter

< ! > This part is recommended for advanced users only!

The step limiter can be considered as a process with fixed step size. It allows to limit the maximum size of step. As for production threshold, the step limiter is defined for a geometrical region and the region can inherit the step limiter of the parent volume. User have to define the step size in the region:

```
/gate/physics/SetMaxStepSizeInRegion [Volume Name] [Step size] [Unit]
```

For example:

```
/gate/physics/SetMaxStepSizeInRegion world  1.0 mm
```

Then, the step limiter process has to be add to particles:

```
/gate/physics/ActivateStepLimiter proton
```

Special cuts

< ! > This part is recommended for advanced users only!

The user can define four cuts to limit the tracking of a particle:

- the maximum total track length
- the maximum total time of flight
- the minimum kinetic energy
- the minimum remaining range

While step limiter is affected to a step, special cuts are affected to a track. When a particle is stopped, the energy is deposited locally. As for production threshold, the special cuts are defined for a geometrical region and the region can inherit the special cuts of the parent volume.

User have to define the values of the special cuts in the region and activate the cuts for particles:

```
/gate/physics/SetMaxToFInRegion world 5 s
/gate/physics/SetMinKineticEnergyInRegion world 1 keV
/gate/physics/SetMaxTrackLengthInRegion world 0.01 mm
/gate/physics/SetMinRemainingRangeInRegion world 0.02 mm
/gate/physics/ActivateSpecialCuts proton
```

The user does not have to define all the cuts. The *ActivateSpecialCuts* command is effective for all the special cuts that are defined.

2.5.2 Variance reduction

Two standard reduction variance techniques are available in GATE: splitting and russian roulette. The weight of secondary particles is recalculated in function of the number of secondaries generated. User can also defined filters to increase the efficiency of these techniques.

< ! > User have to verify that all the tools he used in his simulation take into account particle weight!

Splitting and russian roulette

Splitting

In this technique, the final state of the process is generated N times and the weight of each secondary is 1/N:

```
/gate/physics/processes/Bremsstrahlung/activateSplitting [Particle] [N]

Parameter : [Particle]
Parameter type  : s
Omittable       : False

Parameter : [N]
Parameter type  : i
Omittable       : False
```

Example: to split 100 times the electron bremsstrahlung photon (not that we specify that the e- is the particle which do the bremsstrahlung, but the split is applied on the generated photon):

```
/gate/physics/processes/Bremsstrahlung/activateSplitting e- 100
```

Russian roulette

In this technique, Russian roulette is played on secondary particles. The survival probability is 1/N and the weight of each secondary is N:

```
/gate/physics/processes/Bremsstrahlung/activateRussianRoulette [Particle] [N]

Parameter : [Particle]
Parameter type  : s
Omittable       : False

Parameter : [N]
Parameter type  : i
Omittable       : False
```

Example: to keep 2% of electron bremsstrahlung photon (2/100 = 1/50):

```
/gate/physics/processes/Bremsstrahlung/activateRussianRoulette e- 50
```

Selective splitting and russian roulette

To increase the efficiency of the splitting and the russian roulette technique, user can add selections criteria on the incident (primary) or secondary particles. The selection is done with filters. The filters for splitting and russian

roulette are the same as for Actors. For example, to split bremsstrahlung photons with a vector direction inside a cone of 20 degrees around the x axis:

```
/gate/physics/processes/Bremsstrahlung/addFilter angleFilter secondaries
/gate/physics/processes/Bremsstrahlung/secondaries/angleFilter/setAngle 20
/gate/physics/processes/Bremsstrahlung/secondaries/angleFilter/setDirection 1 0 0
```

There are several filters types: filters on particle, particle ID, energy, direction, volume... See the chapter on Actor for a description of all filters.

TLE and seTLE (Track Length Estimator)

See the TLEDoseActor and SETLEDoseActor *TLE and seTLE (Track Length Estimator)*.

2.6 Source

Table of Contents

To introduce a source into a GATE simulation, the user has to define the type of source (voxelized, linacBeam, phaseSpace, PencilBeam, TPSPencilBeam or GPS) and its feature (angle, energy, and position). Many activity distributions are available in GATE. At each new event, the source manager decides randomly which source decays, and generates for it one or more primary particles.

2.6.1 Creating a source

Different type of sources can be defined in the same GATE simulation. Each source is independent. The command to create a source is given below:

```
/gate/source/NAME
```

where "NAME" defines the name of the source.

Adding a source

The next step is to add the source. For this, user has to type the following GATE command:

```
/gate/source/addSource NAME
```

or:

```
/gate/source/addSource NAME gps
```

In this example, a source "NAME" is added. Once a source has been added, a series of properties must be assigned to it. These properties are: activity, type of particle(s), energy distribution, energy value or bounds, angular emission, spatial distribution and location, and half-life. The commands required to assign these properties are described in the following paragraphs.

Defining the type of source

Ion source

The ion source type can simulate any ion by defining its atomic number (Z), atomic weight (A), ionic charge in units of energy (Q), and its excitation energy in keV (E). It incorporates both the radioactive decay and the atomic de-excitation. This is the most "realistic" way of simulating a radionuclide; however, it is also the slowest.

To use the ion source:

```
/gate/source/NAME/gps/particle ion
/gate/source/NAME/gps/ion 8 15 0 0
/gate/source/NAME/setForcedUnstableFlag true
/gate/source/NAME/useDefaultHalfLife
```

In the above example, an ion source of oxygen-15 has been defined with Z=8, A=15, Q=0, E=0 . If it is too slow, other options are available, as described below.

Simple particles

You can choose from a long list of simple particles (e^-, e^+ , gamma, etc) to use in your simulations (use the help command to obtain the full list). For example:

```
/gate/source/NAME/gps/particle gamma
```

defines a photon source and:

```
/gate/source/NAME/gps/particle e+
```

defines a positron source. If you choose to use a particle, you will have to define more properties. As an example, the correct use of a positron source simulating fluorine-18 is:

```
/gate/source/NAME/gps/particle e+
/gate/source/NAME/gps/energytype Fluor18
/gate/source/NAME/setForcedUnstableFlag true
/gate/source/NAME/setForcedHalfLife 6586 s
```

In the above example, more properties of fluorine-18 have been added by using the helper keyword Fluor18 (see below): half-life and energy distribution. Note that the branching ratios are not respected with this kind of simulation because no decay is simulated, i.e. emission is 100% positron. You may want to lower the activity by an appropriate factor to take this better point into account.

Helper keywords

These keywords are defined to help you to define the particle properties. The first three keywords define an positron energy distribution resulting from common beta emitters, and the last one defines a special two-particle source.

For instance, **Fluor18** defines the positron energy spectrum of fluorine-18. Note that this keyword define only the energy spectrum, you still have to specify the particle e^+ and the half-life.

Back-to-back This keyword is implemented for PET simulations where two annihilation photons are generated at 180 degrees. This type of source is faster to simulate than the ion source or the positron source and allows for selecting emission angle. To use the back-to-back source type:

```
/gate/source/NAME/setType backtoback
```

Note that there are no radioactive decays simulated when using the back-to-back type and that you still have to define the particle (gamma), energy type (Mono) and energy-value (0.511 MeV).

Note: the 'Accolinearity' flag that once exist in Gate is no more valid. See https://github.com/OpenGATE/Gate/issues/381 for details.

FastI124

FastI124 is a special source implementing a simplified decay scheme of the non-pure beta emitter iodine-124 in which positrons are emitted but not neutrinos, there is no nuclear recoil, gammas are emitted if their emission probability is > 1%; and no atomic de-excitation occurs (no x-rays, Auger electrons). These simplifications allow for an increase in speed with respect to the ion source while retaining important features of iodine-124, i.e. gammas may be emitted concurrently with positrons to possibly create "dirty" coincidences. Since decay is simulated, branching ratios are respected hence no activity compensation is necessary.

To use the fastI124 source:

```
/gate/source/NAME/setType fastI124
```

The source takes care of particle definitions (gamma, positron) and energy distribution so that there is no need to specify a particle or mention its energy.

Defining the activity

To define the activity of the given source, the user defines the amount of activity and its unit using the following command:

```
/gate/source/NAME/setActivity 5. becquerel
```

In this example, the total activity of the source referred to as "NAME" is set to 5 Bq. The activity can be defined in Curie (Ci) as well as in Becquerel (Bq).

Defining the energy

Energy distribution

If the source does not take care of the type of energy distribution (e.g. fastI124), then it has to be explicitly defined. This can be achieved either by using a pre-defined spectrum (see helper keywords above) or by using built-in distributions.

Candidates for built-in energy distributions are: mono-energetic "Mono", linear "Lin", powerlaw "Pow", exponential "Exp", Gaussian "Gauss", bremstrahlung "Brem", black-body "Bbody", cosmic diffuse gamma ray "Cdg", user-defined histogram "UserSpectrum", arbitrary point-wise spectrum "Arb", and user-defined energy per nucleon histogram "Epn". Capitalization is important: only strings given exactly as above will be recognized.

In the following example, all particles have the same energy:

```
/gate/source/NAME/gps/energytype Mono
```

Energy value

You may have to specify the energy value (or bounds) depending on the type of energy distribution you have selected. For example, for monoenergetic distributions (like back-to-back sources), you specify the energy value with:

```
/gate/source/NAME/gps/monoenergy 511. keV
```

In the case of ions, the kinetic energy must be 0 since the ions are at rest:

```
/gate/source/NAME/gps/monoenergy 0. ev
```

Any type of energy unit within the International System of Units (SI) can be used: eV, GeV, MeV, keV...

Examples

1) ion source for fluorine-18:

```
/gate/source/NAME/gps/particle ion
/gate/source/NAME/gps/ion 9 18 0 0
/gate/source/NAME/gps/monoenergy 0. keV
/gate/source/NAME/setForcedUnstableFlag true # WARNING - DEBUG - New command line␣
↪to debug the use of ion particle type
/gate/source/F18/useDefaultHalfLife
```

2) positron source for flourine-18:

```
/gate/source/NAME/gps/particle e+
/gate/source/NAME/gps/energytype Fluor18
/gate/source/NAME/setForcedUnstableFlag true
/gate/source/NAME/setForcedHalfLife 6586 s
```

3) backtoback for fluorine-18:

```
/gate/source/NAME/setType backtoback
/gate/source/NAME/gps/particle gamma
/gate/source/NAME/gps/monoenergy 511. keV
/gate/source/NAME/setForcedUnstableFlag true
/gate/source/NAME/setForcedHalfLife 6586 s
```

4) fast iodine-124 source:

```
/gate/source/NAME/setType fastI124
/gate/source/NAME/setForcedUnstableFlag true
/gate/source/NAME/setForcedHalfLife 360806 s
```

Another way to define the energy of a radioactive source is to use the energytype UserSpectrum. You can define 3 different user spectra: a discrete spectrum, a histogram or a linear interpolated spectrum

Table I. Radiation Source Properties

Property	Default values	Ion	particle	backtoback	fastf124
particle type(s)	-	x	x	x	incl.
energy distribution	Mono				incl.
energy value(s)	1.0 MeV				incl.
half-life	-	x	x	x	x
activity	-	x	x	x	x
angular emission	isotropic				
spatial distribution and location	Point source @ (0,0,0)				

Legend	
-	*undetermined*
incl	*included*
X	*must be specified*
(dark)	*may be specified (overrides default value)*

Fig. 2.35: Properties of radioactive source

Example:

```
 ######################## Mode 1: Discrete Spectrum ###################
/gate/source/addsource spectrumLine gps
/gate/source/spectrumLine/gps/particle gamma
/gate/source/spectrumLine/gps/energytype UserSpectrum
/gate/source/spectrumLine/gps/setSpectrumFile ../data/DiscreteSpectrum.txt
/gate/source/spectrumLine/setIntensity 1
 ###################### Mode 1: Discrete Spectrum ####################


 ###################### Mode 2: Histogram ####################
/gate/source/addSource histogram gps
/gate/source/histogram/gps/particle e-
/gate/source/histogram/gps/energytype UserSpectrum
/gate/source/histogram/gps/setSpectrumFile ../data/Histogram.txt
/gate/source/histogram/setIntensity 10
###################### Mode 2: Histogram ####################


#################### Mode 3: Linear interpolation spectrum ###################
/gate/source/addSource interpolationSpectrum gps
/gate/source/interpolationSpectrum/gps/particle e-
/gate/source/interpolationSpectrum/gps/energytype UserSpectrum
/gate/source/interpolationSpectrum/gps/setSpectrumFile ../data/InterpolationSpectrum.
↪txt
/gate/source/interpolationSpectrum/setIntensity 10
#################### Mode 3: Linear interpolation spectrum ###################
```

The user spectra are specified by a text file. The first number on the first line indicates the mode as follows: 1 - discrete, 2 - histogram, and 3 - interpolated spectrum. The second number on the first line specifies the energy, in MeV, of the lower edge of the first bin in histogram mode. (Though ignored in the discrete and interpolated modes, it must be

present for the file to parse correctly.) The remaining lines specify the energy, in MeV, and the associated probability weighting. The probabilities will normalized by the GATE software.

The discrete spectrum generates particles with one of the listed energies:

```
################DiscreteSpectrum.txt ###################
1   0
0.2   0.2
0.4   0.4
0.6   0.6
0.8   0.8
1.0   1.0
1.2   0.8
1.4   0.6
1.6   0.4
1.8   0.2
#####################################################
```

In histogram mode, the energy specified on each line corresponds to the upper edge of the respective bin. The energies of the generated particles will be between the minimum energy, specified on the first line of the file, and the upper edge of the last bin. Within each bin, the energies are distributed uniformly:

```
################## Histogram.txt ####################
2   2
2.2   0.2
2.4   0.4
2.6   0.6
2.8   0.8
3.0   1
3.2   0.8
3.4   0.6
3.6   0.4
3.8   0.2
#####################################################
```

In interpolated mode, the energy of the generated particle will fall between the first and last energy specified, according to the probability distribution created by piecewise-linear interpolation between the points provided:

```
################ InterpolationSpectrum.txt #############
3   0
4.2   0.2
4.4   0.4
4.6   0.6
4.8   0.8
5.0   1
5.2   0.8
5.4   0.6
5.6   0.4
5.8   0.2
#####################################################
```

The following image present the result obtain for the 3 examples (available in example_UserSpectrum repository)

Defining the angular distribution of the emission

An emission angle distribution can be defined with the angular span using:

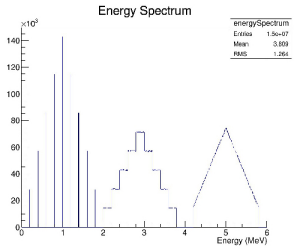


Fig. 2.36: 3 different User spectra

```
/gate/source/NAME/gps/angtype iso
/gate/source/NAME/gps/mintheta 90. deg
/gate/source/NAME/gps/maxtheta 90. deg
/gate/source/NAME/gps/minphi 0. deg
/gate/source/NAME/gps/maxphi 360. deg
```

In this case, all particles have the same polar angle (theta) of 90 degrees. They are all emitted along directions orthogonal to the z-axis. The particles are emitted with an azimuthal angle (phi) between 0 and 360 degrees, along all possible directions.

By default, a full span of 0-180 degrees for the polar angle and 0-360 degrees for the azimuthal angle are defined. The emission span can be reduced for back-to-back sources to speed up the simulation.

Defining the shape of the source

The last step is to define its geometry. The following command defines the type of source distribution:

```
/gate/source/NAME/gps/type Volume
```

In the above description, a volumic source distribution has been chosen. Other types of source distribution can be used: *Point*, *Beam*, *Plane*, or *Surface*. The default value is *Point*.

For a *Plane* source, the source shape type can be *Circle*, *Annulus*, *Ellipsoid*, *Square*, or *Rectangle*. For both *Surface* and *Volume* sources, this can be *Sphere*, *Ellipsoid*, *Cylinder*, or *Para*. The default source is a *Point* source and so *Shape* is not set to any of the above types. Each shape has its own parameters:

```
/gate/source/NAME/gps/shape Cylinder
/gate/source/NAME/gps/radius 1. cm
/gate/source/NAME/gps/halfz 1. mm
```

In the previous commands, the source is a cylinder with a radius of 1 cm and a length of 2 mm. Very often, the half-length is given rather than the full length.

- To define a circle, the radius (*radius*) should be set. (In reality it is not really a circle but a disk).
- To define an annulus, the inner (*radius0*) and outer radii (*radius*) should be given.
- To define an ellipse, square , or rectangle, the half-lengths along x (*halfx*) and y (*halfy*) have to be given.
- To define a sphere, only the radius (*radius*) only has to be specified.
- To define an ellipsoid, its half-lenghths in x (*halfx*), y (*halfy*), and z (*halfz*) have to be given.
- To define a cylinder with its axis along the z-axis, only the radius (*radius*) and the z half-length (*halfz*) have to be specified.

- To define parallelepipeds, the x (*halfx*), y (*halfy*), and z (*halfz*) half-lengths, and the angles alpha (*paralp*), theta (*parthe*), and phi (*parphi*) have to be given.

Define the placement of the source

The position of the source distribution can be defined using:

```
/gate/source/NAME/gps/centre 1. 0. 0. cm
```

In that example, the centre of the source distribution is 1 cm off-centered along the x-axis.

Movement of a source

Attach to a volume

The source can be attached to a volume:

```
/gate/source/[Source name]/attachTo   [Volume Name]
```

If the volume moves during the simulation, the source moves along with the volume. Note that when attaching a source to a volume, the source's placement becomes relative to the volume.

Confining a source

Note: this is the old way of moving a source. It is very inefficient. Please consider using the "Attach to a volume" method instead.

To define sources in movement, the source distribution have to be confined in a Geant4 volume. This volume will be animated using the usual GATE command as described in Chapter 4 of this manual.

The command:

```
/gate/source/NAME/gps/confine NAME_phys
```

specifies that the emission must be confined to a volume of the Geant4 geometry. In this case, the emission distribution is the intersection of the General Particle Source (GPS) and the Geant4 volume. The Geant4 volume must be specified by its physical volume name: GATEname + '_phys'.

One should note that the confinment slows down the simulation, the confinement volume must have an intersection with the GPS shape, and the confinement volume must not be too large as compared to the GPS shape.

A complete example of a moving source can be found in the SPECT benchmark or in the macro hereafter:

```
# Define the shape/dimensions of the moving source
/gate/MovingSource/geometry/setRmax 5. cm
/gate/MovingSource/geometry/setRmin 0. cm
/gate/MovingSource/geometry/setHeight 20. cm
/gate/MovingSource/moves/insert translation
/gate/MovingSource/translation/setSpeed 0 0 0.04 cm/s


# Define the shape/dimensions of the large sourcecontainer
# that should contain the full trajectory of the moving source
/gate/source/SourceContainer/gps/type Volume
```

(continues on next page)

(continued from previous page)

```
/gate/source/SourceContainer/gps/shape Cylinder
/gate/source/SourceContainer/gps/radius 4. cm
/gate/source/SourceContainer/gps/halfz 30. cm
# Define the placement of the SourceContainer
/gate/source/SourceContainer/gps/centre 0. 0. 0. cm
# Define the source as a gamma source
/gate/source/SourceContainer/gps/particle gamma
# Define the gamma energy
/gate/source/SourceContainer/gps/energy 140. keV
# Set the activity of the source
/gate/source/SourceContainer/setActivity 5000. Bq
# Define a confinement and confine the large container to
# the MovingSource at a position defined by the time and
# the translation speed
/gate/source/SourceContainer/gps/confine MovingSource_phys
```

Example: two gammas

The following example gives a script to insert a point source of back-to-back type:

```
# A new source with an arbitrary name #(``twogamma'') is created
/gate/source/addSource twogamma
# The total activity of the source is set
/gate/source/twogamma/setActivity 0.0000001 Ci
# The source emits pairs of particles back-to-back
/gate/source/twogamma/setType backtoback
# The particles emitted by the source are gammas
/gate/source/twogamma/gps/particle gamma
# The gammas have an energy of 511 keV
/gate/source/twogamma/gps/energytype Mono
/gate/source/twogamma/gps/monoenergy 0.511 MeV
# The source is a full sphere with radius 0.1 mm,
# located at the centre of the FOV
/gate/source/twogamma/gps/type Volume
/gate/source/twogamma/gps/shape Sphere
/gate/source/twogamma/gps/radius 0.1 mm
/gate/source/twogamma/gps/centre 0. 0. 0. cm
# The angular distribution of emission angles is isotropic
/gate/source/twogamma/gps/angtype iso
# The parameters below mean that the source emits
# at all angles along the z axis
/gate/source/twogamma/gps/mintheta 0. deg
/gate/source/twogamma/gps/maxtheta 180. deg
# Uncomment the parameters below if you want the source
# to emit in an XY (transverse) plane
/gate/source/twogamma/gps/mintheta 90. deg
/gate/source/twogamma/gps/maxtheta 90. deg
# The parameters below mean that the source emits
# at all angles in the transverse (XY) directions
/gate/source/twogamma/gps/minphi 0. deg
/gate/source/twogamma/gps/maxphi 360. deg
```

Defining a cold source

To define a cold (i.e. with no activity) volume in a phantom, a dedicated command is available.

The command:

```
/gate/source/NAME/gps/Forbid Volume_Name
```

The following example explains how to use this option. First you must define a volume that defines the cold region:

```
/gate/world/daughters/name cold_area
/gate/world/daughters/insert cylinder
/gate/cold_area/vis/forceWireframe
/gate/cold_area/vis/setColor green
/gate/cold_area/geometry/setRmax 3.0 cm
/gate/cold_area/geometry/setHeight 1. cm
```

Then you describe your source with the Forbid command:

```
/gate/source/addSource number1
/gate/source/number1/setActivity 100000. becquerel
/gate/source/number1/gps/particle gamma
/gate/source/number1/setType backtoback
/gate/source/number1/gps/type Volume
/gate/source/number1/gps/shape Cylinder
/gate/source/number1/gps/radius 5. cm
/gate/source/number1/gps/halfz 0.5 cm
/gate/source/number1/gps/centre 0. 0. 0. cm
/gate/source/number1/gps/monoenergy 511. keV
/gate/source/number1/gps/angtype iso
/gate/source/number1/gps/Forbid cold_area_phys
/gate/source/number1/dump 1
/gate/source/list
```

It is important to remember that the /gate/run/initialize command must have been executed prior to using the Forbid command because phantom geometries are not available until after they are initialized.

Visualizing a source

To check that sources are at the right location in the geometry, you can use the following command:

```
/gate/source/[Source name]/visualize
```

along with a real time viewer (e.g. OpenGL). To visualize a source, Gate will randomly pick a certain number of points within the source and display them on the screen, along with the geometry. The full syntax is:

```
/gate/source/[Source name]/visualize count color size
```

where name is the name of the source, count is the number of random points to pick up (must be > 0 and <= 10000), color is the color to assign to those points (valid colors are: white, gray, grey, black, red, green, blue, cyan, magenta, yellow), and size is the screen size (in pixels) of each point (must be > 0 and <= 20).

Depending on the size and shape of the source, more or fewer points may be necessary.

- Example:

```
/gate/source/backgroundSource/visualize 2000 yellow 3
/gate/source/hotRegion/visualize 5000 red 2
```

Intensity

If several sources have been added and no activity is defined, user can use intensity to define the source priorities. A high intensity correspond to a high priority. For each event, the source is randomly selected taking into account the intensity of each sources:

```
/gate/source/MyBeam/setIntensity [value]
```

2.6.2 Pencil Beam source

The simulation source can be a pencil beam. This source allows for characterizing a beam of particles having energy and optical properties. This beam can be used for instance in order to characterize a clinical beam interacting in a passive beam line or to characterize a spot from an active scanning beam delivery system.

Create the source:

```
/gate/source/addSource [Source name] PencilBeam
```

One can select the type of particle used for the pencil beam (proton, e-, etc.):

```
/gate/source/  [Source name]  /setParticleType [particle_type]
```

Alternatively, one can define a specific type of ion, by defining the particle type as "GenericIon" and then specifying the particle parameters of the ion to be generated: Z: AtomicNumber, A: AtomicMass, Q: Charge of Ion (in unit of e), E: Excitation energy (in keV). As an example, the definition of a C12 ion beam is given:

```
/gate/source/  [Source name]  /setParticleType GenericIon
/gate/source/PBS/setIonProperties 6 12 6 0
```

The energy spectrum of the source is Gaussian and is defined by a mean energy and standard deviation:

```
/gate/source/ [Source name] /setEnergy [mean_energy] [Unit]
/gate/source/ [Source name] /setSigmaEnergy [energy_standard_deviation] [Unit]
```

The source position can be set as follows:

```
/gate/source/ [Source name] /setPosition [Pos_X Pos_Y Pos_Z] [Unit]
```

The pencil beam shape is Gaussian. The spot size can is defined by the standard deviation of the normal probability density function in x and y directions. The beam default direction being +z:

```
/gate/source/ [Source name] /setSigmaX [spot_size_X] [Unit]
/gate/source/ [Source name] /setSigmaY [spot_size_Y] [Unit]
```

The beam is also characterized by its divergences: Theta in the XoZ plan and Phi in the YoZ plan. The beam divergence is defined by the standard deviation of the normal probability density function:

```
/gate/source/ [Source name] /setSigmaTheta [divergence_Theta] [Unit]
/gate/source/ [Source name] /setSigmaPhi [divergence_Phi] [Unit]
```

The correlation between spot size and divergence (in the two plans) is characterized by the beam emittance. The beam emittance is defined by the standard deviation of the normal probability density function. The Emittance of the beam has to be lower (or equal) than the ellipse phase space area: [Emittance_X_Theta] <= Pi* [divergence_Theta] * [spot_size_X] and [Emittance_Y_Phi] <= Pi* [divergence_Phi] * [spot_size_Y].

Please note that for emittance, the unit cannot be selected and has to be "mm*mrad":

```
/gate/source/ [Source name] /setEllipseXThetaEmittance [Emittance_X_Theta] mm*mrad
/gate/source/ [Source name] /setEllipseYPhiEmittance [Emittance_Y_Phi] mm*mrad
```

When defining the beam parameters, one can define the beam convergence or divergence in the two plans (XoZ and YoZ), by setting the "RotationNorm" either to "positive" for a convergent beam or to "negative" for a divergent beam:

```
/gate/source/ [Source name] /setEllipseXThetaRotationNorm [negative or positive]
/gate/source/ [Source name] /setEllipseYPhiRotationNorm [negative or positive]
```

Users can also define the direction of the beam, which is by default +z (0 0 1), by rotating the beam along the x, y and z axis. For instance, to rotate the beam direction around the x-axis by 90°:

```
/gate/source/ [Source name] /setRotationAxis 1 0 0
/gate/source/ [Source name] /setRotationAngle 90 deg
```

A TestFlag can be turned on for advanced testing of the source only. It provides additional output:

```
/gate/source/ [Source name] /setTestFlag true
```

The number of particles simulated is defined using the conventional command:

```
/gate/application/setTotalNumberOfPrimaries [number_of_primaries]
```

Example

In the following example, we defined a 180 MeV proton beam, with 1 MeV energy spread. The beam is asymmetrical and convergent. The direction is -Y:

```
/gate/source/addSource PBS PencilBeam
/gate/source/PBS/setParticleType proton
/gate/source/PBS/setEnergy 188.0 MeV
/gate/source/PBS/setSigmaEnergy 1.0 MeV
/gate/source/PBS/setPosition 0 0 0 mm
/gate/source/PBS/setSigmaX 2 mm
/gate/source/PBS/setSigmaY 4 mm
/gate/source/PBS/setSigmaTheta 3.3 mrad
/gate/source/PBS/setSigmaPhi 3.8 mrad
/gate/source/PBS/setEllipseXThetaEmittance 15 mm*mrad
/gate/source/PBS/setEllipseXThetaRotationNorm negative
/gate/source/PBS/setEllipseYPhiEmittance 20 mm*mrad
/gate/source/PBS/setEllipseYPhiRotationNorm negative
/gate/source/PBS/setRotationAxis 1 0 0
/gate/source/PBS/setRotationAngle 90 deg
/gate/application/setTotalNumberOfPrimaries 10
```

Pencil beam source coordinate system

2.6.3 TPS Pencil Beam source

The source of the simulation can be a stack of pencil beams. This source has been designed in order to allow the simulation of real treatment plans for active beam scanning delivery techniques.

For a more practical understanding of the source, the user is invited to execute the TPS source validation procedure available in the GitHub GateContrib of Gate, in the GATE-RTion branch.

Create the source:

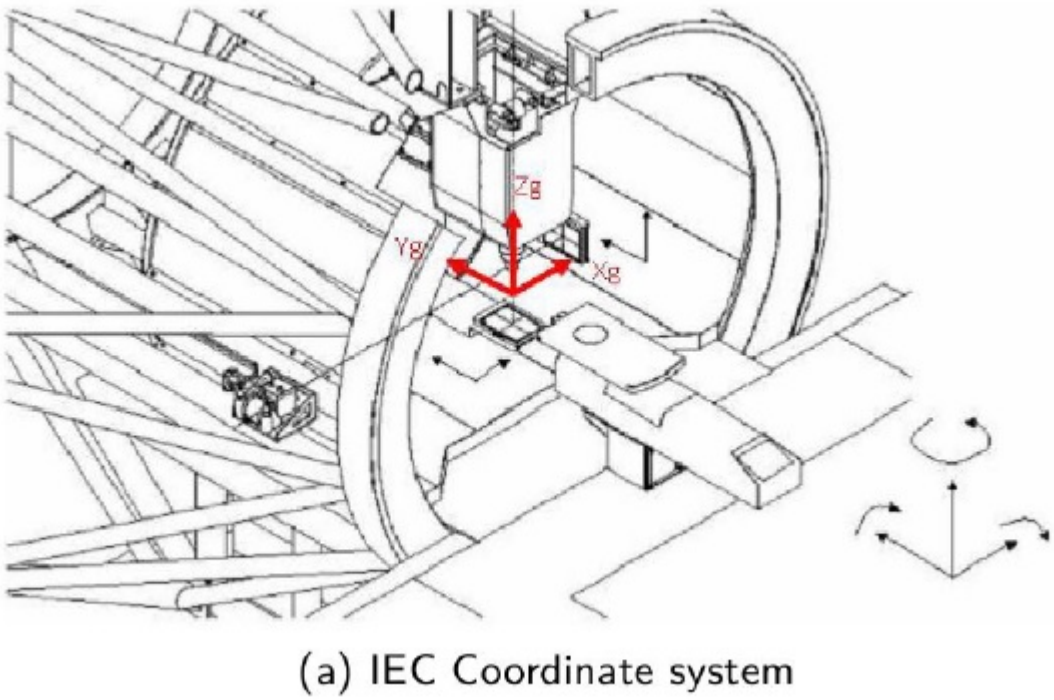


Fig. 2.37: PBS coordinate

```
/gate/source/addSource [Source name]  TPSPencilBeam
```

One can select the type of particle used for the pencil beam (proton, e-, etc.):

```
/gate/source/ [Source name] /setParticleType [particle_type]
```

Alternatively, one can define a specific type of ion, by defining the particle type as "GenericIon" and then specifying the particle parameters of the ion to be generated: Z: AtomicNumber, A: AtomicMass, Q: Charge of Ion (in unit of e), E: Excitation energy (in keV). As an example, the definition of a C12 ion beam is given:

```
/gate/source/  [Source name]  /setParticleType GenericIon
/gate/source/PBS/setIonProperties 6 12 6 0
```

A treatment plan is made of one or multiple fields, each field being described by a gantry angle and a collection of pencil beams having different energies, directions, weights etc. user has to select the "plan description file" of the simulation:

```
/gate/source/ [Source name] /setPlan [plan_description_file]
```

It is possible to simulate all fields simultaneously or only some of them, by using the setting the "setAllowedFieldID" or "setNotAllowedField" commands. In the example below, all fields will be simulated except the field [field_ID_3]:

```
/gate/source/ [Source name] /setNotAllowedFieldID [field_ID_3]
```

In the example below, only the field [field_ID_3] will be simulated:

```
/gate/source/ [Source name] /setAllowedFieldID [field_ID_3]
```

In case a single field for delivery is selected, it is also possible to select a specific layer in that field:

```
/gate/source/ [Source name] /setAllowedFieldID [field_ID]
/gate/source/ [Source name] /selectLayerID [n; n=0 being the first layer]
```

In case a single field and a single layer for delivery are selected, it is also possible to select a specific spot in that field:

```
/gate/source/ [Source name] /setAllowedFieldID [field_ID]
/gate/source/ [Source name] /selectLayerID [n; n=0 being the first layer]
/gate/source/ [Source name] /selectSpotID [m; m=0 being the first spot of that layer]
```

In the "plan description file", each single spot is characterized by its position at treatment isocenter and also by its weight or metersetweight (intensity). In some cases the spot metersetweight provided by in the "treatment plan file" corresponds directly to a number of particles N (first scenario). In this case, the user should use the following command:

```
/gate/source/ [Source name] /setSpotIntensityAsNbIons true
```

In other cases, the spot metersetweight provided by in the "treatment plan file" corresponds to a number of Monitor Units MU (or counts) and the relationship between MU and number of particles N must be inserted in the simulation (second scenario). In this case, the user should use the command below and provide a polynomial description of the calibration curve (N/MU) as a function of energy E into the "source description file" (default option):

```
/gate/source/ [Source name] /setSpotIntensityAsNbIons false
```

It is possible to simulate each spot either with the same probability (flat generation) or stochasticaly by accounting for the spot intensity (probability density function). The second option is strongly advised (for efficiency) and used by default. In case the first option will be selected, the intensity of each spot will be set to 1 and the scoring of each spot will be weighted by the initial spot intensity. It is possible to select the first or second option by setting the "FlatGenerationFlag" to true or false, respectively:

```
/gate/source/ [Source name] /setFlatGenerationFlag [true or false]
```

The delivery of each spot in the treatment plan can either be sorted (from first spot read in the "treatment plan file" to the latest) or randomly (i.e. in a stochastic fashion) accordingly the spot intensity (default option):

```
/gate/source/ [Source name] /setSortedSpotGenerationFlag [true or false]
```

The physical properties of each single pencil beam delivered are computed using the "source description file". This file consists in a set of polynomial equations allowing to define the physical and optical properties of each single pencil beam with energy, as well as the calibration N/MU as a function of energy (in case the option setSpotIntensityAsNbProtons is set to false). Pencil beam properties are those described in the previous section "Pencil Beam source":

```
/gate/source/ [Source name] /setSourceDescriptionFile [source_description_file]
```

Irradiation systems can be tuned with either a convergent beam or a divergent beam. By default, the system is defined as divergent:

```
/gate/source/ [Source name] /setBeamConvergence [true or false]
```

In some cases, it could be that one axis is divergent and the other convergent (or vice-versa). The following options allow setting the convergence/divergence properties of the beam separately:

```
/gate/source/ [Source name] /setBeamConvergenceXTheta [true or false]
/gate/source/ [Source name] /setBeamConvergenceYPhi [true or false]
```

The polynomial function describing the energy spread of the beam can be provided either in percentage of the mean energy (default option) or in absolute MeV:

```
/gate/source/ [Source name] /setSigmaEnergyInMeVFlag [true or false]
```

A TestFlag can be turned on for advanced testing of the source only. It provides additional output:

```
/gate/source/ [Source name] /setTestFlag true
```

The number of particles simulated is defined using the conventional command:

```
/gate/application/setTotalNumberOfPrimaries 10
```

Example

The following example shows how to simulate a proton treatment plan based on the 2 following input files: "MyPlanDescriptionFile.txt" and "MySourceDescriptionFile.txt". The beam is considered convergent and the spot intensities are defined as number of protons:

```
/gate/source/addSource PBS TPSPencilBeam
/gate/source/PBS/setParticleType proton
/gate/source/PBS/setPlan MyPlanDescriptionFile.txt
/gate/source/PBS/setNotAllowedFieldID 1
/gate/source/PBS/setFlatGenerationFlag false
/gate/source/PBS/setSourceDescriptionFile MySourceDescriptionFile.txt
/gate/source/PBS/setSpotIntensityAsNbIons true
/gate/source/PBS/setBeamConvergence true
/gate/application/setTotalNumberOfPrimaries 10
```

About the "source_description_file"

It contains the source to isocenter distance, and scanning magnets distance to isocenter in x- and y-directions. These parameters allow for computing the position and direction of each single pencil beam at the source position defined by the user (nozzle entrance or exit). It contains 9 polynomial equations: 2 describing the energy properties (mean energy in MeV and energy spread in % or MeV), 6 describing the optical properties of the beam (spot size in mm, beam divergence in rad and beam emittance in mm.rad; each in x- and y-directions), 1 describing the beam monitor calibration in number of particles per monitor unit (N/MU). Polynomials are functions of the system energy, which is read in the "plan description file" for each pencil beam. For each polynomial, one has to give the polynomial order and then the polynomial parameters. For instance, for a second order polynomial ($ax^2 + bx + c$), one has to give the polynomial order: 2, followed by the a, b and c parameters in this order. Please note that their is no choice about the units used for the different polynomials!! Please have a look to example "example_Radiotherapy/example5" in the source code. Warning, it is possible to override the definition of the energy spread (% or MeV) directly in the source description file, by adding "%" or "PERCENT" or "percent" or "MeV" directly before the polynomial order in the source description file.

About the "plan_description_file"

It contains many informations about the plan, but not all of them are taken into account for the simulation, as for instance the number of fractions. These additional informations may be used in further releases. Please have a look to example "example_Radiotherapy/example5" and "Gate/examples/example_Radiotherapy/example5/data/PlanDescriptionToGATE.txt" file. Warning, the unused fields of the plan description file cannot be removed. The main parameters of the file are the number of fields, gantry angle for each field, energy of each layer from each field, number of spots in each layer, spot description (position in x- and y- direction at isocenter and intensity) for each spot from each layer.

TPS Pencil Beam source coordinate system in relationship with the Pencil Beam Source

The Pencil Beam source (PBS) is set-up according to IEC coordinate system; i.e. beam direction +Z, spot position in X and Y (see picture below)

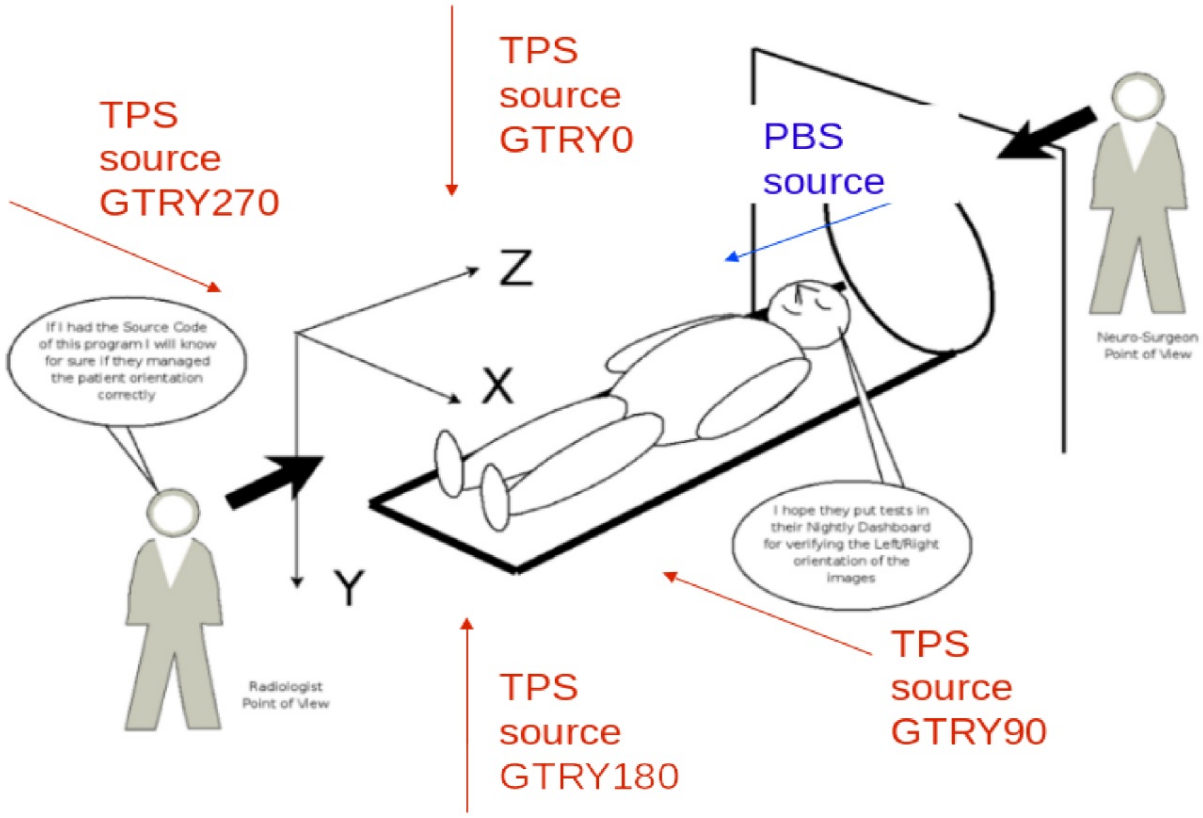


Fig. 2.38: TPS coordinate

The TPS PencilBeam source (TPS) consists in a collection of PencilBeam sources and DICOM image coordinate system is considered as shown in the picture. Relationship between the two sources (from IEC coordinates to DICOM coordinates) is summarized in the table below:

Table 2.4: PBS source

	PBS source (+X, +Y, +Z)
Gantry 0	TPS source (+X, +Z, +Y)
Gantry 90	TPS source (+Y, +Z, -X)
Gantry 180	TPS source (-X, +Z, -Y)
Gantry 270	TPS source (-Y, +Z, +X)

Activated default options: SUMMARY

Table 2.5: Activated default options

TPS source characteristic	Active option by default	Related command
Spot weight (intensity)	Number of Monitor Unit (MU)	setSpotIntensityAsNbIons false
Dose scoring weight	Set to 1	setFlatGenerationFlag false
Spot delivery	Random (stochastic fashion)	setSortedSpotGenerationFlag false
Beam convergence	Divergent	setBeamConvergence false
Beam energy spread*	Percentage	setSigmaEnergyInMeVFlag false

**warning: it can be overwritten based on your source description file!*

2.6.4 The fastY90 source

The fastY90 source will be part of GATE release 8.0, but it is also available in the development versions of GATE 7.2 availiable on GitHub as of June 2016

The *fastY90* source can be used to simulate PET or SPECT imaging of Y90 sources. Rather than simulating the full electron transport of the emitted beta particle, the *fastY90* source uses a pre-calculated bremsstrahlung kernel to generate the photons directly to speed up the simulation. Note that since the kernel has been calculated using a point source in water, simulations that use this source are only valid for modelling sources inside water or materials of similar density and Zeff. For accurate simulation, the attenuating media must also extend somewhat beyond the range of the source by several mm. Although the size of the pre-calculated kernel has a radius of 12 mm, more than 95% of all bremsstrahlung is generated within 6 mm of the source, a higher fraction if only the higher energy bremsstrahlung is considered.

The *fastY90* model includes the positron arising from internal pair production (0+/0+ transition), though not the 2.186 MeV gamma (2+/0+ transition).

To use the fastY90 source:

```
/gate/source/addSource mySource fastY90
```

Simuations with the *fastY90* source can be further sped up by adding a low energy cutoff to the bremsstrahlung generation, effictively ignoring those bremsstrahlung photons with too little energy to play any role in imaging. For example:

```
/gate/source/mySource/setMinBremEnergy  50 keV
```

The Y90 decay produces a positron with a prevalence of about 31.86 ppm. Although the model defaults to this value, it can be modified (for testing purposes, for example) by the setPositronProbabiliity command:

```
/gate/source/mySource/setPositronProbability  0.00003186
```

Using a voxelized distribution with the fastY90 source

The *fastY90* source can be used with a voxelized distribution. The voxelized distribution must be in the Interfile format, with a header file that contains, at minimum, the name of the data file, the matrix size, and the scale factor:

```
/gate/source/mySource/loadVoxelizedPhantom tia_map.h.hdr

!INTERFILE :=
!name of data file :=tia_map.v
matrix size[1] := 256
matrix size[2] := 256
matrix size[3] := 147
scale factor (mm/pixel) [1]:= 1.91
```

(continues on next page)

(continued from previous page)

```
scale factor (mm/pixel) [2]:= -1.91
scale factor (mm/pixel) [3]:= -2.00
```

The data file must be a raw binary containing data in IEEE 32-bit floating point format. The voxelized distribution will be scaled internally to create a 3D probability map of the geometry of the source, but the total activity is set by the setActivity command as for any other source. By default, the location of the voxelized source will be centred at the origin. The position of the voxelized distribution can also be changed using the setVoxelizedPhantomPosition command to specify the position of the first pixel in the data file:

```
gate/source/mySource/setVoxelizedPhantomPosition -3.5 6.0 -10.0 cm
```

2.7 Voxelized source and phantom

Table of Contents

2.7.1 Introduction

Voxelized source and phantom's purpose is the use of digital phantoms or patient data as inhomogeneous anthropomorphic attenuation maps or emission sources so as to simulate and thus reproduce realistic acquisitions.

From its first public release, GATE is able to read digital phantom or clinic data stored in various image file formats so as to allow users to read attenuation maps or emission data from these voxelized phantoms and sources.

To read in voxelized geometry, GATE requires a text file for the description of materials (AttenuationRange.dat for instance) and a 3D image stored in one of the following formats: **ASCII**, **Interfile** (8-bit, 16- or 32-bit Signed and Unsigned, and 32- or 64-bit Real), **Analyze**, **MetaImage** and **DICOM**. The text file must provide a number of subdivisions, define intervals associated to each subdivision and attach them with a correlated material name.

Example of AttenuationRange.dat file:

```
# Number of subdivisions
3
# Define the intervals and attach a correlated material
0 10 Air
11 200 Water
201 255 SpineBone
```

To read in voxelized sources, GATE requires a text file for the description of activity levels (ActivityRange.dat for instance) and a 3D image stored in one of the following formats: **ASCII**, **Interfile** (8-bit, 16- or 32-bit Signed and Unsigned, and 32- or 64-bit Real), **Analyze**, **MetaImage** and **DICOM**. The text file must provide a number of subdivisions, define intervals associated to each subdivision and attach them with a correlated activity (in Bq). The possibility to extend image file formats for voxelized sources is still under development.

Example of ActivityRange.dat file:

```
# Number of subdivisions
6
# Define the intervals and attach a correlated activity (in Bq)
200 210 1.
211 220 3.
221 230 5.
231 240 10.
241 250 20.
251 255 40.
```

2.7.2 Voxelized phantoms

Description of voxelized geometry

To import digital phantom or patient data as a voxelized geometry, GATE to use a special "navigator" algorithm that allow to quickly track particles from voxel to voxel. There are several navigators available. We recommend "**ImageNestedParametrisedVolume**" for most use.

Regular parameterization method

Since Geant4.9.1 a new navigation algorithm, dubbed regular navigation, can be used for the tracking of particles in voxelized volumes. The regular navigation algorithm performs fast direct neighbouring voxel identification without a large memory overhead. This is the major source of acceleration of the implemented regular navigation algorithm. In addition, boundaries between voxels which share the same material can be ignored. Using this option, the geometry only limits tracking at the boundary between voxels with different materials, providing a significant reduction of

the number of propagation steps. The regular navigator uses a new algorithm that performs this search only for the neighbours of the actual voxel. It therefore highly reduces the time spent on this search, as much as the number of voxels is large. It also includes a new method called ComputeStepSkippingEqualMaterials; when a boundary is encountered, the navigator searches for the next voxel it should enter and check if its material is the same as the actual one. If this is the case, this method is directly called again without passing through the navigator manager which loads the new properties of the next voxel, etc. Therefore the fewer the materials, the faster the simulation. In conclusion, the time saved using the regular navigator is directly dependent on the number of voxels and the number of different materials contained in the voxelized phantom. The better acceleration factors were obtained while simulating PET acquisitions (3 different materials: air, water, bone) with finely sampled phantom definitions. This factor could be around 3 in those cases. However in any case, even with a lot of different materials, this navigator will always be faster than using older navigators such as parameterizedBoxMatrix or compressedMatrix. That is the reason why these navigators still be progressively deprecated.

Additionally, as the SkipEqualMaterials method can lead to fewer G4steps, one may want to force the stepping process at each boundary. In that case, the method can be inactivated using the following command:

```
/gate/world/anyname/setSkipEqualMaterials 0
```

Also the particle tracking will inevitably be less effective.

Nested parameterization method

Another method of creating parametrized volumes in Geant4, using nested parametrization and the corresponding navigation algorithm has been available in GATE since version 6.1. Based on parametrized approach, this method allows GATE storing a single voxel representation in memory and dynamically changing its location and composition at run-time during the navigation. The main advantage of this method is high efficiency in memory space. While reusing the same mechanism as parameterized volume, Nested representation also splits the 3D volume along the three principal directions, allowing logarithmic finding of neighbouring voxels. Nested approach supposes geometry has three-dimensional regular reputation of same shape and size of volumes without gap between volumes and material of such volumes are changing according to the position. Instead of direct three-dimensional parameterized volume, one can use replicas for the first and second axes sequentially, and then use one-dimensional parameterization along the third axis. This approach requires much less memory access and consumption for geometry optimization and gives much faster navigation for ultra-large number of voxels. Using Nested representation, images are split into sub-volumes of homogeneous composition, which are parallelepipeds, either of the voxel size or larger. The main drawback is that all the particles are forced to stop at the boundaries of all parallelepipeds, generating a supplementary step and additional time cost, even if the two neighboring parallelepipeds share the same content. Such artificial steps occur very often as human organs are far from being parallelepipedic.

Regionalized parameterization method

Recently, some GATE developers have proposed a new method for efficient particle transportation in voxelized geometry for Monte Carlo simulations, especially for calculating dose distribution in CT images for radiation therapy. The proposed approach, based on an implicit volume representation named segmented volume, coupled with an adapted segmentation procedure and a distance map, allows them to minimize the number of boundary crossings, which slows down simulation. Before being implemented within GATE, the method was developed using the GEANT4 toolkit and compared to four other methods: one box per voxel, parameterized volumes, octree-based volumes, and nested parameterized volumes. For each representation, they compared dose distribution, time, and memory consumption. The proposed method allows them to decrease computational time by up to a factor of 15, while keeping memory consumption low, and without any modification of the transportation engine. Speeding up is related to the geometry complexity and the number of different materials used. They obtained an optimal number of steps with removal of all unnecessary steps between adjacent voxels sharing a similar material. However, the cost of each step is increased. When the number of steps cannot be decreased enough, due for example, to the large number of material boundaries,

such a method is not considered suitable. Thus, optimizing the representation of an image in memory potentially increases computing efficiency.

Warning. In some situations, for example computation of dose distribution, StepLimiter could be required to avoid too large steps. In doubt, use ImageNestedParametrisation.

Fictitious interaction

Important note: so far, this method is available in GATE v7.0 version using Interfile reader only.

For detailed information, please refer to Fictitious interaction section

Description of voxelized phantoms

Regular, Nested and Regionalized parametrization methods together with their corresponding navigation algorithms are available in GATE V7.0 for voxelized phantoms. Note that so far their current implementations do not support voxel visualization attributes on a per material basis.

To create a parameterized phantom object using any of the three above-mentioned methods, one can use the corresponding command lines as follows:

```
/gate/world/daughters/name anyname
/gate/world/daughters/insert  ImageRegularParametrisedVolume
```

Or:

```
/gate/world/daughters/name anyname
/gate/world/daughters/insert ImageNestedParametrisedVolume
```

Or:

```
/gate/world/daughters/name anyname
/gate/world/daughters/insert ImageRegionalizedVolume
```

All these three methods supports 3D images stored in various image file formats, which is automatically defined from their extension:

- ASCII
- Interfile format: header .h33 + raw image .i33
- Analyze format: header .hdr + raw image .img
- MetaImage format: header. mhd + raw image .raw
- DICOM format: a series of .dcm files

Conversion into material definitions

Whatever the navigation algorithm selected, conversion of image grayscales into material definitions is performed as follows:

When already defined in GateMaterials.db file, appropriate material properties are assigned to each voxel encoded value using either a range translator (to be used with images providing label values) or a units to materials conversion descriptor (to be used with images providing label or HU values).

Range translator

Tables read by the range translator have a prescribed format.

The first line defines the number of material subdivisions (i.e. the number of subsequent lines). The following lines describe the intervals (i.e. range) of encoded values (bits or segmented values) associated to each subdivision (i.e material), followed by a material name. A particular material will be assigned to each voxel whose value falls within the material range.

One can keep specifying visibility boolean (true or false) and color attribute values (red, green, blue components and transparency coefficient) within the range translator as he did when using a previous GATE version. However, as previously mentioned, so far current implementations of any of the three new parametrization methods do not support voxel visualization attributes on a per material basis, hence preventing the voxelized phantom from being displayed.

Example of a range translation table (AttenuationRange.dat, for instance):

```
4
0 0 Air false 0.0 0.0 0.0 0.2
4 4 Water true 1.0 0.0 1.0 0.2
5 5 Water true 0.0 1.0 0.0 0.2
6 15 SpineBone true 0.0 0.0 1.0 0.2
```

In this example, the number of material subdivisions is 4. Material Air is assigned to pixels with value 0, Water to pixels with value 4 and 5, and SpineBone to pixels with value between 6 and 15.

Units to materials conversion descriptor

Units to materials conversion descriptor is a simple text file with three columns: Label or HU_start, Label or HU_end and material_name. It allows to associate a material to each label or HU voxel encoded value in the image. This text file can be written by hand or generated with the automated method, especially for HU values (see *For RT applications only: Automated HU stoichiometric calibration*)

Example of a units to materials conversion descriptor (AttenuationRange.dat, for instance):

```
6
0 1  Air
1 4  Water
4 6  Bone
6 16 SpineBone
```

In this example, the material Air is assigned to pixels with value 0, Water to pixels with value between 1 and 3 and 4 and 5, and SpineBone to pixels with value between 6 and 15.

Example of voxelized phantom description macro

Example:

```
# VOXELIZED PHANTOM BASED ON PATIENT DATA
/gate/world/daughters/name                          patient

# INSERT THE PARAMETERIZATION METHOD AND THE CORRESPONDING NAVIGATION ALGORITHM THE␣
↪MOST APPROPRIATE TO YOUR SIMULATION
/gate/world/daughters/insert                        ImageRegularParametrisedVolume
/gate/world/daughters/insert                        ImageNestedParametrisedVolume
/gate/world/daughters/insert                        ImageRegionalizedVolume
```

(continues on next page)

(continued from previous page)

```
# READ IMAGE HEADER FILE (.H33 FOR INTERFILE, .MHD FOR METAIMAGE AND .HDR FOR ANALYZE␣
↪FORMATS)
## In this example, patient.h33 is the header filename of the image stored in␣
↪Interfile file format. This file format is simple. It consists of two files: 1)␣
↪patient.h33 is a ASCII file
with the header description of the image (sizes, spacing and origin and other␣
↪information), and 2) pixels values as binary patient.i33 data.
/gate/patient/geometry/setImage                  data/patient.h33


# [OPTIONAL]
## patient-HUmaterials.db is a text file with patient-related material description.␣
↪If all the wanted material are already into the GateMaterials.db you do not need␣
↪such additional file.
HU means Hounsfield Units because RT applications mainly used CT images. However, any␣
↪image type can be used (with floating point voxel value).
/gate/geometry/setMaterialDatabase               data/patient-HUmaterials.db


# INSERT THE TRANSLATOR THAT WILL CONVERT THE IMAGE FROM DIGITAL VALUES TO MATERIAL␣
↪INFORMATION
# RANGE TRANSLATOR (LABEL VALUES)
/gate/patient/geometry/setRangeToMaterialFile    data/patient-HU2mat.dat
#UNITS TO MATERIALS CONVERSION DESCRIPTOR (LABEL OR HU VALUES)
## When dealing with HU values, this text file can be written by hand or generated␣
↪with the automated method (see Automated HU stoichiometric calibration).
/gate/patient/geometry/setHUToMaterialFile       data/patient-HU2mat.dat


# AS WITH OTHER OBJECTS, ADDITIONAL OPTIONS REGARDING THE POSITION OF THIS PHANTOM␣
↪CAN ALSO BE SPECIFIED
/gate/patient/placement/setTranslation           0. 0. 0. mm
/gate/patient/placement/setRotationAxis          1 0 0
/gate/patient/placement/setRotationAngle         0 deg

# ADD THIS COMMAND LINE, IF YOU WANT TO RETRIEVE INFORMATION ABOUT THE COMPTON AND␣
↪RAYLEIGH INTERACTIONS WITHIN THIS PHANTOM VOLUME
/gate/hof_brain/attachPhantomSD

# FOR IMAGEREGULARPARAMETRISEDVOLUME NAVIGATOR ONLY. COMMAND USED TO SPEED-UP␣
↪NAVIGATION
/gate/hof_brain/setSkipEqualMaterials            1
```

Example of Interfile format header:

```
!INTERFILE :=
!GENERAL IMAGE DATA :=
!type of data := Tomographic
!total number of images := 16
study date := 1997:11:28
study time := 00:00:00
imagedata byte order := LITTLEENDIAN
!number of images/energy window := 16
!process status := Reconstructed
!matrix size [1] := 32
!matrix size [2] := 32
```

(continues on next page)

(continued from previous page)

```
!number format := unsigned integer
!number of bytes per pixel := 2
scaling factor (mm/pixel) [1] := +8.8
scaling factor (mm/pixel) [2] := +8.8
!number of projections := 16
!extent of rotation :=
!time per projection (sec) := 0
study duration (sec) := 0
!maximum pixel count := +2.000000e+02
patient orientation := head_in patient
rotation := supine
!GENERAL DATA :=
!data offset in bytes := 0
!name of data file := brain_phantom.i33
```

Using such an image reader, digital phantom or patient data can be read in as voxelized attenuation geometries. Additionally, when a sensitive detector (phantomSD) is associated to this phantom, the system can retrieve information about the Compton and Rayleigh interactions within this volume.

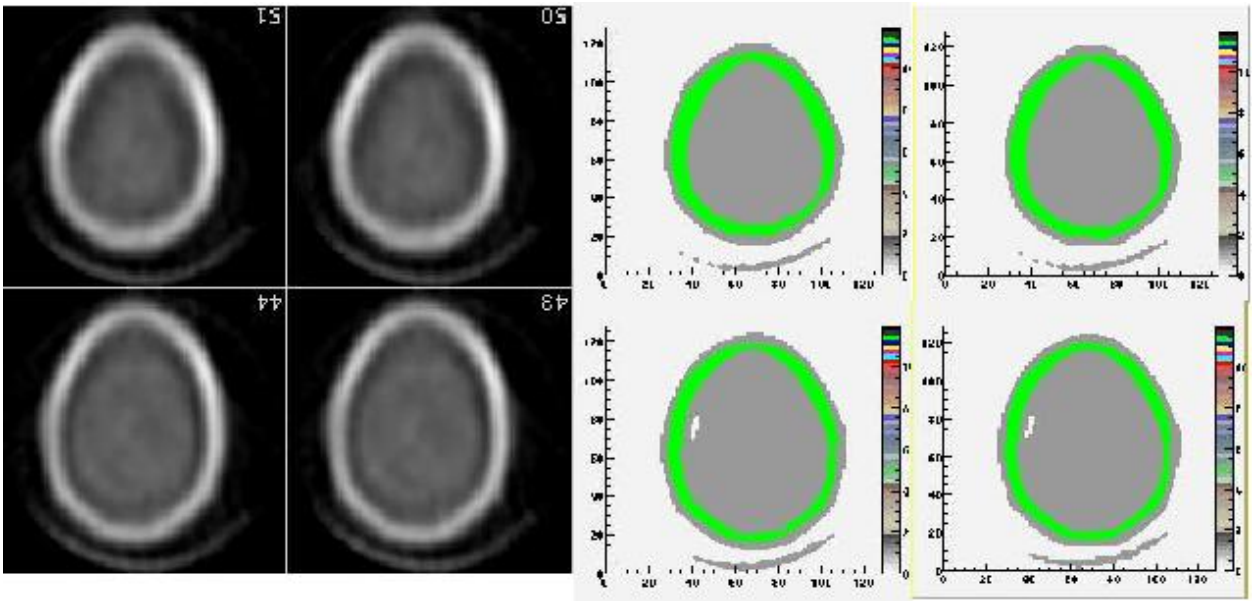


Fig. 2.39: attenuation map from digital Hoffman phantom (left:data - right: after translation to material definition).

For RT applications only: Automated HU stoichiometric calibration

To generate a correspondence between HU (voxel values) and material, you may use the following commands:

```
/gate/HounsfieldMaterialGenerator/SetMaterialTable                  data/
↪Schneider2000MaterialsTable.txt
/gate/HounsfieldMaterialGenerator/SetDensityTable                   data/
↪Schneider2000DensitiesTable.txt
/gate/HounsfieldMaterialGenerator/SetDensityTolerance               0.1 g/cm3
/gate/HounsfieldMaterialGenerator/SetOutputMaterialDatabaseFilename data/patient-
↪HUmaterials.db
/gate/HounsfieldMaterialGenerator/SetOutputHUMaterialFilename       data/patient-
↪HU2mat.txt
```

(continues on next page)

(continued from previous page)

```
/gate/HounsfieldMaterialGenerator/Generate
```

In this case, you need to provide:

- "Schneider2000MaterialsTable.txt" calibration text file allowing to split the HU range into several materials (see [Schneider2000]).
- "Schneider2000DensitiesTable.txt" calibration text file to indicate the relation between HU and mass density (g/cm3). It is normaly given by calibration of you CT scanner. It is critical that you note that density values you provide must match to the HU values you declare, so if you set initial HU values for the materials you have to provide initial density values also. It is a common mistake that you provide the mean density, making Gate overestimate it when perform the interpolation, so be careful.
- the parameter "DensityTolerance" allows the user to define the density tolerance. Even if it is possible to generate a new Geant4 material (atomic composition and density) for each different HU, it would lead to too much different materials, with a long initialization time. So we define a single material for a range of HU belonging to the same material range (in the first calibration Table) and with densities differing for less than the tolerance value.
- the files "patient-HUmaterials.db" and "patient-HU2mat.txt" are generated and can be used with setMaterialDatabase and SetHUToMaterialFile macros.

Examples are available *GateRT*

2.7.3 Voxelized sources

Since release V7.1, possibilities to read in voxellized sources within GATE have been extended. They all require the user to provide 3D image stored in one of the following formats: Interfile (8-bit, 16- or 32-bit Signed and Unsigned, and 32- or 64-bit Real), Analyze, MetaImage and DICOM.

Conversion into activity values

Each voxel of the grayscale image is converted into actual activity value using either a linear or a range (same kind as the voxelized phantom one) translation table.

An example of a range translation table from voxel encoded values to activities (ActivityRange.dat in the example) is shown below:

```
3
4  4    1
5  5    3
14  15  5
```

where you specify the number of subdivisions or intervals (3 in this example), followed by the intervals definition and the correlated activity attached to each interval. If the number in the ASCII file, for a given voxel, is for instance between 14 and 15, then the activity for that voxel is set to 5. Bq. The resulting voxelized source has thus a discretized number of activity values (preliminar segmentation).

Position

By default the activity image is placed in the "first quarter", i.e. the image is placed starting at 0 0 0 mm (x, y, z). This is different from the placing of volumes in Geant4 and GATE where volumes are centered on their geometrical center. In order to align the activity image with a phantom volume, a translation is needed which in the simplest case (where

the activity image has the same size and position as the volume) is a translation of half the size of the volume in the negative direction.

However, when the activity image and volume have different size and/or offset, the above translation should be added to the the image and phantom offset in order to correctly place the phantom and the activity image.

Example below:

```
#Activity image of 100 100 100 mm with an offset of 50 50 50 mm
#Phantom volume of 200 200 200 mm with an offset of 0 0 0 mm
#Phantom placement by GATE will be -100 -100 -100 mm
#The translation to apply is:
#translation = (50 50 50) + (0 0 0) + (-100 -100 -100) = (-50 -50 -50)
/gate/source/activityImage/setPosition -50 -50 -50 mm
```

Note: when reading .mhd images GATE doesn't take into account the Offset keyword. To take the Offset keyword into account for the geometry the method /gate/volumeName/geometry/TranslateTheImageAtThisIsoCenter x y z unit has to be applied. For the source, the method /gate/source/sourceName/TranslateTheSourceAtThisIsoCenter x y z unit has to be applied. These methods can be used to align the source on the geometry.

Example of voxelized source description macro

Example of voxelized source description macro reading in an InterFile image as source distributions is detailed below:

```
!!!WARNING: Macro commands related to voxelized source description have been modified␣
↪in GATE V7.1!!!
Older ones are being deprecated and will be removed from the next release

# DECLARATION OF THE FACT THAT A VOXELIZED SOURCE WILL BE USED
# Always use the keyword voxel to declare the type
/gate/source/addSource                                          hof_brain voxel
# DECLARATION THAT THE VOXELIZED SOURCE WILL BE ENTERED USING IMAGE DATA
/gate/source/hof_brain/reader/insert                            image

# INSERT THE TRANSLATOR THAT WILL CONVERT THE IMAGE FROM DIGITAL VALUES TO ACTIVITY␣
↪VALUES
# Example for a linear translator: this scales all image values directly into␣
↪activities
/gate/source/hof_brain/imageReader/translator/insert            linear
/gate/source/hof_brain/imageReader/linearTranslator/setScale    1. Bq
# Example for a range translator (can not be used simultaneously)
# Here the values of the image file are discretized in intervals and are then␣
↪converted to predefined activities
/gate/source/hof_brain/imageReader/translator/insert            range
/gate/source/hof_brain/imageReader/rangeTranslator/readTable    ActivityRange.dat
/gate/source/hof_brain/imageReader/rangeTranslator/describe     1

# THE FOLLOWING LINE ALLOWS YOU TO INSERT THE IMAGE DATA USING THE APPROPRIATE␣
↪EXTENSION FILE
/gate/source/hof_brain/imageReader/readFile                     hof_brain_phantom.
↪h33
/gate/source/hof_brain/imageReader/verbose                      1

# THE DEFAULT POSITION OF THE VOXELIZED SOURCE IS IN THE 1ST QUARTER
# SO THE VOXELIZED SOURCE HAS TO BE SHIFTED OVER HALF ITS DIMENSION IN THE NEGATIVE␣
↪DIRECTION ON EACH AXIS
/gate/source/hof_brain/setPosition                              -128. -128. 0. mm
```

(continues on next page)

(continued from previous page)

```
/gate/source/hof_brain/dump                                          1

# THE FOLLOWING LINES CHARACTERIZE THE SIZE (NO DIFFERENCE WITH AN ANALYTICAL SOURCE)
/gate/source/voxel/setType backtoback
/gate/source/voxel/gps/particle gamma
/gate/source/voxel/gps/energytype Mono
/gate/source/voxel/gps/monoenergy 140. keV
/gate/source/voxel/gps/angtype iso
/gate/source/voxel/gps/mintheta 0. deg
/gate/source/voxel/gps/maxtheta 90. deg
/gate/source/voxel/gps/minphi 0. deg
/gate/source/voxel/gps/maxphi 360. deg
/gate/source/voxel/gps/confine NULL
```

Using this image file reader any digital phantom or patient data, stored in any image format among ASCII, Interfile, Analyze, MetaImage and DICOM, can be read in as emission distribution. Afterwards, activity levels can be used to determine the number of primary particles for each voxel.

An example of the Hoffman brain phantom, where the gray scales have been translated to activity distributions is shown in Fig. 2.40.

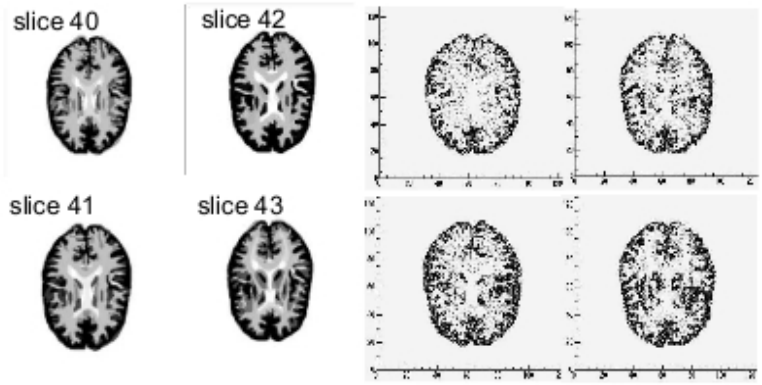


Fig. 2.40: emission map from digital Hoffman phantom (left:data - right: translated activity values).

2.7.4 Dose collection

To collect absorbed dose deposited in the phantom, attach the phantom sensitive detector with the new attachVoxelPhantomSD command (and not attachPhantomSD) and add a dose output module:

```
/gate/anyname/attachVoxelPhantomSD
/gate/anyname/addOutput outputModuleName
```

The output module responds to the following commands:

```
/gate/output/outputModuleName/saveUncertainty [true|false]
/gate/output/outputModuleName/setFileName anyFileName
```

The output file is a binary file (number format is 4 bytes float) containing the absorbed dose in cGy. It has the same dimensions as the phantom. The output module optionally writes a second binary file containing the uncertainty on

absorbed dose expressed as a fraction between 0 and 1. The uncertainty file also has the same dimensions as the phantom and its creation is controlled by the saveUncertainty command. The file name is the same as the absorbed dose file with a capital **U** appended. By default, the output file name is **doseMatrix.bin** and the uncertainty file is not created.

Example:

```
# Create a simple phantom called CCD
/gate/world/daughters/name CCD
/gate/world/daughters/insert parameterizedBoxMatrix

# Read the file : a 300x300x1 array
/gate/CCD/geometry/insertReader image
/gate/CCD/imageReader/insertTranslator tabulated
/gate/CCD/imageReader/tabulatedTranslator/readTable ccdTable.dat
/gate/CCD/imageReader/readFile ccd300Phantom.dat

# Place the phantom and rotate it so that it is in the XZ plane
/gate/CCD/placement/setTranslation 0 -82.269 0 mm
/gate/CCD/placement/setRotationAxis 1 0 0
/gate/CCD/placement/setRotationAngle 90 deg

# Attach the phantom SD and the output module
/gate/CCD/attachVoxelPhantomSD
/gate/CCD/addOutput doseOutput
/gate/output/doseOutput/saveUncertainty true
/gate/output/doseOutput/setFileName ccdDose.bin
```

Comments

Depending on the phantom dimensions, the use of a parameterizedBoxMatrix may increase memory usage by up to a factor of 2 and increase CPU time by 5-50.

If you plan to collect only dose in the phantom, it is suggested that you disable other types of output, for example:

```
/gate/output/ascii/disable
```

Dose calculations

The relative uncertainty on dose is calculated on a per voxel basis. Let $\{d_i\} i = 1, ..., N$ be the sequence of energy deposits in a given voxel, we can calculate the following quantities:

Mean energy deposit:

$\overline{d} = E(d) = \frac{1}{N} \sum_{i=1}^{N} d_i$

Sample variance:

$s^2 = E(d^2) - E(d)^2$

$s^2 = \frac{1}{N^2}[N \sum d_i^2 - (\sum d_i)^2]$

Population variance estimator:

$s^2 = \frac{N}{N-1} s^2$

Standard deviation:

$s = s\sqrt{\frac{N}{N-1}}$

Standard error of the mean:

$\hat{d} = \frac{s}{N} = \frac{s}{\sqrt{N-1}}$

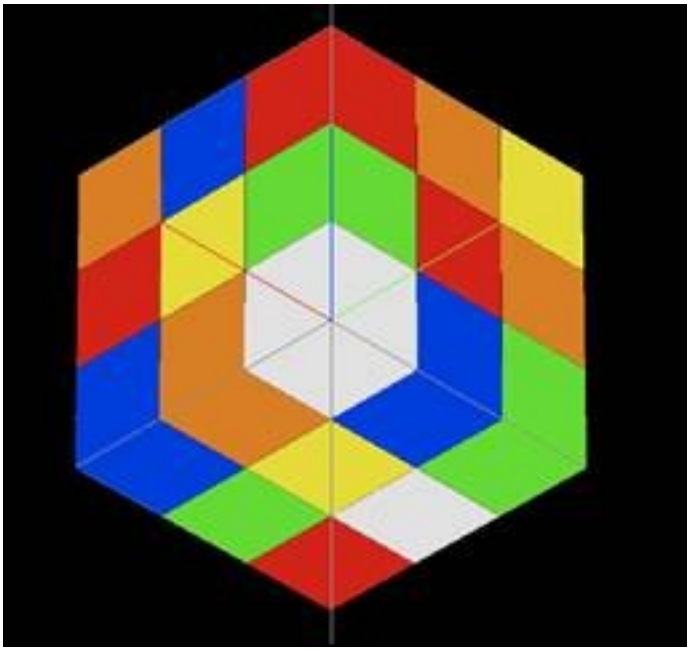


Fig. 2.41: A simple voxellized phantom without transparency

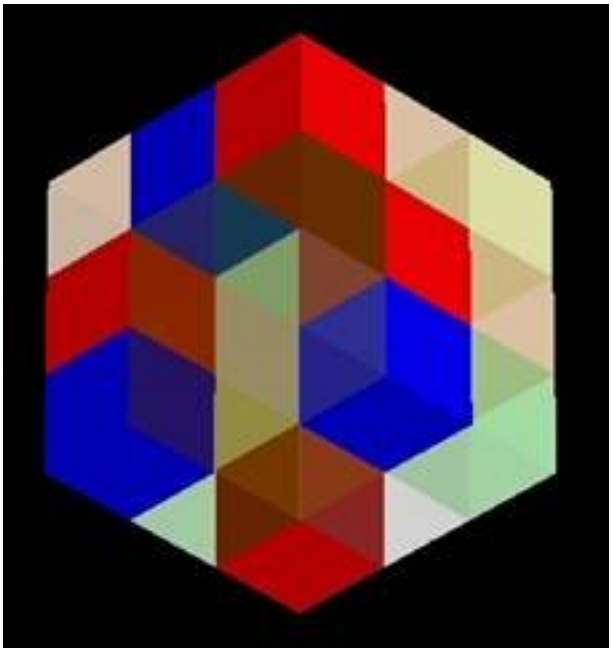


Fig. 2.42: A simple voxellized phantom with transparency

2.7.5 Real-time motion management for voxellized source and phantom

- Generate N frames for the phantom corresponding to the time acquisition desired for example 50 frames for 5s so each frame is for .1 s ;
- I assume the 50 frames are NCAT_frame_.i33, NCAT_frame_2.i33, NCAT_frame_3.i33, NCAT_frame_4.i33 $...$ NCAT_frame_N.i33:

```
/gate/world/daughters/name Ncat
/gate/world/daughters/insert compressedMatrix
/gate/Ncat/geometry/insertReader interfile
/gate/RTPhantom/insert RTVPhantom
/gate/RTVPhantom/AttachTo Ncat
/gate/RTVPhantom/setBaseFileName NCAT
/gate/RTVPhantom/setHeaderFileName NCAT_header.h33
/gate/RTVPhantom/SetNumberOfFrames 50
/gate/RTVPhantom/SetTimePerFrame 0.1 s
/gate/Ncat/interfileReader/insertTranslator range
/gate/Ncat/interfileReader/rangeTranslator/readTable range.dat
/gate/Ncat/interfileReader/rangeTranslator/describe 1
/gate/Ncat/attachPhantomSD
/gate/Ncat/placement/setTranslation  0. 0. 0. mm
/gate/Ncat/interfileReader/describe 1
```

The header NCAT_header.h33 looks like:

```
!matrix size [1] := 128
!matrix size [2] := 128
!number format := unsigned integer
scaling factor (mm/pixel) [1] := +3.125000e+00
scaling factor (mm/pixel) [2] := +3.125000e+00
!number of slices := 128
slice thickness (pixels) := +3.125000e+00
```

For the activity source:

```
# V O X E L    S O U R C E
/gate/source/addSource voxel voxel
/gate/source/voxel/reader/insert interfile
/gate/RTVPhantom/AttachToSource voxel
/gate/source/voxel/interfileReader/translator/insert range
/gate/source/voxel/interfileReader/rangeTranslator/readTable activityRange_test.dat
/gate/source/voxel/interfileReader/rangeTranslator/describe 1
/gate/source/voxel/setType backtoback
/gate/source/voxel/gps/particle gamma
/gate/source/voxel/setForcedUnstableFlag true
/gate/source/voxel/setForcedHalfLife 6586.2 s
/gate/source/voxel/gps/energytype Mono
/gate/source/voxel/gps/monoenergy 0.511 MeV
/gate/source/voxel/setPosition  -200. -200. -200 mm
/gate/source/voxel/gps/confine NULL
/gate/source/voxel/gps/angtype iso
/gate/source/voxel/dump 1

#TIME ACTIVITY option
/gate/source/voxel/interfileReader/SetTimeActivityTablesFrom TimeActivity_Tables.dat
/gate/source/voxel/interfileReader/SetTimeSampling 0.001 s
```

The activityRange_test.dat is a text file looking like this:

```
12
46    46     0
61    61     0
123   123    0
215   215    0
246   246    0
261   261    0
352   352    352
369   369    0
770   770    0
831   831    0
2300  2300   0
950   950    950
```

The TimeActivity_Tables.dat is a text file looking like this:

```
2
950  lesion.dat
352  liver.dat
```

Where the value 950 is the key corresponding in the attenuation map to the lesion and 352 is the key corresponding in the attenuation map to the liver.

Note that lesion.dat is a text file which contains the time activity table curve for the lesion, as explain here:

```
19
0       0
0.01    89.5
0.02    158.8
0.03    199.3
0.04    228.1
0.05    250.4
0.06    268.7
0.08    297.4
0.1     319.7
0.15    360.3
0.2     389.1
0.3     429.6
0.4     458.4
0.5     480.7
0.6     498.9
0.7     514.3
0.8     527.7
0.9     539.5
1       550.0
```

Where first column is the time in second and the second one is the activity in Bq at time t.

2.7.6 Examples

You can find several examples related to the use of voxelized phantom and source in the new repository dedicated to examples (https://github.com/OpenGATE/GateContrib).

- A complete simulation description about How To Use the new voxelized approach for imaging applications

Since GATE v7.0, this new approach is common for both imaging and RT applications. Users need to execute the mainMacro.mac file to start the complete example.

The list of macro files which are involved in this example is the following:

- mainMacro.mac
- MoveVisu.mac
- VoxelizedPhantom.mac
- VoxelizedSource.mac
- MoveVisu.mac
- Verbose.mac

The phantom used in this application (Interfile Format)

- Raw data: brain_phantom.i33
- Header file: brain_phantom.h33

And the associated ASCII files to convert phantom voxel values in material properties and activity values

- range_atten_brain.dat
- activity_range_brain.dat
- Example of photon beam in patient CT image

Two different navigators are tested NestedParameterized and Regionalized, with two number of materials.

Output is a 3D dose distribution map (with associated uncertainty).

- Comparison between MetaImage and DICOM formats

The main simulation files are constructed as follows:

- main-[series name]-[input format].mac
- [series name] : ffp, ffs, hfp, hfs
- [input format]: mhd, dcm

Each main file will generate two mhd images:

- one containing the density dump of the image (density-[series name]-[input format].mhd/.raw)
- one containing the deposited dose in the image (DA-[series name]-[input format]-Dose.mhd/.raw)

Comparison between input formats can be made by the diff command:

```
diff DA-[seriesA]-mhd-Dose.raw DA-[seriesA]-dcm-Dose.raw
```

2.8 Tools to Interact with the Simulation : Actors

Table of Contents

- *Filter on particle type*
- *Filter on particle ID*
- *Filter on volume*
- *Filter on energy*
- *Filter on direction*

Actors are tools which allow to interact with the simulation. They can collect information during the simulation, such as energy deposit, number of particles created in a given volume, etc. They can also modify the behavior of the simulation. Actors use hooks in the simulation : run (begin/end), event(begin/end), track(begin/end), step.

2.8.1 General Purpose

Add an actor

Use the command:

```
/gate/actor/addActor [Actor Type]  [Actor Name]
```

Attach to a volume

Tells that the actor is attached to the volume [Volume Name]. For track and step levels, the actor is activated for step inside the volume and for tracks created in the volume. If no attach command is provided then the actor is activated in any volume. The children of the volume inherit the actor:

```
/gate/actor/[Actor Name]/attachTo   [Volume Name]
```

Save output

This command allow to save the data of the actor to the file [File Name]. The particular behaviour (format, etc.) depends on the type of the actor:

```
/gate/actor/[Actor Name]/save   [File Name]
```

It is possible to save the output every N events with the command:

```
/gate/actor/[Actor Name]/saveEveryNEvents   [N]
```

It is possible to save the output every N seconds with the command:

```
/gate/actor/[Actor Name]/saveEveryNSeconds [N]
```

3D matrix actor (Image actor)

Some actors, such as the *Dose measurement (DoseActor)*, can store some information into a 3D image (or matrix) according to the spatial position of the hit. User can specify the resolution of the 3D matrix (in this case, the size is equal to the size of the bounding box of the attached volume). Alternatively, user can specify the size to allow smaller matrices (never bigger).

- "attachTo" : the scoring value is stored in the 3D matrix only when a hit occur in the attached volume. If the size of the volume is greater than the 3D matrix, hit occurring out of the matrix are not recorded. Conversely, if the 3D matrix is larger than the attached volume, part which is outside the volume will never record hit (even if it occurs) because hit is performed out of the volume.
- "type" : In Geant4, when a hit occurs, the energy is deposited along a step line. A step is defined by two positions the 'PreStep' and the 'PostStep'. The user can choose at which position the actor have to store the information (edep, dose ...) : it can be at PreStep ('pre'), at PostStep ('post'), at the middle between PreStep and PostStep ('middle') or distributed from PreStep to PostStep ('random'). According to the matrix size, such line can be located inside a single dosel or cross several dosels. Preferred type of hit is "random", meaning that a random position is computed along this step line and all the energy is deposited inside the dosel that contains this point.
- the attached volume can be a voxelized image. The scoring matrix volume (dosels) are thus different from the geometric voxels describing the image:

```
/gate/actor/[Actor Name]/attachTo       waterbox
/gate/actor/[Actor Name]/setSize        5 5 5 cm
/gate/actor/[Actor Name]/voxelsize      10 20 5 mm
/gate/actor/[Actor Name]/setPosition    1 0 0 mm
/gate/actor/[Actor Name]/stepHitType    random
```

- If you would like the dose actor to use exactly the same voxels as the input image, then the safest way to configure this is with *setResolution*. Otherwise, when setting *voxelsize*, rounding errors may cause the dosels to be slightly different, in particular in cases where the voxel size is not a nice round number (e.g. 1.03516 mm on a dimension with 512 voxels). Such undesired rounding effects have been observed Gate release 7.2 and may be fixed in a later release.

2.8.2 List of available Actors

Simulation statistic

This actor counts the number of steps, tracks, events, runs in the simulation. If the actor is attached to a volume, the actor counts the number of steps and tracks in the volume. The output is an ASCII file:

```
/gate/actor/addActor SimulationStatisticActor     MyActor
/gate/actor/MyActor/save                          MyOutput.txt
```

Electromagnetic (EM) properties

This actor allows extracting EM properties for all materials defined in a simulation, as listed below:

- Density (mass density in g/cm^3)
- e-density (electronic density in e-/mm^3)
- RadLength (radiation length in mm)
- I (ionization potential in eV)
- EM-DEDX (EM mass stopping power in MeV.cm^2/g)
- Nucl-DEDX (nuclear mass stopping power in MeV.cm^2/g)
- Tot-DEDX (total mass stopping power in MeV.cm^2/g)

EM properties are calculated relative to a specific particle type and energy, as defined by the user. For instance, EM properties corresponding to a 30 MeV neutron can be calculated using the following command lines:

```
/gate/actor/addActor EmCalculatorActor              MyActor
/gate/actor/MyActor/setParticleName                 proton
/gate/actor/MyActor/setEnergy                       150 MeV
/gate/actor/MyActor/save                            MyOutput.txt
```

Dose measurement (DoseActor)

The DoseActor builds 3D images of the energy deposited (edep), dose deposited and the number of hits in a given volume. It takes into account the weight of particles. It can store multiple information into a 3D grid, each information can be enabled by using:

```
/gate/actor/[Actor Name]/enableEdep             true
/gate/actor/[Actor Name]/enableUncertaintyEdep  true
/gate/actor/[Actor Name]/enableSquaredEdep      true
/gate/actor/[Actor Name]/enableDose             true
/gate/actor/[Actor Name]/enableUncertaintyDose  true
/gate/actor/[Actor Name]/enableDose             true
/gate/actor/[Actor Name]/enableUncertaintyDose  true
/gate/actor/[Actor Name]/enableSquaredDose      true
/gate/actor/[Actor Name]/enableNumberOfHits     true
```

Informations can be disable by using “false” instead of “true” (by default all states are false):

```
/gate/actor/[Actor Name]/enableEdep             false
```

The unit of edep is MeV and the unit of dose is Gy. The dose/edep squared is used to calculate the uncertainty when the output from several files are added. The uncertainty is the relative statistical uncertainty. “SquaredDose” flag allows to store the sum of squared dose (or energy). It is very useful when using GATE on several workstations with numerous jobs. To compute the final uncertainty, you only have to sum the dose map and the squared dose map to estimate the final uncertainty according to the uncertainty equations.

It is possible to normalize the maximum dose value to 1:

```
/gate/actor/[Actor Name]/normaliseDoseToMax   true
```

For normalization purposes, further commands are also available:

```
/gate/actor/[Actor Name]/normaliseDoseToWater   true
```

or:

```
/gate/actor/[Actor Name]/normaliseDoseToIntegral   true
```

For the output, the suffixes Edep, Dose, NbOfHits, Edep-Uncertainty, Dose-Uncertainty, Edep-Squared or Dose-Squared are added to the output file name given by the user. You can use several files types: ASCII file (.txt), root file (.root), Analyze (.hdr/.img) and MetaImage (.mhd/.raw) (mhd is recommended !). The root file works only for 1D and 2D distributions:

```
/gate/actor/addActor DoseActor              MyActor
/gate/actor/MyActor/save                    MyOutputFile.mhd
/gate/actor/MyActor/attachTo                MyVolume
/gate/actor/MyActor/stepHitType             random
/gate/actor/MyActor/setSize                 5 5 5 m
/gate/actor/MyActor/setResolution           1 1 3000
/gate/actor/MyActor/enableEdep              true
```

(continues on next page)

(continued from previous page)

```
/gate/actor/MyActor/enableUncertaintyEdep  true
/gate/actor/MyActor/enableSquaredEdep      false
/gate/actor/MyActor/enableDose             false
/gate/actor/MyActor/normaliseDoseToMax     false
```

Water equivalent doses (or dose to water) can be also calculated, in order to estimate doses calculated using water equivalent path length approximations, such as in Treatment Planning Systems (TPS). Command previously presented for the "dose" also work for the "dose to water" as shown below:

```
/gate/actor/[Actor Name]/enableDoseToWater                  true
/gate/actor/[Actor Name]/enableUncertaintyDoseToWater       true
/gate/actor/[Actor Name]/normaliseDoseToWater               true
```

New image format : MHD

Gate now can read and write mhd/raw image file format. This format is similar to the previous hdr/img one but should solve a number of issues. To use it, just specify .mhd as extension instead of .hdr. The principal difference is that mhd store the 'origin' of the image, which is the coordinate of the (0,0,0) pixel expressed in the *physical world* coordinate system (in general in millimetres). Typically, if you get a DICOM image and convert it into mhd (vv can conveniently do this), the mhd will keep the same pixels coordinate system than the DICOM.

In GATE, if you specify the macro "TranslateTheImageAtThisIsoCenter" with the coordinate of the isocenter that is in a DICOM-RT-plan file, the image will be placed such that this isocenter is at position (0,0,0) of the mother volume (often the world). This is very useful to precisely position the image as indicated in a RT plan. Also, when using a DoseActor attached to a mhd file, the output dose distribution can be stored in mhd format. In this case, the origin of the dose distribution will be set such that it corresponds to the attached image (easy superimposition display).

Note however, that the mhd module is still experimental and not complete. It is thus possible that some mhd images cannot be read. Use and enjoy at your own risk, please contact us if you find bugs and be warmly acknowledged if you correct bugs.

Dose by regions

The dose actor can also calculate dose and energy deposited in regions defined by a set of voxels and outputs the result in a text file. These regions are read from a .mhd image file containing labels (integers) which must be of the same size as the dose actor. Each label in the image defines a region where all energies will be summed and the dose calculated during the simulation. A region must contain voxels of the same material for the dose calculation to be correct. This output allows to get the statistical uncertainties for a set of voxels.

To activate this output:

```
/gate/actor/[Actor Name]/inputDoseByRegions     data/regionImage.mhd
/gate/actor/[Actor Name]/outputDoseByRegions    output/DoseByRegions.txt
```

It is possible to define additional regions composed of original regions (of the same material) by specifying a new region label followed by a colon and the list of original region labels:

```
/gate/actor/[Actor Name]/addRegion              1000: 89, 90, 91
/gate/actor/[Actor Name]/addRegion              1001: 92, 93, 94
```

The output ascii file contains one line per region with the following information:

```
#id  vol(mm3)        edep(MeV)       std_edep        sq_edep         dose(Gy)
↪std_dose        sq_dose         n_hits  n_event_hits
0    158092650.2908  13.08421506078  0.053474625991  0.489560086787  1.10061390e-11
↪0.053474625991  3.46402200e-25  40288   814
```

(continues on next page)

(continued from previous page)

An example can be found in the GateContrib GitHub repository under dosimetry/doseByRegions.

Dose calculation algorithms

When storing a dose (D=edep/mass) with the DoseActor, mass is computed by using the material density at the step location and using the volume the dosel. If the size of the image voxel is smaller than the size of the dosel of the DoseActor it can lead to undesired results. Two algorithms are available for the DoseActor.

Volume weighting algorithm

This algorithm is used by default. The absorbed dose of each material is ponderated by the fraction of materials volume:

```
/gate/actor/[Actor Name]/setDoseAlgorithm VolumeWeighting
```

Mass weighting algorithm

This algorithm calculates the dose of each dosels by taking the deposited energy and dividing it by its mass:

```
/gate/actor/[Actor Name]/setDoseAlgorithm MassWeighting
```

Mass image :

Mass images (.txt, .root, .mhd) can be imported and exported to be used by the mass weighting algorithm.

- Exportation:

  ```
  /gate/actor/[Actor Name]/exportMassImage path/to/MassImage
  ```

- Importation:

  ```
  /gate/actor/[Actor Name]/importMassImage path/to/MassImage
  ```

- The unit of mass images is kg.
- When the mass weighting algorithm is used on a unvoxelized volume, depending on the dosel's resolution of the DoseActor the computation can take a very long time.
- **Important note :** If no mass image is imported when using the mass weighting algorithm Gate will calculate the mass during the simulation (this can take a lot of time).

The command 'exportMassImage' can be used to generate the mass image of the DoseActor attached volume one time for all and import it with the 'importMassFile' command.

Limitations :

- **With voxelized phantom :**
  - MassWeighting algorithm work with phantoms imported with *ImageRegularParametrisedVolume* and *ImageNestedParametrisedVolume*.
  - For now it's not possible to choose an actor resolution smaller than the phantom's resolution.
  - It is mandatory to attach directly the phantom to the actor.

- **With unvoxelized geometry :** The dosels resolution must be reasonably low otherwise the time of calculation can be excessively long ! (and can need a lot of memory !)

Tet-Mesh Dose Actor

The **TetMeshDoseActor** can only be attached to 'TetMeshBox' volumes. It scores dose for each tetrahedron of the tetrahedral mesh contained in the TetMeshBox. Example usage:

```
/gate/actor/addActor                TetMeshDoseActor doseSensor
/gate/actor/doseSensor/attachTo     meshPhantom
/gate/actor/doseSensor/save         output/phantom_dose.csv
```

The output of the TetMeshDoseActor is a csv-file tabulating the results, e.g.:

```
# Tetrahedron-ID, Dose [Gy], Relative Uncertainty, Sum of Squared Dose [Gy^2], Volume␣
↪[cm^3], Density [g / cm^3], Region Marker
0, 1.33e-08, 1.30e-01, 3.03e-18, 1.94e-02, 9.49e-01, 1
1, 1.96e-09, 9.99e-01, 3.86e-18, 1.13e-04, 9.49e-01, 1
...
```

Each row corresponds to one tetrahedron. The region marker column identifies to which macroscopic structure a tetrahedron belongs to – it is equal to the region attribute defined for this tetrahedron in the '.ele' file the TetMeshBox is constructed from.

Kill track

This actor kills tracks entering the volume. The output is the number of tracks killed. It is stored an ASCII file:

```
/gate/actor/addActor KillActor       MyActor
/gate/actor/MyActor/save             MyOutputFile.txt
/gate/actor/MyActor/attachTo         MyVolume
```

Stop on script

This actor gets the output of a script and stop the simulation if this output is true:

```
/gate/actor/addActor  StopOnScriptActor     MyActor
/gate/actor/MyActor/save                    MyScript
```

It is possible to save all the other actors before stopping the simulation with the command:

```
/gate/actor/MyActor/saveAllActors           true
```

Track length

This actor stores the length of each tracks in a root file. It takes into account the weight of particles. They are three commands to define the boundaries and the binning of the histogram:

```
/gate/actor/addActor  TrackLengthActor      MyActor
/gate/actor/MyActor/save                    MyOutputFile.root
/gate/actor/MyActor/setLmin                 0 mm
/gate/actor/MyActor/setLmax                 1 cm
/gate/actor/MyActor/setNumberOfBins         200
```

Energy spectrum

This actor builds one file containing N histograms. By default 3 histograms are enabled: The fluence and energy deposition spectra differential in energy and the energy deposition spectrum as a function of LET. Ideally one specifies the lower (Emin) and upper (Emax) boundary of the histogram and the resolution/number of bins:

```
/gate/actor/addActor  EnergySpectrumActor                MyActor
/gate/actor/MyActor/save                                 MyOutputFile.root
/gate/actor/MyActor/energySpectrum/setEmin               0 eV
/gate/actor/MyActor/energySpectrum/setEmax               200 MeV
/gate/actor/MyActor/energySpectrum/setNumberOfBins       2000

/gate/actor/MyActor/enableLETSpectrum                                    true
/gate/actor/MyActor/LETSpectrum/setLETmin                          0 keV/um
/gate/actor/MyActor/LETSpectrum/setLETmax                          100 keV/um
/gate/actor/MyActor/LETSpectrum/setNumberOfBins                      1000

/gate/actor/MyActor/energyLossHisto/setEdepMin               0.0001 keV
/gate/actor/MyActor/energyLossHisto/setEdepMax               200 keV
/gate/actor/MyActor/energyLossHisto/setNumberOfEdepBins       1000
```

By default an equidistant bin width is applied. However, a logarithmic bin width may be enabled:

```
/gate/actor/MyActor/setLogBinWidth                   true
```

In that case the lower boundary of the histogram should not be 0. If 0 is specified as lower boundary, it is replaced with a $\epsilon > 0$ internally.

To normalize the 1D histograms to the number of simulated primary events enable:

```
/gate/actor/MyActor/normalizeToNbPrimaryEvents                   true
```

To score the energy relative to unit particle mass [MeV/u] instead of total energy [MeV] enable:

```
/gate/actor/MyActor/setEnergyPerUnitMass                   true
```

The number of particles entering a volume differential in energy: (this is not fluence):

```
/gate/actor/MyActor/enableNbPartSpectrum                     true
```

The fluence differential in energy corrected by 1/cos(ϕ) with ϕ being the angle of the particle entering a volume. This works only for planes perpendicular to the z direction. No correction for cos(ϕ) = 0 is applied. Only particles entering the volume are scored:

```
/gate/actor/MyActor/enableFluenceCosSpectrum                   true
```

The fluence differential in energy summing up the track length of the particle. The outcome of this vector needs to be divided by the volume of the geometry the actor was attached to:

```
/gate/actor/MyActor/enableFluenceTrackSpectrum                 true
```

The energy deposition differential in energy is scored using GetTotalEnergyDeposit():

```
/gate/actor/MyActor/enableEdepSpectrum                 true
```

the energy deposition per event ('edepHisto'), the energy deposition per track ('edepTrackHisto') and the energy loss per track ('eLossHisto') and the energy deposition per step ('edepStepHisto'). These histograms are stored in a root file. They take into account the weight of particles:

```
/gate/actor/MyActor/enableEdepHisto          true
/gate/actor/MyActor/enableEdepTimeHisto              true
/gate/actor/MyActor/enableEdepTrackHisto             true
/gate/actor/MyActor/enableEdepStepHisto              true
/gate/actor/MyActor/enableElossHisto         true
/gate/actor/MyActor/energyLossHisto/setEmin              0 eV
/gate/actor/MyActor/energyLossHisto/setEmax              15 MeV
/gate/actor/MyActor/energyLossHisto/setNumberOfBins      120
```

To score the energy deposition differential in $Q = charge^2/E_{kin}$:

```
/gate/actor/MyActor/enableQSpectrum                                  true
/gate/actor/MyActor/QSpectrum/setQmin                                0 keV/um
/gate/actor/MyActor/QSpectrum/setQmax                                100 keV/um
/gate/actor/MyActor/QSpectrum/setNumberOfBins                        1000
```

By default histograms are saved as .root files. The histograms will be (in addition) converted to ASCII format files by enabling:

```
/gate/actor/MyActor/saveAsText                              true
```

Production and stopping particle position

This actor stores in a 3D image the position where particles are produced and where particles are stopped. For the output, the suffixes 'Prod' and 'Stop' are added to the output file name given by the user. You can use several files types: ASCII file (.txt), root file (.root), (.mhd/.raw or .hdr/.img). The root file works only for 1D and 2D distribution:

```
/gate/actor/addActor ProductionAndStoppingActor      MyActor
/gate/actor/MyActor/save                             MyOutputFile.mhd
/gate/actor/MyActor/attachTo                         MyVolume
/gate/actor/MyActor/setResolution                    10 10 100
/gate/actor/MyActor/stepHitType                      post
```

< ! > In Geant4, secondary production occurs at the end of the step, the recommended state for 'stepHitType' is 'post'

The "prod" output contains the 3D distribution of the location where particles are created (their first step), and the "stop" contains the 3D distribution of the location where particles stop (end of track). Each voxel of both images thus contains the number of particles that was produced (resp. stopped) in this voxel. Source code is: https://github.com/OpenGATE/Gate/blob/develop/source/digits_hits/src/GateProductionAndStoppingActor.cc

Secondary production

This actor creates a root file and stores the number of secondaries in function of the particle type. Ionisation electrons are dissociated from electrons produced by other processes. Decay positrons are dissociated from positrons produced by other processes. Gammas are classified in four categories: gammas produced by EM processes, gammas produced by hadronic processes, gammas produced by decay processes and other gammas:

```
/gate/actor/addActor  SecondaryProductionActor     MyActor
/gate/actor/MyActor/save                           MyOutputFile.root
/gate/actor/MyActor/attachTo                       MyVolume
```

Delta kinetic energy

This actor sums the relative and absolute Δ (kinetic energy) and stores the results in two files (with suffixes "-RelStopPower" and "-StopPower"). It also stores the map of the hits to allow users to calculate the mean values:

```
/gate/actor/addActor    StoppingPowerActor       MyActor
/gate/actor/MyActor/save                         MyOutputFile.hdr
/gate/actor/MyActor/attachTo                     MyVolume
/gate/actor/MyActor/setResolution                10 10 100
/gate/actor/MyActor/stepHitType                  random
```

Number of particles entering volume

This actor builds a map of the number of particules produced outside of the actor volume and interacting in the volume. The particle is recorded once in each voxel where it interacting:

```
/gate/actor/addActor    ParticleInVolumeActor       MyActor
/gate/actor/MyActor/save                            MyOutputFile.hdr
/gate/actor/MyActor/attachTo                        MyVolume
/gate/actor/MyActor/setResolution                   10 10 100
/gate/actor/MyActor/stepHitType                     post
```

Q-value

This actor calculates the Q-values of interactions:

```
/gate/actor/addActor     QvalueActor         MyActor
/gate/actor/MyActor/save                     MyOutputFile.hdr
/gate/actor/MyActor/attachTo                 MyVolume
/gate/actor/MyActor/setResolution            10 10 100
/gate/actor/MyActor/stepHitType              random
```

CrossSectionProductionActor

The CrossSectionProductionActor derives the production of C-11 or O-15 from the equation proposed by (Parodi et al, 2007). The cross section data are provided directly in the class code. By default, only the production of the C-11 is activated.

WARNING: The size of the image has to be given in mm

The current limit in cross section data is 199 MeV. Other data can be added in the class:

```
/gate/actor/addActor                CrossSectionProductionActor beta
/gate/actor/beta/attachTo           volume
/gate/actor/beta/save               output_dump/test_small.hdr
/gate/actor/beta/addO15             true
/gate/actor/beta/addC11             true
/gate/actor/beta/setVoxelSize       1 1 1 mm
/gate/actor/beta/saveEveryNEvents   100000
```

WashOutActor

The bilogical washout follows the Mizuno model (H. Mizuno et al. Phys. Med. Biol. 48, 2003). The activity distributions of the washout actor associated volume are continuously modified as a function of the acquisition time in terms of the following equation :

$Cwashout(t) = Mf.exp(-t/Tf.ln2) + Mm.exp(-t/Tm.ln2) + Ms.exp(-t/Ts.ln2)$

Where 3 components are defined (fast, medium and slow) with two parameters for each : the half life T and the fraction M (Mf + Mm + Ms = 1).

Users should provide a table as an ASCII file with the washout parameters values for any radioactive source in the associated volume. In order to take into account the physiological properties of each tissue, it is important to highlight that one independent radioactive source should be defined per each material involved in the simulation:

```
/gate/actor/addActor                                  WashOutActor [ACTOR NAME]
/gate/actor/[ACTOR NAME]/attachTo                          [VOLUME NAME]
/gate/actor/[ACTOR NAME]/readTable                 [TABLE FILE NAME]
```

Example of [TABLE FILE NAME]: How to specify different parameters which are associated to the washout model - This ASCII file will be used by the washout Actor:

```
2
[SOURCE 1 NAME]   [MATERIAL 1 NAME]     [Mf VALUE]  [Tf VALUE IN SEC]   [Mm VALUE]
↪[Tm VALUE IN SEC]   [Ms VALUE]  [Ts VALUE IN SEC]
[SOURCE 2 NAME]   [MATERIAL 2 NAME]     [Mf VALUE]  [Tf VALUE IN SEC]   [Mm VALUE]
↪[Tm VALUE IN SEC]   [Ms VALUE]  [Ts VALUE IN SEC]
...
...
```

Fluence Actor (particle counting)

This actor counts the number of time a (new) particle is passing through a volume; output as an image:

```
/gate/actor/addActor FluenceActor      Detector
/gate/actor/Detector/save              output/detector.mhd
/gate/actor/Detector/attachTo          DetectorPlane
/gate/actor/Detector/stepHitType       pre
/gate/actor/Detector/setSize           10 410 410 mm
/gate/actor/Detector/setResolution     1 256 256
/gate/actor/Detector/enableScatter     true
```

TLE and seTLE (Track Length Estimator)

TLE is the Track Length Estimator method initially proposed by [Williamson1997] allowing very fast dose computation for low energy photon beam (about below 1 MeV). About 1000x faster than analog Monte-Carlo. The second method, seTLE for split-exponential TLE, was proposed in [Smekens2014] and is about 15x faster than TLE.

- Williamson J F 1987 Monte Carlo evaluation of kerma at a point for photon transport problems Med. Phys. 14 567–76
- F. Smekens, J. M. Létang, C. Noblet, S. Chiavassa, G. Delpon, N. Freud, S. Rit, and D. Sarrut, “Split exponential track length estimator for Monte-Carlo simulations of small-animal radiation therapy”, Physics in medicine and biology, vol. 59, issue 24, pp. 7703-7715, 2014 pdf

- F. Baldacci, A. Mittone, A. Bravin, P. Coan, F. Delaire, C. Ferrero, S. Gasilov, J. M. Létang, D. Sarrut, F. Smekens, et al., "A track length estimator method for dose calculations in low-energy x-ray irradiations: implementation, properties and performance", Zeitschrift Fur Medizinische Physik, 2014.
- A. Mittone, F. Baldacci, A. Bravin, E. Brun, F. Delaire, C. Ferrero, S. Gasilov, N. Freud, J. M. Létang, D. Sarrut, et al., "An efficient numerical tool for dose deposition prediction applied to synchrotron medical imaging and radiation therapy.", Journal of synchrotron radiation, vol. 20, issue Pt 5, pp. 785-92, 2013

Usage is very simple just replace the DoseActor by TLEDoseActor. See examples/example_Radiotherapy/example10 in the Gate source code:

```
/gate/actor/addActor                    TLEDoseActor  tle
/gate/actor/tle/attachTo              phantom
/gate/actor/tle/stepHitType             random
/gate/actor/tle/setVoxelSize            2 2 2 mm
/gate/actor/tle/enableDose              true
/gate/actor/tle/save                    output/dose-tle.mhd
```

or:

```
/gate/actor/addActor                                SETLEDoseActor setle
/gate/actor/setle/attachTo                          phantom
/gate/actor/setle/setVoxelSize                      2 2 2 mm
/gate/actor/setle/enableHybridino                   true
/gate/actor/setle/setPrimaryMultiplicity            200
/gate/actor/setle/setSecondaryMultiplicity          400
/gate/actor/setle/enableDose                        true
/gate/actor/setle/save                              output/dose-setle.mhd
```

A detailed documentation is available here: http://midas3.kitware.com/midas/download/item/316877/seTLE.pdf

Fixed Forced Detection CT

This actor is a *Variance Reduction Technique* for the simulation of CT.

The fixed forced detection technique (Colijn & Beekman 2004, Freud et al. 2005, Poludniowski et al. 2009) relies on the deterministic computation of the probability of the scattered photons to be aimed at each pixel of the detector. The image of scattered photons is obtained from the sum of these probabilities.

The probability of each scattering point to contribute to the center of the jth pixel is the product of two terms:

- the probability of the photon to be scattered in the direction of the pixel
- the probability of the scattered photon to reach the detector and to be detected

Fixed Forced Detection summary

1) Deterministic simulation of the primary (DRR)
2) Low statistics Monte Carlo simulation Compute scattering points
3) Fixed forced detection (deterministic)

Inputs:

```
/gate/actor/addActor    FixedForcedDetectionActor        MyActor
/gate/actor/MyActor/attachTo                             world
/gate/actor/MyActor/setDetector                          DetectorPlane
/gate/actor/MyActor/setDetectorResolution                128 128
/gate/actor/MyActor/responseDetectorFilename             responseDetector.txt
```

The detector response δ(E) is modeled with a continuous energy-function that describes the average measured signal for a given incident energy E. The output signal in each image depends on the detector response (parameter responseDetectorFilename). For examples, if δ(E)=1, then the output signal is the number of photons, and if δ(E)=E (as responseDetector.txt in the github example), then the output signal is the total energy of photons.

One can separate compton, rayleigh and fluorescence photons, secondary (compton+rayleigh+fluorescence), primary or total (secondary+primary). flatfield is available to compute the measured primary signal if there is no object, which is useful for CT to apply the Beer Lambert law. The attenuation is ln(flatfield/primary) to get the line integral, i.e., the input of most CT reconstruction algorithms. To include the secondary signal (compton+rayleigh+fluorescence) in the attenuation, one can use the images saved by the actor to recompute the attenuation (for example using ITK in Python). The formula for the attenuation would be ln(flatfield / (primary+secondary)).

- **attachTo** Attaches the sensor to the given volume
- **saveEveryNEvents** Save sensor every n Events.
- **saveEveryNSeconds** Save sensor every n seconds.
- **addFilter** Add a new filter
- **setDetector** Set the name of the volume used for detector (must be a Box).
- **setDetectorResolution** Set the resolution of the detector (2D).
- **geometryFilename** Set the file name for the output RTK geometry filename corresponding to primary projections.
- **primaryFilename** Set the file name for the primary x-rays (printf format with runId as a single parameter).
- **materialMuFilename** Set the file name for the attenuation lookup table. Two paramaters: material index and energy.
- **attenuationFilename** Set the file name for the attenuation image (printf format with runId as a single parameter).
- **responseDetectorFilename** Input response detector curve.
- **flatFieldFilename** Set the file name for the flat field image (printf format with runId as a single parameter).
- **comptonFilename** Set the file name for the Compton image (printf format with runId as a single parameter).
- **rayleighFilename** Set the file name for the Rayleigh image (printf format with runId as a single parameter).
- **fluorescenceFilename** Set the file name for the fluorescence image (printf format with runId as a single parameter).
- **secondaryFilename** Set the file name for the scatter image (printf format with runId as a single parameter).
- **enableSquaredSecondary** Enable squared secondary computation
- **enableUncertaintySecondary** Enable uncertainty secondary computation
- **totalFilename** Set the file name for the total (primary + scatter) image (printf format with runId as a single parameter).
- **phaseSpaceFilename** Set the file name for storing all interactions in a phase space file in root format.
- **setInputRTKGeometryFilename** Set filename for using an RTK geometry file as input geometry.
- **noisePrimaryNumber** Set a number of primary for noise estimate in a phase space file in root format.
- **energyResolvedBinSize** Set energy bin size for having an energy resolved output. Default is 0, i.e., off.

An example is available at example_CT/fixedForcedDetectionCT.

The GateHybridForcedDetectionActor works for:

- One voxelized (CT) volume, the rest must be of the same material as the world → No volume between voxelized volume and detector.
- Point sources (plane distribution focused).
- A given detector description.
- With some additional geometric limitations.

The FFD implementation in Gate is based on the Reconstruction Toolkit. The deterministic part, the ray casting, is multi-threaded. One can control the number of threads by setting the environment variable ITK_GLOBAL_DEFAULT_NUMBER_OF_THREADS. If it is not set, the default is to have a many threads as cores in the machine.

Fixed Forced Detection CT with Fresnel phase contrast

Provided that you also compile Gate with GATE_USE_XRAYLIB ON (in addition to RTK), i.e., that you activate the dependency to the xraylib, you can also account for the change of phase in the x-ray wave in the computation of primary images with the following options:

- **materialDeltaFilename** Set the output file name for the refractive index decrement lookup table. Two parameters: material index and energy.
- **fresnelFilename** Set the output file name for the Fresnel diffraction image (printf format with runId as a single parameter).

The output in fresnelFilename is computed following equation (2) of Weber et al, Journal of Microscopy, 2018.

An example is available at GateContrib: Fresnel_FFD.

Fixed Forced Detection SPECT

This actor is a *Variance Reduction Technique* for the simulation of SPECT.

The fixed forced detection technique (Colijn & Beekman 2004, Freud et al. 2005, Poludniowski et al. 2009) relies on the deterministic computation of the probability of the scattered photons to be aimed at each pixel of the detector. The image of scattered photons is obtained from the sum of these probabilities.

The probability of each scattering point to contribute to the center of the jth pixel is the product of two terms:

- the probability of the photon to be scattered in the direction of the pixel
- the probability of the scattered photon to reach the detector and to be detected

Inputs:

```
/gate/actor/addActor    FixedForcedDetectionActor        MyActor
/gate/actor/MyActor/attachTo                             world
/gate/actor/MyActor/setDetector                          DetectorPlane
/gate/actor/MyActor/setDetectorResolution                128 128
/gate/actor/MyActor/setSourceType                        isotropic
/gate/actor/MyActor/generatePhotons                      true
```

or:

```
/gate/actor/MyActor/connectARF                                true
```

- **attachTo** Attaches the sensor to the given volume
- **setDetector** Set the name of the volume used for detector (must be a Box).

- **setDetectorResolution** Set the resolution of the detector (2D).
- **generatePhotons** Generates weighted photons outside of the volume directed at each pixel of the detector.
- **connectARF** Connects the output of the FFD to ARF tables (see *Angular Response Functions to speed-up planar or SPECT simulations*).

An example is available at GateContrib: SPECT_FFD

The GateHybridForcedDetectionActor works for:

- One voxelized (CT) volume, the rest must be of the same material as the world → No volume between voxelized volume and detector.
- With some additional geometric limitations.

PromptGammaTLEActor

This actor is used to investigate prompt gamma production in proton therapy simulations. It provides a speedup factor of around 1000 compared to analog MC. vpgTLE is broken up into three parts. Stage 0 is required to be run once, and each vpgTLE simulation is then broken up into Stage 1 and Stage 2. For each stage, you can find and example in the *examples/vpgTLE* directory.

To understand the background, physics and mathematics of this example, refer to *Accelerated Prompt Gamma estimation for clinical Proton Therapy simulations* by B.F.B. Huisman.

LET Actor

This actor calculates the dose or track averaged linear energy transfer:

```
/gate/actor/addActor    LETActor       MyActor
/gate/actor/MyActor/save               output/myLETactor.mhd
/gate/actor/MyActor/attachTo           phantom
/gate/actor/MyActor/setResolution      1 1 100
/gate/actor/MyActor/setType            DoseAveraged
```

Options: DoseAveraged (default) or TrackAveraged. Both calculation methods use the Geant4 EMCalculator method "GetElectronicStoppingPowerDEDX".

For splitting the simulation into sevaral sub-simulations (e.g. parallel computation) enable:

```
/gate/actor/yActor/doParallelCalculation true
```

The default value is false. Enabling this option will produce 2 output images for each LET actor and run, a file labeled as '-numerator' and one labeled as '-denominator'. Building the quotient of these two images results in the averaged LET image. Note that the numerator and denominator images have to be summed up before the division. The denominator file equals the dose and fluence if DoseAveraged and TrackAveraged is enabled, respectively, after normalizing by the mass or volume.

By default the unrestricted LET is calculated:

```
/gate/actor/MyActor/setRestricted false
```

If the restricted flag is set to true, the restricted LET is calculated, but also the calculation method changes. Instead of using tabulated stopping powers for the mean kinetic energy of the particle, the stopping power is calculated as the quotient of the deposited energy and the step length (ICRU 85). Be aware of potential artifacts (voxel size, step limiter, e- production cuts etc.) reported in literature for this calculation method. The production cut for electrons defines the energy carried away.

By default, the stopping power of the material at the PreStepPoint is used. Often a conversion to the LET (in particular water) is of interest. To convert the stopping power to another material than present in the volume use:

```
/gate/actor/MyActor/setOtherMaterial G4_WATER
```

It may be of interest to separate the LET into several regions. Using following commands
/gate/actor/MyActor/setLETthresholdMin 10 keV/um /gate/actor/MyActor/setLETthresholdMax 100 keV/um

will only score particles having a LET between 10 and 100 keV/um. In this way the average LETd,t in that region can be extracted. Note, when enabling the doParallelCalculation option also the dose and fluence of particles of particles with a certain LET can be extracted.

ID and particle filters can be used:

```
/gate/actor/MyActor/addFilter                    particleFilter
/gate/actor/MyActor/particleFilter/addParticle   proton
```

See: 'Cortes-Giraldo and Carabe, 2014, A critical study on different Monte Carlo scoring method of dose-average-linear energy transfer maps.'

Tissue Equivalent Proportional Counter Actor

The tissue-equivalent proportional counter (TEPC) is a detector dedicated to the measurement of the lineal energy transfer (LET) distribution in a volume at the micrometric scale.These physical data, depending on the beam quality and the location of the detector in the beam, is mainly used to calculate the biological dose for high LET radiation and to characterize the beam quality for radioprotection issues.

A TEPC is very similar to a classical gas ionization chamber. The major difference relies in the sensible volume, which is spherical and filled with low pressure tissue -equivalent gas instead of air. These characteristics allow the TEPC to mimic the shape and composition of the tiny structures in a cell nucleus (about 1 μm of diameter).

Quick use

The principle of the TEPCactor is the same as the EnergySpectrumActor, except that the frequency of lineal energy is stored instead of the deposited energy. In order to obtain the lineal energy, the deposited energy is divided by the mean chord of the TEPC volume ($\overline{L} = \frac{2}{3}\pi\varnothing_{TEPC}$). This imposes creating a sphere as geometry for the TEPC.

Generic commands – The following commands allow to create, attach and save the result in a ROOT file (and a .txt file, if necessary):

```
/gate/actor/addActor TEPCActor  myTEPC
/gate/actor/myTEPC/attachTo     myDetector
/gate/actor/myTEPC/saveAsText   true
/gate/actor/myTEPC/save         output/myLETspectrum.root
```

Pressure command – The pressure of the tissue-equivalent gas (propane-based material) is used to tune the size of the water equivalent sphere represented by the TEPC detector. In the literature, the density of such materials is generally defined for standard pressure and temperature conditions. Although the user can directly create a low pressure and density gas material in the "data/myGateMaterial.db" file, the following command allows to modify in-line the pressure in the TEPC material if this one is defined for standard pressure and temperature conditions:

```
/gate/actor/myTEPC/setPressure   0.044 bar
```

Output commands – This list of commands makes it possible to change the scale of the LET distribution in order to correctly fit with the expected results. As the lineal energy distribution generally extends on several orders of magnitude, the default option is the logarithmic scale:

```
/gate/actor/myTEPC/setLogscale     true
/gate/actor/myTEPC/setNumberOfBins 150
/gate/actor/myTEPC/setEmin         0.01 keV
/gate/actor/myTEPC/setNOrders      6
```

This could be replaced by a linear scale:

```
/gate/actor/myTEPC/setLogscale      false
/gate/actor/myTEPC/setNumberOfBins  150
/gate/actor/myTEPC/setEmin          0 keV
/gate/actor/myTEPC/setEmax          100 keV
```

The last command allows to normalize the distribution by the number of incident particles:

```
/gate/actor/myTEPC/setNormByEvent   true
```

Example

An example of a TEPC actor use is provided in the example repository under dosimetry/TEPCActor folder. In this example, a TEPC detector is placed at different positions in a water tank and irradiated with a 155 MeV mono-energetic proton beam. This setup was used to validate the results against the TEPC measurements published by Kase et al. 2013. In this comparison, our key point was the optimization of the particle cuts and step limiters. Indeed, the lineal energy distribution at the micrometric scale is highly sensible to these two parameters. The particle cuts must be low enough to simulate any significant contribution in the lineal energy distribution and the step limiters must bec correctly tuned in order to avoid boundary effects on geometry elements, while keeping the global simulation time as low as possible. More information regarding the geometry and the physical parameters that were tested to obtain the final macro files are available in the example repository (TEPCactor.pdf).

Phase Space Actor

Example:

```
/gate/actor/addActor PhaseSpaceActor         MyActor
/gate/actor/MyActor/save                     MyOutputFile.IAEAphsp
/gate/actor/MyActor/attachTo                 MyVolume
/gate/actor/MyActor/enableProductionProcess  false
/gate/actor/MyActor/enableDirection          false
/gate/actor/MyActor/useVolumeFrame
```

This actor records information about particles entering the volume which the actor is attached to. They are two file types for the output: root file (.root) and IAEA file (.IAEAphsp and .IAEAheader). The name of the particle, the kinetic energy, the position along the three axes, the direction along the three axes, the weight are recorded. In a IAEA file, each particle is designated by an integer while the full name of the particle is recorded in the root file. Particles in IAEA files are limited to photons, electrons, positrons, neutrons and protons. The root file has two additional pieces of information: the name of the volume where the particle was produced and the name of the process which produced the particle. It is possible to enable or disable some information in the phase space file:

```
/gate/actor/MyActor/enableEkine              false
/gate/actor/MyActor/enableXPosition          false
/gate/actor/MyActor/enableYPosition          false
/gate/actor/MyActor/enableZPosition          false
/gate/actor/MyActor/enableXDirection         false
/gate/actor/MyActor/enableYDirection         false
/gate/actor/MyActor/enableZDirection         false
/gate/actor/MyActor/enableProductionVolume   false
```

(continues on next page)

(continued from previous page)

```
/gate/actor/MyActor/enableProductionProcess  false
/gate/actor/MyActor/enableParticleName       false
/gate/actor/MyActor/enableWeight             false
/gate/actor/MyActor/enableTrackLength        true
```

By default the frame used for the position and the direction of the particle is the frame of the world. To use the frame of the volume which the actor is attached to, the following command should be used:

```
/gate/actor/source/useVolumeFrame
```

By default, the phase space stores particles entering the volume. To store particles exiting the volume, the following command should be used:

```
/gate/actor/MyActor/storeOutgoingParticles true
```

To store all secondary particles created in the volume, use the command:

```
/gate/actor/MyActor/storeSecondaries true
```

Phase spaces built with all secondaries should not be used as source because some particles could be generated several times.

With ROOT files, to avoid very big files, it is possible to restrict the maximum size of the phase space. If a phase space reachs the maximum size, the files is closed and a new file is created. The new file has the same name and a suffix is added. The suffix is the number of the file. For instance, instead of one file of 10 GB, user may prefer 10 files of 1 GB. The value of the maximum size is not exactly the size of the file (value is the size of the TTree):

```
/gate/actor/MyActor/setMaxFileSize [Value] [Unit (B, kB, MB, GB)]
```

The source of the simulation could be a phase space. Gate read two types of phase space: root files and IAEA phase spaces. Both can be created with Gate. However, Gate could read IAEA phase spaces created with others simulations:

```
/gate/source/addSource  [Source name]  phaseSpace
```

User can add several phase space files. All files should have the same informations about particles. The files are chained:

```
/gate/source/[Source name]/addPhaseSpaceFile [File name 1]
/gate/source/[Source name]/addPhaseSpaceFile [File name 2]
```

If particles in the phase space are defined in the world frame, user has to used the command:

```
/gate/source/[Source name]/setPhaseSpaceInWorldFrame
```

If the particle type is not defined in the phase space file, user have to give the particle name. It is supposed that all particles have the same name:

```
/gate/source/[Source name]/setParticleType [Particle name]
```

If user have several phase space sources, each source have the same intensity. User can also choose to give at each source an intensity proportionnal to the number of particles in the files attach to the source:

```
/gate/source/[Source name]/useNbOfParticleAsIntensity true
```

For each run, if the number of events is higher than the number of particles in file, each particle is used several times with the same initial conditions. However, it is possible to rotate the particle position and direction around the z axis

of the volume (make sure your phase space files have a rotational symmetry). The regular rotation is a rotation with a fixed angle:

$\alpha = \frac{2\pi}{N_{used}}$

where N_{used} is the number of time the particle is used:

```
/gate/source/[Source name]/useRegularSymmetry
```

The random rotation is a rotation with a random angle:

```
/gate/source/[Source name]/useRandomSymmetry
```

By default, all particles in a phase space are used. The particles in the the phase space can be preselected in function of their position in the (x, y) plan. For instance, a particle with a origin far from the collimator aperture is not useful and should be ignored. Particles in a r cm-radius circle are selected. The particles outside the circle are ignored:

```
/gate/source/[Source name]/setRmax [r] [unit]
```

Thermal Actor

This actor records the optical photon deposited energy (photons absorbed by the tissue/material) in the volume which the actor is attached to. It also performs the diffusion of the deposited energy. The output file format is a 3D matrix (voxelised image img/hdr). The Pennes bioheat model is used to describe the diffusion of hear in biological perfused tissues. The Pennes equation is solved analytically via Fourier transformations and convolution theorem. The solution of the diffusion equation is equivalent to convolving the initial conditions (3D energy map) with a Gaussian with a standard deviation $\sigma = \sqrt{2tK_1}$, with t the diffusion time, K_1 the tissue thermal diffusivity. The blood perfusion term appears in the solution via an exponential function:

```
/gate/actor/addActor ThermalActor                MyActor
/gate/actor/MyActor/save                         3DMap.hdr
/gate/actor/MyActor/attachTo                     phantom
/gate/actor/MyActor/stepHitType                  random
/gate/actor/MyActor/setPosition                  0. 0. 0. mm
/gate/actor/MyActor/setVoxelSize                 0.5 0.5 0.5 mm
```

Tissue thermal property:

```
/gate/actor/MyActor/setThermalDiffusivity        0.32 mm2/s
```

Density and heat capacity should just be in the same unit for both blood and tissue. In the following example, the density is in kg/mm3 and the heat capacity in mJ kg-1 C-1:

```
/gate/actor/MyActor/setBloodDensity              1.06E-6
/gate/actor/MyActor/setBloodHeatCapacity         3.6E6
/gate/actor/MyActor/setTissueDensity             1.04E-6
/gate/actor/MyActor/setTissueHeatCapacity        3.65E6
/gate/actor/MyActor/setBloodPerfusionRate        0.004
```

During light illumination of a tissue, the thermal heat produced by the optical photons deposited energy does not accumulate locally in the tissue; it diffuses in biological tissues during illumination. This dynamic effect has been taken into account in the GATE code. The n seconds light illumination simulation is sampled into p time frame 3D images by setting the simulation parameter setNumberOfTimeFrames to p. Each of the p sample images is diffused for a duration of [1, 2, . . . , p-1] x n/p seconds. The final image illustrating the heat distribution in the tissues at the end of the illumination time is obtained by adding all diffused images to the last n/p seconds illumination image. This thermal energy (or heat) map will continue to diffuse after illumination by setting the parameter setDiffusionTime to the value

of interest. At a certain point in time after the initial temperature boost induced by nanoparticles, the temperature of the tissues will go back to its initial value due to diffusion. This boundary condition is taken into account in a post processing-step of the GATE simulation:

```
/gate/actor/MyActor/setNumberOfTimeFrames          5
/gate/actor/MyActor/setDiffusionTime               5 s
```

Merged Volume Actor

Since GATE V8.0, the user has to possibility to add a G4VSolid (or a analytical solid such as: box, cylinder, tessellated, sphere etc...) within a voxellized volume (defined by ImageRegularParametrisedVolume or ImageNestedParametrisedVolume).

To be done, the user needs an actor and MUST declare the volumes in a specific order.

Here is a schematic procedure:

1) Declaring a volume containing the voxellized phantom AND the volume(s) to merge with the voxellized phantom
2) Declaring the voxellized phantom
3) Declaring all the analytical solid to add within the voxellized phantom

Here is a simple example:

```
# THE CONTAINER VOLUME
/gate/world/daughters/name GlobalVol
/gate/world/daughters/insert box
/gate/GlobalVol/geometry/setXLength 90. mm
/gate/GlobalVol/geometry/setYLength 90. mm
/gate/GlobalVol/geometry/setZLength 90. mm
/gate/GlobalVol/placement/setTranslation 0.0 0.0 0.0 mm
/gate/GlobalVol/placement/setRotationAxis 1 0 0
/gate/GlobalVol/placement/setRotationAngle 0 deg
/gate/GlobalVol/setMaterial Air
/gate/GlobalVol/vis/setColor cyan
/gate/GlobalVol/describe

# THE VOXELLIZED PHANTOM
/gate/GlobalVol/daughters/name PhantomTest
/gate/GlobalVol/daughters/insert ImageRegularParametrisedVolume
/gate/PhantomTest/geometry/setImage phantom_test_without_box.h33
/gate/PhantomTest/geometry/setRangeToMaterialFile range.dat
/gate/PhantomTest/placement/setTranslation 0. 0. 0. mm
/gate/PhantomTest/placement/setRotationAxis 1 0 0
/gate/PhantomTest/placement/setRotationAngle 0 deg
/gate/PhantomTest/setSkipEqualMaterials 0
/gate/PhantomTest/describe

# 2 ANALYTICAL VOLUMES TO MERGE WITHIN VOXELLIZED PHANTOM
# FIRST VOLUME
/gate/GlobalVol/daughters/name BoxAir
/gate/GlobalVol/daughters/insert box
/gate/BoxAir/geometry/setXLength 10.0 mm/gate/BoxAir/geometry/setYLength 10.0 mm
/gate/BoxAir/geometry/setZLength 10.0 mm
/gate/BoxAir/placement/setTranslation -30.0 0.0 0.0 mm
/gate/BoxAir/placement/setRotationAxis 1 0 0
```

(continues on next page)

(continued from previous page)

```
/gate/BoxAir/placement/setRotationAngle 0 deg
/gate/BoxAir/setMaterial Air
/gate/BoxAir/vis/setColor cyan
/gate/BoxAir/describe
# SECOND VOLUME
/gate/GlobalVol/daughters/name BoxLung
/gate/GlobalVol/daughters/insert box
/gate/BoxLung/geometry/setXLength 10.0 mm
/gate/BoxLung/geometry/setYLength 10.0 mm
/gate/BoxLung/geometry/setZLength 10.0 mm
/gate/BoxLung/placement/setTranslation -10.0 0.0 0.0 mm
/gate/BoxLung/placement/setRotationAxis 1 0 0
/gate/BoxLung/placement/setRotationAngle 0 deg
/gate/BoxLung/setMaterial Lung
/gate/BoxLung/vis/setColor red
/gate/BoxLung/describe
```

The final step is to declare the actor. This actor MUST be the first actor declared in the GATE macro. This actor is like a navigator and its influence during the simulation is very important. Here is the declaration of the actor associated to the above example:

```
/gate/actor/addActor MergedVolumeActor mergedVol
/gate/actor/mergedVol/attachTo GlobalVol
/gate/actor/mergedVol/volumeToMerge BoxAir,BoxLung
```

For this actor, the order of the declared volume and the declared actor is very important. In the case of dosimetry, the user could add the dosimetry actor (after the MergedVolumeActor) to retrieve the energy deposit in the volume as follows:

```
/gate/actor/addActor DoseActor doseMeasurement
/gate/actor/doseMeasurement/attachTo GlobalVol
/gate/actor/doseMeasurement/save output/merged_volume.mhd
/gate/actor/doseMeasurement/stepHitType random
/gate/actor/doseMeasurement/setPosition 0 0 0 mm
/gate/actor/doseMeasurement/setVoxelSize 0.5 0.5 0.5 mm
/gate/actor/doseMeasurement/setSize 90.5 90.5 90.5 mm
/gate/actor/doseMeasurement/enableEdep true
/gate/actor/doseMeasurement/enableUncertaintyEdep true
/gate/actor/doseMeasurement/enableSquaredEdep true
/gate/actor/doseMeasurement/enableDose false
/gate/actor/doseMeasurement/enableUncertaintyDose false
/gate/actor/doseMeasurement/enableSquaredDose false
/gate/actor/doseMeasurement/enableNumberOfHits true
```

A complete example is provided here.

Proton Nuclear Information Actor

This actor records information on proton nuclear interactions (number and type). The information can be stored in a phase space file, as illustrated in imaging/ProtonRadiography.

MuMapActor

In PET recon, it need MuMap to attenuation correction, people can use MuMapActor to get MuMap and sourceMap. Note: voxel Mu Uint(default) is cm-1:

```
/gate/actor/addActor MuMapActor getMuMap
/gatt/actor/getMuMap/attachTo world
/gate/actor/getMuMap/save myMapFileName.mhd
/gate/actor/getMuMap/setPosition 0 0 0 mm
/gate/actor/getMuMap/setVoxelSize 2 2 2 mm
/gate/actor/getMuMap/setResolution 128 128 100
/gate/actor/getMuMap/setEnergy 511 keV
/gate/actor/getMuMap/setMuUnit 1 1/mm ##assign Mu uint
```

2.8.3 Filters

Filters are used to add selectrion criteria on actors. They are also used with reduction variance techniques. They are filters on particle type, particle ID, energy, direction. . . .

Filter on particle type

With this filter it is possible to select particle with the name [Particle Name]:

```
/gate/actor/[Actor Name]/addFilter                      particleFilter
/gate/actor/[Actor Name]/particleFilter/addParticle     [Particle Name]
```

User can select various particles. It is also possible to select particles which has a parent with the name [Particle Name]:

```
/gate/actor/[Actor Name]/addFilter                          particleFilter
/gate/actor/[Actor Name]/particleFilter/addParentParticle   [Particle Name]
```

For ions, user should use the Geant4 nomenclature (C12[0.0], c11[0.0]. . .). These names are different from those used for physics. To select all ions except alpha, deuton and triton, there is the key word 'GenericIon'.

It is also possible to filter on the atomic number (Z) and the mass number (A):

```
/gate/actor/[Actor Name]/addFilter                      particleFilter
/gate/actor/[Actor Name]/particleFilter/addParticleZ     Z
/gate/actor/[Actor Name]/particleFilter/addParticleA     A
```

with A and Z being integer values.

To address all particles with atomic number Z1 OR atomic number Z2, Z3 . . . :

```
/gate/actor/[Actor Name]/addFilter                      particleFilter
/gate/actor/[Actor Name]/particleFilter/addParticleZ     Z1
/gate/actor/[Actor Name]/particleFilter/addParticleZ     Z2
/gate/actor/[Actor Name]/particleFilter/addParticleZ     Z3
```

Within atomic number the logical connection on multiple entries is OR, whereas the two types of particle filters, atomic and mass number filter, are connected with logical AND.

To filter on the PDG number of a particle:

```
/gate/actor/[Actor Name]/addFilter                      particleFilter
/gate/actor/[Actor Name]/particleFilter/addParticlePDG     PDG
```

Hence, there are 3 possibilities to filter (for example) for protons:

```
/gate/actor/[Actor Name]/addFilter                          particleFilter
/gate/actor/[Actor Name]/particleFilter/addParticleZ        1
/gate/actor/[Actor Name]/particleFilter/addParticleA        1
```

or:

```
/gate/actor/[Actor Name]/addFilter                          particleFilter
/gate/actor/[Actor Name]/particleFilter/addParticle         proton
```

or:

```
/gate/actor/[Actor Name]/addFilter                          particleFilter
/gate/actor/[Actor Name]/particleFilter/addParticlePDG      2212
```

Example: To kill electrons and positrons in the volume MyVolume:

```
/gate/actor/addActor     KillActor                      MyActor
/gate/actor/MyActor/save                                MyOutputFile.txt
/gate/actor/MyActor/attachTo                            MyVolume
/gate/actor/MyActor/addFilter                           particleFilter
/gate/actor/MyActor/particleFilter/addParticle          e-
/gate/actor/MyActor/particleFilter/addParticle          e+
```

Filter on particle ID

In an event, each track has an unique ID. The incident particle has an ID equal to 1. This filter select particles with the ID [Particle ID] or particles which has a parent with the ID [Particle ID]. As for particle filter, user can select many IDs:

```
/gate/actor/[Actor Name]/addFilter                IDFilter
/gate/actor/[Actor Name]/IDFilter/selectID        [Particle ID]

/gate/actor/[Actor Name]/addFilter                      IDFilter
/gate/actor/[Actor Name]/IDFilter/selectParentID        [Particle ID]
```

Example: To kill all particle exept the incident particle in the volume MyVolume (all particles are the children of the incident particle exept the incident particle itself):

```
/gate/actor/addActor    KillActor                     MyActor
/gate/actor/MyActor/save                              MyOutputFile.txt
/gate/actor/MyActor/attachTo                          MyVolume
/gate/actor/MyActor/addFilter                         IDFilter
/gate/actor/MyActor/IDFilter/selectParentID           1
```

You cannot combine ID and particleFilter.

Filter on volume

This actor is especially useful for reduction variance techniques or for selections on daughter volumes.

Example: To kill particles in volume A and B, children of the volume MyVolume:

```
/gate/actor/addActor   KillActor                        MyActor
/gate/actor/MyActor/save                                MyOutputFile.txt
```

(continues on next page)

(continued from previous page)

```
/gate/actor/MyActor/attachTo                                MyVolume
/gate/actor/MyActor/addFilter                               volumeFilter
/gate/actor/MyActor/volumeFilter/addVolume                  A
/gate/actor/MyActor/volumeFilter/addVolume                  B
```

Filter on energy

This filter allows to select particles with a kinetic energy above a threshold Emin and/or below a threshold Emax:

```
/gate/actor/[Actor Name]/addFilter              energyFilter
/gate/actor/[Actor Name]/energyFilter/setEmin   [Value]  [Unit]
/gate/actor/[Actor Name]/energyFilter/setEmax   [Value]  [Unit]
```

Example: To kill particles with an energy above 5 MeV:

```
/gate/actor/addActor   KillActor                        MyActor
/gate/actor/MyActor/save                                MyOutputFile.txt
/gate/actor/MyActor/attachTo                            MyVolume
/gate/actor/MyActor/addFilter                           energyFilter
/gate/actor/MyActor/energyFilter/setEmin                5 MeV
```

Filter on direction

This filter is used to select particle with direction inside a cone centered on the reference axis. The angle between the axis and the edge of the cone is in degree. The axis is defined with the (x,y,z) directions:

```
/gate/actor/[Actor Name]/addFilter                    angleFilter
/gate/actor/[Actor Name]/angleFilter/setAngle         [Value]
/gate/actor/[Actor Name]/angleFilter/setDirection     [x] [y] [z]
```

Example: To kill particles in a cone of 20 degrees around x axis:

```
/gate/actor/addActor    KillActor                       MyActor
/gate/actor/MyActor/save                                MyOutputFile.txt
/gate/actor/MyActor/attachTo                            MyVolume
/gate/actor/MyActor/addFilter                           angleFilter
/gate/actor/MyActor/angleFilter/setAngle                20
/gate/actor/MyActor/angleFilter/setDirection            1 0 0
```

2.9 How to run Gate

Table of Contents

2.9.1 Interactive mode

To start Gate in interactive mode, simply type:

```
$ Gate
```

and the following output (or something similar) will appear on the screen:

```
1  [G4] *************************************************************
2  [G4]  Geant4 version Name: geant4-10-03    (4-December-2015)
3  [G4]                       Copyright : Geant4 Collaboration
4  [G4]                       Reference : NIM A 506 (2003), 250-303
5  [G4]                             WWW : http://cern.ch/geant4
6  [G4] *************************************************************
7  [G4]
8  [Core-0] Initialization of geometry
9  [Core-0] Initialization of physics
10 [Core-0] Initialization of actors
11 [Core-0]
12 [Core-0] ********************************************************************
13 [Core-0]  GATE version name: gate_v8.0
14 [Core-0]                     Copyright : OpenGATE Collaboration
15 [Core-0]                     Reference : Phys. Med. Biol. 49 (2004) 4543-4561
16 [Core-0]                     Reference : Phys. Med. Biol. 56 (2011) 881-901
17 [Core-0]                     WWW : http://www.opengatecollaboration.org
18 [Core-0] ********************************************************************
19 [Core-0]
20 [Core-0] You are using Geant4 version 10.3.0
21 Idle>
```

This output will vary depending on your Gate installation, that is which version of Geant4 software was installed and how it was installed.

Line 2 indicates the version of the Geant4 software in your installation. Lines 8-10 are initialization messages from Geant4. Line 13 indicates the version of the Gate software you are using. Finally, and if everything went right, then Gate outputs the interpreter command prompt (line 21). This means Gate is ready to read commands entered by the user.

If you are not yet familiar with Gate commands, you can get help by typing *ls*:

```
1    Idle> ls
2    Command directory path : /
3     Sub-directories :
4       /control/   UI control commands.
5       /units/   Available units.
6         /process/   Process Table control commands.
7         /gate/   GATE detector control.
```

(continues on next page)

(continued from previous page)

```
8          /particle/   Particle control commands.
9          /geometry/   Geometry control commands.
10         /tracking/   TrackingManager and SteppingManager control commands.
11         /event/   EventManager control commands.
12         /cuts/   Commands for G4VUserPhysicsList.
13         /run/   Run control commands.
14         /random/   Random number status control commands.
15         /material/   Commands for materials
16         /hits/   Sensitive detectors and Hits
17         /vis/   Visualization commands.
18         /digi/   DigitizerModule
19         /gps/   ...Title not available...
Commands :
```

When the *Sub-directories* names end with a \ (slash), it means you can go deeper in that sub-directory. For instance, let's say you want to find out more about how to run macros:

```
1       Idle> ls /control
2       Command directory path : /control/
3
4
5       Guidance :
6       UI control commands.
7
8        Sub-directories :
9          /control/cout/   Control cout/cerr for local thread.
10         /control/matScan/   Material scanner commands.
11       Commands :
12         macroPath * Set macro search pathwith colon-separated list.
13         execute * Execute a macro file.
14         loop * Execute a macro file more than once.
15         foreach * Execute a macro file more than once.
16         suppressAbortion * Suppress the program abortion caused by G4Exception.
17         verbose * Applied command will also be shown on screen.
18         saveHistory * Store command history to a file.
19         stopSavingHistory * Stop saving history file.
20         alias * Set an alias.
21         unalias * Remove an alias.
22         listAlias * List aliases.
23         getEnv * Get a shell environment variable and define it as an alias.
24         getVal * Get the current value of the UI command and define it as an alias.
25         echo * Display the aliased value.
26         shell * Execute a (Unix) SHELL command.
27         manual * Display all of sub-directories and commands.
28         createHTML * Generate HTML files for all of sub-directories and commands.
29         maximumStoredHistory * Set maximum number of stored UI commands.
30         if * Execute a macro file if the expression is true.
31         doif * Execute a macro file if the expression is true.
32         add * Define a new alias as the sum of two values.
33         subtract * Define a new alias as the subtraction of two values.
34         multiply * Define a new alias as the multiplification of two values.
35         divide * Define a new alias as the division of two values.
36         remainder * Define a new alias as the remainder of two values.
```

A * at the end of the *Sub-directories* names means that it is the last level for that subdirectory. In line 13, it is explained that the command **/control/execute** executes a macro file. This command basically reads the macro file and executes the lines as they appear in the file. Suppose, you have a file named *myScanner.mac* that contains all the necessary

commands to run a particular simulation. Then type:

```
1   Idle> /control/execute myScanner.mac
```

to run the macro file. The macro file *myScanner.mac* can contain additional **/control/execute** commands to run other macro files and so on. Gate will read and execute those files in the order in which they appear. Notice that **/control/execute** does not start a simulation (data acquisition), it simply reads the commands and executes them. The command that starts the actual simulation is **/gate/application/startDAQ**, which is usually the last command found in you macro files.

Depending on the level of verbosity that you have specified in your macro, you will see more or less messages about the simulation. If your simulation contains visualization commands, you will see an OpenGL window appear with a beautiful picture of your scanner.

At the end of your simulation, the command line interpreter prompt will appear again. To exit the interpreter, type:

```
Idle> exit
Graphics systems deleted.
Visualization Manager deleting...
```

You can display the help typing this command:

```
$ Gate -h
```

and the following output (or something similar) will appear on the screen:

```
[Core-0] Gate command line help
[Core-0] Usage: Gate [OPTION]... MACRO_FILE
[Core-0]
[Core-0] Mandatory arguments to long options are mandatory for short options too.
[Core-0]   -h, --help             print the help
[Core-0]   -v, --version          print the version
[Core-0]   -a, --param            set alias. format is '[alias1,value1] [alias2,
↪value2] ...'
[Core-0]   --d                    use the DigiMode
[Core-0]   --qt                   use the Qt visualization mode
```

2.9.2 Running GATE in Qt mode

First you need to compile Geant4 with the variable 'GEANT4_USE_QT' setting to 'ON'. You can visualize the position of your system using the Qt mode.

Then you need to type the following command to your console:

```
$ Gate --qt
```

A window will display:

Afterwards you can launch your macro GATE using the 'Session:' section and typing /control/execute benchPET.mac. You will be able to visualize your system in the viewer windows:

In order to use Qt you have to write this line in your GATE macro:

```
/vis/open OGLSQt
```

IMPORTANT!!!: Qt visualization mode is a visualization after the simulation. In fact you could zoom, translate, etc... only at the end of the simulation.

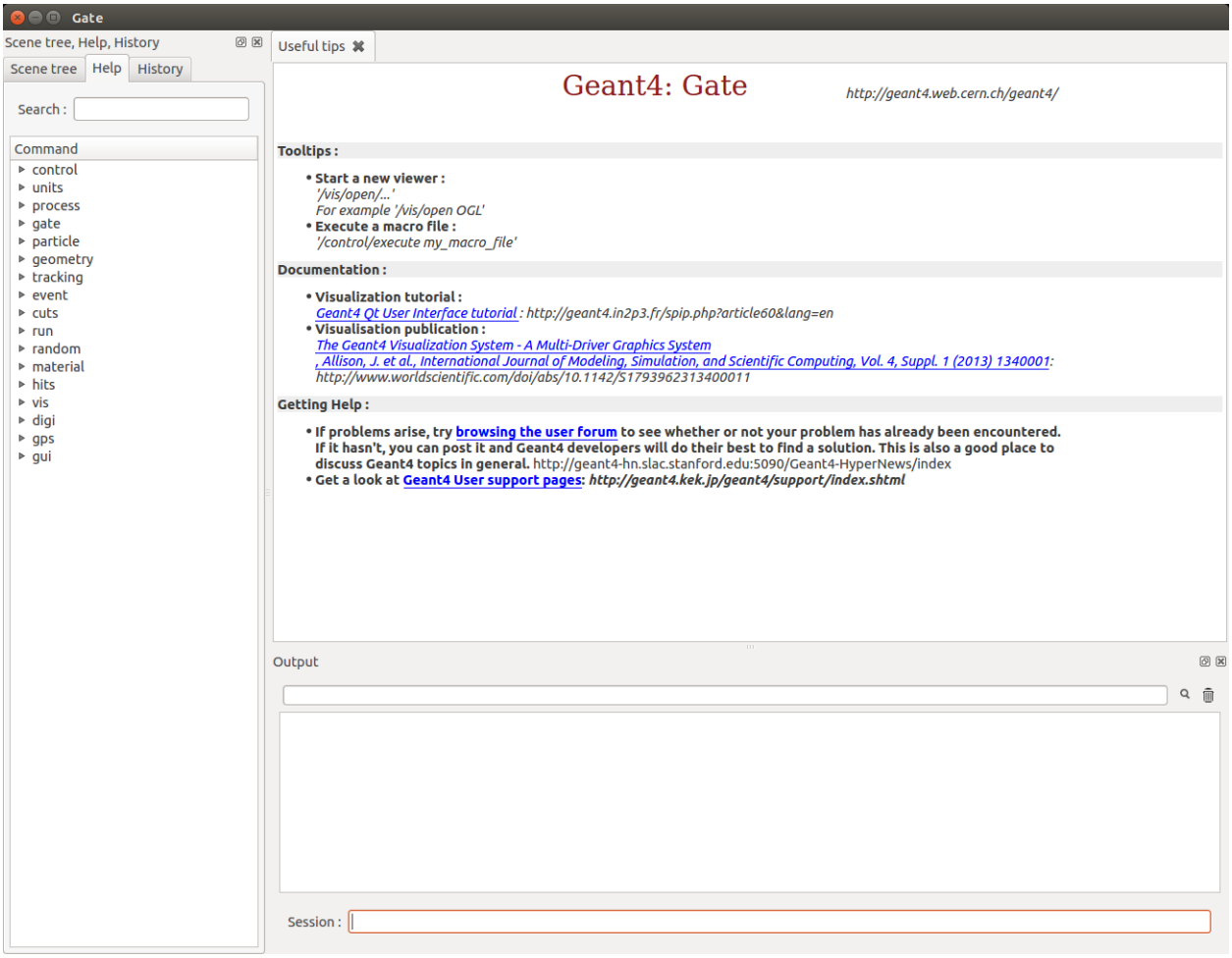


Fig. 2.43: Gate Qt

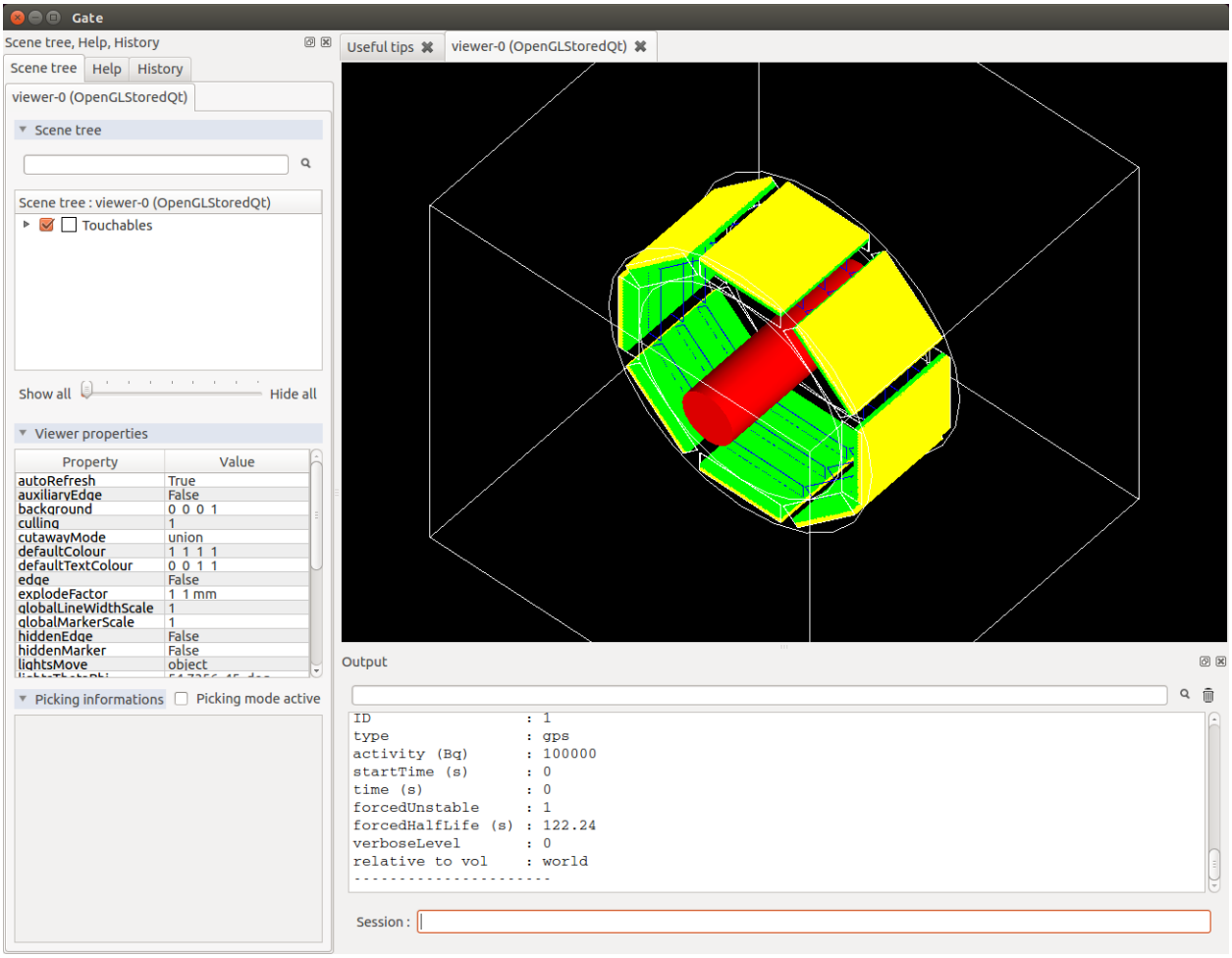


Fig. 2.44: Gate Qt

2.9.3 Running parameterized macros

It is very common for users to run several simulations that differ in only a few parameters. For instance, a user might have designed a small animal PET scanner and would like to estimate its performance for five different crystal materials and three energy windows. In that case, the user does not need to write a complete set of macros for each simulation scenario. Instead, he can write parameterized macros. The actual values of the parameters are specified on the command line when running Gate or they can be defined with the interpreter.

For instance, suppose we want to parameterize the lower and upper level energy discriminators and the length of coincidence window. Here is the corresponding macro command lines:

```
#       D I G I T I Z E R
1    /gate/digitizer/Singles/insert adder
2    /gate/digitizer/Singles/insert readout
3    /gate/digitizer/Singles/readout/setDepth 1
4    /gate/digitizer/Singles/insert blurring
5    /gate/digitizer/Singles/blurring/setResolution 0.26
6    /gate/digitizer/Singles/blurring/setEnergyOfReference 511. keV
7    /gate/digitizer/Singles/insert thresholder
8    /gate/digitizer/Singles/thresholder/setThreshold {lld} keV
9    /gate/digitizer/Singles/insert upholder
10   /gate/digitizer/Singles/upholder/setUphold {uld} keV
#       C O I N C I D E N C E   S O R T E R
11   /gate/digitizer/Coincidences/setWindow {CoincWindow} ns
```

Lines 8, 10, and 11 define aliases for the lower level discriminator, the upper level discriminator, and the length of the coincidence window, respectively.

An alias is always specified between **{** and **}** (curled brackets) and it can consist of any set of characters.

To pass actual values to the macro file, you can run Gate, for instance, as follows:

```
$ Gate -a [CoincWindow,10][lld,350][uld,650]
```

It is worth emphasizing the following points about aliases:

- The order of the aliases at the command line does not matter.
- Aliases are case sensitive, so **[lld,350]** is not the same as **[LLD,350]**.
- All aliases in your macro file(s) must be defined when you run Gate. If some are undefined the simulation will fail.

2.9.4 How to launch *DigiGate*

GATE offers an operating mode dedicated to digitizer optimization, known as *DigiGate* (see *Digitizer and readout parameters*). *DigiGate* works by re-reading a previously generated ROOT hit-file.

The use of **DigiGate** consists of two steps.

- In the first step, the simulation runs according to **MacroTest.mac**. This macro file should save the **Hits** data in the root output file with the name **gate.root** (which is the default name).
- In the second step, the digitizer modifications are made in **MacroTest.mac** (like a new module for the energy resolution, or a different dead-time...), and then the analysis is repeated by using the **gate.root** file as an input file for the program **DigiGate**. This is achieved by launching **Gate** with a '-d' option:

```
Gate < MacroTest.mac
```

-> a root output file is produced with *Hits* information.

-> the digitizer of MacroTest.mac is changed along with the name of the root output file:

```
Gate --d < MacroTest.mac
```

-> a new root output file is produced which incorporates the changes due to a different digitizer without having to repeat the particle generation and its propagation. user can use the following GATE command to read the hit file replaced the name **gate.root**:

```
/gate/hitreader/setFileName FileName
```

2.9.5 How to separate the phantom and detector tracking - Phase space approach

To speed-up the simulation, it is possible to split and separate the particle tracking. This is a phase space approach with the possibility to store the phantom tracking particle history in a root file and to use it as an input file for the detector tracking.

Using Gate in the tracker mode: phantom tracking

Basically, as illustrated in the folder example_TrackerDetector, 3 major command lines are available to use the phantom tracker mode:

```
# Selection of the tracking mode
#
/gate/stepping/SetMode Tracker
#
# Setting of the policy regarding the tracker mode
#
/gate/stepping/SetPolicy Option1
/gate/stepping/SetPolicy Option2
```

The **Option1** variable can be chosen from the following list:

- StopOnPhantomBoundary (Default): the particles are tracked until the last hit before the phantom boundary ;
- StopAfterPhantomBoundary: the particles are traked until the first hit after phantom boundary ;
- KillTrackAndSecondaries: StopOnPhantomBoundary + no secondary production.

The **Option2** variable may be chosen from:

- KeepAll (Default): all particles (primary and secondary) are stored
- KeepOnlyPrimaries: only source particles are stored
- KeepOnlyPhotons: only photons are stored
- KeepOnlyElectrons: only electrons are stored

Two additional command line options are also available:

```
/gate/stepping/SetEnergyThreshold aThreshold keV
```

WIth this option, only particles reaching the phantom boundary with an energy greater than a threshold in energy of **aThreshold** (expressed in keV) will be stored:

```
/gate/stepping/SetTextOutput status
```

with a status flag set as On or Off. This command will print *Tracks* information in the PostStepInfo.txt file.

Finally the tracker mode acquisition will generate N root files named OutPutRoot_TrackerData_number.root. The base output name, which is OutPutRoot in the case of this example, is chosen by the user with the usual output command line to set the file name:

```
/gate/output/root/setFileName OutPutRoot
```

Using Gate in the detector mode: detector tracking

During the tracker mode acquisition, N files are generated with the following name architecture:

- OutPutRoot_TrackerData.root
- OutPutRoot_TrackerData_1.root
- OutPutRoot_TrackerData_2.root
- ...
- OutPutRoot_TrackerData_(N-1).root

To use the Detector Mode, the user must select the mode and specify that N TrackerData files were generated during the tracker mode. All this can be done using the 2 following command lines:

```
/gate/stepping/SetMode Detector
/gate/stepping/SetNumberOfTrackerDataFiles N
```

New commands in detector mode

In Detector Mode, we need to tell GATE that N TrackerData files were generated during tracker mode and we should use these command lines:

```
/gate/stepping/SetMode Detector
/gate/stepping/SetNumberOfTrackerDataFiles N
```

2.9.6 Batch mode

It is possible to run a Gate simulation in *batch* mode, i.e. without the need to enter the interpreter and run the **/control/execute** and **exit** commands each time.

If you want to run simulations in *batch* mode, you can do so by typing the alias values or (–qt parameter) before the file name of the macro you want to run. For example:

```
$ Gate -a [CoincWindow,10][lld,350][uld,650] myScanner.mac
```

Or:

```
$ Gate --qt myScanner.mac
```

In order to return to command prompt, the last line in **myScanner.mac** file must be:

```
exit
```

This is very important, especially when you are running a series of simulations in sequence. If Gate does not find the exit command, it will return to the user interface prompt and the rest of the simulations will not run.

It is recommended to redirect the terminal output of the simulation (listing of physics processes, sources, run time, etc.) by writing it to a text file instead of printing it in the terminal. This allows one to store the terminal output of each simulation for later viewing. For example:

```
$ Gate -a [CoincWindow,10][lld,350][uld,650] myScanner.mac > terminal_output.txt
```

The above command creates a file named terminal_output.txt and does not print to the terminal window.

When running multiple simulations simultaneously from the command line in batch mode, it is often desirable to have the process run in the background. This can be accomplished by inserting an ampersand "&" symbol at the end of the command. For example:

```
$ Gate -a [CoincWindow,10][lld,350][uld,650] myScanner.mac > terminal_output.txt &
```

It is recommended (although not compulsory) to avoid running visualization commands in batch mode.

2.10 Visualization

Table of Contents

2.10.1 Introduction

The visualization options in GATE provide the same functionalities as provided in GEANT4. Most options in GEANT4 to visualize detector geometry, particle trajectories, tracking steps, etc. are also available in GATE. The graphics systems that can be selected in GATE are: DAWNFILE, VRMLFILE (versions 1 and 2) and OpenGL in stored and immediate mode, with OpenGL required as an external library. Most of the libraries are freely available.

2.10.2 Important Hints

When loading digital images using OpenGL, the OpenGL Immediate-X viewer is recommended instead of the frequently used Stored-X viewer.

Using DAWN and VRMLVIEW, complicated geometries, like a huge number of crystals in a cylindrical PET system, may take very long to get rendered. To decrease the file size and to speed up the visualization, the following option may be used:

```
/gate/crystal/vis/setVisible 0
```

Using that option, the individual crystals are not rendered but they are shown as a wireframe instead.

2.10.3 Command Lines

Basic commands provided by the GEANT4-package can be used to set basic vizualisation options, as shown below.

Visualization with OpenGL

The best method is to let Geant4 choose the best OpenGL mode with **/vis/open OGL** and not force a mode which can be present or not on user system.

- All OpenGL commands are available here:

http://geant4.slac.stanford.edu/Presentations/vis/G4OpenGLTutorial/G4OpenGLTutorial.html

For example, if you wish to change the center of your simulation in order to zoom to a specific part of it, you can use the pan command and zoom in:

```
/vis/viewer/panTo -5 -1
/vis/viewer/zoom 4.

# V I E W E R #
/vis/open OGL
# define the zoom factor
/vis/viewer/zoom 1.5
# Set the viewing angle
/vis/viewer/set/viewpointThetaPhi 5 60
# Set the drawing style
/vis/viewer/set/style surface
# Tell the viewer to draw the volumes
/vis/drawVolume
# The trajectories for each run should be drawn together
# don't store trajectories = 0; store trajectories = 1
/tracking/storeTrajectory 1
# Requests viewer to refresh hits, tracks, etc., at end of event.
# Or to accumulate drawings. Detector remains or is redrawn.
/vis/scene/endOfEventAction accumulate
```

The following commands implement additional options relevant within GATE:

```
# draw object in WireFrame Mode
/gate/block/vis/forceWireframe
```

or:

```
# draw object to appear as a solid
/gate/block/vis/forceSolid

# define object color
/gate/block/vis/setColor blue
```

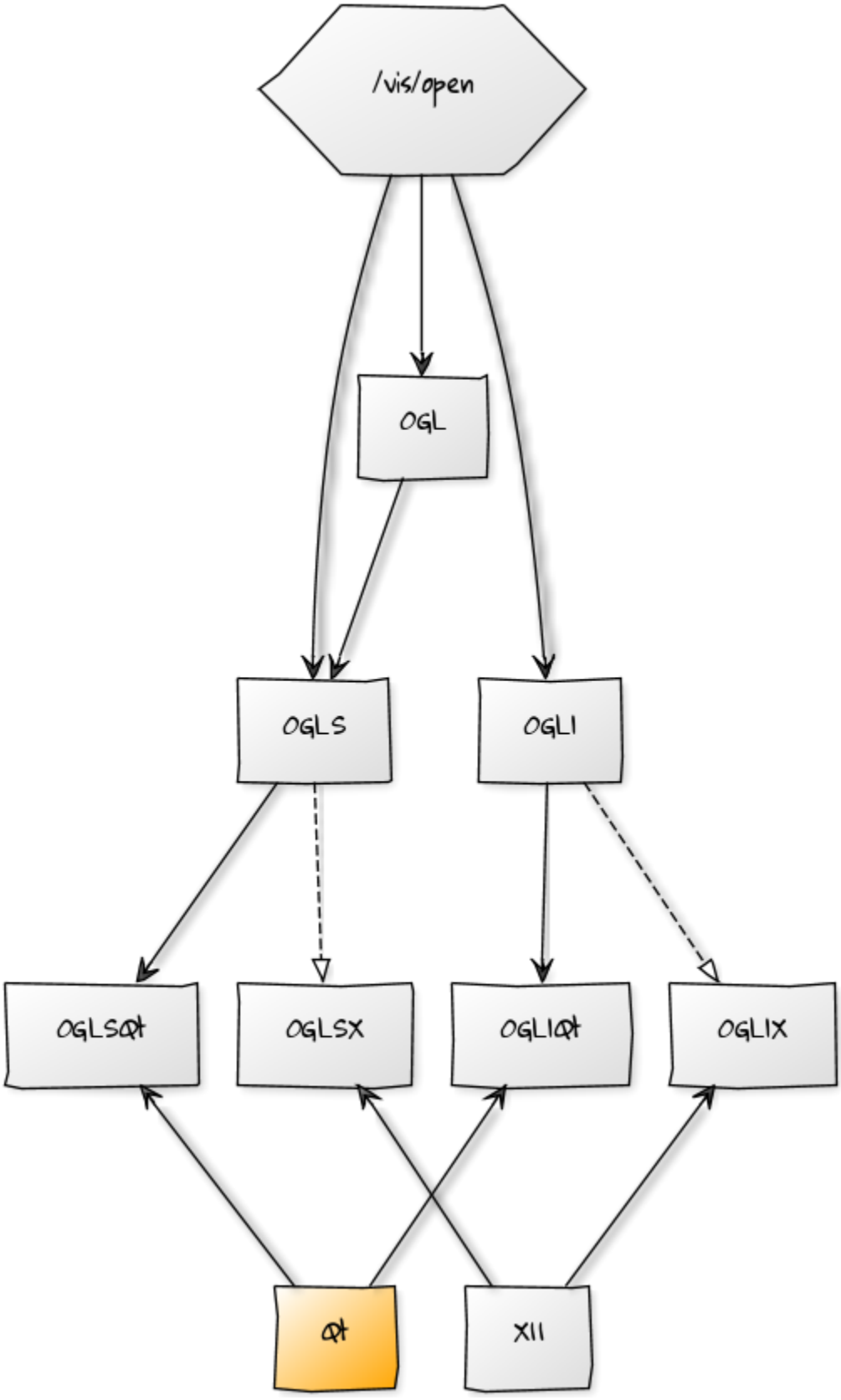


Fig. 2.45: OGL

Instead of block as in the example here, objects like crystal, source, scanner can be assigned specific vizualization properties.

The "autoUpdate" capability has been removed since version 6.0.0 and manual geometry update has to be used instead. This can be done using the following command at any moment in the GATE command prompt environment:

```
/gate/geometry/rebuild
```

Otherwise the complete geometry building is done when /gate/run/initialize command is given.

2.10.4 Visualization of Images

Since 7.0, GATE can show images with OpenGL but only in Immediate mode and the volume of the image must be in WireFrame mode. This functionality works with X11 and Qt.

2.10.5 Immediate mode

Command:

```
/vis/open OGLI
```

2.10.6 Stored mode

Command:

```
/vis/open OGLS
```

Visualization with DAWN

Instead of real-time visualization based on OpenGL, storing images in a file (mostly eps) for further processing might be useful. DAWN offers such options.

The package can be downloaded from the Internet and installed following the instruction given at http://geant4.kek.jp/~tanaka/src/dawn_3_85e.taz

To use DAWN and DAWNFILE in your macro, a specific open command should be used, in replacement of the opening of OpenGL:

```
/vis/open DAWNFILE
/vis/viewer/reset
/vis/viewer/set/viewpointThetaPhi 30 0
/vis/viewer/zoom 1.5
/vis/drawVolume
/tracking/storeTrajectory 1
/vis/scene/endOfEventAction accumulate
/vis/viewer/update /vis/viewer/refresh
```

Specific environment variables have to be set in your shell script to have access to DAWN inside GATE. For instance, in a C-shell:

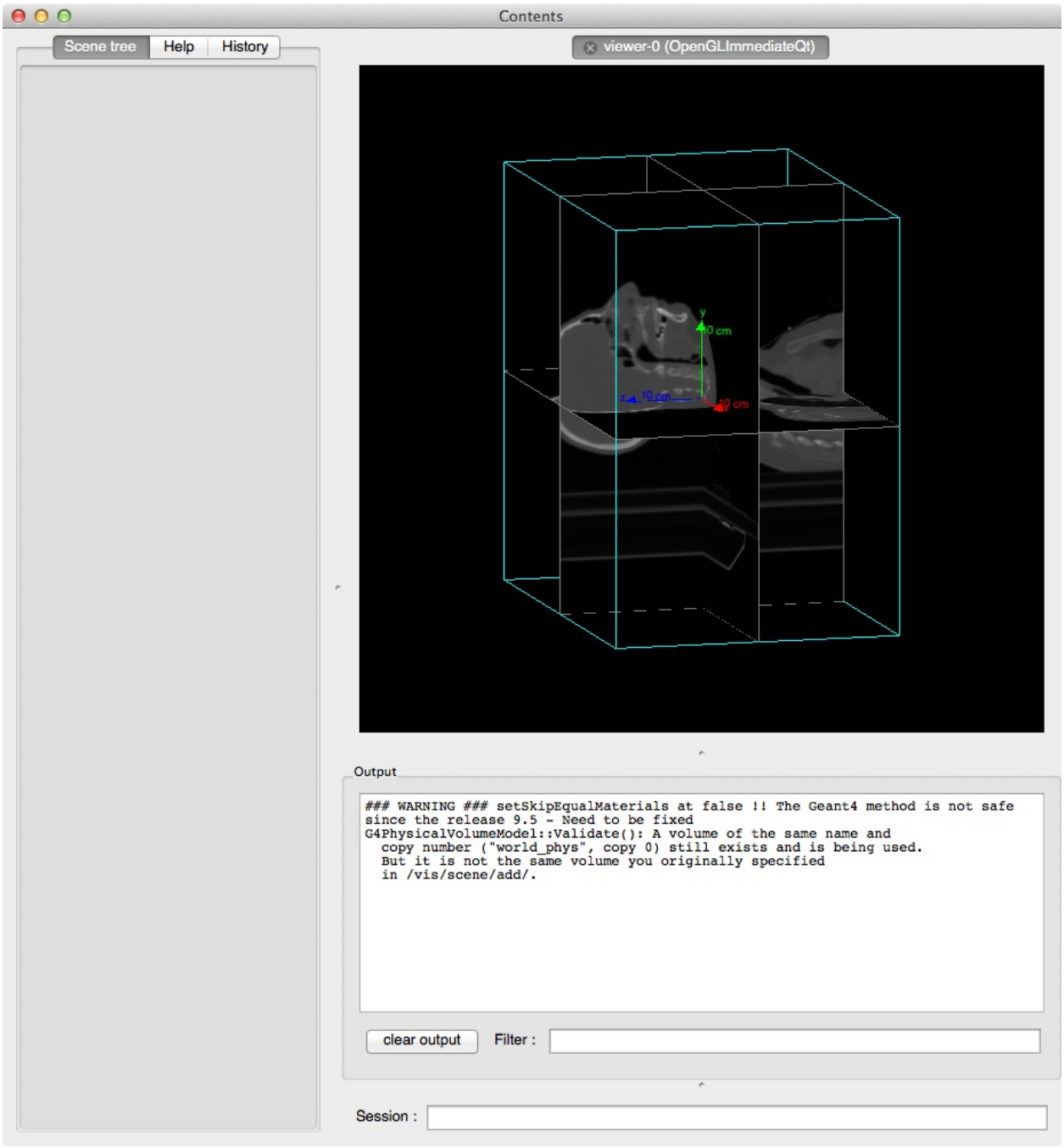


Fig. 2.46: OpenGL Immediate mode

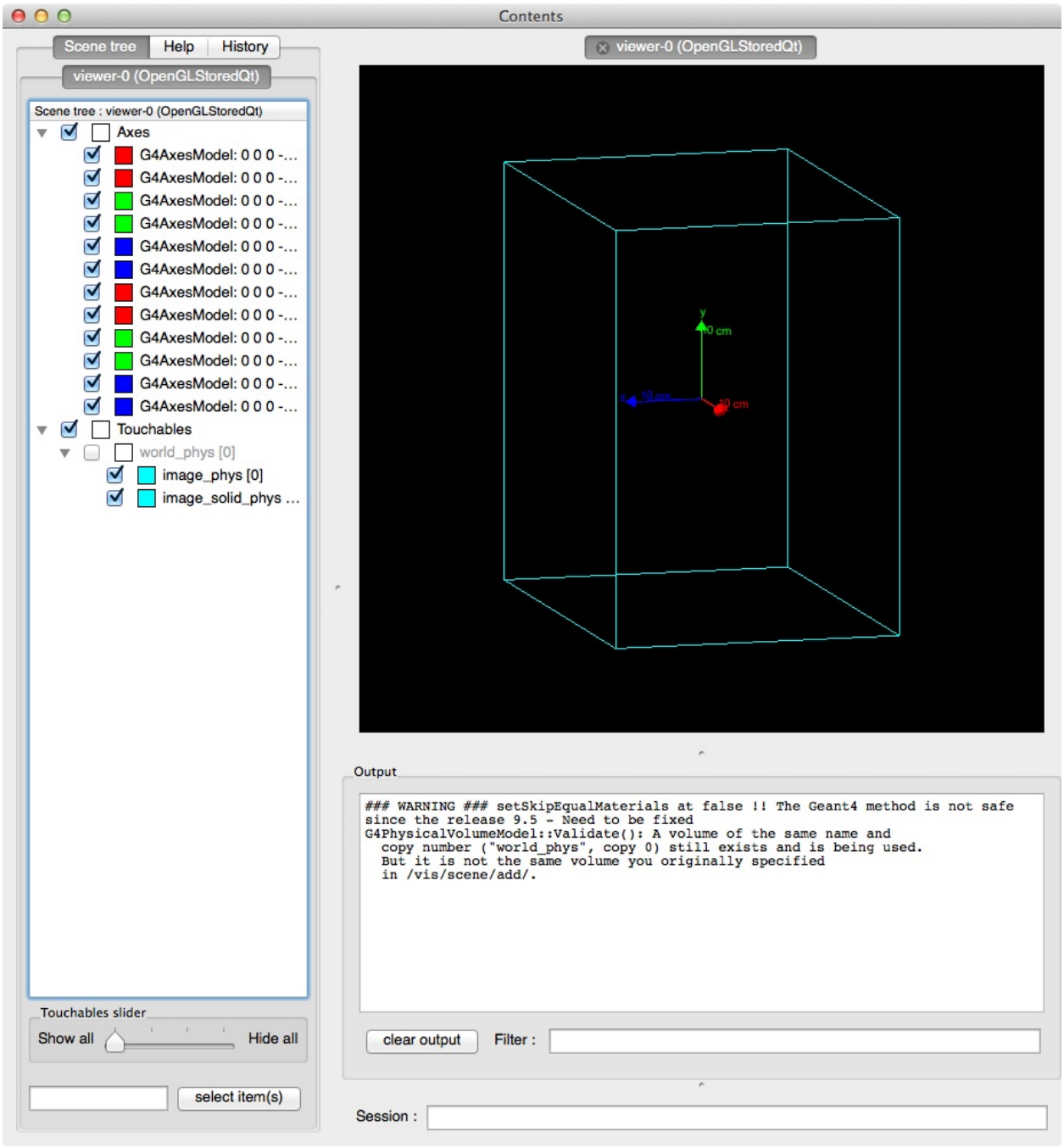


Fig. 2.47: OpenGL Stored mode

```
if ( Xn == Xy ) then
setenv G4VIS_BUILD_DAWN_DRIVER 1
echo "On this machine the G4VIS_BUILD_DAWN_DRIVER= G4VIS_BUILD_DAWN_DRIVER"
endif
```

and also:

```
if ( Xn == Xy ) then
setenv G4VIS_USE_DAWN 1$
echo "On this machine the G4VIS_USE_DAWN= G4VIS_USE_DAWN"
endif
```

Visualization with VRML

Sometimes, it may be helpful to check a geometry setup by interactively manipulating the visualized scene. These features are offered by the option VRML2FILE in connection with an appropriate viewer like vrmlview. Such a viewer can be freely downloaded from: http://www.sim.no/products/VRMLview/

A specific environment variable has to be set first:

```
setenv G4VRMLFILE_VIEWER vrmlview
```

For using this option in Gate, the following line has to be added to the macro instead of the corresponding OpenGL opening:

```
/vis/open VRML2FILE
```

Again, the environment variables have to be properly set (here C-schell example):

```
if [ Xn = Xy ] ;
then G4VIS_BUILD_VRML_DRIVER=1
export G4VIS_BUILD_VRML_DRIVER
echo "On this machine the G4VIS_BUILD_VRML_DRIVER=$G4VIS_BUILD_VRML_DRIVER"
fi

if [ Xn = Xy ] ;
then G4VIS_USE_VRML=1
export G4VIS_USE_VRML
echo "On this machine the G4VIS_USE_VRML=$G4VIS_USE_VRML"
fi
```

During processing in GATE, a file is written with the extension wrl.

Axes

Any position in the *world* is defined with respect to a three-axis system: X, Y and Z. These three axes can be seen in the display window using:

```
/vis/scene/add/axes
```

CHAPTER 3

Imaging application

3.1 Defining a system

Table of Contents

3.1.1 Definition

A System is a key-concept of GATE. It provides a *template* of a predefined geometry to simulate a scanner. A system can be used to model several scanners shating the same general geometrical characteristics. It can be considered as sort of a template described by key components organized in a certain way, what is called a *tree level structure*, each component having its own specific role or ordering.

For instance, in the cylindricalPET scanner system, the geometrical *volumes* containing crystals are grouped in matrices, themselves assembled in submodules and modules. At the top level of this structure, the sectors composed of modules are *repeated* on a cylindrical surface to build up the whole device. Thus, a family of PET scanners obeying this structure can be described using this system, illustrated in Fig. 3.1, composed of volumes called *rsectors*, *modules*, *submodules*, *crystal* and finaly (crystal) *layer*.

Different systems are available in GATE : *scanner*, *SPECTHead*, *cylindricalPET*, *ecat*, *CPET*, *OPET* and *OpticalSystem*, which can be used to simulate most of the existing imaging devices.

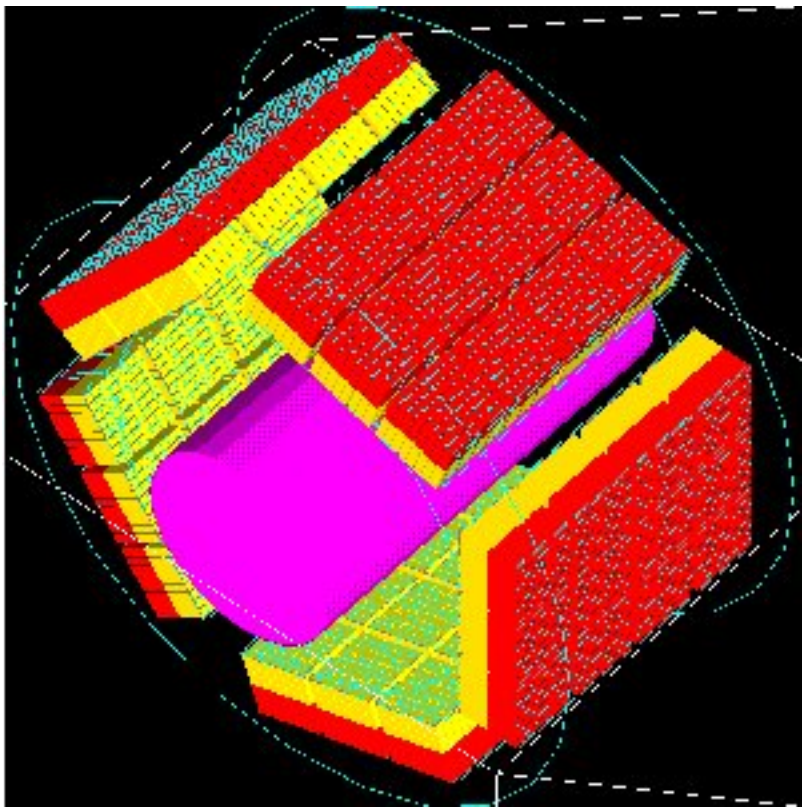


Fig. 3.1: Picture of a phantom and a cylindricalPET system composed of 5 rsectors, 4 modules (repeated along Z axis), 3 submodules (repeated along Y axis), 64 crystals (8 x 8) and 2 layers (red and yellow)

3.1.2 Choice of the system

It is possible to use GATE without using a system, but in that case, no information regarding particle interaction in the detector will be available. The reason is that the volumes where the *hits* (interactions that occur inside the detector parts of the scanner, see *Digitizer and readout parameters*) are recorded only for volumes belonging to a defined system (those volumes are declared as *crystalS*, *SD* for *sensitive detector*, see *Attaching the sensitive detectors*). When the user is only testing a scanner geometry, the use of a predefined system is not necessary. But if the user wants to record information related to the particle history inside the detector, the geometry has to be associated with a system. This section explains the elements and rules to associate a geometry with a system.

Geometry constraints

Except for the general system *scanner*, one should first take into account the *geometrical* shape of the different components (gantry, sector, bucket, etc.) and also the shape of the crystal or the detector material (*e.g.* scintillators).

Each *level* has to be assigned to a physical volume of the geometry. A level volume has to be fully enclosed in the upper level volume.

The number of levels has to be set and must conform to the specifications listed in Table 3.1. The numbering of different sensitive volumes is completely set by the choice of the system and conforms to a specific output format.

The maximum number of components in each level depends on the output format since it can be limited by the number of bits used for the numbering of the crystals. See *Data output* for further details.

Constraints related to the simulation of the DAQ electronics

Several points have to be considered when designing the simulation of the electronics cards. First, the whole readout electronic components should be analyzed in order to define its main components. This concerns not only the single channel simulation, with effects like thresholder response, but also the crosstalk between different channels including the electronic or the optical crosstalk among components in a same level. For a PET scanner, the coincidence between two channels have to be simulated, based on the *single component* simulations. In GATE, it is possible to introduce all these signal processing steps through the digitizer modules (see *Digitizer and readout parameters*), operating at different levels or depths, as shown in Table 3.2. The depth value is used here to tag a group of similar components operating at a certain level, which could be the scintillator block level (crystal with depth=5, or a group of crystal matrices with depth=1, depth values given in these examples refer to the cylindricalPET system).

To simulate the electronic processing consistenly with the system used to model the detector, the following procedure should be used:

- Regroup the detector electronic components in different levels.
- List the signal processing to be used for rach of the resulting groups (see *adder*, *readout*, *dead time* in *Digitizer and readout parameters*),
- Combine the signals coming from different volumes with, for example, the readout module for the signals summation of a volume, or the crosstalk and/or the coincidence between signals and coincidence.

NOTE : One or several crosstalk processing can be applied to components of different levels, for instance crosstalk between crystals, followed by crosstalk between modules. Such processing involves components at the same level. For PET scanners, coincidences are validated by testing the number difference in the uppermost level (as defined as depth = 1 in table). This test can reject accidental coincidence between adjacent logic structures. When the user builds a geometry, this logic organisation should correspond to the fisrt level of a system to use this coincidence sorting (see *Digitizer and readout parameters*).

3.1.3 How to connect the geometry to a system

The connection between the geometry and a system is performed in several steps:

- The geometrical structure needs first to be defined, keeping in mind that it must fulfill some constraints, as described before.
- The system geometry has then to be introduced, or *attached*, in the simulation process with the "attach" command and a specific *keyword* argument corresponding to one level of the geometrical structure. The general macro line for this attachment is:

```
systems/SystemName/Level/attach UserVolumeName
```

where :

- *SystemName* is the specific name of the system (one of the entry in column 1),
- *Level* is the specific name of the level (see column 2),
- *UserVolumeName* is the name the user gave to a volume, according to the conventions of *Defining a geometry*.
- Finally, the specific output corresponding to the system has to be defined for further data analysis (see *Data output*).

Table 3.1: Different systems available in GATE and their characteristics. In the second column are listed some of the keyword that are also used at in the macro (see also table 2 for a complete list). The shape in the third column describe the mother volume, composed of "daughter" volumes as described in Chap. 3 : a box means a box shaped mother volume containing an array of daughter boxes, a cylinder mother volumes will contains cylinders. Cylinders are understood here as tube sectors defined by an inner and outer radius.

System	Components and Shape		Available Outputs
scanner or PETscanner	level1	geometry not fixed	Basic output: ASCII or ROOT. Coincidences are only available for the "PETscanner" system The "level5" key was introduced after Gate v8.2. For Gate v8.2, the key layer0 and layer1 were used
	level2		
	level3		
	level4		
	level5		
CTscanner	module	box	Raw Data, ASCII, ROOT
	cluster	box	
	pixel	box	
CPET	sector	cylinder	Basic Output: ASCII, ROOT
	cassette	cylinder	
	module	box	
	crystal	box	
	layer	box	
cylindricalPET	rsector	box	Basic Output: ASCII, ROOT and Raw. Specific: LMF
	module	box	
	submodule	box	
	crystal	box	
	layer	box	

Continued on next page

Table 3.1 – continued from previous page

System	Components and Shape		Available Outputs
SPECThead	crystal	geometry not fixed	Basic Output: ASCII, ROOT and Raw. Specific: PROJECTIONSET or INTEFILE, no coincidences
	pixel		
ecat	block	box	Basic Output: ASCII, ROOT and Raw. Specific: SINOGRAM or ECAT7
	crystal		
ecatAccel	block	box	Basic Output: ASCII, ROOT and Raw. Specific: SINOGRAM or ECAT7
	crystal		
OPET	rsector	box	Basic Output: ASCII, ROOT and Raw. Specific: LMF
	module	box	
	submodule	box	
	crystal	box	
	layer	wedge	
OpticalSystem	crystal	geometry not fixed	Basic Output: ROOT and Raw. Specific: PROJECTIONSET
	pixel		

Table 3.2: Keywords corresponding to system components definition to be used with an "attach" command. At least one level has to be attached to the system. If necessary, these level's names can be possibly used as input to digitizers modules: for example, different electronic dead times for each level's electronics can be modelised. The two last lines, listed here for information, are related to "hits" which apply only for "sensitive" volume. Please refer to Chap. 5 for more details on this topic.

System	Attach Keyword Argument	Depth for readout segmentation
scanner	"level1"	1
	"level2"	2
	"level3"	3
	"level4"	4
	"level5"	5
CTscanner	"module"	1
	"cluster_0...2"	2
	"pixel_0...2"	3
cylindricalPET	"rsector"	1
	"module"	2
	"submodule"	3
	"crystal"	4
	"layer[i], i=0,3"	5
CPET	"sector"	1
	"cassette"	2
	"module"	3
	"crystal"	4
	"layer[i], i=0,3"	5
SPECThead	"crystal"	1
	"pixel"	2
ecat	"block"	1

Continued on next page

Table 3.2 – continued from previous page

System	Attach Keyword Argument	Depth for readout segmentation
	"crystal"	2
ecatAccel	"block"	1
	"crystal"	2
OPET	"rsector"	1
	"module"	2
	"submodule"	3
	"crystal"	4
	"layer[i], i=0,7"	5
OpticalSystem	"crystal"	1
	"pixel"	2

3.1.4 Different types of systems

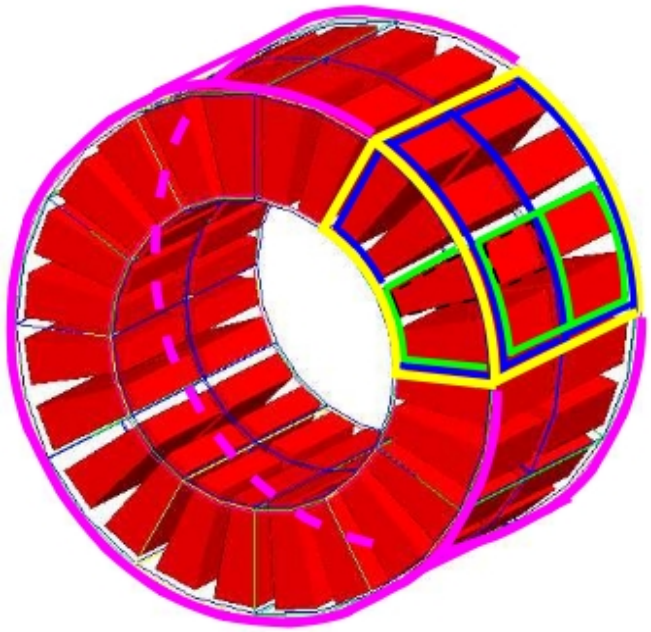


Fig. 3.2: Illustration of the scanner system. The different volumes, in particular the sensitive ones, can be of any shape, here cylindrical sector crystals, instead of boxes in other systems. The scanner cylinder is drawn in magenta, whereas one of the sector components : Level1, Level2, Level3, Level4 is shown in yellow, blue, green, red, respectively. The "Detector" volumes of cylindrical sector shapes are shown in plain red.

Scanner

Description

The *scanner* system is the most generic system in Gate. There is no geometrical constraints on the five different components.

Use

Different shapes of the volumes inside the tree level can be choosen, among those listed in Table 2.2

Fig. 3.7 illustrates the kind of detector that can be simulated with this system without any geometry constraint. On the other hand, there is no specific output format associated with this system and information regarding the hits are only available in ROOT or ASCII format.

CTscanner

The CTscanner system allows you to simulate a simple CT scanner. It has three possible levels:

- **module** component, that can be linearly repeated along the Y axis.
- **cluster** component, repeated inside the module, allowing you to simulate many kind of pixels
- **pixel** component, repeated inside the cluster. Raw data are the standard imageCT output to store the simulated CT projections and to produce it at each time slice. The image type is a float matrix (32-bits) of dimension given by the product of the number of pixels in X and Y, the content corresponds to the number of counts per pixel per acquisition (time slice).

Three types of simulations are proposed to the user:

- **Complete simulation:** The modules, the clusters, and the pixels are user defined. All volumes are created by Geant4 and the digitalization can be made at the pixel level (level 3).
- **Fast simulation:** Only the module level is defined. Geant4 creates one volume corresponding to the CT module (only one block possible) and the digitalization is made by the output module. The number of pixels per module are given through the output module messenger. This mode is faster since only one Geant4 volume is simulated, but obviously, only a rather approximated scanner response can be garanteed.
- **Complete simulation with a Variance Reduction Technique (VRT):** In the same way as the complete simulation, the components (pixels, clusters, and modules) are user defined. Unlike the complete simulation, using Geant4 in the detector, the user handles the particle on the surface of the CT scanner. For more informations see the part below.

This variance reduction technique (VRT) has been developped with the aim to making the simulation time faster. Here are the successive steps of the implementation:

- **Generation and Propagation** of the particles through the World, then detection of those on the surface of the detector. The propagation of the particles through the detector are 'killed', in order to handle ourself the detection and not by Geant4.
- **Computation of the mean free path** (MFP) of the particle through the detector with the standard model (the compatibility with the low energy model being not implemented yet)
- **Computation of the path** of the particle in the detector:

$PATH = MFP * -log(1 - R)$

R being a distribution uniformly random number between 0 and 1

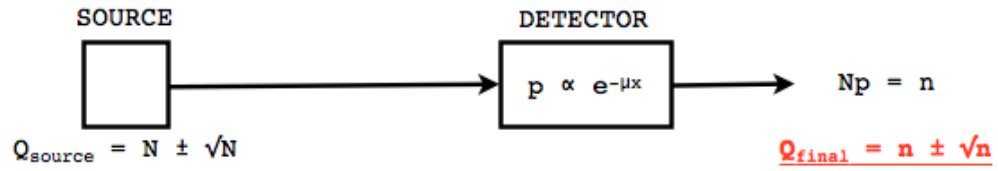
The user may perform this last step for each particle K times, in order to decrease the simulation time by avoiding a new generation and propagation of the particle. However it has an influence on the variance of the output data. The following scheme shows the differences with a simulation with a without VRT:

N: mean of the generated particles, and $\sqrt{N}$ its standard deviation.

p: binomial probability of detection of the particle.

n = Np: mean of the number of detected particles, and $\sqrt{n} = \sqrt{Np}$ its standard deviation.

- Without VRT:



- With VRT:

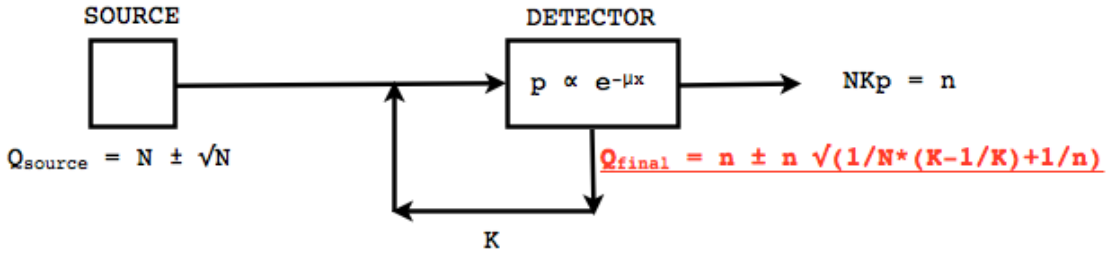


Fig. 3.3: VRT sheme

The simulation time decreases linearly with K. But K:math:*geqslant 10*, because of the reduction of the variance should be avoided. For deeper insight, see the following table and graph.

VRT	Activity (Bq)	Mean (ph./ pix.)	Standard deviation	n (number of photons in each pixel, before the detection)	Model
0	20000	9 798,77	99,43	19 252,87	98,99
	15000	7 349,11	85,60	14 440,17	85,73
	10000	4 899,66	69,88	9 626,28	70,00
	5000	2 449,67	49,50	4 814,34	49,49
2	10000	9 798,82	119,61	9 626,28	140,61
	7500	7 349,21	103,20	7 220,27	121,77
	5000	4 899,16	85,32	4 814,34	99,42
	2500	2 449,97	60,27	2 407,34	70,31
5	4000	9 800,17	173,64	3 851,30	186,35
	3000	7 350,01	150,06	2 888,70	161,38
	2000	4 901,09	122,61	1 925,97	131,78
	1000	2 449,44	84,72	963,18	93,14
10	2000	9 799,77	237,58	1 925,97	244,16
	1500	7 348,27	202,52	1 444,38	211,41
	1000	4 896,89	165,64	963,18	172,53
	500	2 448,43	116,61	481,60	122,00

Fig. 3.4: VRT table

The simulation corresponding to the table is:

- A detector with 10,000 pixels (0.5x0.5x0.5 mm3) in Silicon (Si)
- A monochromatic source (17.6 keV)
- A time of exposition of 1 second

Use

example 1: complete CT simulation:

```
###############
# CT SCANNER #
###############
/gate/world/daughters/name CTscanner
/gate/world/daughters/insert box
/gate/CTscanner/placement/setTranslation 0.00 0.00 100 mm
/gate/CTscanner/geometry/setXLength 100 mm
/gate/CTscanner/geometry/setYLength 100 mm
```

(continues on next page)

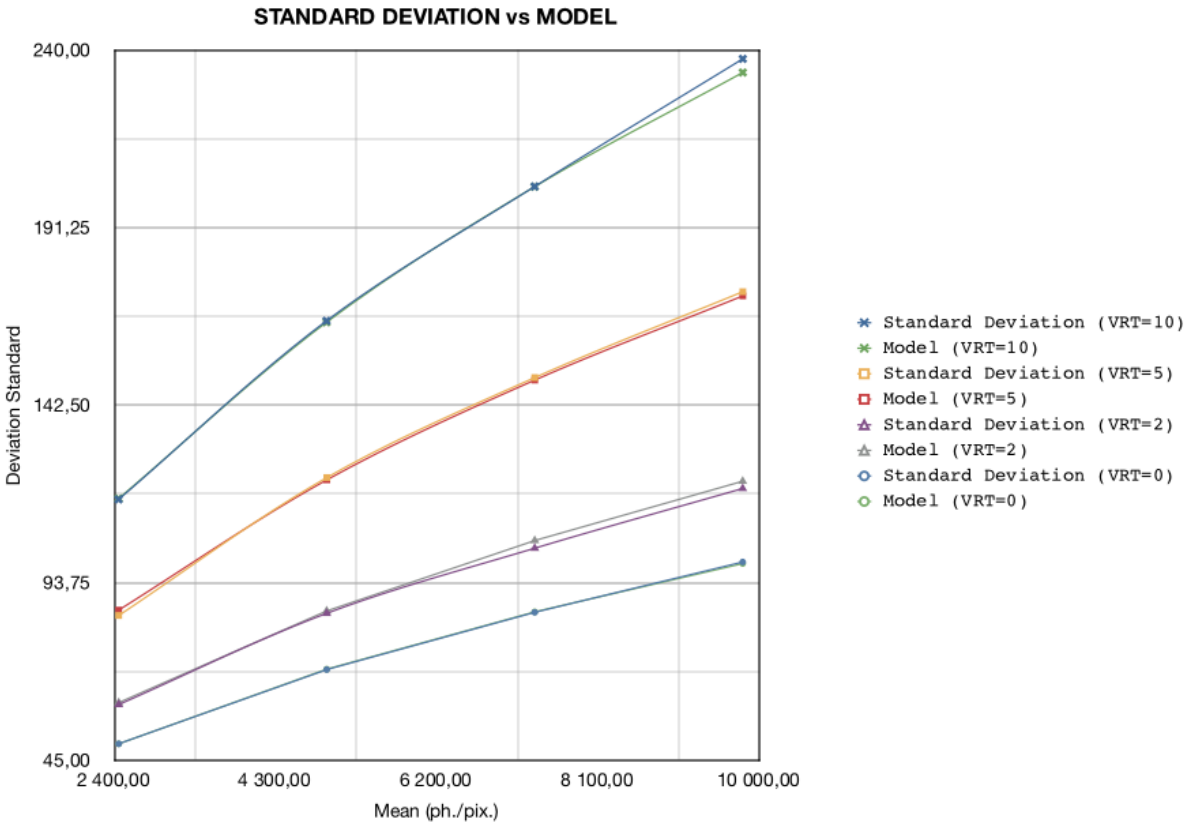


Fig. 3.5: VRT graph

(continued from previous page)

```
/gate/CTscanner/geometry/setZLength 0.5 mm
/gate/CTscanner/setMaterial Air
/gate/CTscanner/vis/forceWireframe
/gate/CTscanner/vis/setColor white

#############           ############
# CTSCANNER #   ---->   #  MODULE  #
#############           ############
/gate/CTscanner/daughters/name module
/gate/CTscanner/daughters/insert box
/gate/module/geometry/setXLength 100 mm
/gate/module/geometry/setYLength 100 mm
/gate/module/geometry/setZLength 0.5 mm
/gate/module/setMaterial Air
/gate/module/vis/forceWireframe
/gate/module/vis/setColor white

############           #############
#  MODULE  #   ---->   # CLUSTER_0 #
############           #############
/gate/module/daughters/name cluster
/gate/module/daughters/insert box
/gate/cluster/geometry/setXLength 100 mm
/gate/cluster/geometry/setYLength 100 mm
/gate/cluster/geometry/setZLength 0.5 mm
/gate/cluster/setMaterial Air
/gate/cluster/vis/forceWireframe
/gate/cluster/vis/setColor white

############             #############           ###########
#  MODULE  #   ---->     # CLUSTER_0 #   ---->    # PIXEL_0 #
############             #############           ###########
/gate/cluster/daughters/name pixel
/gate/cluster/daughters/insert box
/gate/pixel/geometry/setXLength 1 mm
/gate/pixel/geometry/setYLength 1 mm
/gate/pixel/geometry/setZLength 1 mm
/gate/pixel/setMaterial Silicon
/gate/pixel/vis/setColor red

# REPEAT PIXEl_0
/gate/pixel/repeaters/insert cubicArray
/gate/pixel/cubicArray/setRepeatNumberX 100
/gate/pixel/cubicArray/setRepeatNumberY 100
/gate/pixel/cubicArray/setRepeatNumberZ   1
/gate/pixel/cubicArray/setRepeatVector 1 1 0.0 mm
/gate/pixel/cubicArray/autoCenter true

# ATTACH SYSTEM
/gate/systems/CTscanner/module/attach module
/gate/systems/CTscanner/cluster_0/attach cluster
/gate/systems/CTscanner/pixel_0/attach pixel

# ATTACH LAYER
/gate/pixel/attachCrystalSD
```

example 2: complete CT simulation with VRT

In the same way as the complete simulation, the difference is the output (K = 5):

```
/gate/output/imageCT/vrtFactor 5
```

Exemple 3 : Fast CT simulation:

```
###############
# CT SCANNER #
###############
/gate/world/daughters/name CTscanner
/gate/world/daughters/insert box
/gate/CTscanner/placement/setTranslation 0.00 0.00 100 mm
/gate/CTscanner/geometry/setXLength 1.00 mm
/gate/CTscanner/geometry/setYLength 1.00 mm
/gate/CTscanner/geometry/setZLength 0.50 mm
/gate/CTscanner/setMaterial Air
/gate/CTscanner/vis/forceWireframe
/gate/CTscanner/vis/setColor white

##############            ############
# CTSCANNER #    ---->    #  MODULE  #
##############            ############
/gate/CTscanner/daughters/name module
/gate/CTscanner/daughters/insert box
/gate/module/geometry/setXLength 1. mm
/gate/module/geometry/setYLength 1. mm
/gate/module/geometry/setZLength 0.50 mm
/gate/module/setMaterial Air
/gate/module/vis/forceWireframe
/gate/module/vis/setColor white

#############           #############
#  MODULE  #    ---->   # CLUSTER_0 #
#############           #############
/gate/module/daughters/name cluster
/gate/module/daughters/insert box
/gate/cluster/geometry/setXLength 1. mm
/gate/cluster/geometry/setYLength 1. mm
/gate/cluster/geometry/setZLength 0.50 mm
/gate/cluster/setMaterial Air
/gate/cluster/vis/forceWireframe
/gate/cluster/vis/setColor white

#############             #############           ############
#  MODULE  #    ---->     # CLUSTER_0 #    ---->   # PIXEL_0 #
#############             #############           ############
/gate/cluster/daughters/name pixel
/gate/cluster/daughters/insert box
/gate/pixel/geometry/setXLength 1. mm
/gate/pixel/geometry/setYLength 1. mm
/gate/pixel/geometry/setZLength 0.5 mm
/gate/pixel/setMaterial Silicon
/gate/pixel/vis/setColor red


# ATTACH SYSTEM
/gate/systems/CTscanner/module/attach module
/gate/systems/CTscanner/cluster_0/attach cluster
```

(continues on next page)

(continued from previous page)

```
/gate/systems/CTscanner/pixel_0/attach pixel

# ATTACH LAYER
/gate/pixel/attachCrystalSD

/gate/output/imageCT/numFastPixelX 100
/gate/output/imageCT/numFastPixelY 100
/gate/output/imageCT/numFastPixelZ 1
```

CylindricalPET

Description

CylindricalPET is a PET system that can describe most of the small animal PET scanners. The main specificity of *cylindricalPET* is the possibility to record output data in the List Mode Format (LMF) developed by the Crystal Clear Collaboration. A complete description of LMF is can be found in *LMF output*.

A CylindricalPET is based on a cylindrical geometry, and consists of 5 hierarchic levels, from mother to daughter, as defined below:

- **world cylindricalPET** is defined as a cylinder in the world, with a non zero inner radius.
- **rsector** (depth=1) is defined as a box, and repeated with a *ring repeater* in cylindricalPET.
- **module** (depth=2) is a box inside *rsector*. It is repeated by a *cubicarray repeater* with no X repetition (*repeatNumberX = 1*). This level is optional.
- **submodule** (depth=3) is a box inside *module*. It is repeated by a *cubicarray repeater* with no X repetition (*repeatNumberX = 1*). This level is optional.
- **crystal** (depth=4) is a box inside *submodule*. It is repeated by a *cubicarray repeater* with no X repetition (*repeatNumberX = 1*).
- **layer** (depth=5) is a (or several, in the case of a phoswich system) radially arranged box(es) inside *crystal*. A repeater should not be used for layers, but the should be build them one by one in the macro. *layer* must be set as a sensible detector with the macro command:

```
/attachCrystalSD
```

The words in bold characters are dedicated. See also keywords in Table 3.2.

Material of *layer* (s) must be the material of the detector, for instance LSO or BGO + GSO for a double layer phoswich system. Materials of other levels (crystals, submodules, modules, rsectors, cylindricalPET) can be anything else.

IMPORTANT : Visualization should help you build this geometry with no overlap. GATE performs a test for detecting volume overlap, but with a limited precision. This test is performed at the end of the initialization of Gate (see *Getting Started*):

```
/run/initialize
/geometry/test/recursive_test
```

Users should carefully check that volumes are not bigger than the mother volume they are included in.

Use

An example of definition of a PET scanner following the CylindricalPET system structure is given below. The definition of the scanner should be performed at the beginning of the macro, before initializations:

```
# W O R L D
/gate/world/geometry/setXLength 40 cm
/gate/world/geometry/setYLength 40. cm
/gate/world/geometry/setZLength 40. cm

# M O U S E
/gate/world/daughters/name mouse
/gate/world/daughters/insert cylinder
/gate/mouse/setMaterial Water
/gate/mouse/vis/setColor red
/gate/mouse/geometry/setRmax 18.5 mm
/gate/mouse/geometry/setRmin 0. mm
/gate/mouse/geometry/setHeight 68. mm

# C Y L I N D R I C A L
/gate/world/daughters/name cylindricalPET
/gate/world/daughters/insert cylinder
/gate/cylindricalPET/setMaterial Water
/gate/cylindricalPET/geometry/setRmax 145 mm
/gate/cylindricalPET/geometry/setRmin 130 mm
/gate/cylindricalPET/geometry/setHeight 80 mm
/gate/cylindricalPET/vis/forceWireframe

# R S E C T O R
/gate/cylindricalPET/daughters/name rsector
/gate/cylindricalPET/daughters/insert box
/gate/rsector/placement/setTranslation 135 0 0 mm
/gate/rsector/geometry/setXLength 10. mm
/gate/rsector/geometry/setYLength 19. mm
/gate/rsector/geometry/setZLength 76.6 mm
/gate/rsector/setMaterial Water
/gate/rsector/vis/forceWireframe

# M O D U L E
/gate/rsector/daughters/name module
/gate/rsector/daughters/insert box
/gate/module/geometry/setXLength 10. mm
/gate/module/geometry/setYLength 19. mm
/gate/module/geometry/setZLength 19. mm
/gate/module/setMaterial Water
/gate/module/vis/forceWireframe
/gate/module/vis/setColor gray

# C R Y S T A L
/gate/module/daughters/name crystal
/gate/module/daughters/insert box
/gate/crystal/geometry/setXLength 10. mm
/gate/crystal/geometry/setYLength 2.2 mm
/gate/crystal/geometry/setZLength 2.2 mm
/gate/crystal/setMaterial Water
/gate/crystal/vis/forceWireframe
/gate/crystal/vis/setColor magenta
```

(continues on next page)

(continued from previous page)

```
# L A Y E R
/gate/crystal/daughters/name LSO
/gate/crystal/daughters/insert box
/gate/LSO/geometry/setXLength 10. mm
/gate/LSO/geometry/setYLength 2.2 mm
/gate/LSO/geometry/setZLength 2.2 mm
/gate/LSO/placement/setTranslation 0 0 0 mm
/gate/LSO/setMaterial LSO
/gate/LSO/vis/setColor yellow

# R E P E A T C R Y S T A L
/gate/crystal/repeaters/insert cubicArray
/gate/crystal/cubicArray/setRepeatNumberX 1
/gate/crystal/cubicArray/setRepeatNumberY 8
/gate/crystal/cubicArray/setRepeatNumberZ 8
/gate/crystal/cubicArray/setRepeatVector 10. 2.4 2.4 mm

# R E P E A T M O D U L E
/gate/module/repeaters/insert cubicArray
/gate/module/cubicArray/setRepeatNumberZ 4
/gate/module/cubicArray/setRepeatVector 0. 0. 19.2 mm

# R E P E A T R S E C T O R
/gate/rsector/repeaters/insert ring
/gate/rsector/ring/setRepeatNumber 42

# A T T A C H S Y S T E M
/gate/systems/cylindricalPET/rsector/attach rsector
/gate/systems/cylindricalPET/module/attach module
/gate/systems/cylindricalPET/crystal/attach crystal
/gate/systems/cylindricalPET/layer0/attach LSO

# A T T A C H L A Y E R SD
/gate/LSO/attachCrystalSD
/gate/mouse/attachPhantomSD
```

CPET

This system was defined for the simulation of a CPET-like scanner (C-PET Plus, Philips Medical Systems, the Netherlands), with one ring of NaI crystal with a curved shape. For this scanner, a single level in addition to the system level is enough to describe the volume hierarchy.

Description

This system has the particularity to have cylindrical shaped sector components, based on the *cylinder* shape (see Fig. 3.6 and *Defining a geometry* for definitions), whereas these components are generally boxes in other systems.

Use

Described below is an example of code appropriate for modeling a one ring scanner of NaI crystal with a curved shape:

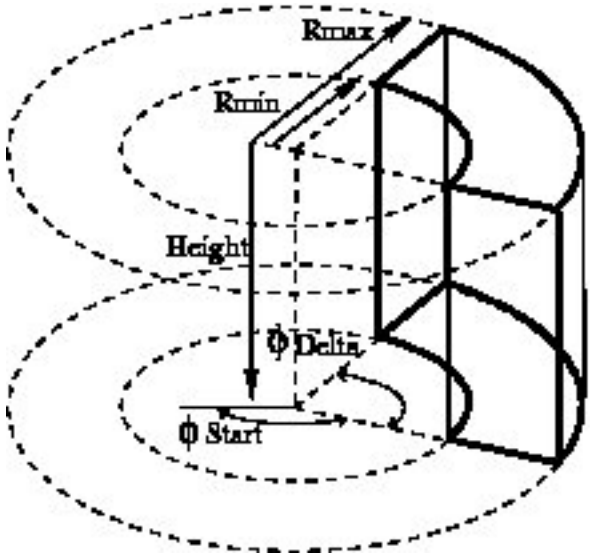


Fig. 3.6: Definition of a CPET sector volume. This system allows one to define sectors with a cylindrical shape instead of sectors with a box shape,like in other PET systems

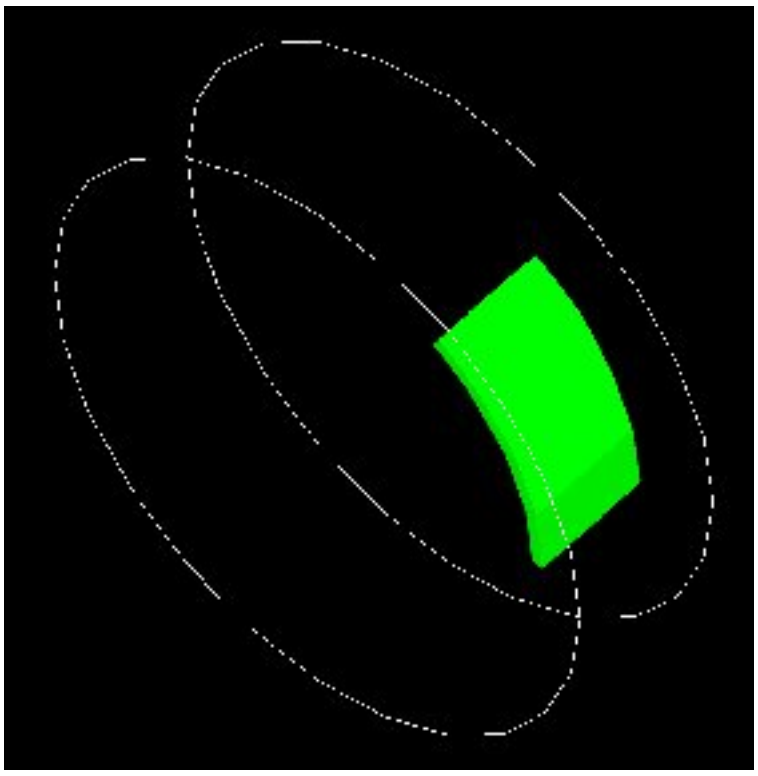


Fig. 3.7: One NaI crystal with a curved shape

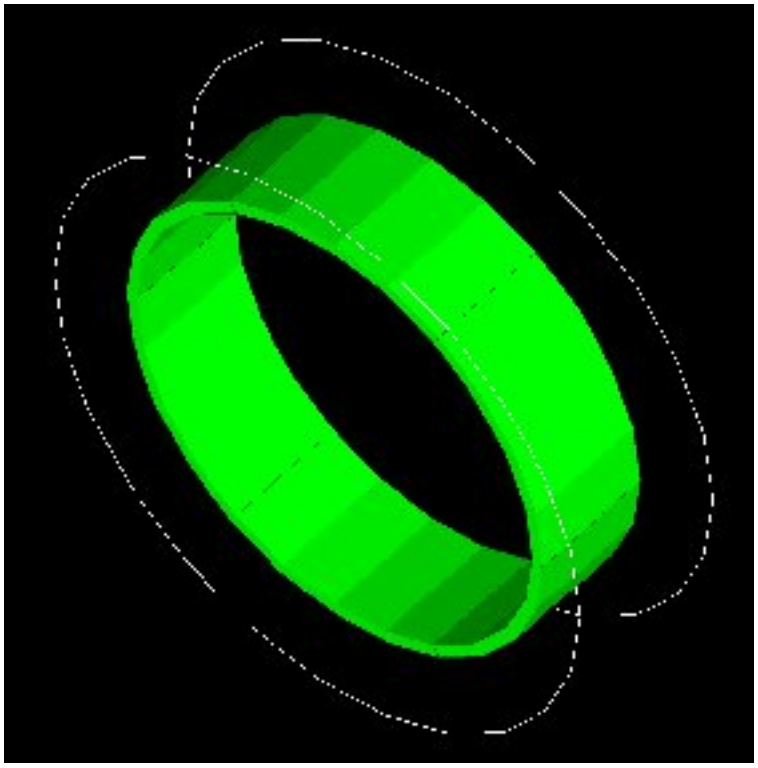


Fig. 3.8: After the ring repeater, the scanner consists of 6 NaI crystals

```
# BASE = CPET SYSTEM
/gate/world/daughters/name CPET
/gate/world/daughters/insert cylinder
/gate/CPET/setMaterial Air
/gate/CPET/geometry/setRmax 60 cm
/gate/CPET/geometry/setRmin 0.0 cm
/gate/CPET/geometry/setHeight 35.0 cm
/gate/CPET/vis/forceWireframe

# FIRST LEVEL = CRYSTAL
/gate/CPET/daughters/name crystal
/gate/CPET/daughters/insert cylinder
/gate/crystal/geometry/setRmax 47.5 cm
/gate/crystal/geometry/setRmin 45.0 cm
/gate/crystal/geometry/setHeight 25.6 cm
/gate/crystal/geometry/setPhiStart 0 deg
/gate/crystal/geometry/setDeltaPhi 60 deg

# REPEAT THE CURVE SECTOR INTO THE WHOLE RING
/gate/crystal/repeaters/insert ring
/gate/crystal/ring/setRepeatNumber 6

# CRYSTAL VOLUME IS MADE OF NAI
/gate/crystal/setMaterial NaI
/gate/crystal/vis/setColor green
```

The object *crystal* is then attached to its corresponding component in the CPET system (level 1 : the sector level for CPET system - see previous section for details):

```
/gate/systems/CPET/sector/attach crystal
```

The crystals are set as sensitive detectors (see *The crystalSD*):

```
/gate/crystal/attachCrystalSD
```

The digitizer part (see *Digitizer modules*) is made of the *adder* module and some blurring module (see *Digitizer and readout parameters*).

Ecat

Description

The ecat system is a simplified version of *cylindricalPET* and was named *ecat* because it is appropriate for modelling PET scanners from the **ECAT** family, from CPS Innovations (Knoxville, TN, U.S.A.). Such scanners are based on the block detector principle, consisting in an array of crystals, typically 8 x 8 read by few photomultipliers, typically 4. The blocks are organized along an annular geometry to yield multi-ring detectors.

An example of macro with an ecat definition is provided in:

https://github.com/OpenGATE/GateContrib/blob/master/imaging/PET/PET_Ecat_System.mac

The ecat system has only three hierarchical levels: one is for the entire detector (*base*), one for the block (*block*), and one for the crystals within the block (*crystal*).

In addition to the standard output modules (*ASCII* and *root*), two additional output modules are specifically associated to the *ecat* system, and correspond to sinogram formats. These are the *sinogram* and the *ecat7* output modules and are discussed in *Sinogram output* and *Ecat7 output*.

Use

Described below is an example of code for modeling a four block-ring scanner.

It has to be named after the selected system (*ecat* here) and is defined as a volume daughter of the *world*. It has a ring shape and should include all detectors (see Fig. 3.9):

```
/gate/world/daughters/name ecat
/gate/world/daughters/insert cylinder
/gate/ecat/setMaterial Air
/gate/ecat/geometry/setRmax 442.0 mm
/gate/ecat/geometry/setRmin 412.0 mm
/gate/ecat/geometry/setHeight 155.2 mm
/gate/ecat/setTranslation 0.0 0.0 0.0 mm
```

The following commands set the size and the position of the first block within the base *ecat*. It is a rectangular parallelepiped and should include all crystals within a block. For a multiple block-ring system centered axially on the base *ecat*, the axial position of this first block should be set to zero (see Fig. 3.9):

```
/gate/ecat/daughters/name block
/gate/ecat/daughters/insert box
/gate/block/placement/setTranslation 427.0 0.0 0.0 mm
/gate/block/geometry/setXLength 30.0 mm
/gate/block/geometry/setYLength 35.8 mm
/gate/block/geometry/setZLength 38.7 mm
/gate/block/setMaterial Air
```

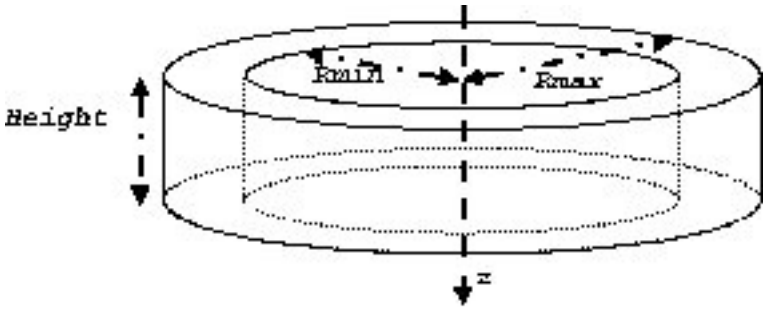


Fig. 3.9: Definition of the base

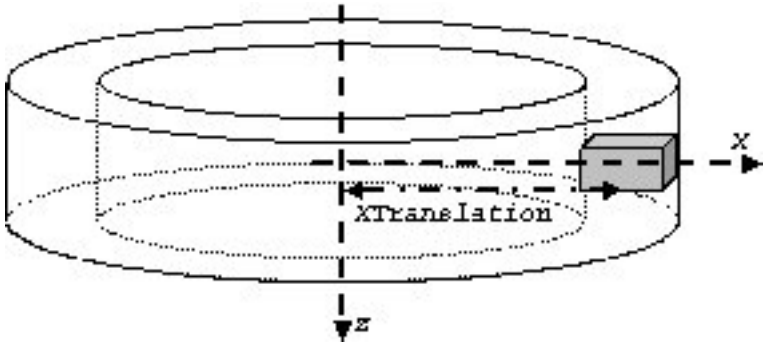


Fig. 3.10: Definition of the block

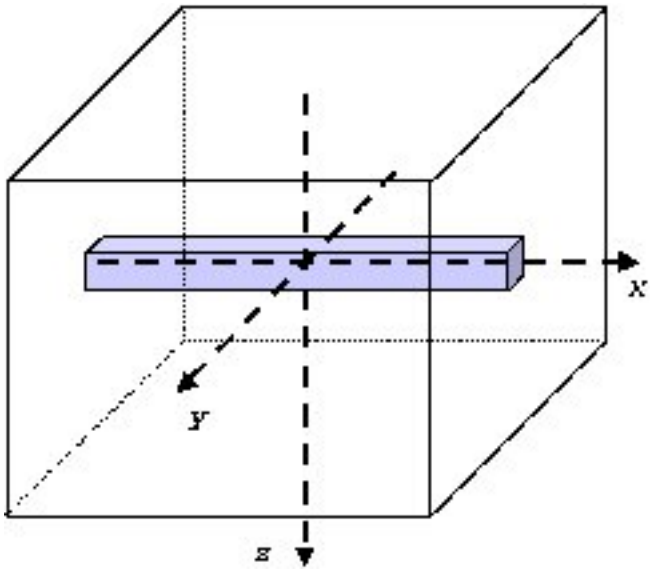


Fig. 3.11: Definition of the crystal

The next commands set the size and the position of the first crystal within the *block*. For a crystal array centered on the *block*, the position of this first crystal should be at the center of the block (see Fig. 3.11):

```
/gate/block/daughters/name crystal
/gate/block/daughters/insert box
/gate/crystal/placement/setTranslation 0.0 0.0 0.0 mm
/gate/crystal/geometry/setXLength 30.0 mm
/gate/crystal/geometry/setYLength 4.4 mm
/gate/crystal/geometry/setZLength 4.75 mm
/gate/crystal/setMaterial BGO
```

Finally, the crystal array is described. The sampling of the crystals within a block is defined, together with the size and the sampling of the crystal array within one block. The crystal array is centered on the position of the original crystal:

```
/gate/crystal/repeaters/insert cubicArray
/gate/crystal/cubicArray/setRepeatNumberX 1
/gate/crystal/cubicArray/setRepeatNumberY 8
/gate/crystal/cubicArray/setRepeatNumberZ 8
/gate/crystal/cubicArray/setRepeatVector 0. 4.45 4.80 mm
```

To create the full scanner, the rings have then to be defined. The following commands set the number of blocks per block-ring and the number of block-rings. Multiple block-ring systems will be centered axially on the axial position of the original block:

```
/gate/block/repeaters/insert linear
/gate/block/linear/setRepeatNumber 4
/gate/block/linear/setRepeatVector 0. 0. 38.8 mm
/gate/block/repeaters/insert ring
/gate/block/ring/setRepeatNumber 72
```

This description results in a 4 block-ring scanner, *i.e.* a 32 crystal-ring scanner, with 576 crystals per crystal-ring.

Command lines are then used to attach the objects *block* and *crystal* to their corresponding components in the ecat system:

```
systems/ecat/block/attach block
systems/ecat/crystal/attach crystal
```

To detect events, the crystals are finally set as sensitive detectors (see *The crystalSD*):

```
/gate/crystal/attachCrystalSD
```

The digitizer part (see *Digitizer modules*) can be the same as for the cylindricalPET system.

ecatAccel

Description

The ecatAccel system was introduced to model a new PET scanner family ECAT ACCEL (from CPS Innovations, Knoxville, TN, U.S.A.). The ecatAccel system differs from the ecat system by its geometrical shape : the detection blocks are arranged along a spherical ring whereas they are arranged along annular rings for the ecat system. As data processing and output format are highly dependent on the scanner geometry, it was necessary to introduce a new system even though it has many common features with the ecat system. The same hierarchical levels (base, block and crystal) as for the ecat system are used to describe the geometry of the ecatAccel system, and the same standard output modules (ASCII and root) and specific outputs (sinogram and ecat7) are also available. Please refer to *Sinogram output* and *Ecat7 output* for further information on sinogram and ecat7 outputs for the ecatAccel system.

Use

Described below is an example of code for modeling the ACCEL PET scanner of the BIOGRAPH-LSO (SIEMENS - CTI) PET-CT scanner.

The scanner is named after the selected system (ecatAccel here) and is defined as a volume daughter of the world. As for the ecat system, it has a ring shape and should include all detectors (see Fig. 3.9). For the BIOGRAPH, it can be described as follows:

The base is described:

```
/gate/world/daughters/name ecatAccel
/gate/world/daughters/insert cylinder
/gate/ecatAccel/setMaterial Air
/gate/ecatAccel/geometry/setRmax 437.0 mm
/gate/ecatAccel/geometry/setRmin 412.0 mm
/gate/ecatAccel/geometry/setHeight 162. mm
/gate/ecatAccel/setTranslation 0.0 0.0 0.0 mm
```

The block is then described: the size and the position of the first block are set within the base ecatAccel. As for the ecat system, it is a rectangular parallelepiped and should include all crystals within a block. For a multiple block-ring system centered axially on the base ecatAccel, the axial position of this first block should be set to zero:

```
/gate/ecatAccel/daughters/name block
/gate/ecatAccel/daughters/insert box
/gate/block/geometry/setXLength 51.6 mm
/gate/block/geometry/setYLength 25.0 mm
/gate/block/geometry/setZLength 51.6 mm
/gate/block/setMaterial Air
```

The crystal geometry and settings are then specified. The following commands set the size and the position of the first crystal within the block. For a crystal array centered on the block, the position of this first crystal should be at the center of the block:

```
/gate/block/daughters/name crystal
/gate/block/daughters/insert box
/gate/crystal/placement/setTranslation 0.0 0.0 0.0 mm
/gate/crystal/geometry/setXLength 6.45 mm
/gate/crystal/geometry/setYLength 25.0 mm
/gate/crystal/geometry/setZLength 6.45 mm
/gate/crystal/setMaterial LSO
```

Then the crystal array is described within a block. The crystal array will be centered on the position of the original crystal:

```
/gate/crystal/repeaters/insert cubicArray
/gate/crystal/cubicArray/setRepeatNumberX 8
/gate/crystal/cubicArray/setRepeatNumberY 1
/gate/crystal/cubicArray/setRepeatNumberZ 8
/gate/crystal/cubicArray/setRepeatVector 6.45 0.0 6.45 mm
```

Finally, the different rings of the scanner are described. The number of blocks per block-ring (command setRepeatNumberWithTheta) is indicated as well as the number of block-rings (command setRepeatNumberWithPhi). The angle between two adjacent blocks in a block-ring is set with the command setThetaAngle and the angle between two adjacent blocks belonging to two neighbouring rings in the axial direction is set with the command setPhiAngle. Multiple block-ring will be centered axially on the axial position of the original block:

```
/gate/block/repeaters/insert sphere
/gate/block/sphere/setRadius 424.5 mm
/gate/block/sphere/setRepeatNumberWithTheta 3ionSource
/gate/block/sphere/setRepeatNumberWithPhi 48
/gate/block/setThetaAngle 7.5 deg
/gate/block/setPhiAngle 7.5 deg
```

This description results in a 3 block-ring scanner, i.e. a 24 crystal-ring scanner, with 384 crystals per crystal-ring.

The objects block and crystal are attached to their corresponding components in the ecatAccel system:

```
/gate/systems/ecatAccel/block/attach block
/gate/systems/ecatAccel/crystal/attach crystal
```

The sensitive detector is set to the crystals as for the ecat system and the digitizer part remains the same as for the cylindricalPET system.

OPET

Description

The OPET system was introduced to model a new PET prototype.

Use

Described below is an example of code for modeling the OPET PET scanner:

```
# R S E C T O R (Create a box to put the crystals in: one PMT)
/gate/OPET/daughters/name rsector
/gate/OPET/daughters/insert box
/gate/rsector/placement/setTranslation 20.4955 0 0 mm
/gate/rsector/geometry/setXLength 10 mm
/gate/rsector/geometry/setYLength 17.765 mm
/gate/rsector/geometry/setZLength 17.765 mm
/gate/rsector/setMaterial Air
/gate/rsector/vis/setVisible False
/gate/rsector/vis/forceWireframe
/gate/rsector/vis/setColor yellow

# M O D U L E (Make a box for one row of 8 crystals)
/gate/rsector/daughters/name module
/gate/rsector/daughters/insert box
/gate/module/geometry/setXLength 10 mm
/gate/module/geometry/setYLength 17.765 mm
/gate/module/geometry/setZLength 2.162 mm
/gate/module/setMaterial Air
/gate/module/vis/setVisible False
/gate/module/vis/forceWireframe
/gate/module/vis/setColor black

# Daughter crystal inside mother crystal
/gate/module/daughters/name crystal0
/gate/module/daughters/insert box
/gate/crystal0/geometry/setXLength 10 mm
```

(continues on next page)

(continued from previous page)

```
/gate/crystal0/geometry/setYLength 2.1620 mm
/gate/crystal0/geometry/setZLength 2.1620 mm
/gate/crystal0/placement/setTranslation 0. -7.8015 0. mm
/gate/crystal0/setMaterial Air
/gate/crystal0/vis/setColor black
/gate/crystal0/vis/setVisible false

# L A Y E R (Put the LSO in the small box)
/gate/crystal0/daughters/name LSO0
/gate/crystal0/daughters/insert wedge
/gate/LSO0/geometry/setXLength 10 mm
/gate/LSO0/geometry/setNarrowerXLength 8.921 mm
/gate/LSO0/geometry/setYLength 2.1620 mm
/gate/LSO0/geometry/setZLength 2.1620 mm
/gate/LSO0/placement/setRotationAxis 0 1 0
/gate/LSO0/placement/setRotationAngle 180 deg
/gate/LSO0/placement/setTranslation 0.2698 0. 0. mm
/gate/LSO0/setMaterial BGO
/gate/LSO0/vis/setColor yellow

# Daughter crystal inside mom crystal
/gate/module/daughters/name crystal1
/gate/module/daughters/insert box
/gate/crystal1/geometry/setXLength 10 mm
/gate/crystal1/geometry/setYLength 2.1620 mm
/gate/crystal1/geometry/setZLength 2.1620 mm
/gate/crystal1/placement/setTranslation 0. -5.5725 0. mm
/gate/crystal1/setMaterial Air
/gate/crystal1/vis/setColor magenta
/gate/crystal1/vis/forceWireframe
/gate/crystal1/vis/setVisible false

# L A Y E R (Put the LSO in the small box)
/gate/crystal1/daughters/name LSO1
/gate/crystal1/daughters/insert wedge
/gate/LSO1/geometry/setXLength 8.921 mm
/gate/LSO1/geometry/setNarrowerXLength 8.193 mm
/gate/LSO1/geometry/setYLength 2.1620 mm
/gate/LSO1/geometry/setZLength 2.1620 mm
/gate/LSO1/placement/setRotationAxis 0 1 0
/gate/LSO1/placement/setRotationAngle 180 deg
/gate/LSO1/placement/setTranslation 0.7215 0. 0. mm
/gate/LSO1/setMaterial BGO
/gate/LSO1/vis/setColor red

# Daughter crystal inside mom crystal
/gate/module/daughters/name crystal2
/gate/module/daughters/insert box
/gate/crystal2/geometry/setXLength 10 mm
/gate/crystal2/geometry/setYLength 2.1620 mm
/gate/crystal2/geometry/setZLength 2.1620 mm
/gate/crystal2/placement/setTranslation 0. -3.3435 0. mm
/gate/crystal2/setMaterial Air
/gate/crystal2/vis/setColor black
/gate/crystal2/vis/setVisible false

# L A Y E R (Put the LSO in the small box)
```

(continues on next page)

(continued from previous page)

```
/gate/crystal2/daughters/name LSO2
/gate/crystal2/daughters/insert wedge
/gate/LSO2/geometry/setXLength 8.193 mm
/gate/LSO2/geometry/setNarrowerXLength 7.773 mm
/gate/LSO2/geometry/setYLength 2.1620 mm
/gate/LSO2/geometry/setZLength 2.1620 mm
/gate/LSO2/placement/setRotationAxis 0 1 0
/gate/LSO2/placement/setRotationAngle 180 deg
/gate/LSO2/placement/setTranslation 1.0085 0. 0. mm
/gate/LSO2/setMaterial BGO
/gate/LSO2/vis/setColor green

# Daughter crystal inside mom crystal
/gate/module/daughters/name crystal3
/gate/module/daughters/insert box
/gate/crystal3/geometry/setXLength 10 mm
/gate/crystal3/geometry/setYLength 2.1620 mm
/gate/crystal3/geometry/setZLength 2.1620 mm
/gate/crystal3/placement/setTranslation 0. -1.1145 0. mm
/gate/crystal3/setMaterial Air #
/gate/crystal3/vis/forceWireframe
/gate/crystal3/vis/setColor black
/gate/crystal3/vis/setVisible false

# L A Y E R (Put the LSO in the small box)
/gate/crystal3/daughters/name LSO3
/gate/crystal3/daughters/insert wedge
/gate/LSO3/geometry/setXLength 7.773 mm
/gate/LSO3/geometry/setNarrowerXLength 7.637 mm
/gate/LSO3/geometry/setYLength 2.1620 mm
/gate/LSO3/geometry/setZLength 2.1620 mm
/gate/LSO3/placement/setRotationAxis 0 1 0
/gate/LSO3/placement/setRotationAngle 180 deg
/gate/LSO3/placement/setTranslation 1.1475 0. 0. mm
/gate/LSO3/setMaterial BGO
/gate/LSO3/vis/setColor blue

# Daughter crystal inside mom crystal
/gate/module/daughters/name /gate/crystal4/
/gate/module/daughters/insert box
/gate/crystal4//geometry/setXLength 10 mm
/gate/crystal4//geometry/setYLength 2.1620 mm
/gate/crystal4//geometry/setZLength 2.1620 mm
/gate/crystal4//placement/setTranslation 0. 1.1145 0. mm
/gate/crystal4//setMaterial Air
/gate/crystal4//vis/setColor black
/gate/crystal4//vis/setVisible false

# L A Y E R (Put the LSO in the small box)
/gate/crystal4//daughters/name /gate/LSO4
/gate/crystal4//daughters/insert wedge
/gate/LSO4/geometry/setXLength 7.773 mm
/gate/LSO4/geometry/setNarrowerXLength 7.637 mm
/gate/LSO4/geometry/setYLength 2.1620 mm
/gate/LSO4/geometry/setZLength 2.1620 mm
/gate/LSO4/placement/setRotationAxis 0 0 1
/gate/LSO4/placement/setRotationAngle 180 deg
```

(continues on next page)

(continued from previous page)

```
/gate/LSO4/placement/setTranslation 1.1475 0. 0. mm
/gate/LSO4/setMaterial BGO
/gate/LSO4/vis/setColor blue

# Daughter crystal1 inside mom crystal
/gate/module/daughters/name crystal5
/gate/module/daughters/insert box
/gate/crystal5/geometry/setXLength 10 mm
/gate/crystal5/geometry/setYLength 2.1620 mm
/gate/crystal5/geometry/setZLength 2.1620 mm
/gate/crystal5/placement/setTranslation 0. 3.3435 0. mm
/gate/crystal5/setMaterial Air
/gate/crystal5/vis/setColor black
/gate/crystal5/vis/setVisible false

# L A Y E R (Put the LSO in the small box)
/gate/crystal5/daughters/name LSO5
/gate/crystal5/daughters/insert wedge
/gate/LSO5/geometry/setXLength 8.193 mm
/gate/LSO5/geometry/setNarrowerXLength 7.773 mm
/gate/LSO5/geometry/setYLength 2.1620 mm
/gate/LSO5/geometry/setZLength 2.1620 mm
/gate/LSO5/placement/setRotationAxis 0 0 1
/gate/LSO5/placement/setRotationAngle 180 deg
/gate/LSO5/placement/setTranslation 1.0085 0. 0. mm
/gate/LSO5/setMaterial BGO
/gate/LSO5/vis/setColor green

# Daughter crystal1 inside mom crystal
/gate/module/daughters/name /gate/crystal6
/gate/module/daughters/insert box
/gate/crystal6/geometry/setXLength 10 mm
/gate/crystal6/geometry/setYLength 2.1620 mm
/gate/crystal6/geometry/setZLength 2.1620 mm
/gate/crystal6/placement/setTranslation 0. 5.5725 0. mm
/gate/crystal6/setMaterial Air
/gate/crystal6/vis/forceWireframe
/gate/crystal6/vis/setColor black
/gate/crystal6/vis/setVisible false

# L A Y E R (Put the LSO in the small box)
/gate/crystal6/daughters/name /gate/LSO6
/gate/crystal6/daughters/insert wedge
/gate/LSO6/geometry/setXLength 8.921 mm
/gate/LSO6/geometry/setNarrowerXLength 8.193 mm
/gate/LSO6/geometry/setYLength 2.1620 mm
/gate/LSO6/geometry/setZLength 2.1620 mm
/gate/LSO6/placement/setRotationAxis 0 0 1
/gate/LSO6/placement/setRotationAngle 180 deg
/gate/LSO6/placement/setTranslation 0.7215 0. 0. mm
/gate/LSO6/setMaterial BGO
/gate/LSO6/vis/setColor red

# Daughter crystal1 inside mom crystal
/gate/module/daughters/name /gate/crystal7
/gate/module/daughters/insert box
/gate/crystal7/geometry/setXLength 10 mm
```

(continues on next page)

(continued from previous page)

```
/gate/crystal7/geometry/setYLength 2.1620 mm
/gate/crystal7/geometry/setZLength 2.1620 mm
/gate/crystal7/placement/setTranslation 0. 7.8015 0. mm
/gate/crystal7/setMaterial Air
/gate/crystal7/vis/forceWireframe
/gate/crystal7/vis/setColor black
/gate/crystal7/vis/setVisible false

# L A Y E R (Put the LSO in the small box)
/gate/crystal7/daughters/name /gate/LSO7
/gate/crystal7/daughters/insert wedge
/gate/LSO7/geometry/setXLength 10 mm
/gate/LSO7/geometry/setNarrowerXLength 9.07 mm
/gate/LSO7/geometry/setYLength 2.1620 mm
/gate/LSO7/geometry/setZLength 2.1620 mm
/gate/LSO7/placement/setTranslation 0.2698 0. 0. mm
/gate/LSO7/placement/setRotationAxis 0 0 1
/gate/LSO7/placement/setRotationAngle 180 deg
/gate/LSO7/setMaterial BGO
/gate/LSO7/vis/setColor yellow

# Repeat 8 time the level2 to get 8 rings (Z direction)
/gate/module/repeaters/insert linear
/gate/module/linear/setRepeatNumber 8
/gate/module/linear/setRepeatVector 0. 0. 2.2275 mm

/gate/rsector/repeaters/insert ring
/gate/rsector/ring/setRepeatNumber 6

/gate/OPET/placement/setRotationAxis 0 0 1 #
/gate/OPET/placement/setRotationAngle 30 deg

# A T T A C H S Y S T E M
/gate/systems/OPET/rsector/attach rsector
/gate/systems/OPET/module/attach module
/gate/systems/OPET/layer0/attach LSO0
/gate/systems/OPET/layer1/attach LSO1
/gate/systems/OPET/layer2/attach LSO2
/gate/systems/OPET/layer3/attach LSO3
/gate/systems/OPET/layer4/attach /gate/LSO4
/gate/systems/OPET/layer5/attach LSO5
/gate/systems/OPET/layer6/attach /gate/LSO6
/gate/systems/OPET/layer7/attach /gate/LSO7

#A T T A C H L A Y E R SD : definition of your sensitive detector
/gate/LSO0/attachCrystalSD
/gate/LSO1/attachCrystalSD
/gate/LSO2/attachCrystalSD
/gate/LSO3/attachCrystalSD
/gate/LSO4/attachCrystalSD
/gate/LSO5/attachCrystalSD
/gate/LSO6/attachCrystalSD
/gate/LSO7/attachCrystalSD
```

Fig. 3.12 shows the final OPET scanner.

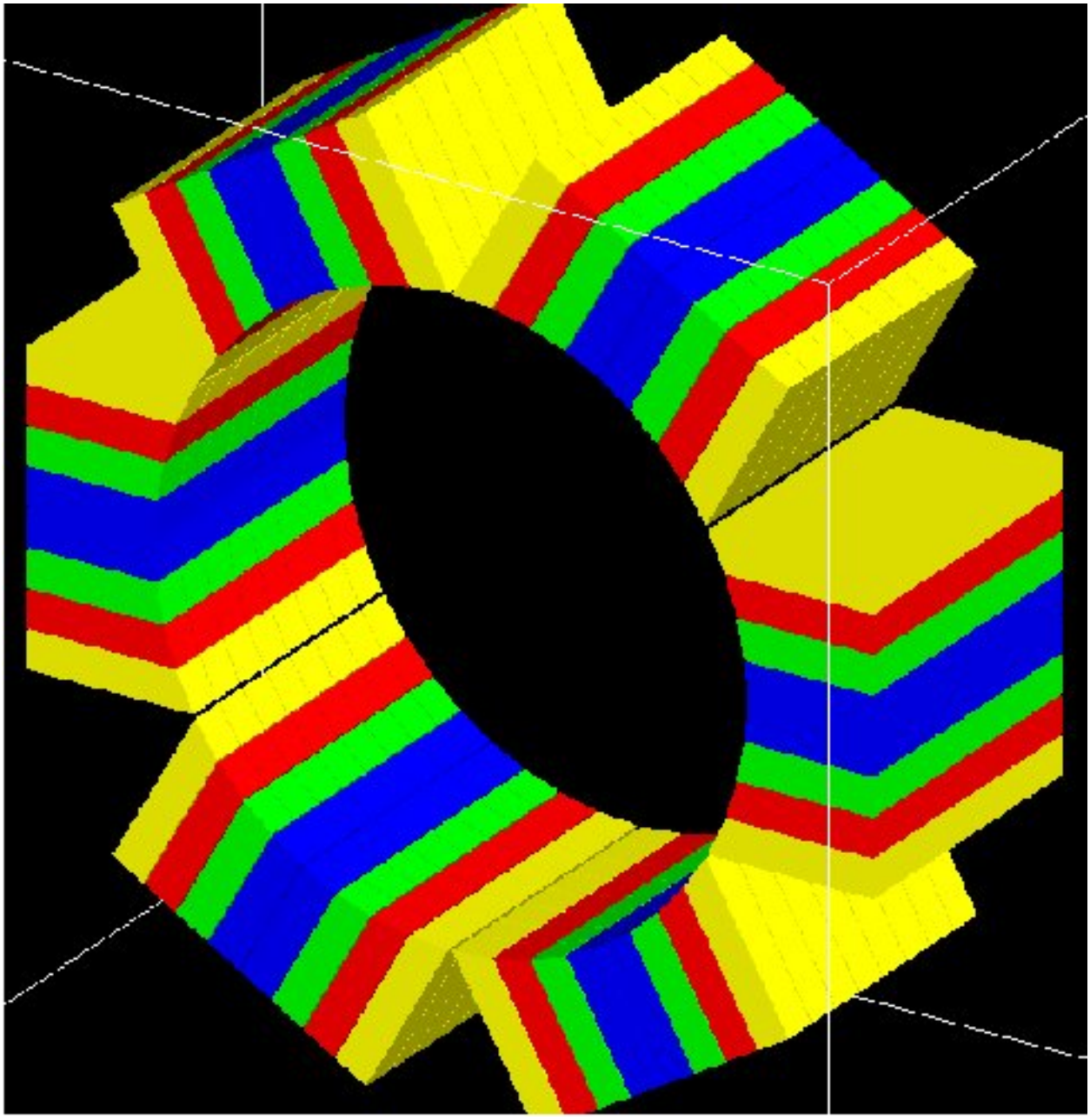


Fig. 3.12: The OPET scanner

SPECTHead

Description

SPECTHead is a SPECT system appropriate to model SPECT dedicated scanners within GATE. The main reason for specifying SPECThead is that it can be coupled to the InterFile output which is discussed in *Interfile output of projection set*. An example macro defining a typical SPECT scanner can be found in:

https://github.com/OpenGATE/GateContrib/blob/master/imaging/SPECT/SPECT.mac

wherein the specific Interfile output module is called.

A SPECThead system is a box-shaped geometry element and consists of three hierarchic levels:

- **base** which is always attached to the volume SPECThead, which is a dedicated word.
- **crystal** which is coupled to the main detector block.
- **pixel** which can be used for modeling a pixelated detector.

If a uniform detector block is being used, then the *crystal* material should be that of the detector. If the detector is pixelated, then the *pixel* material definition should correspond to the detector material, while the crystal material can be anything non-specific.

Use

Below is part of the SPECT benchmark macro, which is distributed with the GATE software, and which involves the SPECTHead system:

```
# World
# Define the world dimensions
/gate/world/ dimensions
/gate/world/geometry/setXLength 100 cm
/gate/world/geometry/setYLength 100 cm
/gate/world/geometry/setZLength 100 cm

# SPECThead is the name of the predefined SPECT system

# Create the SPECT system, which will yield
# an Interfile output of the projection data
/gate/world/daughters/name /gate/SPECThead
/gate/world/daughters/insert box

# Define the dimensions
/gate/SPECThead/geometry/setXLength 7. cm
/gate/SPECThead/geometry/setYLength 21. cm
/gate/SPECThead/geometry/setZLength 30. cm

# Define the position
/gate/SPECThead/placement/setTranslation 20.0 0. 0. cm

# Set the material associated with the main volume
/gate/SPECThead/setMaterial Air

# Replicate the head (around the Z axis by default)
# to get a hypothetical four-headed system
/gate/SPECThead/repeaters/insert ring
/gate/SPECThead/ring/setRepeatNumber 4
```

(continues on next page)

(continued from previous page)

```
/gate/SPECThead/ring/setAngularPitch 90. deg

#  Define the rotation speed of the head
#  Define the orbiting around the Z axis
/gate/SPECThead/moves/insert orbiting
/gate/SPECThead/orbiting/setSpeed 0.15 deg/s
/gate/SPECThead/orbiting/setPoint1 0 0 0 cm
/gate/SPECThead/orbiting/setPoint2 0 0 1 cm

# Define visualisation options
/gate/SPECThead/vis/forceWireframe

# Collimator
# Create a full volume defining the shape of
# the collimator (typical for SPECT)
/gate/SPECThead/daughters/name /gate/collimator
/gate/SPECThead/daughters/insert box

# Define the dimensions of the collimator volume
/gate/collimator/geometry/setXLength 3. cm
/gate/collimator/geometry/setYLength 19. cm
/gate/collimator/geometry/setZLength 28. cm

# Define the position of the collimator volume
/gate/collimator/placement/setTranslation -2. 0. 0. cm

# Set the material of the collimator volume
/gate/collimator/setMaterial Lead

# Define some visualisation options
/gate/collimator/vis/setColor red
/gate/collimator/vis/forceWireframe

# Insert the first hole of air in the collimator
/gate/collimator/daughters/name /gate/hole
/gate/collimator/daughters/insert hexagone
/gate/hole/geometry/setHeight 3. cm
/gate/hole/geometry/setRadius .15 cm
/gate/hole/placement/setRotationAxis 0 1 0
/gate/hole/placement/setRotationAngle 90 deg
/gate/hole/setMaterial Air

# Repeat the hole in an array
/gate/hole/repeaters/insert cubicArray
/gate/hole/cubicArray/setRepeatNumberX 1
/gate/hole/cubicArray/setRepeatNumberY 52
/gate/hole/cubicArray/setRepeatNumberZ 44
/gate/hole/cubicArray/setRepeatVector 0. 0.36 0.624 cm

# Repeat linearly these holes
/gate/hole/repeaters/insert linear
/gate/hole/linear/setRepeatNumber 2
/gate/hole/linear/setRepeatVector 0. 0.18 0.312 cm
/gate/hole/attachPhantomSD

# Crystal
# Create the crystal volume
```

(continues on next page)

(continued from previous page)

```
/gate/SPECThead/daughters/name crystal
/gate/SPECThead/daughters/insert box

# Define the dimensions of the crystal volume
/gate/crystal/geometry/setXLength 1. cm
/gate/crystal/geometry/setYLength 19. cm
/gate/crystal/geometry/setZLength 28. cm

# Define the position of the crystal volume
/gate/crystal/placement/setTranslation 0. 0. 0. cm

# Set the material associated with the crystal volume
/gate/crystal/setMaterial NaI
/gate/crystal/attachCrystalSD

# The SPECThead system is made of three levels: base (for the head),
#crystal (for the crystal and crystal matrix) and pixel
#(for individual crystals for pixelated gamma camera)

/gate/systems/SPECThead/crystal/attach crystal

# Look at the system
/gate/systems/SPECThead/describe
```

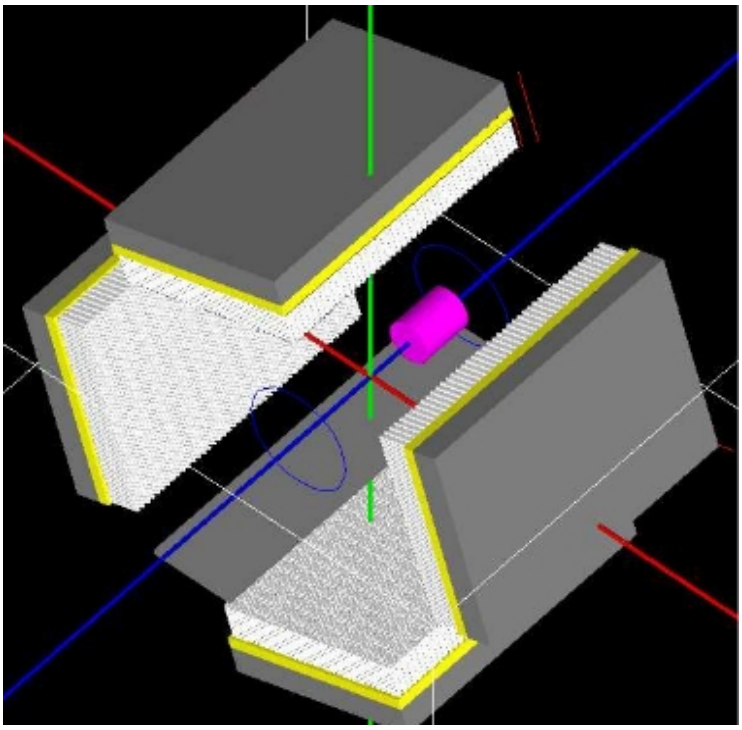


Fig. 3.13: Example of a hypothetical four-headed SPECThead system. The detectors are not pixelated in this example

Modelling the collimator

SPECT systems need collimator. A parameterized collimator setup was developed for both parallel hole collimators and fan beam collimators. It is based on the GEANT4 replica system in which a single volume represents multiple copies of a volume (the air holes) within its mother volume (the collimator itself). SPECT collimator geometries are built using this approach in less than a second.

Example for parallel hole collimators:

```
/gate/SPECThead/daughters/name colli

#specify that the parallel beam collimator setup must be used
/gate/SPECThead/daughters/insert parallelbeam

#set the collimator material
/gate/colli/setMaterialName Lead

#set the collimator dimensions
/gate/colli/geometry/setDimensionX 70 cm
/gate/colli/geometry/setDimensionY 80 cm

#set the thickness of the collimator
/gate/colli/geometry/setHeight 3 cm

#specify the hole radius
/gate/colli/geometry/setInnerRadius 0.5 cm

#set the septal thickness to the required distance between the holes
/gate/colli/geometry/setSeptalThickness 0.2 cm
/gate/colli/placement/alignToX
/gate/colli/placement/setRotationAxis 0 0 1
/gate/colli/placement/setRotationAngle -90 deg
```

! Fan Beam option is out of order - Should be debug ASAP !

Example of code for modelling fanbeam collimators:

```
/gate/SPECThead/daughters/name fanbeam
/gate/SPECThead/daughters/insert collimator

#set the material for the collimator
/gate/fanbeam/setMaterial Lead #define the X and Y size of the collimator
/gate/fanbeam/geometry/setDimensionY 53.5 cm
/gate/fanbeam/geometry/setDimensionX 25.0 cm

#specify the focal length
/gate/fanbeam/geometry/setFocalDistanceY 0.0 cm
/gate/fanbeam/geometry/setFocalDistanceX 35.0 cm

#specify the thickness of the collimator
/gate/fanbeam/geometry/setHeight 5.8 cm

#set the septal thickness to the required distance between the holes
/gate/fanbeam/geometry/setSeptalThickness 0.8 cm

#specify the hole radius
/gate/fanbeam/geometry/setInnerRadius 1.70 cm
/gate/fanbeam/placement/setRotationAxis 0 0 1
```

(continues on next page)

(continued from previous page)

```
/gate/fanbeam/placement/setRotationAngle -90 deg
/gate/fanbeam/vis/setColor blue
/gate/fanbeam/vis/forceWireframe
```

Septal Penetration

If one wants to record, for every photon detected, how many times they crossed the collimator septa, the command recordSeptalPenetration must be turned on (default value is false) and the septal volume name must be attached to a PhantomSD:

```
/gate/output/analysis/recordSeptalPenetration true
/gate/output/analysis/setSeptalVolumeName collimator
```

If the septal volume name does not exist, the simulation is aborted.

OpticalSystem

Description

OpticalSystem is appropriate to model Optical Imaging within GATE. An example macro defining a typical Optical Imaging system can be found in:

https://github.com/OpenGATE/GateContrib/blob/master/imaging/Optical/Optical_System.mac

An OpticalSystem is a box-shaped geometry element and consists of three hierarchic levels:

- **base** which is always attached to the volume OpticalSystem.
- **crystal** which is coupled to the main detector block.
- **pixel** which can be used for modeling a pixelated detector.

Use

Below is part of the Optical Imaging benchmark macro, which is distributed with the GATE softwar:

```
# World
# Define the world dimensions
/gate/world/geometry/setXLength       100. cm
/gate/world/geometry/setYLength       100. cm
/gate/world/geometry/setZLength       100. cm
/gate/world/setMaterial               Air

# Create the Optical Imaging system, which will yield
# a binary output of the projection data
/gate/world/daughters/name                      OpticalSystem
/gate/world/daughters/insert                    box

# Define the dimensions, position and material
/gate/OpticalSystem/geometry/setXLength         10.5 cm
/gate/OpticalSystem/geometry/setYLength         10.5 cm
/gate/OpticalSystem/geometry/setZLength         2.0 cm
/gate/OpticalSystem/placement/setTranslation    0 0 0 cm
```

(continues on next page)

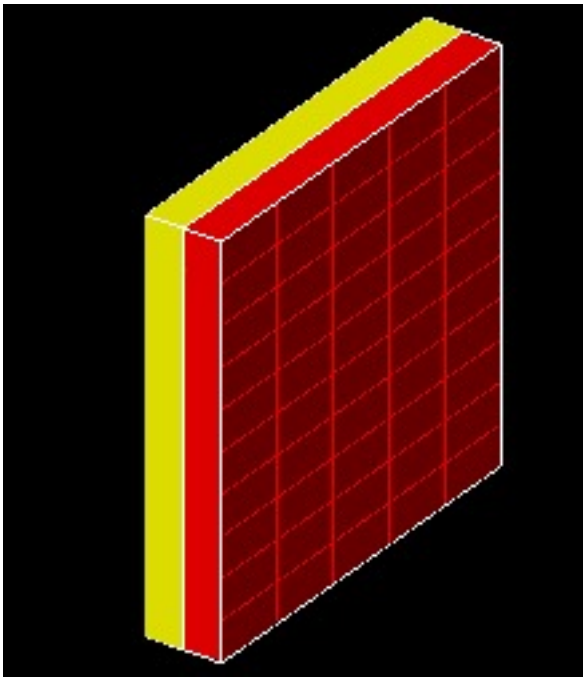


Fig. 3.14: Example of a hypothetical OpticalImaging pixelated system. A pixelated camera is simulated in red. The additional volume in yellow could represent some electronic board.

(continued from previous page)

```
/gate/OpticalSystem/setMaterial                 Air

# Define pixelated detector:
/gate/OpticalSystem/daughters/name              crystal
/gate/OpticalSystem/daughters/insert            box
/gate/crystal/geometry/setXLength               10.5 cm
/gate/crystal/geometry/setYLength               10.5 cm
/gate/crystal/geometry/setZLength               1.0 cm
/gate/crystal/placement/setTranslation          0 0 -0.5 cm
/gate/crystal/setMaterial                       Air
/gate/crystal/vis/forceWireframe                1
/gate/systems/OpticalSystem/crystal/attach      crystal

/gate/crystal/daughters/name                    pixel
/gate/crystal/daughters/insert                  box
/gate/pixel/geometry/setXLength                 2.0  cm
/gate/pixel/geometry/setYLength                 2.0  cm
/gate/pixel/geometry/setZLength                 1.0 cm
/gate/pixel/setMaterial                         Air
/gate/pixel/placement/setTranslation            0 0 0 cm
/gate/pixel/vis/setColor                        red
/gate/pixel/repeaters/insert                    cubicArray
/gate/pixel/cubicArray/setRepeatNumberX         5
/gate/pixel/cubicArray/setRepeatNumberY         5
/gate/pixel/cubicArray/setRepeatNumberZ         1
/gate/pixel/cubicArray/setRepeatVector          2.1 2.1 0 cm
/gate/pixel/vis/forceSolid
/gate/pixel/attachCrystalSD
```

(continues on next page)

(continued from previous page)

```
/gate/systems/OpticalSystem/pixel/attach        pixel

# Define an additional volume behind the pixels
/gate/OpticalSystem/daughters/name              Electronics
/gate/OpticalSystem/daughters/insert            box
/gate/Electronics/geometry/setXLength           10.5  cm
/gate/Electronics/geometry/setYLength           10.5  cm
/gate/Electronics/geometry/setZLength           1.0 cm
/gate/Electronics/setMaterial                   Air
/gate/Electronics/placement/setTranslation      0 0 0.5 cm
/gate/Electronics/vis/setColor                  yellow
/gate/Electronics/vis/forceSolid
```

3.1.5 How to define a multi-system detector

To simulate a multi-system device consisting of several detectors (PET,SPECT,CT,..) you need to add in your macro special commands as explained below. This will allow you to simultaneously register Hits inside every detector.

Defining the systems

The standard definition of a GATE system is done according to the command:

```
/gate/world/daughters/name  SystemName
```

Where SystemName must be one of available system names in GATE (see Table 3.1). Unfortunately, defining a system with this command prevent you to insert more than one system of the same type. Another method has been inserted in GATE to define more than one system at a time. Using this more general method, users can simulate several systems simultaneously. A system is now defined by its own name and its type according to the next two command lines:

```
/gate/world/daughters/name AnyName
/gate/world/daughters/systemType SystemType
```

Where AnyName can be any name as for any Geant4 volume name and SystemType must be one the names of GATE systems mentioned in Table 3.1.

How to connect the geometry to the systems

By using more than one system, we have to change the attachment commands to connect the geometrical elements of every system with its defined components, to do that we use the next command:

```
/gate/systems/SystemName/Level/attach UserVolumeName
```

This command has the same form as for one system, but the essential difference is that SystemName here can be any name gave by the user to the system.

Example of multi-system

An example for the creation of three systems, one system of type “cylindricalPET” and two systems of type “scanner” is explained below. Note that it is not necessary to use the “ systemType” command for cylindricalPET system because there is only one system of this type:

```
# W O R L D
/gate/world/geometry/setXLength 40. cm
/gate/world/geometry/setYLength 40. cm
/gate/world/geometry/setZLength 60. cm

# M O U S E
/gate/world/daughters/name mouse
/gate/world/daughters/insert cylinder
/gate/mouse/setMaterial Water
/gate/mouse/vis/setColor red
/gate/mouse/geometry/setRmax 18.5 mm
/gate/mouse/geometry/setRmin 0. mm
/gate/mouse/geometry/setHeight 68. mm

# SYSTEM 1: cylindricalPET
/gate/world/daughters/name cylindricalPET   # standard definition
/gate/world/daughters/insert cylinder
/gate/cylindricalPET/setMaterial Water
/gate/cylindricalPET/geometry/setRmax 145 mm
/gate/cylindricalPET/geometry/setRmin 130 mm
/gate/cylindricalPET/geometry/setHeight 80 mm
/gate/cylindricalPET/vis/forceWireframe

#cylindricalPET => rsector
/gate/cylindricalPET/daughters/name rsector
/gate/cylindricalPET/daughters/insert box
/gate/rsector/placement/setTranslation 135 0 0 mm
/gate/rsector/geometry/setXLength 10. mm
/gate/rsector/geometry/setYLength 19. mm
/gate/rsector/geometry/setZLength 76.6 mm
/gate/rsector/setMaterial Water
/gate/rsector/vis/forceWireframe

#cylindricalPET => module
/gate/rsector/daughters/name module
/gate/rsector/daughters/insert box
/gate/module/geometry/setXLength 10. mm
/gate/module/geometry/setYLength 19. mm
/gate/module/geometry/setZLength 19. mm
/gate/module/setMaterial Water
/gate/module/vis/forceWireframe
/gate/module/vis/setColor gray

#cylindricalPET => crystal
/gate/module/daughters/name crystal
/gate/module/daughters/insert box
/gate/crystal/geometry/setXLength 10. mm
/gate/crystal/geometry/setYLength 2.2 mm
/gate/crystal/geometry/setZLength 2.2 mm
/gate/crystal/setMaterial Water
/gate/crystal/vis/forceWireframe
/gate/crystal/vis/setColor magenta

#cylindricalPET => LSO
/gate/crystal/daughters/name LSO
/gate/crystal/daughters/insert box
/gate/LSO/geometry/setXLength 10. mm
```

(continues on next page)

(continued from previous page)

```
/gate/LSO/geometry/setYLength 2.2 mm
/gate/LSO/geometry/setZLength 2.2 mm
/gate/LSO/placement/setTranslation 0 0 0 mm
/gate/LSO/setMaterial LSO
/gate/LSO/vis/setColor yellow

# R E P E A T C R Y S T A L
/gate/crystal/repeaters/insert cubicArray
/gate/crystal/cubicArray/setRepeatNumberX 1
/gate/crystal/cubicArray/setRepeatNumberY 8
/gate/crystal/cubicArray/setRepeatNumberZ 8
/gate/crystal/cubicArray/setRepeatVector 10. 2.4 2.4 mm

# R E P E A T M O D U L E
/gate/module/repeaters/insert cubicArray
/gate/module/cubicArray/setRepeatNumberZ 4
/gate/module/cubicArray/setRepeatVector 0. 0. 19.2 mm

# R E P E A T R S E C T O R
/gate/rsector/repeaters/insert ring
/gate/rsector/ring/setRepeatNumber 42


# SYSTEM 2: Scanner_1
/gate/world/daughters/name Scanner_1       #  System name  definition for scanner 1
/gate/world/daughters/systemType scanner   #  System type  definition for scanner 1
/gate/world/daughters/insert box
/gate/Scanner_1/setMaterial Air
/gate/Scanner_1/geometry/setXLength 20  cm
/gate/Scanner_1/geometry/setYLength 20 cm
/gate/Scanner_1/geometry/setZLength 10 cm
/gate/Scanner_1/placement/setTranslation  0 0 -18 cm
/gate/Scanner_1/vis/forceWireframe
/gate/Scanner_1/vis/setVisible 1

# Scanner_1 => Cryostat_1
/gate/Scanner_1/daughters/name Cryostat_1
/gate/Scanner_1/daughters/insert box
/gate/Cryostat_1/placement/setTranslation  0 0 0 cm
/gate/Cryostat_1/geometry/setXLength 19  cm
/gate/Cryostat_1/geometry/setYLength 19  cm
/gate/Cryostat_1/geometry/setZLength 9   cm
/gate/Cryostat_1/setMaterial Stainless
/gate/Cryostat_1/vis/setColor yellow
/gate/Cryostat_1/vis/forceWireframe

# Scanner_1 => ActiveZone_1
/gate/Cryostat_1/daughters/name ActiveZone_1
/gate/Cryostat_1/daughters/insert box
/gate/ActiveZone_1/placement/setTranslation 0 0 0 cm
/gate/ActiveZone_1/geometry/setXLength 18   cm
/gate/ActiveZone_1/geometry/setYLength 18   cm
/gate/ActiveZone_1/geometry/setZLength 8    cm
/gate/ActiveZone_1/setMaterial LXenon
/gate/ActiveZone_1/vis/setColor white
```

(continues on next page)

(continued from previous page)

```
# SYSTEM 3: Scanner_2
/gate/world/daughters/name Scanner_2      #  System name  definition for scanner 2
/gate/world/daughters/systemType scanner  #  System Type  definition for scanner 2

/gate/world/daughters/insert box
/gate/Scanner_2/setMaterial Air
/gate/Scanner_2/geometry/setXLength 20  cm
/gate/Scanner_2/geometry/setYLength 20 cm
/gate/Scanner_2/geometry/setZLength 10 cm
/gate/Scanner_2/placement/setTranslation  0 0 18 cm
/gate/Scanner_2/vis/forceWireframe
/gate/Scanner_2/vis/setVisible 1

# Scanner_2 => Cryostat_2
/gate/Scanner_2/daughters/name Cryostat_2
/gate/Scanner_2/daughters/insert box
/gate/Cryostat_2/placement/setTranslation  0 0 0 cm
/gate/Cryostat_2/geometry/setXLength 19  cm
/gate/Cryostat_2/geometry/setYLength 19  cm
/gate/Cryostat_2/geometry/setZLength 9   cm
/gate/Cryostat_2/setMaterial Stainless
/gate/Cryostat_2/vis/setColor yellow
/gate/Cryostat_2/vis/forceWireframe

# Scanner_2 => ActiveZone_2
/gate/Cryostat_2/daughters/name ActiveZone_2
/gate/Cryostat_2/daughters/insert box
/gate/ActiveZone_2/placement/setTranslation 0 0 0 cm
/gate/ActiveZone_2/geometry/setXLength 18   cm
/gate/ActiveZone_2/geometry/setYLength 18   cm
/gate/ActiveZone_2/geometry/setZLength 8    cm
/gate/ActiveZone_2/setMaterial LXenon
/gate/ActiveZone_2/vis/setColor white


# A T T A C H S Y S T E M S
/gate/systems/cylindricalPET/rsector/attach rsector
/gate/systems/cylindricalPET/module/attach module
/gate/systems/cylindricalPET/crystal/attach crystal
/gate/systems/cylindricalPET/layer0/attach LSO

# New attachment commands
/gate/systems/Scanner_1/level1/attach Cryostat_1
/gate/systems/Scanner_1/level2/attach ActiveZone_1

/gate/systems/Scanner_2/level1/attach Cryostat_2
/gate/systems/Scanner_2/level2/attach ActiveZone_2


# A T T A C H L A Y E R SD
/gate/LSO/attachCrystalSD
/gate/mouse/attachPhantomSD

/gate/ActiveZone_1/attachCrystalSD

/gate/ActiveZone_2/attachCrystalSD
```

Notes

1) The command "systemType" is optional in case of using only one system, but the system name must be one GATE systems (first column in Table 3.1) as for standard definition
2) Same remark, in the case where all systems have different types.
3) In general, one has to use the "systemType" command only for simulating more than one system of the same type.

3.2 Attaching the sensitive detectors

Table of Contents

3.2.1 General purpose

Once a model has been defined for the scanner through the construction of a system (see *Defining a system*), the next step is to attach a **sensitive detector** (SD) to some volumes of the geometry. As in any Geant4 simulation, these sensitive detectors are used to store information regarding interactions of a particle in the matter (*hits*) using information from the steps occuring along the particle track. A hit is a snapshot of a physical interaction of a track in a sensitive region of the detector. Fig. 3.15 illustrates these notions. Hits contain various pieces of information associated to a step object, such as the energy deposition of a step, geometrical information, position and time of a step, etc.

GATE records and stores information related to the hits only for those volumes that are attached to a sensitive detector. All information regarding the interactions occuring in *non-sensitive* volumes is lost.

The crystalSD

The *crystalSD* is used to record information regarding interactions taking place inside the volumes belonging to a scanner for instance (crystals or collimators) : energy deposition, positions of interaction, origin of the particle (emission vertex), type of interaction (name of the physical processes involved), etc.

A *crystalSD* can be attached only to those volumes that belong to a given system. Once a *crystalSD* has been attached, it is considered as attached to this system. This sensitive detector can be attached using the command **attachCrystalSD**. These volumes are essentially meant to be scintillating elements (crystals) but can also be attached to non-scintillating elements such as collimators, shields or septa.

Below is an example of command lines that should be included in a macro using the *crystalSD*. These command lines must be inserted after the description of the attachment to the system:

The first command is used to attach the scintillation crystal to the detection level *crystal* of the SPECThead system:

```
/systems/SPECThead/crystal/attach crystal
```

Then, the second command attaches the crystalSD to the volume representing the scintillation crystal in the geometry:

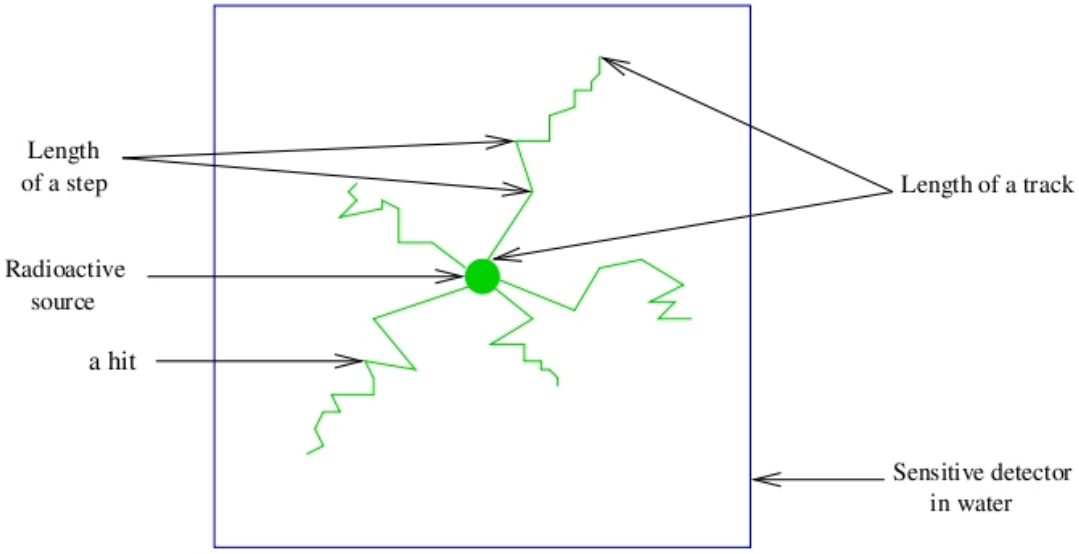


Fig. 3.15: Particle interactions in a sensitive detector

```
/gate/crystal/attachCrystalSD
```

The phantomSD

The *phantomSD* plays a crucial role in GATE simulations, as it is used to record information regarding Compton and Rayleigh interactions taking place in the volumes before the detection in the scanner system (e.g., for a SPECT camera, these volumes can be the table, phantom, and collimator, in which it can be relevant to retrieve information about Compton and Rayleigh interactions). These data can then be used to estimate whether a photon reaching a detector is a direct or a Compton-scattered photon. Thus, in PET, the *phantomSD* is currently the only way to discriminate scattered from true coincidences. To simulate low energy X-ray acquisitions (for example mammography acquisitions from 7 to 28 keV), information concerning Rayleigh interactions is significant.

Using this type of sensitive detector, it is possible to retrieve two pieces of information related to the hits:

- The number of Compton and Rayleigh interactions occuring in all the volumes attached to the *phantomSD* : **nPhantomCompton** and **nPhantomRayleigh**. These pieces of information are also available for the *crystalSD* with the variables **nCrystalCompton** and **nCrystalRayleigh**. It is impossible to attach both sensitive detectors to the same volume.
- The last volume attached to the *phantomSD* in which a Compton or a Rayleigh interaction occured: **compVolName** and **RayleighVolName**.

IMPORTANT: To retrieve data output information regarding hits occuring in the *phantomSD* (nPhantomCompton and compVolName), a *crystalSD* has to be defined in the simulation. Otherwise, data output variables will be created but will be empty. When all these conditions are satisfied, any interaction taking place within the field of view (FOV) of the scanner is automatically recorded by the *phantomSD*, so that the number of Compton interactions for each photon can be accurately computed.

This procedure does not take into account Compton interactions taking place within the detectors, so that inter-crystal cross-talk via Compton interactions is not detected. Here is an example of command-lines that should be included within the macro in order to use the *phantomSD*. These command lines must be inserted after the description of the

attachment to the system. First, commands are used to attach the scattering volumes to the detection level *base* of the SPECThead system:

```
/systems/SPECThead/base/attach FOV
/systems/SPECThead/base/attach head
/systems/SPECThead/base/attach body
```

Then, we can attach the phantomSD to the volumes representing the scattering volumes in the geometry:

```
/FOV/attachPhantomSD
/head/attachPhantomSD
/body/attachPhantomSD
```

Finally, the last commands are used to attach the scintillation crystal to the detection level *crystal* of the SPECThead system and to attach the crystalSD to the volume representing the scintillation crystal in the geometry:

```
/systems/SPECThead/crystal/attachCrystalSD
/gate/crystal/attachCrystalSD
```

In the case of a voxellized matrix: Previous commands to attach sensitive detectors are used for the volumes created using the geometry commands of GATE (see *Defining a geometry*). In order to record the same information concerning the interactions occuring in a voxellized matrix, see *Voxelized source and phantom*.

3.3 Digitizer and readout parameters

Table of Contents

3.3.1 General Purpose

The purpose of the digitizer module is to simulate the behaviour of the scanner detectors and signal processing chain.

From particle detection to coincidences in GATE

GATE uses Geant4 to generate particles and transport them through the materials. This mimics *physical* interactions between particles and matter. The information generated during this process is used by GATE to simulate the detector pulses (*digits*), which correspond to the *observed data*. The digitizer represents the series of steps and filters that make up this process.

The typical data-flow for an event is as follows:

- A particle is generated, with its parameters, such as initial type, time, momentum, and energy.
- An elementary trajectory step is applied. A step corresponds to the trajectory of a particle between discrete interactions (i.e. photoelectric, Compton, pair production, etc). During a step, the changes to particle's energy and momentum are calculated. The length of a step depends upon the nature of the interaction, the type of particle and material, etc. The calculation of the step length is complex and is mentioned here only briefly. For more details, please refer to the Geant4 documentation.
- If a step occurs within a volume corresponding to a *sensitive* detector, the interaction information between the particle and the material is stored. For example, this information may include the deposited energy, the momentum before and after the interaction, the name of the volume where the interaction occurred, etc. This set of information is referred to as a *Hit*.
- Steps 2 and 3 are repeated until the energy of the particle becomes lower than a predefined value, or the particle position goes outside the predefined limits. The entire series of steps form a simulated trajectory of a particle, that is called a *Track* in Geant4.
- The amount of energy deposited in a crystal is filtered by the digitizer module. The output from the digitizer corresponds to the signal after it has been processed by the Front End Electronics (FEE). Generally, the FEE is made of several processing units, working in a serial and/or in parallel. This process of transforming the energy of a *Hit* into the final digital value is called *Digitization* and is performed by the GATE digitizer. Each processing unit in the FEE is represented in GATE by a corresponding digitizer module. The final value obtained after filtering by a set of these modules is called a *Single*. *Singles* can be saved as output. Each transient value, between two modules, is called a *Pulse*.

This process is repeated for each event in the simulation in order to produce one or more sets of Singles. These *Singles* can be stored into an output file (as a ROOT tree, for example).

In case of PET systems, a second processing stage can be inserted to sort the *Singles* list for coincidences. To do this, the algorithm searches in this list for a set of *Singles* that are detected within a given time interval (the so called 'coincident events').

Finally, the coincidence data may be filtered-out to mimic any possible data loss which could occur in the coincidence logical circuit or during the data transportation. As for the *Singles*, the processing is performed by specifying a list of generic modules to apply to the coincidence data flow.

Definition of a hit in Geant4

A hit is a snapshot of the physical interaction of a track within a sensitive region of a detector. The information given by a hit is

- Position and time of the step
- Momentum and energy of the track
- Energy deposition of the step
- Interaction type of the hit
- Volume name containing the hit

As a result, the history of a particle is saved as a series of *hits* generated along the particles trajectory. In addition to the physical hits, Geant4 saves a special *hit*. This *hit* takes place when a particle moves from one volume to another (this type of *hit* deposits zero energy). The *hit* data represents the basic information that a user has with which to construct the physically observable behaviour of a scanner. To see the information stored in a *hit*, see the file *GateCrystalHit.hh*.

Role of the digitizer

As mentioned above, the information contained in the *hit* does not correspond to what is provided by a real detector. To simulate the digital values (*pulses*) that result from the output of the Front End Electronics, the sampling methods of the signal must be specified. To do this, a number of digitizer modules are available and are described below.

The role of *the digitizer* is to build, from the *hit* information, the physical observables, which include energy, position, and time of detection for each particle. In addition, the digitizer must implement the required logic to simulate coincidences during PET simulations. Typical usage of digitizer module includes the following actions:

- simulate detector response
- simulate readout scheme
- simulate trigger logic

These actions are accomplished by inserting *digitizer* modules into GATE, as explained in the next sections.

Disabling the digitizer

If you want to disable the digitizer process and all output (that are already disabled by default), you can use the following commands:

```
/gate/output/analysis/disable
/gate/output/digi/disable
```

3.3.2 Digitizer modules

The digitization consists of a series of signal processors. The output at each step along the series is defined as a *pulse*. At the end of the chain, the output *pulses* are named *singles*. These *Singles* realistically simulate the physical observables of a detector response to a particle interacting with it. An example is shown in Fig. 3.16.

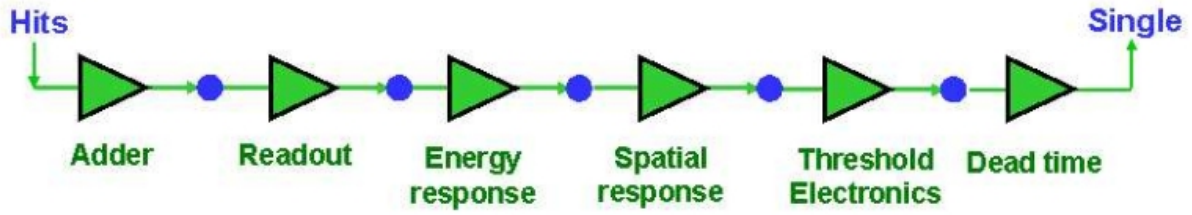


Fig. 3.16: The digitizer is organized as a chain of modules that begins with the hit and ends with the single which represents the physical observable seen from the detector.

To specify a new signal-processing module (i.e. add a new processing unit in the readout scheme) the following command template should be used:

```
/gate/digitizer/insert MODULE
```

where **MODULE** is the name of the digitizer module. The order of the module declaration should make sense. The data flow follows the same order as the module declaration in the macro. In a typical scanner, the following sequence works well, athough it is not mandatory (the module names will be explained in the rest of the section):

- insert adder before readout
- insert readout before thresholder/upholder
- insert blurring before thresholder/upholder

Distributions

Since many of the modules presented below have to deal with functions or probability density, a generic tool is provided to describe such mathematical objects in GATE. Basically, a distribution in GATE is defined by its name, its type (Gaussian, Exponential, etc...) and the parameters specifics to each distribution type (such as the mean and the standard deviation of a Gaussian function). Depending on the context, these objects are used directly as functions, or as probability densities into which a variable is randomly chosen. In the following, the generic term of distribution will be used to describe both of these objects, since their declaration is unified under this term into GATE.

Five types of distribution are available in GATE, namely:

- Flat distributions, defined by the range into which the function is not null, and the value taken within this range.
- Gaussian distributions, defined by a mean value and a standard deviation.
- Exponential distributions, defined by its power.
- Manual distributions, defined by a discrete set of points specified in the GATE macro file. The data are linearly interpolated to define the function in a continuous range.
- File distribution, acting as the manual distribution, but where the points are defined in a separate ASCII file, whose name is given as a parameter. This method is appropriate for large numbers of points and allows to describe any distribution in a totally generic way.

A distribution is declared by specifying its name then by creating a new instance, with its type name:

```
/gate/distributions/name my_distrib
/gate/distributions/insert Gaussian
```

The possible type name available corresponds to the five distributions described above, that is *Flat*, *Gaussian*, *Exponential*, *Manual* or *File*. Once the distribution is created (for example a Gaussian), the related parameters can be set:

```
/gate/distributions/my_distrib/setMean 350 keV
/gate/distributions/my_distrib/setSigma 30 keV
```

Table 3.3: Summary of the parameters for each distribution type

Parameter name	Description
FLAT DISTRIBUTION	
setMin	set the low edge of the range where the function is not null (default is 0)
setMax	set the high edge of the range where the function is not null (default is 1)
setAmplitude	set the value taken by the function within the non null range (default is 1)
GAUSSIAN DISTRIBUTION	
setMean	set the mean value of the distribution (default is 0)
setSigma	set the standard deviation of the distribution (default is 1)
setAmplitude	set the amplitude of the distribution (default is 1)
EXPONENTIAL DISTRIBUTION	
setLambda	set the power of the distribution (default is 1)
setAmplitude	set the amplitude of the distribution (default is 1)
MANUAL DISTRIBUTION	
setUnitX	set the unit for the x axis
setUnitY	set the unit for the y axis
insertPoint	insert a new point, giving a pair of (x,y) values
addPoint	add a new point, giving its y value, and auto incrementing the x value
autoXstart	in case of auto incremental x value, set the first x value to use
FILE DISTRIBUTION	
setUnitX	set the unit for the x axis
setUnitY	set the unit for the y axis
autoX	specify if the x values are read from file or if they are auto-incremented
autoXstart	in case of auto incremental x value, set the first x value to use
setFileName	the name of the ASCII file where the data have to be read
setColumnX	which column of the ASCII file contains the x axis data
setColumnY	which column of the ASCII file contains the y axis data
read	do read the file (should be called after specifying all the other parameters)

Adder

One particle often creates multiple interactions, and consequently multiple *hits*, within a crystal. The first step of the digitizer is to sum all the *hits* that occur within the same crystal (i.e. the same volume). This is due to the fact that the electronics always measure an integrated signal, and do not have the time or energy resolution necessary to distinguish between the individual interactions of the particle within a crystal. This digitizer action is completed by a module called the adder. The adder should be the first module of a digitizer chain. It acts on the lowest level in the system hierarchy, as explained in *Defining a system*:

- A system must be used to describe the geometry (also the mother volume name must corresponds to a system name)
- The lowest level of this system must be attached to the detector volume and must be declared as a *sensitive detector*

The adder regroups *hits* per volume into a *pulse*. If one particle that enters a detector makes multiple *hits* within two different crystal volumes before being stopped, the output of the adder module will consists of two *pulses*. Each *pulse* is computed as follows : the energy is taken to be the total of energies in each volume, the position is obtained with an energy-weighted centroid of the different *hit* positions. The time is equal to the time at which the first *hit* occured.

The command to use the adder module is:

```
/gate/digitizer/Singles/insert adder
```

Adder Compton

The adderCompton module has a different behavior than the classic adder, which performs an energy-weighted centroid addition of all electronic and photonic hits. Instead, for each electronic energy deposition, the energy is added to the previous photonic hit in the same volume ID (or discarded if none), but the localization remains that of the photonic interaction. That way, the Compton kinematics becomes exact for photonic interations, enabling further studies. The user must use the classic adder afterwards, to handle multiple photonic interactions in the same crystal. The commands to use the adder module are:

```
/gate/digitizer/Singles/insert adderCompton
/gate/digitizer/Singles/insert adder
```

Readout

With the exception of a detector system where each crystal is read by an individual photo-detector, the readout segmentation is often different from the basic geometrical structures of the detector. The readout geometry is an artificial geometry that is usually associated with a group of sensitive detectors. There are two ways of modelling this readout process : either a winner-takes-all approach that will somewhat model APD-like readout, or a energy-centroid approach that will be closer to the block-PMT readout. Using the winner-takes-all policy, the grouping has to be determined by the user through a variable named *depth* corresponding to the component in the volume hierarchy at which pulses are summed together. Using this variable, the *pulses* are summed if their volume ID's are identical to this level of depth. Using the energy-centroid policy, the depth of the grouping is forced to occur at the 'crystal' level whatever the system used, so the depth variable is ignored. This means that the pulses in a same level just above the crystal level are summed together.

The readout module regroups pulses per block (group of *sensitive detectors*). For both policy, the resulting pulse in the block has the total energy of all pulses summed together. For the winner-takes-all policy, the position of the pulse is the one with the maximum energy. For the energy-centroid policy, the position is determined by weighting the crystal indices of each pulse by the deposited energy in order to get the energy centroid position. In this case, only the crystal index is determined, and the actual cartesian coordinates of the resulting pulse are reset to the center of this crystal. If a sub-level of the crystal is used (different layers), then the final sub-level is determined by the one having the maximum energy deposited (so a winner-takes-all approach for these sub-levels of the crystal is used):

```
/gate/digitizer/Singles/insert readout
/gate/digitizer/Singles/readout/setPolicy myPolicy
/gate/digitizer/Singles/readout/setDepth X
```

The parameter *myPolicy* can be *TakeEnergyWinner* for the winner-takes-all policy or *TakeEnergyCentroid* for the energy centroid policy. If the energy centroid policy is used, the depth is forced to be at the level just above the crystal level, whatever the system used. If the winner-takes-all policy is used, then the user must choose the *depth* at which the readout process takes place. If the *setPolicy* command is not set, then the winner-takes-all policy is chosen by default in order to be back-compatible with previous Gate releases.

Fig. 3.17 illustrates the actions of both the *adder* and *readout* modules. The *adder* module transforms the *hits* into a *pulse* in each individual volume and then the *readout* module sums a group of these *pulses* into a single *pulse* at the level of depth as defined by the user for the winner-takes-all policy.

The importance of the *setDepth* command line when using the winner-takes-all policy is illustrated through the following example from a PET system (see *Defining a system*). In a *cylindricalPET* system, where the first volume level is *rsector*, and the second volume level is *module*, as shown in Fig. 3.18, the *readout depth* depends upon how the electronic readout functions.

If one PMT reads the four modules in the axial direction, the *depth* should be set with the command:

```
/gate/digitizer/Singles/readout/setDepth 1
```

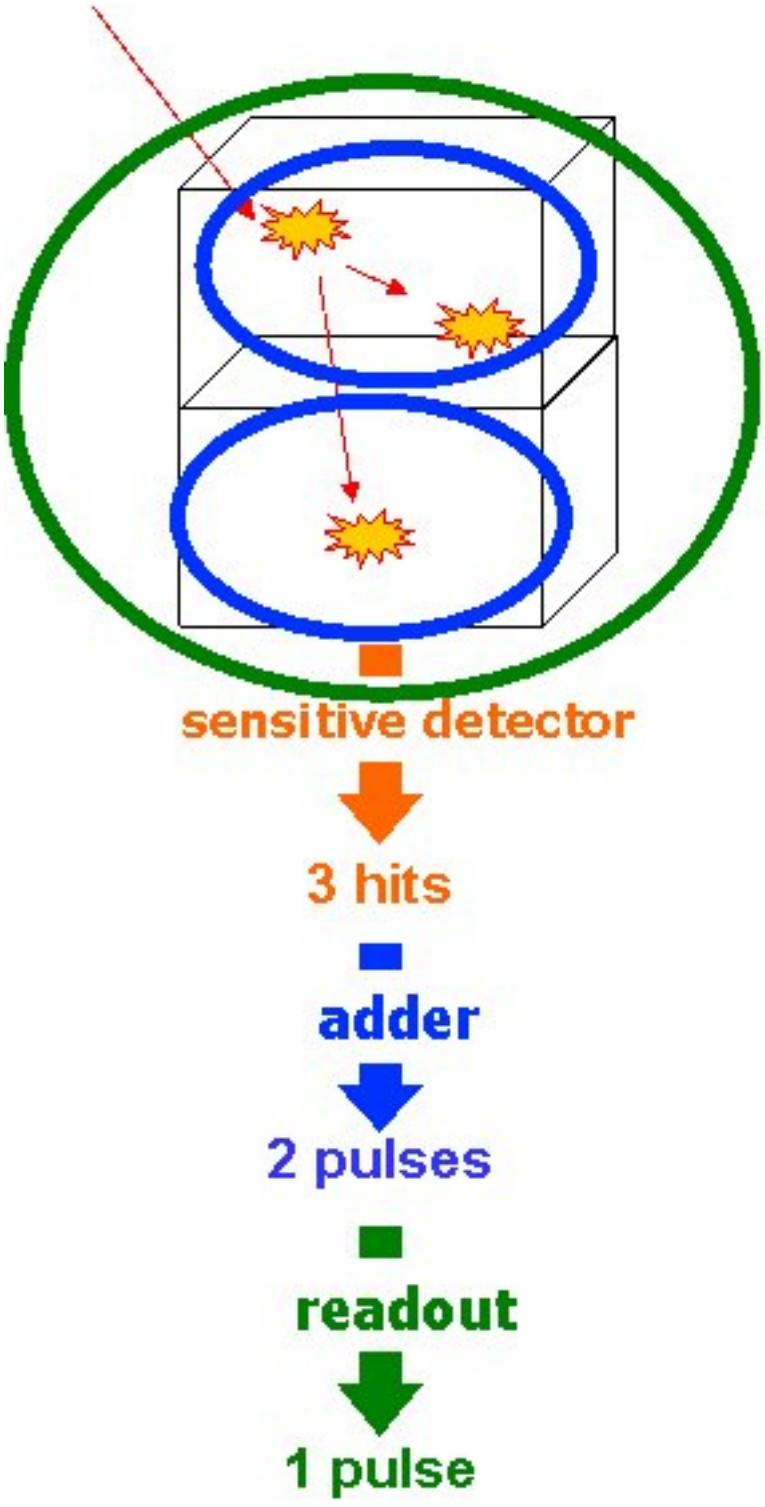


Fig. 3.17: Actions of the *it adder* and *it readout* modules

The energy of this *single* event is the sum of the energy of the pulses inside the white rectangle (*rsector*) of Fig. 3.18. However, if individual PMTs read each module (group of crystals), the *depth* should be set with the command:

```
/gate/digitizer/Singles/readout/setDepth 2
```

In this case, the energy of the *single* event is the sum of the energies of the pulses inside the red box (*module*) of Fig. 3.18.

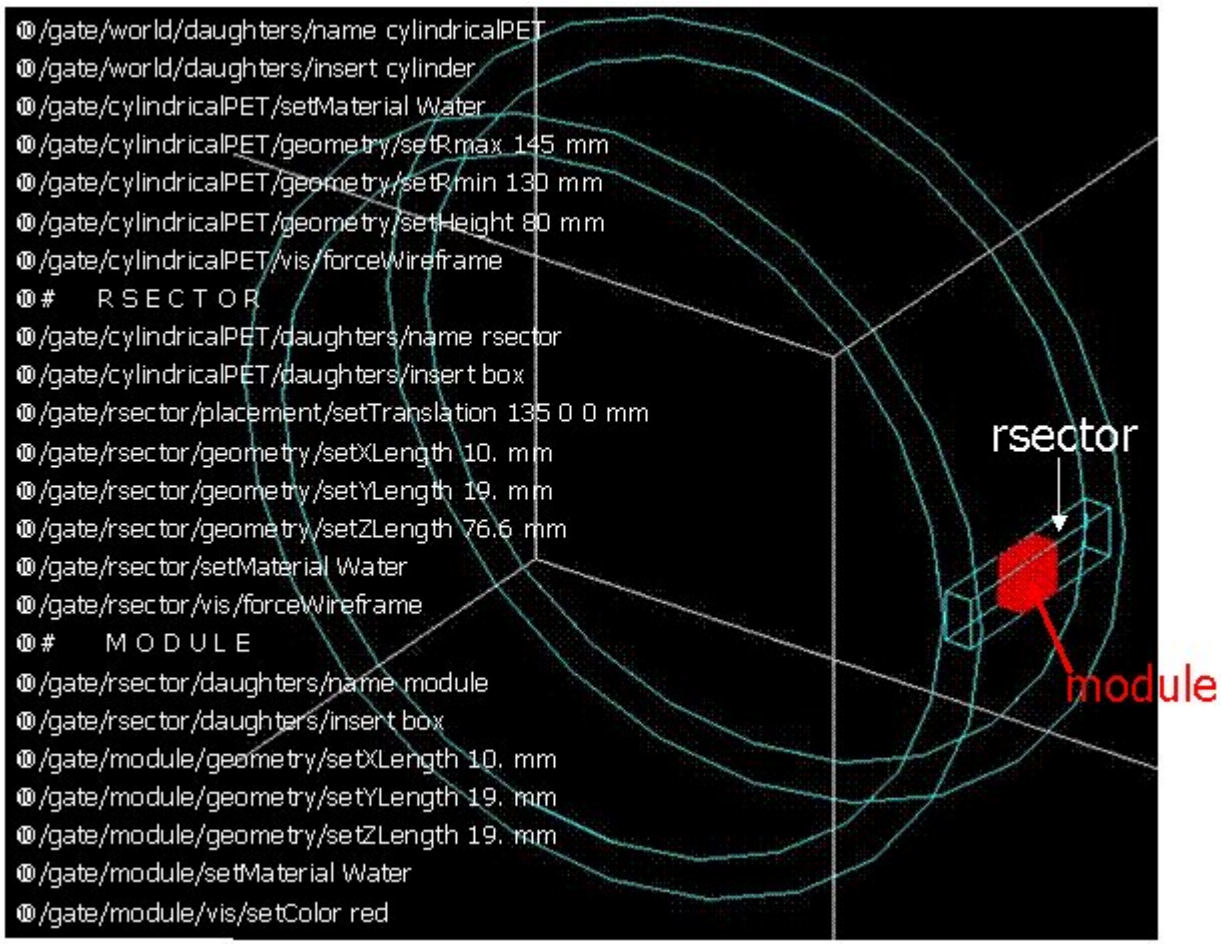


Fig. 3.18: Setting the *readout depth* in a CylindricalPET system

The next task is to transform this output *pulse* from the readout module into a *single* which is the physical observable of the experiment. This transformation is the result of the detector response and should mimic the behaviors of the photo-detector, electronics, and acquisition system.

Blurring : Energy blurring

The *blurring* pulse-processor module simulates Gaussian blurring of the energy spectrum of a pulse after the *readout* module. This is accomplished by introducing a resolution, R_0 (FWHM), at a given energy, E_0. According to the camera, the energy resolution may follow different laws, such as an inverse square law or a linear law.

For inverse square law ($R = R_0 \frac{\sqrt{E_0}}{\sqrt{E}}$), one must specify the inverse square law and fix the attributes like the energy of reference and the resolution (example of a 15% resolution @ 511 KeV):

```
/gate/digitizer/Singles/blurring
/gate/digitizer/Singles/blurring/setLaw inverseSquare
/gate/digitizer/Singles/blurring/inverseSquare/setResolution 0.15
/gate/digitizer/Singles/blurring/inverseSquare/setEnergyOfReference 511. keV
```

For linear law, one must specify the linear law and fix the attributes like the energy of reference, the resolution and the slope:

```
/gate/digitizer/Singles/blurring
/gate/digitizer/Singles/blurring/setLaw linear
/gate/digitizer/Singles/blurring/linear/setResolution 0.15
/gate/digitizer/Singles/blurring/linear/setEnergyOfReference 511. keV
/gate/digitizer/Singles/blurring/linear/setSlope -0.055 1/MeV
```

Blurring : crystal blurring

This type of blurring is used for the scanners where all the detectors are made of the same type of crystal. In this case, it is often useful to assign a different energy resolution for each crystal in the detector block, between a minimum and a maximum value. To model the efficiency of the system, a coefficient (between 0 and 1) can also be set.

As an example, a random blurring of all the crystals between 15% and 35% at a reference energy of 511 keV, and with a quantum efficiency of 90% can be modelled using the following commands:

```
/gate/digitizer/Singles/insert crystalblurring
/gate/digitizer/Singles/crystalblurring/setCrystalResolutionMin 0.15
/gate/digitizer/Singles/crystalblurring/setCrystalResolutionMax 0.35
/gate/digitizer/Singles/crystalblurring/setCrystalQE 0.9
/gate/digitizer/Singles/crystalblurring/setCrystalEnergyOfReference 511.keV
```

In this example, for each interaction the program randomly chooses a crystal resolution between 0.15 and 0.35. The crystals are not assigned a constant resolution. The crystal quantum efficiency is set using **setCrystalQE** and represents the probability for the event to be detected by the photo-detector. This parameter represents the effect of the transfer efficiency of the crystal and of the quantum efficiency of the photo-detector.

Blurring : Local energy blurring for different crystals

The LocalBlurring module is very similar to the energy *blurring* module, but different energy resolutions are applied to different volumes. This type of blurring is useful for detectors with several layers of different scintillation crystals (e.g. depth of interaction measurement with a phoswich module in a CylindricalPET system).

If a detector has a resolution of 15.3% @ 511 KeV for a crystal called crystal1 and has a resolution of 24.7% @ 511 KeV for another crystal (crystal2) in a phoswich configuration, the following commands should be used:

```
/gate/digitizer/Singles/insert localBlurring
/gate/digitizer/Singles/localBlurring/chooseNewVolume crystal1
/gate/digitizer/Singles/localBlurring/crystal1/setResolution 0.153
/gate/digitizer/Singles/localBlurring/crystal1/setEnergyOfReference 511 keV
/gate/digitizer/Singles/localBlurring/chooseNewVolume crystal2
/gate/digitizer/Singles/localBlurring/crystal2/setResolution 0.247
/gate/digitizer/Singles/localBlurring/crystal2/setEnergyOfReference 511 keV
```

BEWARE: crystal1 and crystal2 must be valid *Sensitive Detector* volume names !!

Blurring: Intrinsic resolution blurring with crystals of different compositions

This blurring pulse-processor simulates a local Gaussian blurring of the energy spectrum (different for different crystals) based on the following model:

$R = \sqrt{2.35^2 \cdot \frac{1+\bar{\nu}}{N_{ph} \cdot \bar{\epsilon} \cdot \bar{p}} + {R_i}^2}$

where $N_{ph} = LY \cdot E$ and LY, $\bar{p}$ and $\bar{\epsilon}$, are the Light Yield, Transfer, and Quantum Efficiency for each crystal.

$\bar{\nu}$ is the relative variance of the gain of a Photo Multiplier Tube (PMT) or of an Avalanche Photo Diode (APD). It is hard-codded and set to 0.1.

If the intrinsic resolutions, (R_i), of the individual crystals are not defined, then they are set to one.

To use this *digitizer* module properly, several modules must be set first. These digitizer modules are **GateLightYield**, **GateTransferEfficiency**, and **GateQuantumEfficiency**. The light yield pulse-processor simulates the crystal light yield. Each crystal must be given the correct light yield. This module converts the *pulse* energy into the number of scintillation photons emitted, N_{ph}. The transfer efficiency pulse-processor simulates the transfer efficiencies of the light photons in each crystal. This digitizer reduces the "pulse" energy (by reducing the number of scintillation photons) by a transfer efficiency coefficient which must be a number between 0 and 1. The quantum efficiency pulse-processor simulates the quantum efficiency for each channel of a photo-detector, which can be a Photo Multiplier Tube (PMT) or an Avalanche Photo Diode (APD).

The command lines are illustrated using an example of a phoswich module made of two layers of different crystals. One crystal has a light yield of 27000 photons per MeV (LSO crystal), a transfer efficiency of 28%, and an intrinsic resolution of 8.8%. The other crystal has a light yield of 8500 photons per MeV (LuYAP crystal), a transfer efficiency of 24% and an intrinsic resolution of 5.3%

In the case of a *cylindricalPET* system, the construction of the crystal geometry is truncated for clarity (the truncation is denoted by ...). The *digitizer* command lines are:

```
# LSO layer
/gate/crystal/daughters/name LSOlayer ....

# BGO layer
/gate/crystal/daughters/name LuYAPlayer ....

# A T T A C H S Y S T E M ....
/gate/systems/cylindricalPET/crystal/attach crystal
/gate/systems/cylindricalPET/layer0/attach LSOlayer
/gate/systems/cylindricalPET/layer1/attach LuYAPlayer

# A T T A C H C R Y S T A L S D
/gate/LSOlayer/attachCrystalSD
/gate/LuYAPlayer/attachCrystalSD

# In this example the phoswich module is represented by the *crystal* volume and is␣
↪made of two different material layers.
# To apply the resolution blurring of equation , the parameters discussed above must␣
↪be defined for each layer
#(i.e. Light Yield, Transfer, Intrinsic Resolution, and the Quantum Efficiency).
# DEFINE TRANSFER EFFICIENCY FOR EACH LAYER
/gate/digitizer/Singles/insert transferEfficiency
/gate/digitizer/Singles/transferEfficiency/chooseNewVolume LSOlayer
/gate/digitizer/Singles/transferEfficiency/LSOlayer/setTECoef 0.28
/gate/digitizer/Singles/transferEfficiency/chooseNewVolume LuYAPlayer
/gate/digitizer/Singles/transferEfficiency/LuYAPlayer/setTECoef 0.24

# DEFINE LIGHT YIELD FOR EACH LAYER
```

(continues on next page)

(continued from previous page)

```
/gate/digitizer/Singles/insert lightYield
/gate/digitizer/Singles/lightYield/chooseNewVolume LSOlayer
/gate/digitizer/Singles/lightYield/LSOlayer/setLightOutput 27000
/gate/digitizer/Singles/lightYield/chooseNewVolume LuYAPlayer
/gate/digitizer/Singles/lightYield/LuYAPlayer/setLightOutput 8500

# DEFINE INTRINSIC RESOLUTION FOR EACH LAYER
/gate/digitizer/Singles/insert intrinsicResolutionBlurring
/gate/digitizer/Singles/intrinsicResolutionBlurring/ chooseNewVolume LSOlayer
/gate/digitizer/Singles/intrinsicResolutionBlurring/ LSOlayer/setIntrinsicResolution␣
↪0.088
/gate/digitizer/Singles/intrinsicResolutionBlurring/ LSOlayer/setEnergyOfReference␣
↪511 keV
/gate/digitizer/Singles/intrinsicResolutionBlurring/ chooseNewVolume LuYAPlayer
/gate/digitizer/Singles/intrinsicResolutionBlurring/ LuYAPlayer/
↪setIntrinsicResolution 0.053
/gate/digitizer/Singles/intrinsicResolutionBlurring/ LuYAPlayer/setEnergyOfReference␣
↪511 keV

# DEFINE QUANTUM EFFICIENCY OF THE PHOTODETECTOR
/gate/digitizer/Singles/insert quantumEfficiency
/gate/digitizer/Singles/quantumEfficiency/chooseQEVolume crystal
/gate/digitizer/Singles/quantumEfficiency/setUniqueQE 0.1
```

Note: A complete example of a phoswich module can be in the PET benchmark.

Note for Quantum Efficiency

With the previous commands, the same quantum efficiency will be applied to all the detector channels. The user can also provide lookup tables for each detector module. These lookup tables are built from the user files.

To set multiple quantum efficiencies using files (*fileName1*, *fileName2*, ... for each of the different modules), the following commands can be used:

```
/gate/digitizer/Singles/insert quantumEfficiency
/gate/digitizer/Singles/quantumEfficiency/chooseQEVolume crystal
/gate/digitizer/Singles/quantumEfficiency/useFileDataForQE fileName1
/gate/digitizer/Singles/quantumEfficiency/useFileDataForQE fileName2
```

If the *crystal* volume is a daughter of a *module* volume which is an array of 8 x 8 crystals, the file *fileName1* will contain 64 values of quantum efficiency. If several files are given (in this example two files), the program will choose randomly between theses files for each *module*.

Important note

After the introduction of the lightYield (LY), transferEfficiency ($\bar{p}$) and quantumEfficiency} ($\bar{\epsilon}$) modules, the energy variable of a *pulse* is not in energy unit (MeV) but in number of photoelectrons N_{pe}.

$N_{phe} = N_{ph} \cdot \bar{\epsilon} \cdot \bar{p} = LY \cdot E \cdot \bar{\epsilon} \cdot \bar{p}$

In order to correctly apply a threshold on a phoswhich module, the threshold should be based on this number and not on the real energy. In this situation, to apply a threshold at this step of the digitizer chain, the threshold should be applied as explained in *Thresholder & Upholder*. In this case, the GATE program knows that these modules have been used, and will apply threshold based upon the number N_{pe} rather than energy. The threshold set with this sigmoidal function in energy unit by the user is translated into number N_{pe} with the lower light yield of the phoswish module. To retrieve the energy it is necessary to apply a calibration module.

Calibration

The Calibration module of the pulse-processor models a calibration between N_{phe} and $Energy$. This is useful when using the class(es) GateLightYield, GateTransferEfficiency, and GateQuantumEfficiency. In addition, a user specified calibration factor can be used. To set a calibration factor on the energy, use the following commands:

```
/gate/digitizer/Singles/insert calibration
/gate/digitizer/Singles/setCalibration VALUE
```

If the calibration digitizer is used without any value, it will correct the energy as a function of values used in GateLightYield, GateTransferEfficiency, and GateQuantumEfficiency.

Crosstalk

The crosstalk module simulates the optical and/or electronic crosstalk of the scintillation light between neighboring crystals. Thus, if the input pulse arrives in a crystal array, this module creates pulses around it (in the edge and corner neighbor crystals). The percentage of energy that is given to the neighboring crystals is determined by the user. To insert a crosstalk module that distributes 10% of input pulse energy to the adjacent crystals and 5% to the corner crystals, the following commands can be used:

```
/gate/digitizer/Singles/insert crosstalk
/gate/digitizer/Singles/crosstalk/chooseCrosstalkVolume crystal
/gate/digitizer/Singles/crosstalk/setEdgesFraction 0.1
/gate/digitizer/Singles/crosstalk/setCornersFraction 0.05
```

In this example, a pulse is created in each neighbor of the crystal that received the initial pulse. These secondary pulses have 10% (5% for each corner crystals) of the initial energy of the pulse.

BEWARE: this module works only for a chosen volume that is an array repeater!!!

Thresholder & Upholder

The *Thresholder/Upholder* modules allow the user to apply an energy window to discard low and high energy photons. The low energy cut, supplied by the user, represents a threshold response, below which the detector remains inactive. The user-supplied high energy cut is the maximum energy the detector will register. In both PET and SPECT analysis, the proper setting of these windows is crucial to mimic the behavior of real scanners, in terms of scatter fractions and count rate performances for instance. In a typical PET scanner, the energy selection for the photo-peak is performed using the following commands. A low threshold of 0 keV allows the user to see all the events, and is often useful for debugging a simulation:

```
/gate/digitizer/Singles/insert thresholder
/gate/digitizer/Singles/thresholder/setThreshold 250. keV
/gate/digitizer/Singles/insert upholder
/gate/digitizer/Singles/upholder/setUphold 750. keV
```

Energy windows

In SPECT analysis, subtractive scatter correction methods such as the dual-energy-window or the triple-energy-window method may be performed in post processing on images obtained from several energy windows. If one needs multiple energy windows, several digitizer branches will be created. Furthermore, the projections associated to each energy window can be recorded into one interfile output. In the following example, 3 energy windows are defined separately with their names and thresholds (see *Thresholder & Upholder*):

```
/gate/digitizer/name Window1
/gate/digitizer/insert singleChain
/gate/digitizer/Window1/setInputName Singles
/gate/digitizer/Window1/insert thresholder
/gate/digitizer/Window1/thresholder/setThreshold 315 keV
/gate/digitizer/Window1/insert upholder
/gate/digitizer/Window1/upholder/setUphold 328 keV

/gate/digitizer/name Window2
/gate/digitizer/insert singleChain
/gate/digitizer/Window2/setInputName Singles
/gate/digitizer/Window2/insert thresholder
/gate/digitizer/Window2/thresholder/setThreshold 328 keV
/gate/digitizer/Window2/insert upholder
/gate/digitizer/Window2/upholder/setUphold 400 keV

/gate/digitizer/name Window3
/gate/digitizer/insert singleChain
/gate/digitizer/Window3/setInputName Singles
/gate/digitizer/Window3/insert thresholder
/gate/digitizer/Window3/thresholder/setThreshold 400 keV
/gate/digitizer/Window3/insert upholder
/gate/digitizer/Window3/upholder/setUphold 416 keV
```

When specifying the interfile output (see *Interfile output of projection set*), the different window names must be added with the following commands:

```
/gate/output/projection/setInputDataName Window1
/gate/output/projection/addInputDataName Window2
/gate/output/projection/addInputDataName Window3
```

Sigmoidal thresholder

The *Sigmoidal thresholder* models a threshold discriminator based on a sigmoidal function. A sigmoidal function is an S-shaped function of the form, $\sigma(x) = \frac{1}{1+c\exp(-ax)}$, which acts as an exponential ramp from 0 to 1:

$\sigma(E) = \frac{1}{1+\exp\left(\alpha\ \frac{E-E_0}{E_0}\right)}$

where the parameter α is proportional to the slope at symmetrical point $E_0(\sigma(E_0) = 1/2)$.

For this type of threshold discriminator, the user chooses the threshold **setThreshold**, the percentage of acceptance for this threshold **setThresholdPerCent**, and the α parameter **setThresholdAlpha**. With these parameters and the input *pulse* energy, the function is calculated. If the result is bigger than a random number generated between 0 and 1, the *pulse* is accepted and copied into the output pulse-list. On the other hand, if this criteria is not met, the input *pulse* is discarded:

```
/gate/digitizer/Singles/insert sigmoidalThresholder
/gate/digitizer/Singles/sigmoidalThresholder/setThreshold 250 keV
/gate/digitizer/Singles/sigmoidalThresholder/setThresholdAlpha 60.
/gate/digitizer/Singles/sigmoidalThresholder/setThresholdPerCent 0.95
```

Time resolution

The *temporal resolution* module introduces a Gaussian blurring in the time domain. It works in the same manner as the *blurring* module, but with time instead of energy. To set a Gaussian temporal resolution (FWHM) of 1.4 ns, use

the following commands:

```
/gate/digitizer/Singles/insert timeResolution
/gate/digitizer/Singles/timeResolution/setTimeResolution 1.4 ns
```

Spatial blurring

For SPECT simulations, the spatial resolution is assumed to follow a Gaussian distribution defined by its width σ:

```
/gate/digitizer/Singles/insert spblurring
/gate/digitizer/Singles/spblurring/setSpresolution 2.0 mm
/gate/digitizer/Singles/spblurring/verbose 1
```

In PET analysis, coincidence events provide the lines of response (LOR) needed for the image reconstruction. Only the two crystal numbers are transferred by the simulation. The determination of these crystal numbers is based on the crystal in which the highest energy has been deposited. Without additional spatial blurring of the crystal, simulation results will always have a better spatial resolution than experimental measurements. This module is only available for the *ecat* system. The spatial blurring is based on a 2D Gaussian function:

```
# E C A T 7
/gate/output/sinogram/enable
/gate/output/sinogram/RadialBins Your_Sinogram_Radial_Bin_Number
/gate/output/sinogram/setTangCrystalBlurring Your_Value_1 mm
/gate/output/sinogram/setAxialCrystalBlurring Your_Value_2 mm
```

Noise

Different sources of background noise exist in a PET/SPECT architecture. For example, the electronics can introduce its own noise, or some crystals used for the detection, such as LSO, contains radioactive nucleus, which can contribute to the background detection count rate. Within GATE, the *noise* module adds such background events, in a totally generic way, so that any kind of source of noise can be simulated. To do so, the energy and the inter-event time interval are chosen randomly, for each event, into user defined distributions, by using the mechanism described in *Distributions*.

In the following example, a noise source is introduced, whose energy is distributed according to a Gaussian law, and whose time distribution follows a Poisson process. To do this, one first defines the two necessary distributions. Since the noise description uses the distribution of the time interval between consecutive events, one has to define an exponential distribution. Indeed, if the probability of detecting k events in a time interval of t is distributed along a Poisson law $P_1(k,t) = e^{-\lambda t}\frac{(\lambda t)^k}{k!}$, then the probability density of having a time interval in the range $[t; t+dt]$ between two consecutive events is given by $dP_2(t) = \lambda e^{-\lambda t} dt$:

```
/gate/distributions/name energy_distrib
/gate/distributions/insert Gaussian
/gate/distributions/energy_distrib/setMean 450 keV
/gate/distributions/energy_distrib/setSigma 1 keV

/gate/distributions/name dt_distrib
/gate/distributions/insert Exponential
/gate/distributions/dt_distrib/setLambda 7.57 mus

/gate/digitizer/Singles/insert noise
/gate/digitizer/Singles/noise/setDeltaTDistribution dt_distrib
/gate/digitizer/Singles/noise/setEnergyDistribution energy_distrib
```

The special event ID, **event_ID=-2**, is assigned to these noise events.

Local efficiency

The different crystals, or groups of crystals, composing a PET/SPECT system can be characterized by their own efficiency. GATE offers a method to describe such efficiency per crystal or volume. To define the efficiency distribution in the scanner, one can specify which level of the volume hierarchy of the system are differentiated (see the examples in *Command line*). Then the distribution of efficiency, for each differentiated volume, is specified via a generic distribution, as described in *Distributions*.

In the following examples, one assumes that the system is composed of 8 blocks (level1) of 64 crystals (level2). The first example shows how to specify one efficiency per block, defined in a file named **eff_per_block.dat**, containing 8 values (one per block):

```
/gate/distributions/name block_eff_distrib
/gate/distributions/insert File
/gate/distributions/block_eff_distrib/autoX true
/gate/distributions/block_eff_distrib/setFileName eff_per_block.dat
/gate/distributions/block_eff_distrib/read

/gate/digitizer/Singles/insert localEfficiency
/gate/digitizer/Singles/localEfficiency/enableLevel 1
/gate/digitizer/Singles/localEfficiency/disableLevel 2
/gate/digitizer/Singles/localEfficiency/setEfficiency block_eff_distrib
```

In the second example, one specifies a different efficiency for each crystal inside a block, but the scheme is repeated from one block to another. So a pattern of 64 efficiency values is defined in the file **eff_within_block.dat**:

```
/gate/distributions/name within_block_eff_distrib
/gate/distributions/insert File
/gate/distributions/within_block_eff_distrib/autoX true
/gate/distributions/within_block_eff_distrib/setFileName eff_within_block.dat
/gate/distributions/within_block_eff_distrib/read

/gate/digitizer/Singles/insert localEfficiency
/gate/digitizer/Singles/localEfficiency/disableLevel 1
/gate/digitizer/Singles/localEfficiency/enableLevel 2
/gate/digitizer/Singles/localEfficiency/setEfficiency within_block_eff_distrib
```

Finally, in the next example, each crystal has its own efficiency, described in the file **eff_per_crystal.dat** containing 8 x 64 elements:

```
/gate/distributions/name crystal_eff_distrib
/gate/distributions/insert File
/gate/distributions/crystal_eff_distrib/autoX true
/gate/distributions/crystal_eff_distrib/setFileName eff_per_crystal.dat
/gate/distributions/crystal_eff_distrib/read

/gate/digitizer/Singles/insert localEfficiency
/gate/digitizer/Singles/localEfficiency/enableLevel 1
/gate/digitizer/Singles/localEfficiency/enableLevel 2
/gate/digitizer/Singles/localEfficiency/setEfficiency crystal_eff_distrib
```

Memory buffers and bandwidth

To mimic the effect of limited transfer rate, a module models the data loss due to an overflow of a memory buffer, read periodically, following a given reading frequency. This module uses two parameters, the reading frequency ν . Moreover, two reading methods can be modelled, that is, in an event per event basis (an event is read at each reading

clock tick), or in a full buffer reading basic (at each reading clock tick, the whole buffer is emptied out). In the first reading method, the data rate is then limited to ν , while in the second method, the data rate is limited to $D \cdot \nu$. When the size limit is reached, any new pulse is rejected, until the next reading clock tick arrival which frees a part of the buffer. In such a case, a non null buffer depth allows to manage a local rise of the input data flow. To specify a buffer, read at 10 MHz, with a buffer depth of 64 events, in a mode where the whole buffer is read in one clock tick, one can use:

```
/gate/digitizer/Your_Single_chain/insert buffer
/gate/digitizer/Your_Single_chain/buffer/setBufferSize 64 B
/gate/digitizer/Your_Single_chain/buffer/setReadFrequency 10 MHz
/gate/digitizer/Your_Single_chain/buffer/setMode 1
```

The chain *Your_Single_chain* can be the default chain *Singles* or any of single chain that the user has defined. The size of the buffer represents the number of elements, 64 Singles in this example, that the user can store in a buffer. To read the buffer in an event by event basis, one should replace the last line by **setMode = 0.**

Pile-up

An important characteristic of a detector is its response time, which is the time that the detector takes to form the signal after the arrival of the radiation. The duration of the signal is also important. During this period, if a second event can be accepted, this second signal will *pile up* on the first. The resulting pulse is a combinaison in terms of time and energy, of the two signals. If N pulses enter in the time window of the same sensitive volume (set by the depth of the system level), the output pulse of the pile-up module will be a pulse with an output energy defined by the sum of the energies ($E_{out} = \sum_{i=0}^{N} E_i$) and a time set to the last time of the last pulse participating to the pile-up $t_{out} = t_N$. Since multiple events are grouped into a unique event with the pile-up effect, one can consider this as a loss of events occuring during a given time length, which can be seen as a dead time effect. Moreover, since the pile-up end time is always updated with the last single occuring, the effect is more or less represented by a paralysable dead-time. To insert a pile-up corresponding to a signal formation time of 100 ns in a module corresponding to the crystal group as described by the 4th level of the system, one should use:

```
/gate/digitizer/Singles/insert pileup
/gate/digitizer/Singles/pileup/setDepth 4
/gate/digitizer/Singles/pileup/setPileup 100 ns
```

Dead time

Due to the shaping time of signals or for any other reason, each detection of a single event can hide the subsequent single detected on the same electronic module. This loss lasts a certain amount of time, depending on the characteristics of the detectors used as well as of the readout electronics. The dead time can be modelled in GATE as shown below. Two models of the dead-time have been implemented in the digitizer: *paralysable* and *nonparalysable* response. These models can be implemented *event by event* during a simulation. The detailed method underlying these models can be found in Knoll 1979 (Radiation detection and measurement, John Wiley & Sons, New York). The fundamental assumptions made by these two models are illustrated in Fig. 3.19.

The dead time module is applied to a specific volume within the Sensitive Detector system hierarchy. All events taking place within this volume level will trigger a dead-time detector response. This action of the digitizer simulates the time during which this detector, busy at processing a particle, will not be able to process the next one. Moreover, one can simulate the case where data are accumulated into a buffer, which is written to a mass storage having a time access, during which no other data can be processed. In such a case, the dead time is not started after the first data, but once the buffer is full. This case can also be simulated in GATE.

To apply a dead-time to the volume_name (which has to be previously attached to a level of the system), the following commands can be used:

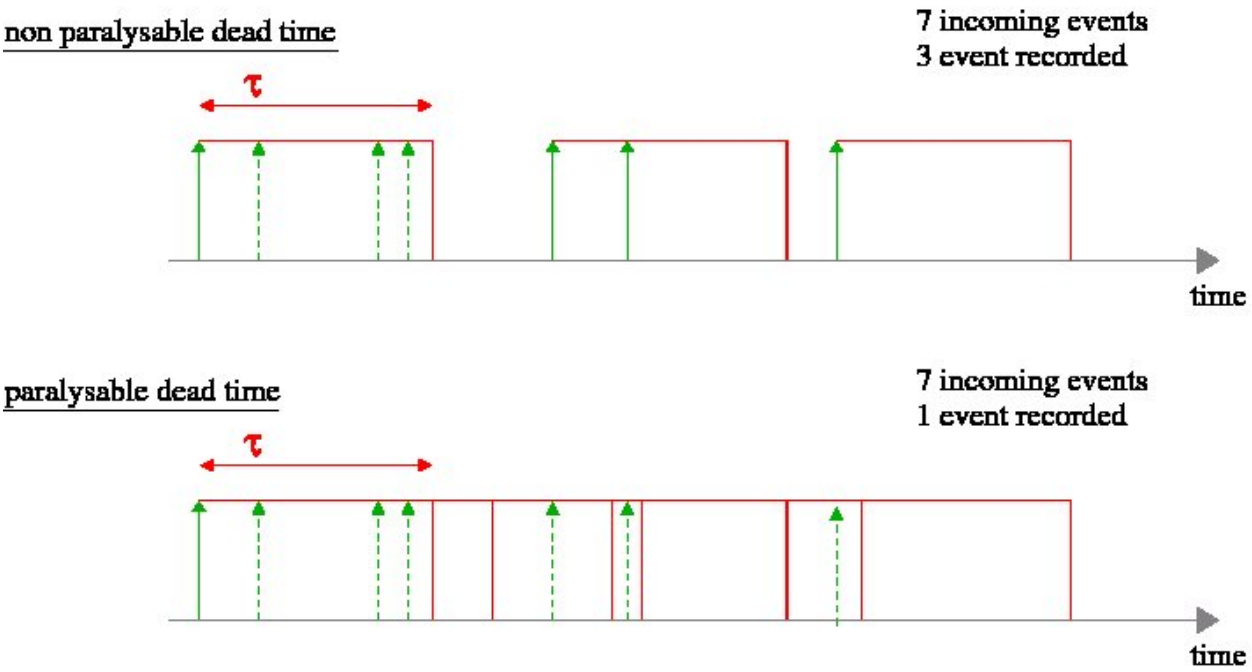


Fig. 3.19: For 7 incoming particles and a fixed dead-time τ, the *nonparalysable* electronic readout will accept 3 particles, and the *paralysable* will accept only 1 particle (the dashed arrows represents the removed events, while the solid arrows are the accepted singles)

```
# ATTACHEMENT TO THE SYSTEM
/gate/systems/system_name/system_level_name/attach volume_name
..
..
# DEADTIME
/gate/digitizer/Singles/insert deadtime
/gate/digitizer/Singles/deadtime/setDeadTime 100000. ns
/gate/digitizer/Singles/deadtime/setMode paralysable
/gate/digitizer/Singles/deadtime/chooseDTVolume volume_name
```

The name *system_name* and its corresponding *system_level_name* do not exist and have to be chosen in the tables given in *Defining a system*.

In the second example, a dead time corresponding to a disk access of 1 µs for a memory buffer of 1 Mbyte is given. The *setMode* command specifies the behavior of the dead time during the disk access. If this mode is set to 0, the memory buffer is assumed to be a shared resource for the computer, and thus is not available during the disk writing. So, no data can fill the buffer during the disk access. On the other hand, in case of model 1, the buffer is immediately freed after being sent to the disk controller. Data are thus not rejected, unless the buffer is filled up again, before the disk access is finished. In such a case, the dead time module will be totally transparent (ie. will not reject any data), unless the counting rate is high enough to fill the buffer in a time lower than the disk access dead time:

```
# ATTACHEMENT TO THE SYSTEM
/gate/systems/system_name/system_level_name/attach volume_name
..
..
# DEADTIME
/gate/digitizer/Singles/insert deadtime
/gate/digitizer/Singles/deadtime/setDeadTime 1 mus
```

(continues on next page)

(continued from previous page)

```
/gate/digitizer/Singles/deadtime/setMode nonparalysable
/gate/digitizer/Singles/deadtime/chooseDTVolume volume_name
/gate/digitizer/Singles/deadtime/setBufferSize 1 MB
/gate/digitizer/Singles/deadtime/setBufferMode 0
```

3.3.3 Multiple processor chains

The use of multiple processor chains makes the design of the digitizer and data output system extremely flexible. The manager for the pulse-processors is called the *GatePulseProcessorChain*, and has a messenger called the *GatePulseProcessorChainMessenger*. By default, all the digitizer components are stored in one processor-chain called *digitizer/Singles*. New processor chains can be created that specify the source of their data. For instance, the following sequence of commands will generate three outputs:

- *Singles* with no energy cut
- *LESingles* with a low-energy window
- *HESingles* with a high-energy window

For a standard PET (with BGO crystals), the components of the standard processor chain will consist in the following commands:

```
/gate/digitizer/Singles/insert adder
/gate/digitizer/Singles/insert readout
/gate/digitizer/Singles/readout/setDepth 1
```

To add the blurring filter to the "Single" branch:

```
/gate/digitizer/Singles/insert blurring
/gate/digitizer/Singles/blurring/setResolution 0.26
/gate/digitizer/Singles/blurring/setEnergyOfReference 511. keV
```

The following commands create a low-energy chain branching from the output of "Singles" chain:

```
/gate/digitizer/name LESingles
/gate/digitizer/insert singleChain
/gate/digitizer/LESingles/setInputName Singles
/gate/digitizer/LESingles/insert thresholder
/gate/digitizer/LESingles/thresholder/setThreshold 50. keV
/gate/digitizer/LESingles/insert upholder
/gate/digitizer/LESingles/upholder/setUphold 350. keV
```

These next commands create a high-energy chain branching from the output of the "Singles" chain:

```
/gate/digitizer/name HESingles
/gate/digitizer/insert singleChain
/gate/digitizer/HESingles/setInputName Singles
/gate/digitizer/HESingles/insert thresholder
/gate/digitizer/HESingles/thresholder/setThreshold 350. keV
/gate/digitizer/HESingles/insert upholder
/gate/digitizer/HESingles/upholder/setUphold 650. keV
```

3.3.4 Coincidence sorter

The coincidence sorter searches, into the singles list, for pairs of coincident singles. Whenever two or more *singles* are found within a coincidence window, these *singles* are grouped to form a *Coincidence* event. Two methods are possible to find coincident singles within GATE. In the first method, when a single is detected, it opens a new coincidence window, and search for a second single occurring during the length of the window. In this method, as long as the window opened by the first single is not closed, no other single can open its own coincidence window. In the second method, all singles open their own coincidence window, and a logical OR is made between all the individual signals to find coincidences. The two methods are available in GATE, and can lead to slightly different results, for a given window width. A comparison of the difference of these two behaviors in a real case is sketched in Fig. 3.20.

To exclude coincidence coming from the same particle that scattered from a block to an adjacent block, the proximity of the two blocks forming the coincidence event is tested. By default, the coincidence is valid only if the difference in the block numbers is greater or equal to two, but this value can be changed in GATE if needed.

Delayed coincidences

Each *Single* emitted from a given source particle is stored with an event ID number, which uniquely identifies the decay from which the single is coming from. If two event ID numbers are not identical in a coincidence event, the event is defined as a *Random* coincidence.

An experimental method used to estimate the number of random coincidences consists of using a delayed coincidence window. By default, the coincidence window is opened when a particle is detected (i.e. when a *Single* is created). In this method, a second coincidence window is created in parallel to the normal coincidence window (which in this case is referred to as the prompt window). The second window (usually with the same width) is open, but is shifted in time. This shift should be long enough to ensure that two particles detected within it are coming from different decays. The resulting number of coincidences detected in this delayed window approximates the number of random events counted in the prompt window. GATE offers the possibility to specify an offset value, for the coincidence sorter, so that prompts and/or delayed coincidence lines can be simulated.

Multiple coincidences

When more than two *singles* are found in coincidence, several type of behavior could be implemented. GATE allows to model 9 different rules that can be used in such a case. The list of rules along with their explanation are given in Table 3.4, and a comparison of the effects of each processing rule for various cases of multiple coincidences is shown in Fig. 3.21. If no policy is specified, the default one used is: keepIfAllAreGoods.

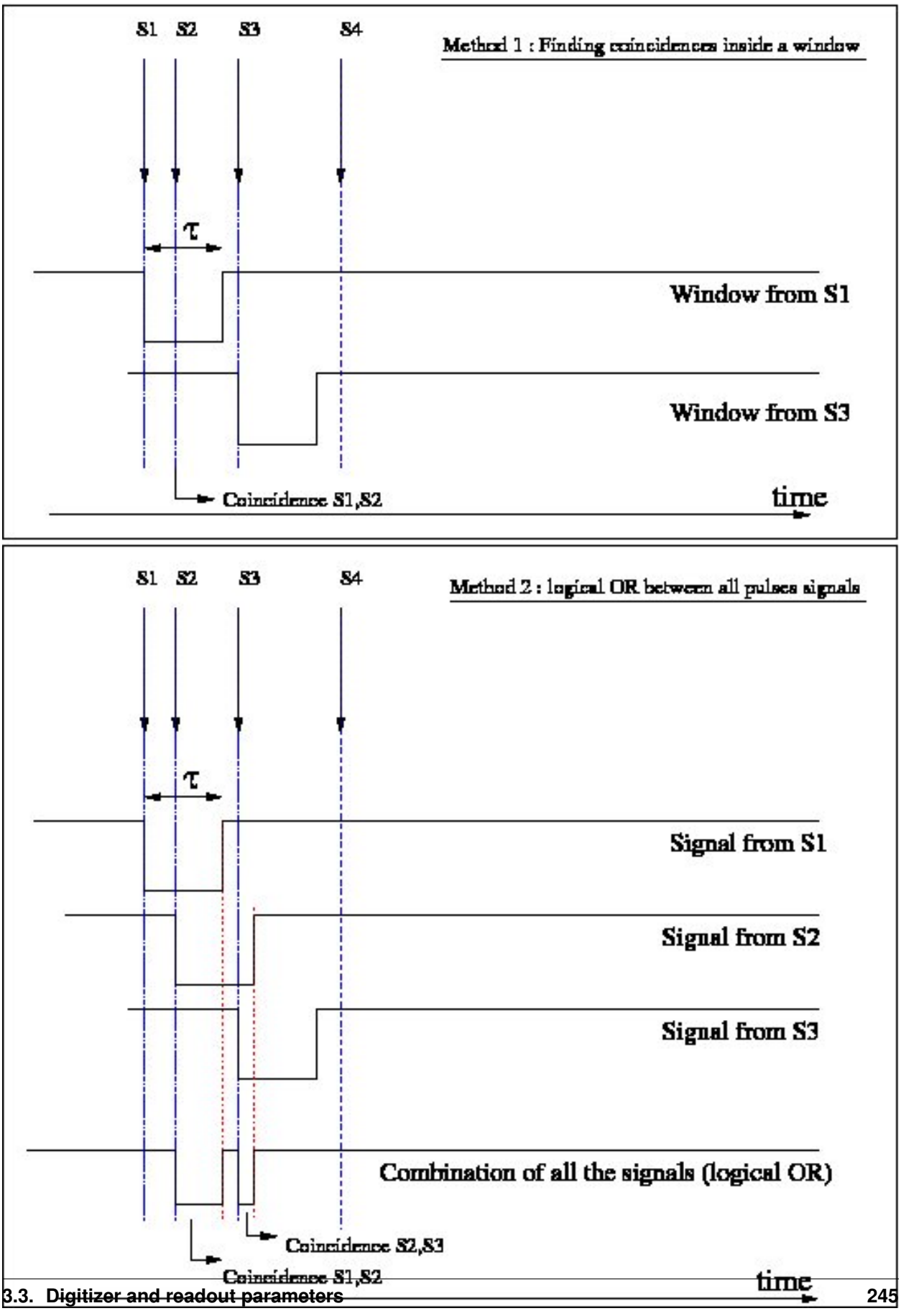


Fig. 3.20: Comparison between the two methods of coincidence sorting, for a given succession of singles. In the first one (top), the S2 single does not open its own window since its arrival time is within the window opened by S1. With

Table 3.4: Available multiple policy and associated meaning. When a multiple coincidence involving n *singles* is peocessed, it is first decomposed into a list of n·(n1) pairs which are analyzed individually. In this table, the term "good" means that a pair of singles are in coincidence and that the 2 singles are separated by a number of blocks greater than or equal to the **minSectorDifference** parameter of the coincidence sorter. The prefix "take" means that 1 or more pairs of coincidences will be stored, while the prefix "keep" means that a unique coincidence, composed of at least three singles will be kept in the data flow and is called "multicoincidence". In the latter case, the multicoincidence will not be written to the disk, but may participate to a possible deadtime or bandwidth occupancy. The user may clear the multicoincidence at any desired step of the acquisition, by using the multipleKiller pulse processor (described in #Multiple coincidence removal). The "kill" prefix means that all events will be discarded and will not produce any coincidence.

Policy name	Description
takeAllGoods	Each good pairs are considered
takeWinnerOfGoods	Only the good pair with the highest energy is considered
takeWinnerIfIsGood	If the pair with the highest energy is good, take it, otherwise, kill the event
takeWinnerIfAllAre-Goods	If all pairs are goods, take the one with the highest energy
keepIfOnlyOneGood	If exactly one pair is good, keep the multicoincidence
keepIfAnyIsGood	If at least one pair is good, keep the multicoincidence
keepIfAllAreGoods	If all pairs are goods, keep the multicoincidence
killAllIfMultipleGoods	If more than one pairs is good, the event is seen as a real "multiple" and thus, all events are killed
killAll	No multiple coincidences are accepted, no matter how many good pairs are present

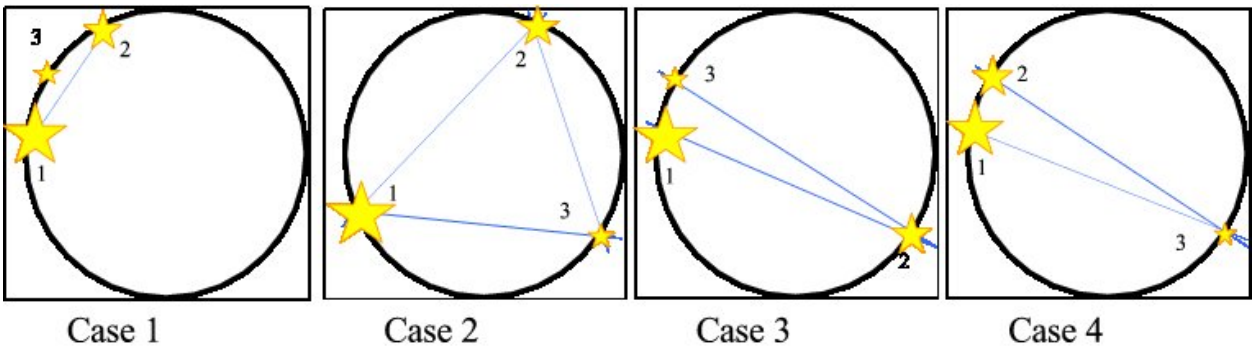


Fig. 3.21: Comparison of the behavior of the available multiple processing policies, for various multiple coincidence situations. The stars represent the detected singles. The size of the star, as well as the number next to it, indicate the energy level of the single (ie. single no 1 has more energy than single no 2, which has itself more energy than the single no 3). The lines represent the possible good coincidences (ie. with a sector difference higher than or equal to the minSectorDifference of the coincidence sorter). In the table, a minus(-) sign indicates that the event is killed (ie. no coincidence is formed). The sign indicates that all the singles are kept into a unique multicoincidence, which will not be written to disk, but which might participate to data loss via dead time or bandwidth occupancy. In the other cases, the list of pairs which are written to the disk (unless being removed thereafter by possible filter applied to the coincidences) is indicated

Table 3.5: Table associated with Fig. 3.21

Policy name	Case 1	Case 2	Case 3	Case 4
takeAllGoods	(1,2)	(1,2); (1,3); (2,3)	(1,2); (2,3)	(1,3); (2,3)
takeWinnerOfGoods	(1,2)	(1,2)	(1,2)	(1,3)
takeWinnerIfIsGood	(1,2)	(1,2)	(1,2)	-
takeWinnerIfAllAreGoods	-	(1,2)	-	-
keepIfOnlyOneGood	*	-	-	-
keepIfAnyIsGood	*	*	*	*
keepIfAllAreGoods	-	*	-	-
killAllIfMultipleGoods	(1,2)	-	-	-
killAll	-	-	-	-

Command line

To set up a coincidence window of 10 ns, the user should specify:

```
/gate/digitizer/Coincidences/setWindow 10. ns
```

To change the default value of the minimum sector difference for valid coincidences (the default value is 2), the command line should be used:

```
/gate/digitizer/Coincidences/minSectorDifference <number>
```

By default, the offset value is equal to 0, which corresponds to a prompt coincidence sorter. If a delayed coincidence sorter is to be simulated, with a 100 ns time shift for instance, the offset value should be set using the command:

```
/gate/digitizer/Coincidences/setOffset 100. ns
```

To specify the depth of the system hierarchy for which the coincidences have to be sorted, the following command should be used:

```
/gate/digitizer/Coincidences/setDepth <system's depth (1 by default)>
```

As explained in Fig. 3.20, there are two methods for building coincidences. The default one is the method 1. To switch to method 2, one should use:

```
/gate/digitizer/Coincidences/allPulseOpenCoincGate true
```

So when set to false (by default) the method 1 is chosen, and when set to true, this is the method 2. **Be aware that the method 2 is experimental and not validated at all, so potentially containing non-reported bugs.**

Finally, the rule to apply in case of multiple coincidences is specified as follows:

```
/gate/digitizer/Coincidences/setMultiplePolicy <policyName>
```

The default multiple policy is keepIfAllAreGoods.

3.3.5 Multiple coincidence sorters

Multiple coincidence sorters can be used in GATE. To create a coincidence sorter, the sorter must be named and a location specified for the input data. In the example below, three new coincidence sorters are created:

- One with a very long coincidence window:

```
/gate/digitizer/name LongCoincidences
/gate/digitizer/insert coincidenceSorter
/gate/digitizer/LongCoincidences/setInputName Singles
/gate/digitizer/LongCoincidences/setWindow 1000. ns
```

- One for low-energy singles:

```
/gate/digitizer/name LECoincidences
/gate/digitizer/insert coincidenceSorter
/gate/digitizer/LECoincidences/setWindow 10. ns
/gate/digitizer/LECoincidences/setInputName LESingles
```

- One for high-energy-singles:

```
/gate/digitizer/name HECoincidences
/gate/digitizer/insert coincidenceSorter
/gate/digitizer/HECoincidences/setWindow 10. ns
/gate/digitizer/HECoincidences/setInputName HESingles
```

A schematic view corresponding to this example is shown in Fig. 3.22.

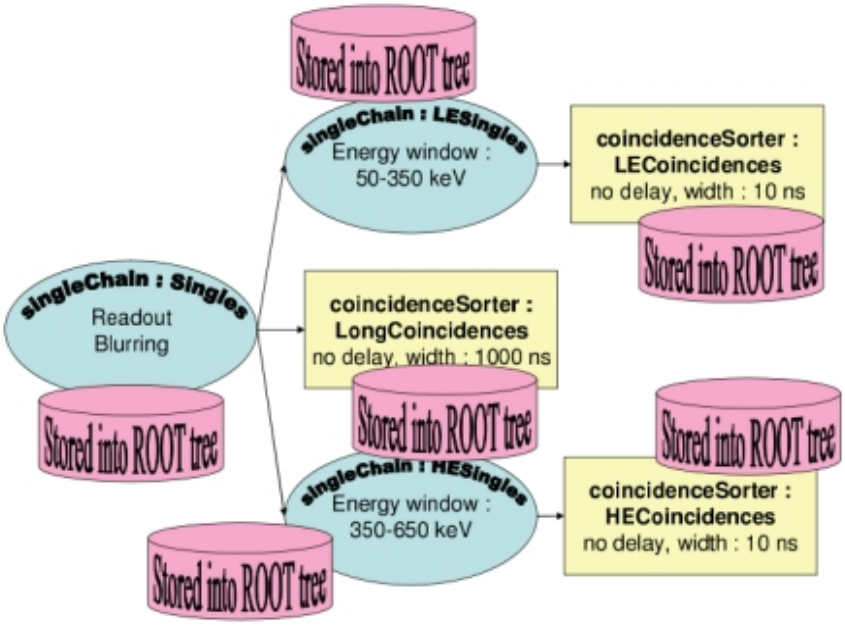


Fig. 3.22: Readout scheme produced by the listing from the sections

3.3.6 Coincidence processing and filtering

Coincidence pulse processors

Once the coincidences are identified, further processing can be applied to mimic count losses that may occur because of the acquisition limitations, such as dead time. Count loss may also be due to the limited bandwidth of wires or buffer capacities of the I/O interface. The modelling of such effects within GATE is explained below. Moreover, for PET scanners using a delayed coincidence line, data coming from the two types of coincidences (ie. prompts and delayed) can be processed by a unique coincidence sorter. If so, the rate of a coincidence type can affect the rate of the

other. For instance, a prompt coincidence can produce dead time which will hide a delayed coincidence. The prompt coincidence events can also saturate the bandwidth, so that the random events are partially hidden.

The modelling of such effects consists in grouping the two different coincidence types into a unique one, which is then processed by a unique filter.

A coincidence pulse processor is a structure that contains the list of coincidence sources onto which a set of filters will be applied, along with the list of filters themselves. The order of the list of coincidence may impact the repartition of the data loss between the prompt and the delay lines. For example, if the line of prompt coincidences has priority over the line of delayed coincidences, then the events of the latter have more risk to be rejected by a possible buffer overflow than those of the former. This kind of effects can be suppressed by specifying that, inside an event, all the coincidences arriving with the same time flag are merged in a random order.

To implement a coincidence pulse processor merging two coincidence lines into one, and apply an XXX module followed by another YYY module on the total data flow, one should use the following commands, assuming that the two coincidence lines named *prompts* and *delayed* are already defined:

```
/gate/digitizer/name myCoincChain
/gate/digitizer/insert coincidenceChain
/gate/digitizer/myCoincChain/addSource prompts
/gate/digitizer/myCoincChain/addSource delayed
/gate/digitizer/myCoincChain/insert XXX
# set parameter of XXX....
/gate/digitizer/myCoincChain/insert YYY
# set parameter of YYY....
```

To specify that two coincidences arriving with the same time flag have to be processed in random order, one should use the command:

```
/gate/digitizer/myCoincChain/usePriority false
```

Coincidence dead time

The dead time for coincidences works in the same way as that acting on the *singles* data flow. The only difference is that, for the *single* dead time, one can specify the hierarchical level to which the dead time is applied on (corresponding to the separation of detectors and electronic modules), while in the coincidence dead time, the possibility to simulate separate coincidence units (which may exist) is not implemented. Apart from this limitation, the command lines for coincidence dead time are identical to the ones for *singles* dead time, as described in *Pile-up*. When more than one coincidence can occur for a unique GEANT4 event (if more than one coincidence line are added to the coincidence pulse processor, or if multiple coincidences are processed as many coincidences pairs), then the user can specify that the whole event is kept or rejected, depending on the arrival time of the first coincidence. To do so, one should use the command line: :

/gate/digitizer/myCoincChain/deadtime/conserveAllEvent true

Coincidence buffers

The buffer module for affecting coincidences uses exactly the same command lines and functionalities as the ones used for single pulse lists, and described in section *Local efficiency*.

Multiple coincidence removal

If the multiple coincidences are kept and not splitted into pairs (ie. if any of the **keepXXX** multiple coincidence policy is used), the multicoincidences could participate to dataflow occupancy, but could not be written to the disk. Unless

otherwise specified, any multicoincidence is then cleared from data just before the disk writing. If needed, this clearing could be performed at any former coincidence processing step, by inserting the **multipleKiller** module at the required level. This module has no parameter and just kill the multicoincidence events. Multiple coincidences split into many pairs are not affected by this module and cannot be distinguished from the normal "simple" coincidences. To insert a multipleKiller, one has to use the syntax:

```
/gate/digitizer/myCoincChain/insert multipleKiller
```

3.3.7 Example of a digitizer setting

Here, the digitizer section of a GATE macro file is analyzed line by line. The readout scheme produced by this macro, which is commented on below, is illustrated in Fig. 3.23.

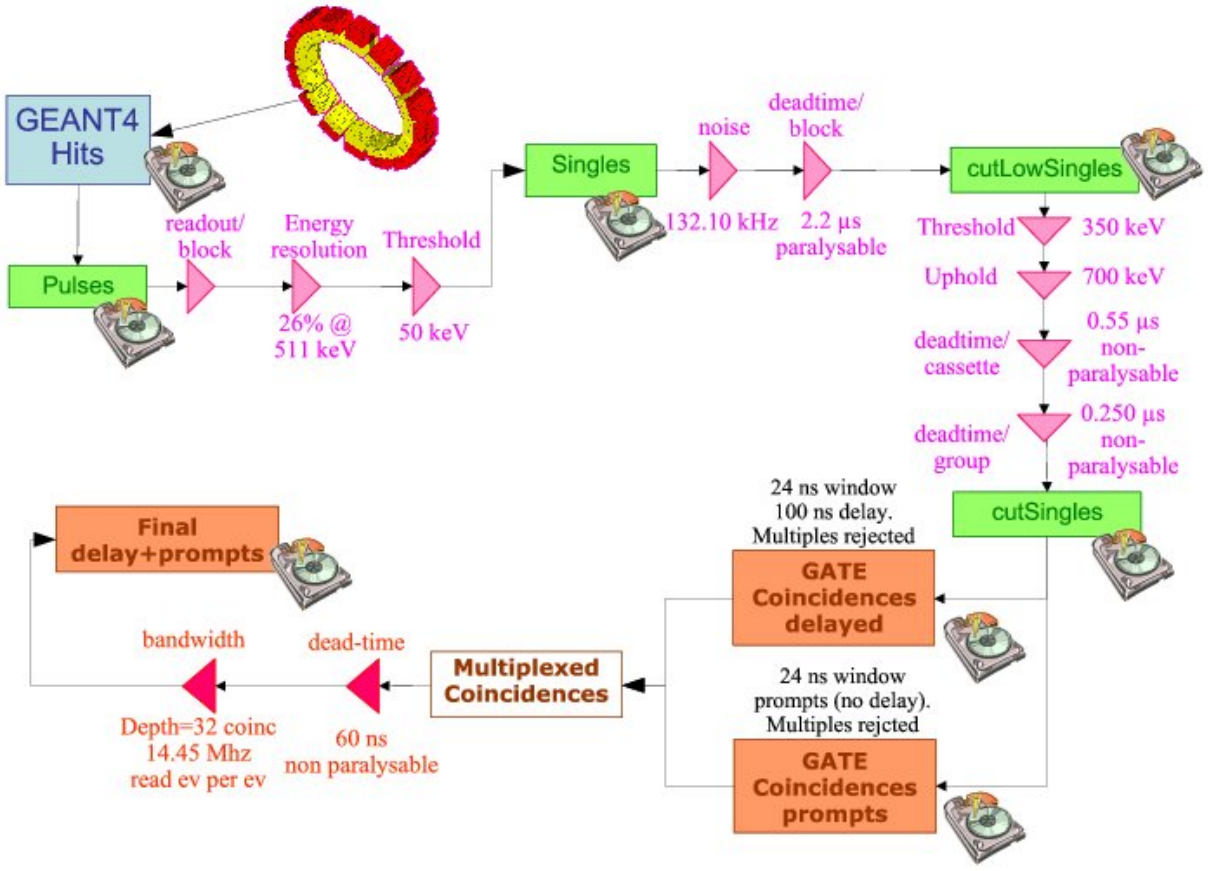


Fig. 3.23: Readout scheme produced by the listing below. The disk icons represent the data written to the GATE output files

Example:

```
1 # A D D E R
2 /gate/digitizer/Singles/insert adder
3
4 # R E A D O U T
```

(continues on next page)

(continued from previous page)

```
/gate/digitizer/Singles/insert readout
/gate/digitizer/Singles/readout/setDepth

# E N E R G Y B L U R R I N G
/gate/digitizer/Singles/insert blurring
/gate/digitizer/Singles/blurring/setResolution 0.26
/gate/digitizer/Singles/blurring/setEnergyOfReference 511. keV

# L O W E N E R G Y C U T
/gate/digitizer/Singles/insert thresholder
/gate/digitizer/Singles/thresholder/setThreshold 50. keV

/gate/digitizer/name cutLowSingles
/gate/digitizer/insert singleChain
/gate/digitizer/cutLowSingles/setInputName Singles

# N O I S E

/gate/distributions/name energy_distrib
/gate/distributions/insert Gaussian
/gate/distributions/energy_distrib/setMean 450 keV
/gate/distributions/energy_distrib/setSigma 30 keV

/gate/distributions/name dt_distrib
/gate/distributions/insert Exponential
/gate/distributions/dt_distrib/setLambda 7.57 mus

/gate/digitizer/cutLowSingles/insert noise
/gate/digitizer/cutLowSingles/noise setDeltaTDistributions dt_distrib
/gate/digitizer/cutLowSingles/noise setEnergyDistributions energy_distrib

 # D E A D T I M E
/gate/digitizer/cutLowSingles/insert deadtime
/gate/digitizer/cutLowSingles/deadtime/setDeadTime 2.2 mus
/gate/digitizer/cutLowSingles/deadtime/setMode paralysable
/gate/digitizer/cutLowSingles/deadtime/chooseDTVolume module

# H I G H E N E R G Y C U T
/gate/digitizer/name cutSingles
/gate/digitizer/insert singleChain
/gate/digitizer/cutSingles/setInputName cutLowSingles
/gate/digitizer/cutSingles/name highThresh
/gate/digitizer/cutSingles/insert thresholder
/gate/digitizer/cutSingles/highThresh/setThreshold 350. keV
/gate/digitizer/cutSingles/insert upholder
/gate/digitizer/cutSingles/upholder/setUphold 700. keV

/gate/digitizer/cutSingles/name deadtime_cassette
/gate/digitizer/cutSingles/insert deadtime
/gate/digitizer/cutSingles/deadtime_cassette/setDeadTime 0.55 mus
/gate/digitizer/cutSingles/deadtime_cassette/setMode nonparalysable
/gate/digitizer/cutSingles/deadtime_cassette/chooseDTVolume cassette
/gate/digitizer/cutSingles/name deadtime_group
/gate/digitizer/cutSingles/insert deadtime
/gate/digitizer/cutSingles/deadtime_group/setDeadTime 0.250 mus
/gate/digitizer/cutSingles/deadtime_group/setMode nonparalysable
/gate/digitizer/cutSingles/deadtime_group/chooseDTVolume group
```

(continues on next page)

(continued from previous page)

```
62
63
64 # C O I N C I S O R T E R 65
65 /gate/digitizer/Coincidences/setInputName cutSingles
66 /gate/digitizer/Coincidences/setOffset 0. ns
67 /gate/digitizer/Coincidences/setWindow 24. ns
68 /gate/digitizer/Coincidences/minSectorDifference 3
69
70 /gate/digitizer/name delayedCoincidences
71 /gate/digitizer/insert coincidenceSorter
72 /gate/digitizer/delayedCoincidences/setInputName cutSingles
73 /gate/digitizer/delayedCoincidences/setOffset 100. ns
74 /gate/digitizer/delayedCoincidences/setWindow 24. ns
75 /gate/digitizer/delayedCoincidences/minSectorDifference 3
76
77 /gate/digitizer/name finalCoinc
78 /gate/digitizer/insert coincidenceChain
79 /gate/digitizer/finalCoinc/addInputName delay
80 /gate/digitizer/finalCoinc/addInputName Coincidences
81 /gate/digitizer/finalCoinc/usePriority true
82 /gate/digitizer/finalCoinc/insert deadtime
83 /gate/digitizer/finalCoinc/deadtime/setDeadTime 60 ns
84 /gate/digitizer/finalCoinc/deadtime/setMode nonparalysable
85 /gate/digitizer/finalCoinc/deadtime/conserveAllEvent true
86 /gate/digitizer/finalCoinc/insert buffer
87 /gate/digitizer/finalCoinc/buffer/setBufferSize 32 B
88 /gate/digitizer/finalCoinc/buffer/setReadFrequency 14.45 MHz
89 /gate/digitizer/finalCoinc/buffer/setMode 0
```

Lines 1 to 15: The branch named “Singles” contains the result of applying the adder, readout, blurring, and threshold (50 keV) modules.

Lines 17 to 20: A new branch (line 18) is defined, named “cutLowSingles” (line 17), which follows the “Singles” branch in terms of data flow (line 19).

Lines 21 to 35: Two distributions are created, which will be used for defining a background noise. The first distributions, named energy_distribution (line 23) is a Gaussian centered on 450 keV and of 30 keV standard deviation, while the second one is an exponential distribution with a power of 7.57 mu s. These two distributions are used to add noise. The energy distribution of this source of noise is Gaussian, whileThe exponential distribution represents the distribution of time interval between two consecutive noise events (lines 32-34).

Lines 37 to 40: A paralysable (line 39) dead time of 2.2 mu s is applied on the resulting signal+noise events.

Lines 43 to 62: Another branch (line 44) named “cutSingles” (line 43) is defined. This branch contains a subset of the “cutLowSingles” branch (line 45) (after dead-time has been applied), composed of those events which pass through the 350 keV/700 keV threshold/uphold window (lines 46-50). In addition, the events tallied in this branch must pass the two dead-time cuts (lines 52 to 61) after the energy window cut.

Lines 65 to 68: The “default” coincidence branch consists of data taken from the output of the high threshold and two dead-time cuts (“cutSingles”) (line 65). At this point, a 24 ns window with no delay is defined for this coincidence sorter.

Lines 70 to 75: A second coincidence branch is defined (line 71), which is named “delayedCoincidences”. This branch takes its data from the same output (“cutSingles”), but is defined by a delayed coincidence window of 24 ns, and a 100 ns delay (line 73).

Lines 77 to 89: The delayed and the prompts coincidence lines are grouped (lines 79-80). Between two coincidences coming from these two lines and occuring within a given event, the priority is set to the delayed line, since it is

inserted before the prompt line, and the priority is used (line 81). A non-paralysable dead time of 60 ns is applied on the delayed+prompt coincidences (lines 82-85). If more than one coincidence occur inside a given event, the dead time can kill all of them or none of them, depending on the arrival time of the first one. As a consequence, if a delay coincidence is immediately followed by a prompt coincidence due to the same photon, then, the former will not hide the latter (line 85). Finally, a memory buffer of 32 coincidences, read at a frequency of 14.45 MHz, in an event-by-event basis (line 89) is applied to the delayed+prompt sum (lines 86-89).

3.3.8 Digitizer optimization

In GATE standard operation mode, primary particles are generated by the source manager, and then propagated through the attenuating geometry before generating *hits* in the detectors, which feed into the digitizer chain. While this operation mode is suited for conventional simulations, it is inefficient when trying to optimize the parameters of the digitizer chain. In this case, the user needs to compare the results obtained for different sets of digitizer parameters that are based upon the same series of hits. Thus, repeating the particle generation and propagation stages of a simulation is unnecessary for tuning the digitizer setting.

For this specific situation, GATE offers an operation mode dedicated to digitizer optimization, known as *DigiGATE*. In this mode, *hits* are no longer generated: instead, they are read from a hit data-file (obtained from an initial GATE run) and are fed directly into the digitizer chain. By bypassing the generation and propagation stages, the computation speed is significantly reduced, thus allowing the user to compare various sets of digitizer parameters quickly, and optimize the model of the detection electronics. *DigiGATE* is further explained in chapter 13.

3.3.9 Angular Response Functions to speed-up planar or SPECT simulations

The ARF is a function of the incident photon direction and energy and represents the probability that a photon will either interact with or pass through the collimator, and be detected at the intersection of the photon direction vector and the detection plane in an energy window of interest. The use of ARF can significantly speed up planar or SPECT simulations. The use of this functionality involves 3 steps.

Calculation of the data needed to derive the ARF tables

In this step, the data needed to generate the ARF tables are computed from a rectangular source located at the center of FOV. The SPECT head is duplicated twice and located at roughly 30 cm from the axial axis.

The command needed to compute the ARF data is:

```
/gate/systems/SPECThead/arf/setARFStage generateData
```

The ARF data are stored in ROOT format as specified by the GATE command output:

```
/gate/output/arf/setFileName testARFdata
```

By default the maximum size of a ROOT file is 1.8 Gbytes. Once the file has reached this size, ROOT automatically closes it and opens a new file name testARFdata_1.root. When this file reaches 1.8 Gb, it is closed and a new file testARFdata_2.root is created etc. A template macro file is provided in https://github.com/OpenGATE/GateContrib/blob/master/imaging/ARF/generateARFdata.mac which illustrates the use of the commands listed before.

Computation of the ARF tables from the simulated data

Once the required data are stored in ROOT files, the ARF tables can be calculated and stored in a binary file:

```
/gate/systems/SPECThead/arf/setARFStage computeTables
```

The digitizer parameters needed for the computation of the ARF table are defined by:

```
/gate/systems/SPECThead/ARFTables/setEnergyDepositionThreshHold 328. keV
/gate/systems/SPECThead/ARFTables/setEnergyDepositionUpHold 400. keV
/gate/systems/SPECThead/ARFTables/setEnergyResolution 0.10
/gate/systems/SPECThead/ARFTables/setEnergyOfReference 140. keV
```

In this example, we shot photons with 364.5 keV as kinetic energy. We chose an energy resolution of 10% @ 140 keV and the energy window was set to [328-400] keV. The simulated energy resolution at an energy Edep will be calculated by:

$FWHM = 0.10 * \sqrt{140 * Edep}$ where Edep is the photon deposited energy.

If we want to discard photons which deposit less than 130 keV, we may use:

```
/gate/systems/SPECThead/setEnergyDepositionThreshHold 130. keV
```

The ARF tables depend strongly on the distance from the detector to the source used in the previous step. The detector plane is set to be the half-middle plan of the detector part of the SPECT head. In our example, the translation of the SPECT head was 34.5 cm along the X direction (radial axis), the detector was 2 cm wide along X and its center was located at x = 1.5 cm with respect to the SPECThead frame. This is what we call the detector plane (x = 1.5 cm) so the distance from the source to the detector plane is 34.5 + 1.5 = 36 cm:

```
# DISTANCE FROM SOURCE TO DETECTOR PLANE TAKEN TO BE THE PLANE HALF-WAY THROUGH THE␣
↪CRYSTAL RESPECTIVELY TO THE SPECTHEAD FRAME : here it is 34.5 cm + 1.5 cm
/gate/systems/SPECThead/ARFTables/setDistanceFromSourceToDetector 36 cm
```

The tables are then computed from a text file which contains information regarding the incident photons called ARF-Data.txt which is provided in https://github.com/OpenGATE/GateContrib/tree/master/imaging/ARF

```
# NOW WE ARE  READY TO COMPUTE THE ARF TABLES
/gate/systems/SPECThead/ARFTables/ComputeTablesFromEnergyWindows ARFData.txt
```

The text file reads like this:

```
#   this file contains all the energy windows computed during first step
#   with their associated root files
#   it has to be formatted the following way
# [Emin,Emax] is the energy window of the incident photons
# the Base FileName is the the name of the root files name.root, name_1.root name_2.
↪root ...
# the number of these files has to be given as the last parameter
#
# enter the data for each energy window
# Incident Energy Window: Emin - Emax (keV) | Root FileName | total file number
                        0.     365.         test1head            20
```

Here we have only one incident energy window for which we want to compute the corresponding ARF tables. The data for this window are stored inside 20 files whose base file name is test1head. These ARF data files were generated from the first step and were stored under the names of test1head.root, test1head_1.root ... test1head_19.root.

Finally the computed tables are stored to a binary file:

```
/gate/systems/SPECThead/ARFTables/list
# SAVE ARF TABLES TO A BINARY FILE FOR PRODUCTION USE
/gate/systems/SPECThead/ARFTables/saveARFTablesToBinaryFile ARFSPECTBench.bin
```

Use of the ARF tables

The command to tell GATE to use ARF tables is:

```
/gate/systems/SPECThead/arf/setARFStage useTables
```

The ARF sensitive detector is attached to the SPECThead:

```
/gate/SPECThead/attachARFSD
```

These tables are loaded from binary files with:

```
/gate/systems/SPECThead/ARFTables/loadARFTablesFromBinaryFile ARFTables.bin
```

3.3.10 Multi-system approaches: how to use more than one system in one simulation set-up ?

Singles arriving from different systems request different treatment in the digitizer. So we have to use multiple digitizer chains and to separate between theses singles according to their systems.

SystemFilter

The systemFilter module separates between the singles coming from systems. This module have one parameter which is the name of the system:

```
/gate/digitizer/SingleChain/insert systemFilter
/gate/digitizer/SingleChain/systemFilter/selectSystem SystemName
```

SingleChain is the singles chain, Singles by default, and SystemName is the name of the selected system.

Suppose that we have two systems with the names “cylindricalPET” and “Scanner_1”, so to select singles in cylindricalPET system we use:

```
/gate/digitizer/Singles/insert systemFilter
/gate/digitizer/Singles/systemFilter/selectSystem cylindricalPET
```

Note we didn’t insert a singles chain because we have the default chain “Singles”, on the other side for “Scanner_1” we to insert a new singles chain “Singles_S1”:

```
/gate/digitizer/name Singles_S1
/gate/digitizer/insert singleChain
```

Then we insert the system filter:

```
/gate/digitizer/Singles_S1/insert systemFilter
/gate/digitizer/Singles_S1/systemFilter/selectSystem Scanner_1
```

How to manage the coincidence sorter with more than one system: the Tri Coincidence Sorter approach

The aim of this module is to obtain the coincidence between coincidence pulses and singles of another singles chain (between TEP camera and scanner for example). In this module we search the singles, of the concerned singles chain, which are in coincidence with the coincidence pulses according to a time window. In fact, this module, from the point

of view of the code, is coincidence pulses processor. So we have a coincidence chain as input with a singles chain. We have also to define a time window to search the coincident coincidence-singles within this window. We test the time difference between the average time of the tow singles of the coincidence pulse and the time of every single of the singles chain in question. We have as output of this module two trees of ROOT: one tree contain the coincidences which have at first one coincident single with every one of them. The second tree contain the coincident singles and it is generated automatically with name of coincidence tree+"CSingles". For example if we call the coincidence "triCoinc", so the name of the singles tree will be " triCoincCSingles". This singles tree contain the same branches as any singles tree with an additional branch named "triCID" from (tri coincidence ID) and it has the same value for all singles which are in coincidence with one coincidence pulse.

In this module we store the singles event by event in a singles buffer that have a semi-constant size that the user can adjust. When we have a coincidence pulse we compare between the average time of this coincidence pulse and the time of each single pulse in the buffer. When we have coincident singles with the coincidence pulse, we store them in two trees as mentioned above.

Define the classical coincidence sorter applied on the cylindrical PET system:

```
/gate/digitizer/Coincidences/setWindow 30. ns
```

Define now a "coincidenceChain" where you will plug the first coincidence sorter (named 'Coincidences' in that case):

```
/gate/digitizer/name TriCoinc
/gate/digitizer/insert coincidenceChain
/gate/digitizer/TriCoinc/addInputName Coincidences
```

Finally, we call the 'triCoincProcessor' module and we plug on it the second system which define in that case by the 'Singles_S1' chain:

```
/gate/digitizer/TriCoinc/insert triCoincProcessor
/gate/digitizer/TriCoinc/triCoincProcessor/setSinglesPulseListName Singles_S1
/gate/digitizer/TriCoinc/triCoincProcessor/setWindow 15 ns
/gate/digitizer/TriCoinc/triCoincProcessor/setSinglesBufferSize 40
```

3.4 Data output

Table of Contents

In GATE, there are several types of output format, which can be enabled. By default all outputs are disabled and moreover there is no default output file name. You **must** give a file name, otherwise the output module will be automatically disabled.

All the output commands must always be written after the initialization line:

```
/gate/output/...
```

3.4.1 ASCII and binary outputs

The GateToASCII/GateToBinary classes enable the ASCII/**binary** file output. It allows you to process your raw data with your own tools. On the other hand, this output is not compressed and the output files are very large. If the ASCII/**binary** files are not needed for analysis, it is strongly recommended not using this output to speed up the simulation and save space:

```
/gate/output/ascii(**binary**)/enable
/gate/output/ascii/setFileName   test

# enable ascii(**binary**) output for hits
/gate/output/ascii(**binary**)/setOutFileHitsFlag   1

# enable ascii(**binary**) output for Singles (end of digitizer chain)
/gate/output/ascii(**binary**)/setOutFileSinglesFlag   1

# enable ascii(**binary**) output for coincidences
/gate/output/ascii(**binary**)/setOutFileCoincidencesFlag   1

# enable ascii(**binary**) output for singles (after a digitizer module)
/gate/output/ascii(**binary**)/setOutFileSingles< name of the digitizer module >Flag
↪ 1
```

The names of the digitizer module are : *Adder*, *Readout*, *Spblurring*, *Blurring*, *Thresholder*, *Upholder*. Their actions are explained in *Digitizer and readout parameters*.

To disable these ASCII(**binary**) files which can be large, the macro should contain the following lines:

```
/gate/output/ascii(**binary**)/setOutFileHitsFlag            0
/gate/output/ascii(**binary**)/setOutFileSinglesFlag         0
/gate/output/ascii(**binary**)/setOutFileCoincidencesFlag    0
```

Only the file *gateRun.dat* which contain the number of decay per run will then be created.

Description of the ASCII(binary) file content

The units are :

- MeV (energy)
- mm (position)
- s (time)
- deg (angle)

Hits file : gateHits.dat(.bin)

Each line is a hit and the columns represent :

- Column 1 : ID of the run (i.e. time-slice) **(4-bytes, G4int)**
- Column 2 : ID of the event **(4-bytes, G4int)**
- Column 3 : ID of the primary particle whose descendant generated this hit **(4-bytes, G4int)**
- Column 4 : ID of the source which emitted the primary particle **(4-bytes, G4int)**
- Columns 5 to N+4 : Volume IDs at each level of the hierarchy of a system, so the number of columns depends on the system used.

Example : cylindricalPET system N=6

- Column 5 : ID of volume attached to the "base" level of the system **(4-bytes, G4int)**
- Column 6 : ID of volume attached to the "rsector" level of the system **(4-bytes, G4int)**

- Column 7 : ID of volume attached to the “module” level of the system **(4-bytes, G4int)**
- Column 8 : ID of volume attached to the “submodule” level of the system **(4-bytes, G4int)**
- Column 9 : ID of volume attached to the “crystal” level of the system **(4-bytes, G4int)**
- Column 10 : ID of volume attached to the “layer” level of the system **(4-bytes, G4int)**

Example : SPECTHead system N=3

- Column 5 : ID of volume attached to the “base” level of the system **(4-bytes, G4int)**
- Column 6 : ID of volume attached to the “crystal” level of the system **(4-bytes, G4int)**
- Column 7 : ID of volume attached to the “pixel” level of the system **(4-bytes, G4int)**
- Column N+5 : Time stamp of the hit **(8-bytes, G4double)**
- Column N+6 : Energy deposited by the hit **(8-bytes, G4double)**
- Column N+7 : Range of particle which has generated the hit **(8-bytes, G4double)**
- Column N+8, N+9 ,N+10 : XYZ position of the hit in the world referential **(8-bytes, G4double)**
- Column N+11 : Geant4 code of the particle which has generated the hit (11 for Electrons & 22 for Photons) **(4-bytes, G4int)**
- Column N+12 : ID of the particle which has generated the hit **(4-bytes, G4int)**
- Column N+13 : ID of the mother of the particle which has generated the hit **(4-bytes, G4int)**
- Column N+14 : ID of the photon giving the particle which has generated the hit **(4-bytes, G4int)**
- Column N+15 : Number of Compton interactions in phantoms before reaching the detector **(4-bytes, G4int)**
- Column N+16 : Number of Rayleigh interactions in phantoms before reaching the detector **(4-bytes, G4int)**
- Column N+17 : Name of the process which has generated the hit **(8-bytes, G4string)**
- Column N+18 : Name of the last volume where a Compton effect occurred **(8-bytes, G4string)**
- Column N+19 : Name of the last volume where a Rayleigh effect occurred **(8-bytes, G4string)**

Singles files : gateSingles.dat(.bin)

The system is set as a cylindricalPET system. Each line is a single and the columns are :

- Column 1 : ID of the run (i.e. time-slice) **(4-bytes, G4int)**
- Column 2 : ID of the event **(4-bytes, G4int)**
- Column 3 : ID of the source **(4-bytes, G4int)**
- Column 4, 5, 6 : XYZ position of the source in world referential **(8-bytes, G4double)**
- Column 7 to 12 : Volume IDs*(cf. columns 5-10 of sec 11.) **(4-bytes, G4int)**
- Column 13 : Time stamp of the single **(8-bytes, G4double)**
- Column 14 : Energy deposited by the single **(8-bytes, G4double)**
- Column 15 to 17 : XYZ position of the single in the world referential **(8-bytes, G4double)**
- Column 18 : Number of Compton interactions in phantoms before reaching the detector **(4-bytes, G4int)**
- Column 19 : Number of Compton interactions in detectors before reaching the detector **(4-bytes, G4int)**
- Column 20 : Number of Rayleigh interactions in phantoms before reaching the detector **(4-bytes, G4int)**

- Column 21 : Number of Rayleigh interactions in detectors before reaching the detector **(4-bytes, G4int)**
- Column 22 : Name of the phantom where a Compton effect occured **(8-bytes, G4string)**
- Column 23 : Name of the phantom where a Rayleigh effect occured **(8-bytes, G4string)**

Coincidences files : gateCoincidences.dat(.bin)

The system is set as a cylindricalPET system. Each line is a coincidence created with two singles and the columns are :

- Column 1 : ID of the run (i.e. time-slice) (first single) **(4-bytes, G4int)**
- Column 2 : ID of the event (first single) **(4-bytes, G4int)**
- Column 3 : ID of the source (first single) **(4-bytes, G4int)**
- Column 4 to 6 : XYZ position of the source in world referential (first single) **(8-bytes, G4double)**
- Column 7 : Time stamp (first single) (8-bytes, G4double) **(8-bytes, G4double)**
- Column 8 : Deposited energy (first single) (8-bytes, G4double) **(8-bytes, G4double)**
- Column 9 to 11 : XYZ position in the world referential (first single) **(8-bytes, G4double)**
- Column 12 to 17 : volume IDs* (first single)
- For binary : Column 12 and 13 **(8-bytes, G4double)**
- For binary : Column 14 **(8-bytes, G4double)**
- For binary : Column 15 to 17 **(4-bytes, G4int)**
- Column 18 : Number of Compton interactions in phantoms before reaching the detector (first single) **(4-bytes, G4int)**
- Column 19 : Number of Compton interactions in detectors before reaching the detector (first single) **(4-bytes, G4int)**
- Column 20 : Number of Rayleigh interactions in phantoms before reaching the detector (first single) **(4-bytes, G4int)**
- Column 21 : Number of Rayleigh interactions in detectors before reaching the detector (first single) **(4-bytes, G4int)**
- Column 22 : Scanner axial position (first single) **(8-bytes, G4double)**
- Column 23 : Scanner angular position (first single) **(8-bytes, G4double)**
- Column 24 : ID of the run (i.e. time-slice) (second single) **(4-bytes, G4int)**
- Column 25 : ID of the event (second single) **(4-bytes, G4int)**
- Column 26 : ID of the source (second single) **(4-bytes, G4int)**
- Column 27 to 29 : XYZ position of the source in world referential (second single) **(8-bytes, G4double)**
- Column 30 : Time stamp (second single) **(8-bytes, G4double)**
- Column 31 : Energy deposited (second single) **(8-bytes, G4double)**
- Column 32 to 34 : XYZ position in the world referential (second single) **(8-bytes, G4double)**
- Column 35 to 40 : volume IDs
- For binary : Column 35 and 36 **(8-bytes, G4double)**

- For binary : Column 37 **(8-bytes, G4double)**
- For binary : Column 38 to 40 **(4-bytes, G4int)**

The number of different volumeIDs depends on the complexity of the system geometry (6 IDs for cylindricalPET system, 3 for ECAT system, ...). Then, the number of column of your ASCII file is not constant, but system-dependent.

- Column 41 : Number of Compton interactions in phantoms before reaching the detector (second single) **(4-bytes, G4int)**
- Column 42 : Number of Compton interactions in detectors before reaching the detector (second single) **(4-bytes, G4int)**
- Column 41 : Number of Rayleigh interactions in phantoms before reaching the detector (second single) **(4-bytes, G4int)**
- Column 42 : Number of Rayleigh interactions in detectors before reaching the detector (second single) **(4-bytes, G4int)**
- Column 45 : Scanner axial position (second single) **(8-bytes, G4double)**
- Column 46 : Scanner angular position (second single) **(8-bytes, G4double)**

Selection of the variables in Singles/Coincidences ASCII and binary outputs

The user can select which variables he/she wants in the ASCII(**binary**) file. The mechanism is based on a series of 0/1, one for each variable. By default all variables are enabled, but one can choose to enable only some of the variables listed in 10.4.1:

```
/gate/output/ascii(**binary**)/setCoincidenceMask       1 0 1 0 1 1
/gate/output/ascii(**binary**)/setSingleMask            0 0 1 1
```

Note: the VolumeID variables are enabled/disabled together, as a group. The component of the 3D vectors, instead, like the positions (components x,y,z), are enabled/disabled one by one.

Large files: automatic file swap

When a user defined limit is reached by the Coincidence or Single ASCII(**binary**) output file, by default Gate closes the file and opens another one with the same name but a suffix _1 (and then _2, and so on). By default the file limit is set to 2000000000 bytes. One can change the number of bytes with a command like:

```
/gate/output/ascii(**binary**)/setOutFileSizeLimit 30000
```

If the value is < 10000, no file swapping is made (to avoid creating thousands of files by mistake).

For example, if one does not have any limit in the Operating System, one can put the number to 0, and there will be only one large (large) file at the end.

In case of high statistics applications, one might consider enabling only the ROOT output (see *Root output*), which contains the same information as the binary one, but automatically compressed and ready for analysis.

What is the file gateRun.dat(.bin)?

This file is the list of the number of decays generated by the source for each run (one by line). The Output manager is called for each event, even if the particle(s) of the decay do not reach the detector. Note that the number of processed decays can be slighly different from the expected number $\mathrm{N} = \mathrm{A} \times \Delta\mathrm{t}$ where A is the activity and $\Delta\mathrm{t}$ is the time

of the acquisition, due to the random character of the decay which governs the event generation (Poisson law). Gate generates the time delay from the previous event, if it is out of the time slice it stops the event processing for the current time slice and if needed it starts a new time slice.

3.4.2 Root output

Example:

```
/gate/output/root/enable
/gate/output/root/setFileName FILE_NAME
```

which will provide you with a FILE_NAME.root file. By default, this root file will contain: 2 Trees for SPECT systems (Hits and Singles) or 3 Trees for PET systems (Coincidences, Hits and Singles) in which several variables are stored.

If needed, and for a matter of file size, you could choose not to generate all trees. In this case, just add the following lines in your macro:

```
/gate/output/root/setRootHitFlag            0
/gate/output/root/setRootSinglesFlag        0
/gate/output/root/setRootCoincidencesFlag   0
/gate/output/root/setRootNtupleFlag         0
```

By turning to 1 (or 0) one of this tree flag, you will fill (or not) the given tree.

In a debug mode, it can be useful to store in a Tree the informations after the action of one particular module of the digitizer chain. The following flags exist to turn on or off these intermediate trees:

```
/gate/output/root/setOutFileSinglesAdderFlag          0
/gate/output/root/setOutFileSinglesReadoutFlag        0
/gate/output/root/setOutFileSinglesSpblurringFlag     0
/gate/output/root/setOutFileSinglesBlurringFlag       0
/gate/output/root/setOutFileSinglesThresholderFlag    0
/gate/output/root/setOutFileSinglesUpholderFlag       0
```

If you want to disable the whole ROOT output, just do not call it, or use the following command:

```
/gate/output/root/disable
```

Using TBrowser To Browse ROOT Objects

The ROOT graphical user interface TBrowser is a useful tool to interactively inspect and visualize produced simulation data.

Since Gate 8.0 new branches are included in the ROOT Hits Tree: trackLength, trackLocalTime, momDirX, momDirY and momDirZ. The additional information that is now available can be used for applications like timing resolution and surface treatment studies of scintillation crystals when surfaces are defined (see *Defining surfaces*).

When launching ROOT with the command in a terminal:

```
root FILE_NAME.root
root [1] TBrowser t
```

you can easily see the content of your ROOT data file.

Select desired outputfile (.root).

The trees (Hits, Singles etc.) will be filled according to the flags set to 1 in your .mac-file:

```
/gate/output/root/setRootHitFlag                1
```

The Hits tree is opened and shows many branches. Select a tree. Either double click on each branch to see histogrammed/plotted data or use root commands like:

```
Hits->Draw( "posX:posY:posZ")
```

This command plots the position of Hits in 3D.

Add conditions to specify your histogram e.g:

```
Hits->Draw("posX:posY:posZ","PDGEncoding==0")
```

This command plots the position of Hits that are optical photons(PDGEncoding=0) in 3D:

```
Hits->Draw("posX:posY:posZ","PDGEncoding==0 && time<=1 ")
```

Multiple conditions can be added e.g.: 3D position of optical photons in the first second of the simulation.

- PDGEncoding (Particle Data Group): The type of particle can be obtained (e.g.: "0" optical photon; "22" gamma particle; for a complete list visit: http://pdg.lbl.gov/2007/reviews/montecarlorpp.pdf).
- trackLocalTime[s]: (available starting Gate 8.0) The time that it takes a particle to complete a track.

$t_0 =$ start of particles path

$t_{max} =$ end of path

It correlates directly to the trackLength according to the following formula:

$trackLocalTime[s] = \frac{trackLength[mm]*10^{-3}*n}{c}$

$n =$ refractive index of medium

$c =$ speed of light $= 2.99792458 * 10^8 m$

- time[s]: The absolute time of a hit in the sensitive detector.

$t_0 =$ start of particles path

$t_{max} =$ end of path

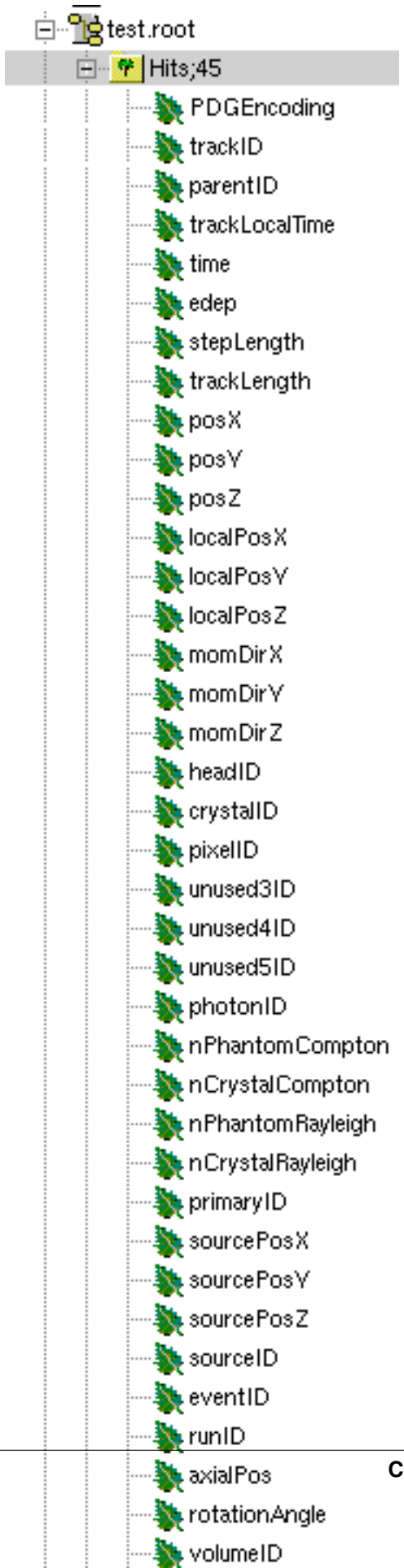
- stepLength[mm]: The distance between two interactions of a particle (e.g.: distance between a gamma particle entering a sensitive volume and being scattered)
- trackLength[mm]: (available starting Gate 8.0) The total distance of one particle often including multiple steps. Can also be derived by the trackLocalTime.
- momDirX,Y,Z: (available starting Gate 8.0) The momentum direction of a detected/absorbed particle in the sensitive detector consisting of three components that make a 3D vector.

Use:

```
Hits->Draw("momDirX: momDirY: momDirZ")
```

to look at vectors in 3D.

- processName: The process by which the particle ended its path in the sensitive detector (e.g.: Transportation ("T"), Optical Absorption("O"), Comptonscatter("C"), PhotoElectric("P"), RaleighScattering("R")). You might be interested in distinguishing between particles that are detected at the detector("T") and those that were absorbed("O"). A particle that undergoes Comptonscatter("C") is counted as two hits when it splits up.


test.root
Hits;45
PDGEncoding
trackID
parentID
trackLocalTime
time
edep
stepLength
trackLength
posX
posY
posZ
localPosX
localPosY
localPosZ
momDirX
momDirY
momDirZ
headID
crystalID
pixelID
unused3ID
unused4ID
unused5ID
photonID
nPhantomCompton
nCrystalCompton
nPhantomRayleigh
nCrystalRayleigh
primaryID
sourcePosX
sourcePosY
sourcePosZ
sourceID
eventID
runID
axialPos
rotationAngle
volumeID

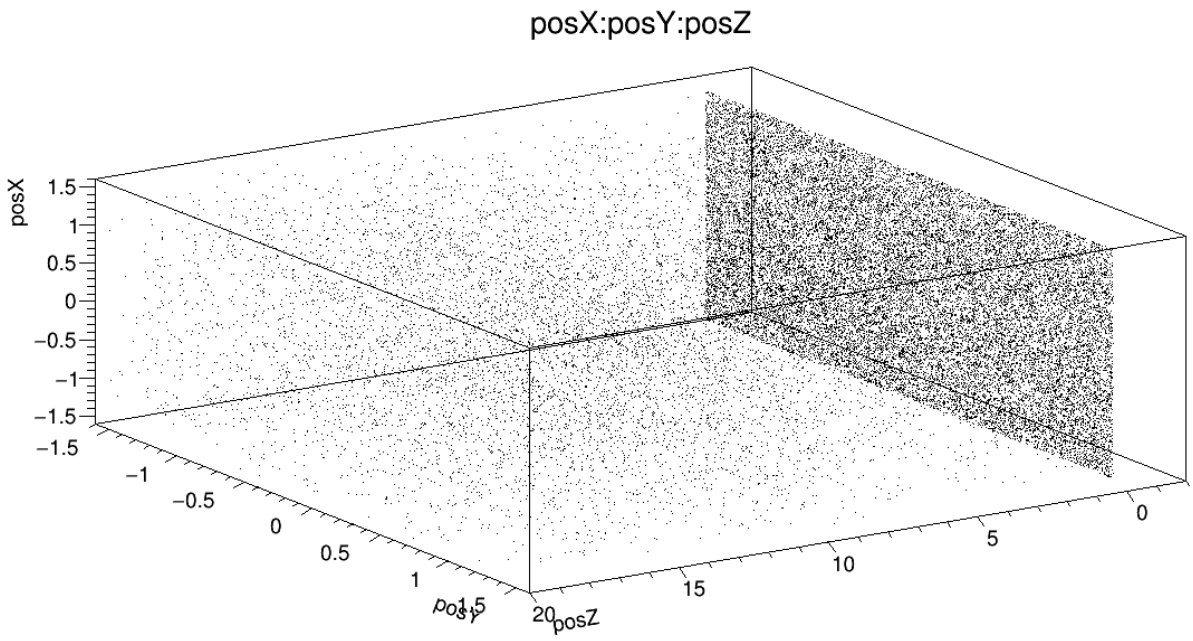


Fig. 3.25: Position of Hits in 3D

(for more information http://www-root.fnal.gov/root/GettingStarted/GettingStarted.htm)

How to merge Root files?

Two or more Root files can be merged into one single file by using the **hadd** utility on the command line:

```
hadd MergedFile.root file1.root file2.root ... fileN.root
```

How to analyze the Root output

You can either plot the variables directly from the browser, or through a macro file (e.g. called PET_Analyse.C). Analysis macros are available in https://github.com/OpenGATE/GateContrib/tree/master/imaging/ROOT_Analyse

In this case, after launching ROOT:

```
root [0] .x PET_Analyse.C
```

You may also use the root class called **MakeClass** (http://root.cern.ch/download/doc/ROOTUsersGuideHTML/ch12s21.html) which generates a skeleton class designed to **loop over the entries of a tree** from your root file. Please consult the ROOT Homepage: http://root.cern.ch/ for more details. In the location of your output.root file, launch root and do the following:

```
root [0] TChain chain("Hits");          <<<=== name of the tree of interest : Hits
root [1] chain.Add("output1.root");
root [1] chain.Add("output2.root");
root [2] chain.MakeClass("MyAnalysis"); <<<==== name of your macro : MyAnalysis.C
```

MakeClass() will automatically create 2 files : **MyAnalysis.h** (a header file) and **MyAnalysis.C** (template to loop over your events). You can run this code in ROOT by doing:

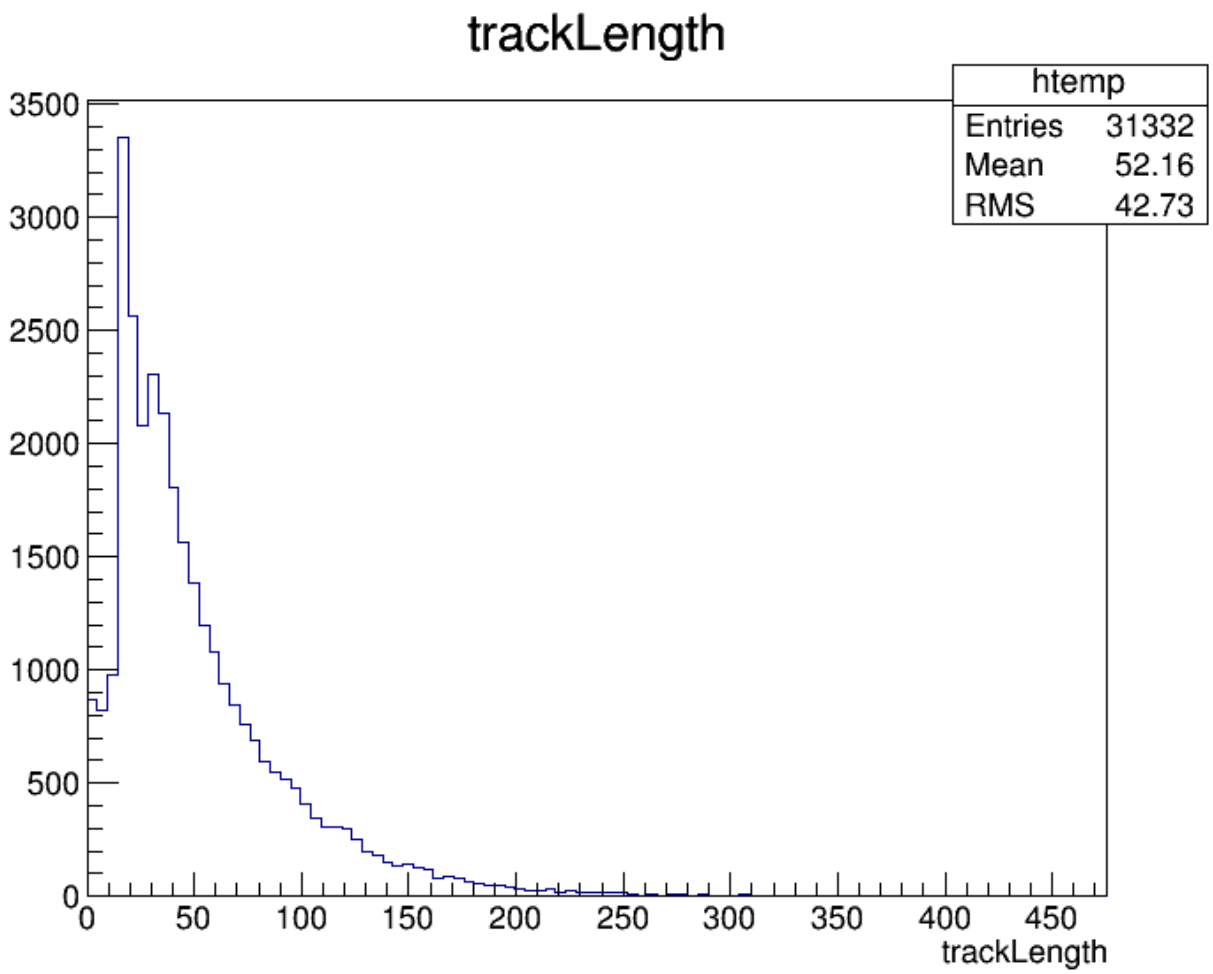


Fig. 3.26: trackLength

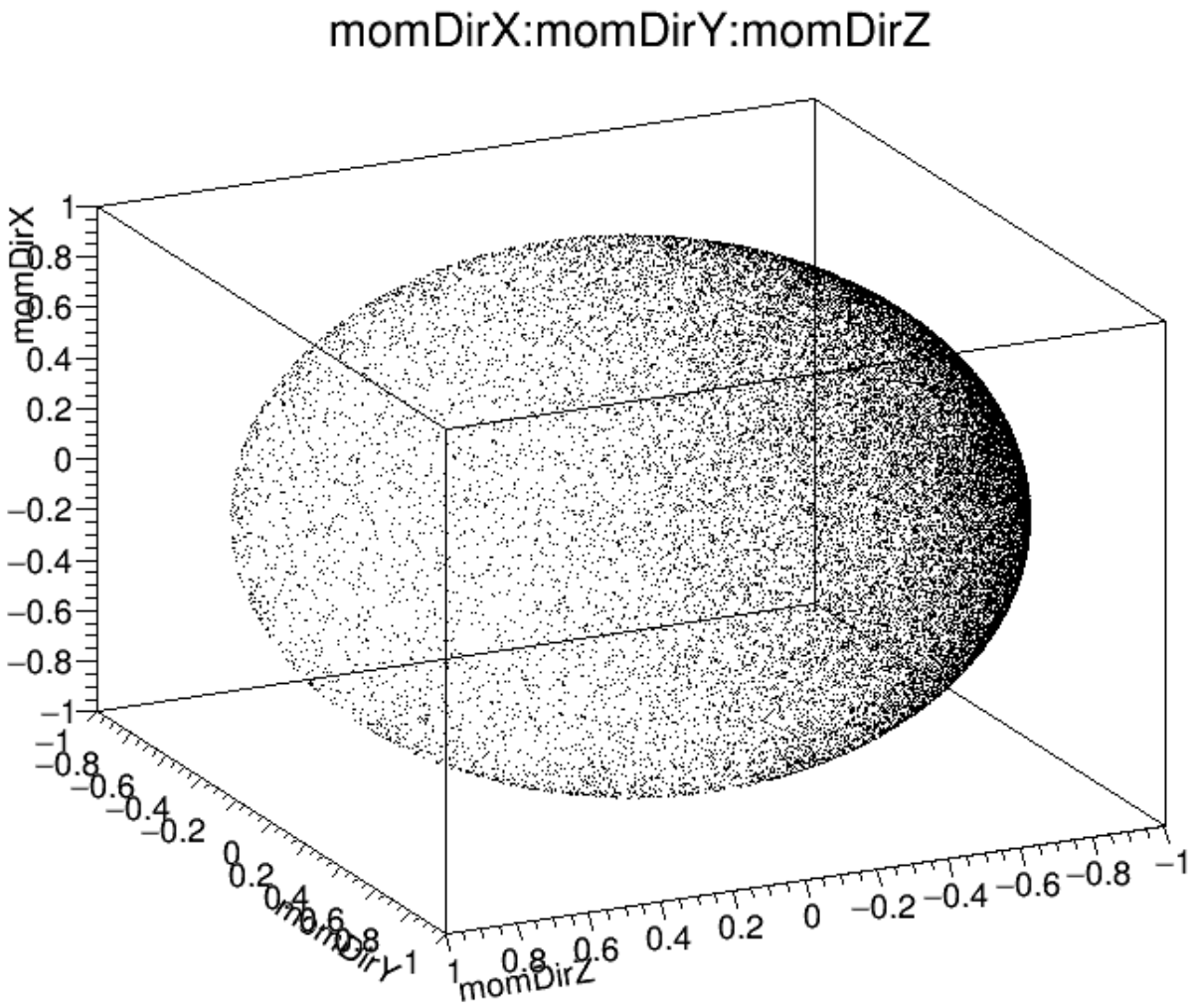


Fig. 3.27: Momentum direction of particles.

```
Root >  .L MyAnalysis.C
Root >  MyAnalysis t
Root >  t.Loop();
```

Writing a counter in ROOT

You can modify/improve the MyAnalysis.C macro by adding a counter as shown below:

```
void MyAnalysis::Loop()
{
if (fChain == 0) return;
Long64_t nentries = fChain->GetEntriesFast();
Long64_t nbytes = 0, nb = 0;
Int_t num_INITIAL = 0;
Int_t num_DETECTED = 0;

// Loop over photons
for (Long64_t jentry=0; jentry Long64_t ientry = LoadTree(jentry);
if (ientry < 0) break;
nb = fChain->GetEntry(jentry); nbytes += nb;
num_INITIAL++; // number of photons in the tree
if(HitPos_Y == 0.3)       <== here you could apply some cuts which are analysis␣
↪dependent
num_DETECTED++;
}
}// End Loop over the entries.

// You can print some results on the screen :
std::cout<<"***************************** Results *****************************" <<␣
↪std::endl;
std::cout<<"Number of Generated Photons: " << num_INITIAL << std::endl;
std::cout<<"Number of Detected Photons: " << num_DETECTED << std::endl;
```

Draw all ROOT tree branches in a postscript file

If you look at the GATE code optical example directory (https://github.com/OpenGATE/GateContrib/tree/master/imaging/Optical), you will see a macro named **DrawBranches.C**. If you modify it so it points to your root file and execute it in root:

```
root> .x DrawBranches.C
```

This will draw/plot all the branches of your tree into a postscript file. That might be helpful.

Convert a ROOT file to a text file

This link shows how to convert the data in a root file to a text file for further analysis: http://root.cern.ch/phpBB3/viewtopic.php?f=3&t=16590

```
// Name this file "dump.cxx" and use as:
// root [0] .x dump.cxx(); > dump.txt
// Produces "dump.txt" and "dump.xml" files.

void dump(const char *fname = "dna.root",
```

(continues on next page)

(continued from previous page)

```
const char *nname = "ntuple")      // <=== If needed, change this line.
{
if (!fname || !(*fname) || !nname || !(*nname)) return; // just a precaution

TFile *f = TFile::Open(fname, "READ");
if (!f) return; // just a precaution

TTree *t; f->GetObject(nname, t);
if (!t) { delete f; return; } // just a precaution

// See:
// http://root.cern.ch/root/html/TTreePlayer.html#TTreePlayer:Scan
// http://root.cern.ch/root/html/TTree.html#TTree:Scan
t->SetScanField(0);
t->Scan("*");

// See:
// http://root.cern.ch/root/html/TObject.html#TObject:SaveAs
t->SaveAs("dump.xml");
// t->SaveAs(TString::Format("%s.xml", nname));

delete f; // no longer needed (automatically deletes "t")
}
```

The ROOT online plotter

GATE provides a very convenient tool called the online plotter, which enables online display of several variables. This online analysis is available even if the root output is disabled in your macro, for instance because the user does not want to save a large root file. **But Gate have to be compiled with certain options to have this output available.** The online plotter can be easily used with the following macro:

```
/gate/output/plotter/enable
/gate/output/plotter/showPlotter
/gate/output/plotter/setNColumns                2                              <===␣
↪sets the number of display windows to be used
/gate/output/plotter/setPlotHeight              250
/gate/output/plotter/setPlotWidth               300
/gate/output/plotter/addPlot hist               Ion_decay_time_s               <===␣
↪plots an histogram previously defined in GATE
/gate/output/plotter/addPlot hist               Positron_Kinetic_Energy_MeV    <===␣
↪plots a variable from one of the GATE trees
/gate/output/plotter/addPlot tree Singles       comptonPhantom
/gate/output/plotter/addPlot tree Coincidences  energy1
/gate/output/plotter/listPlots
```

Fig. 3.28 presents an example of online plotter, obtained with the above macro.

3.4.3 Interfile output of projection set

The Interfile format is especially suited for acquisition protocol using a multiple headed rotating gamma camera. The total description of the Interfilev3.3 format can be found on the Interfile website: http://www.medphys.ucl.ac.uk/interfile/index.htm.

When images are acquired in multiple windows (e.g. energy windows, time windows, multiple heads), the images

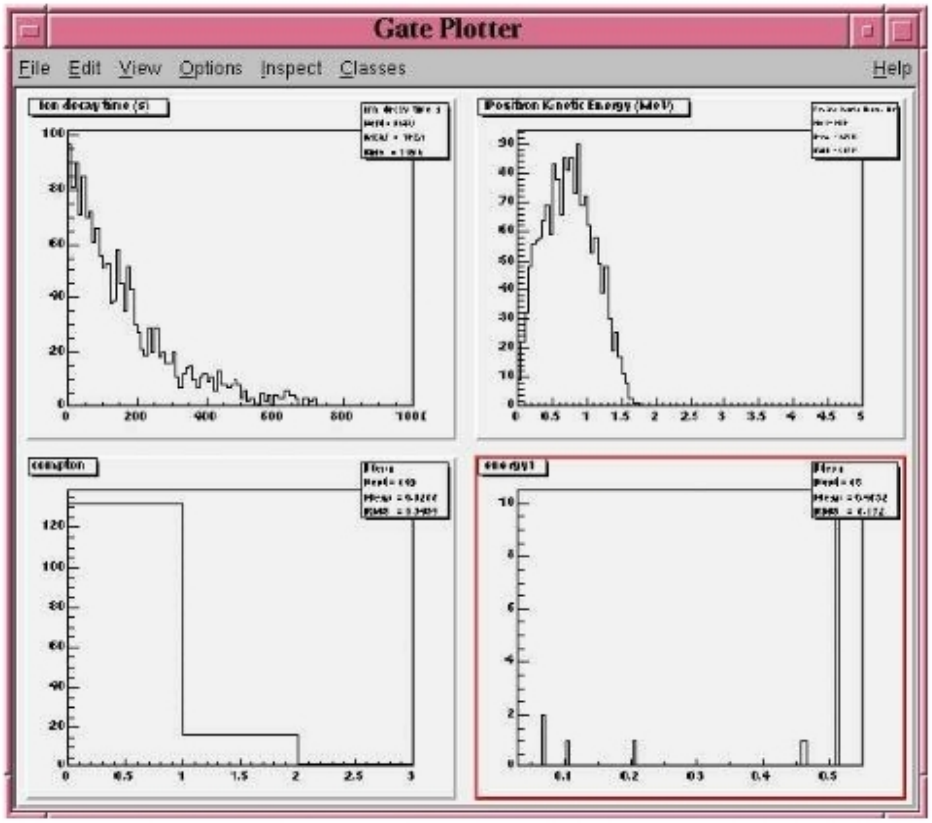


Fig. 3.28: The Online Plotter

are recorded according to the order in which the corresponding keys are defined. Thus if multiple energy windows are used, all image data for the first window must be given first, followed by the image data for the second window, etc. This loop structure is defined in the Interfile syntax by the use of the 'for' statement. Two files are created when using the Interfile/Projection output: *your_file.hdr* and *your_file.sin*. The header file contains all information about the acquisition while the *your_file.sin* file contains the binary information. An example of such a header is:

```
!INTERFILE :=
!imaging modality := nucmed
!version of keys := 3.3
date of keys := 1992:01:01
;
!GENERAL DATA :=
data description := GATE simulation
!data starting block := 0
!name of data file := your_file.sin
;
!GENERAL IMAGE DATA :=
!type of data := TOMOGRAPHIC
!total number of images := 64
study date := 2003:09:15
study time := 11:42:34
imagedata byte order := LITTLEENDIAN
number of energy windows := 1
;
!SPECT STUDY (general) :=
number of detector heads := 2
;
!number of images/energy window := 64
!process status := ACQUIRED
!number of projections := 32
!matrix size [1] := 16
!matrix size [2] := 16
!number format := UNSIGNED INTEGER
!number of bytes per pixel := 2
!scaling factor (mm/pixel) [1] := 1
!scaling factor (mm/pixel) [2] := 1
!extent of rotation := 180
!time per projection (sec) := 10
study duration (elapsed) sec : = 320
!maximum pixel count : = 33
;
!SPECT STUDY (acquired data) :=
!direction of rotation := CW
start angle := 0
first projection angle in data set := 0
acquisition mode := stepped
orbit := circular
camera zoom factor := 1
;
!number of images/energy window := 64
!process status := ACQUIRED
!number of projections := 32
!matrix size [1] := 16
!matrix size [2] := 16
!number format := UNSIGNED INTEGER
!number of bytes per pixel := 2
!scaling factor (mm/pixel) [1] := 1
!scaling factor (mm/pixel) [2] := 1
```

(continues on next page)

(continued from previous page)

```
!extent of rotation := 180
!time per projection (sec) := 10
study duration (elapsed) sec : = 320
!maximum pixel count : = 36
;
!SPECT STUDY (acquired data) :=
!direction of rotation := CW
start angle := 180
first projection angle in data set := 180
acquisition mode := stepped
orbit := circular
camera zoom factor := 1
;
GATE GEOMETRY :=
head x dimension (cm) := 30
head y dimension (cm) := 80
head z dimension (cm) := 70
head material := Air
head x translation (cm) := -25
head y translation (cm) := 0
head z translation (cm) := 0
crystal x dimension (cm) := 1.5
crystal y dimension (cm) := 60
crystal z dimension (cm) := 50
crystal material := NaI
;
GATE SIMULATION :=
number of runs := 32
;
!END OF INTERFILE :=
```

To use the Interfile output, the following lines have to be added to the macro:

```
# PROJECTION
/gate/output/projection/enable
/gate/output/projection/setFileName      your_file
/gate/output/projection/projectionPlane  YZ
/gate/output/projection/pixelSizeY       1. mm
/gate/output/projection/pixelSizeX       1. mm
/gate/output/projection/pixelNumberY     16
/gate/output/projection/pixelNumberX     16
```

The projectionPlane should be chosen correctly, according to the simulated experiment. The pixelSize and the pixelNumber are always described in a fixed XY-axes system.

Reading an interfile image with ImageJ

The Interfile Output is available as a ".sin" and ".hdr" files directly into the folder of concern. Several software may be used to read the data, among them the software ImageJ is quite often used. The procedure to use is the following:

Once ImageJ is opened, click on the thumb **File** and select **Import -> Raw**. A window appears into which the **name.sin** can be selected.

Once the image is selected, select the following information:

- Image Type: *16-bit Unsigned*

- *Width & Height & Number of Images* can be read into the **.hdr** files if unknown.
- Tick the case: *Little Endian byte Order*
- Tick the case: *Use Virtual Stack* if the data had multiple projection windows.

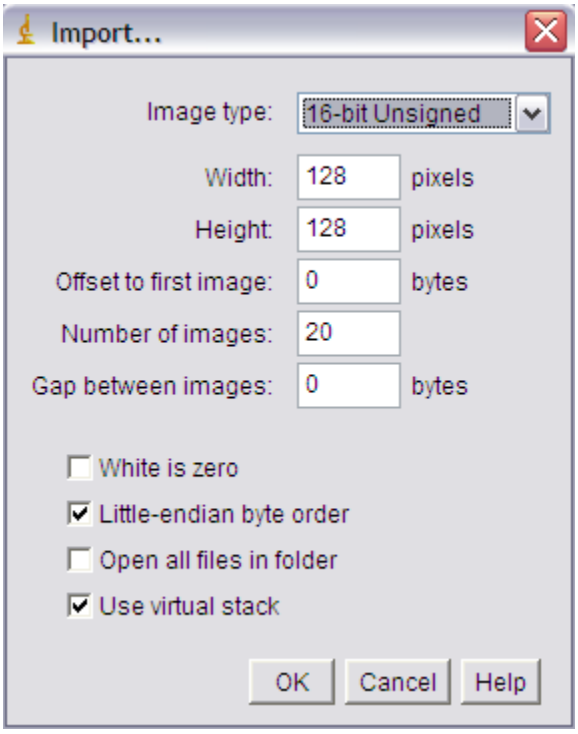


Fig. 3.29: Window snapshot in ImageJ for .sin files.

However one must be careful with this editing. Some users complained that the image in tomographic views provided image in stack in a strange fashion.

A second way to read Interfile images is to use this plugin with ImageJ Interfile Plugin Decoder. The advantage is that the plugin seeks all the information in the .hdr files by itself.

Reading an interfile image with IDL

For a planar projection, the image projections created with GATE may also be read with IDL with the function *Read_Binary"". In the example below, the projection *name.sin* has to be inserted into the IDL main folder. The image size must be detailed into the READ_BINARY function which might lead to a false image if not specified properly. If in doubt, the image size information is to be obtained in the .hdr files.

- **IDL>** file = 'name.sin'
- **IDL>** SizeIMageX = 128
- **IDL>** SizeImageZ = 128
- **IDL>** data=READ_BINARY(file,DATA_DIMS=[SizeIMageX,SizeIMageY],DATA_TYPE=12,ENDIAN='Little')

3.4.4 Sinogram output

If the ecat system or the ecatAccel system have been selected (see *Ecat*), the sinogram output module can be enable with the following commands:

For the **ecat** system:

```
/gate/output/sinogram/enable
/gate/output/sinogram/setFileName MySinogramFileName
```

For the **ecatAccel** system:

```
/gate/output/sinoAccel/enable
/gate/output/sinoAccel/setFileName MySinogramFileName
```

Using this format, the coincidence events are stored in an array of 2D sinograms. There is one 2D sinogram per pair of crystal-rings. For example, for the ECAT EXACT HR+ scanner (32 crystal-rings) from CPS Innovations, there are 1024 2D sinograms. The number of radial bins is specified using the command:

For the ecat system:

```
/gate/output/sinogram/RadialBins 256
```

For the ecatAccel system:

```
/gate/output/sinoAccel/RadialBins 256
```

There is a one-to-one correspondence between the sinogram bins and the lines-of-response (LOR) joining two crystals in coincidence. The sinogram bin assignment is not based on the true radial and azimuthal position of the LOR, but on the indexing of the crystals. This means that the sinograms are subject to curvature effects. By default, all coincident events are recorded, regardless of their origin (random, true unscattered or true scattered coincidence). It is possible to discard random events:

For the ecat system:

```
/gate/output/sinogram/TruesOnly true
```

For the ecatAccel system:

```
/gate/output/sinoAccel/TruesOnly true
```

In the trues, both scattered and unscattered coincidences are included. There is no simulation of a delayed coincidence window. At the beginning of each run, the content of the 2D sinograms is reset to zero. At the end of each run, the contents of the 2D sinograms can be optionally written to a raw file (one per run). This feature has to be enabled:

For the ecat system:

```
/gate/output/sinogram/RawOutputEnable
```

For the ecatAccel system:

```
/gate/output/sinoAccel/RawOutputEnable
```

Three files are written per run:

- the raw data (unsigned short integer) in MySinogramFileName.ima
- a mini ASCII header in MySinogramFileName.dim **<=== contains the minimal information required to read MySinogram-FileName.ima**
- an information file in MySinogramFileName.info **<=== describes the ordering of the 2D sinograms in MySinogram-FileName.ima.**

Here is an example of a header file with the default settings for the ECAT EXACT HR+ scanner:

```
288 288 1024     <=== size of the matrix : 1024 2D sinograms with 288 radial bins and␣
↪288 azimuthal bins
-type U16        <=== format : unsigned short integer
-dx 1.0          <=== size of x-bin; set arbitrarly to 1.
-dy 1.0          <=== size of y-bin; set arbitrarly to 1.
-dz 1.0          <=== size of z-bin; set arbitrarly to 1.
```

Here is an example of the information file with the default settings for the ECAT EXACT HR+ scanner:

```
1024 2D sinograms
[RadialPosition;AzimuthalAngle;AxialPosition;RingDifference]
RingDifference varies as 0,+1,-1,+2,-2, ...,+31,-31
AxialPosition varies as |RingDifference|,...,62-|RingDifference| per increment of 2
AzimuthalAngle varies as 0,...,287 per increment of 1
RadialPosition varies as 0,...,287 per increment of 1
Date type : unsigned short integer (U16)
```

Each 2D sinogram is characterized by the two crystal-rings in coincidence ring1 and ring2 . Instead of indexing the 2D sinograms by ring1 and ring2 , they are indexed by the ring difference ring2 ring1 and the axial position ring2 + ring1:

```
for RingDifference = 0,+1,-1,+2,-2,....,+31,-31
 for AxialPosition = |RingDifference|; AxialPosition <= 62-|RingDifference|;␣
↪AxialPosition += 2
   ring_1 = (AxialPosition - RingDifference)/2
   ring_2 = RingDifference + (AxialPosition - RingDifference)/2
   Write Sinogram(ring_1;ring_2)
```

In addition to the sinogram output module, there is a conversion of the 2D sinograms to an ecat7 formatted 3D sinogram in the ecat7 output module. This 3D sinogram is then written to an ecat7 matrix file.

3.4.5 Ecat7 output

If and only if both the ecat system and the sinogram output module have been selected, the ecat7 output module can be enable using the following commands:

```
/gate/output/ecat7/enable
/gate/output/ecat7/setFileName MySinogramFile
```

This module writes the content of the 2D sinograms defined in the sinogram output module to an ecat7 formatted matrix scan file, the native file format from CPS Innovations (Knoxville (TN), U.S.A.) for their *ECAT* scanner family. Due to the large size of a full 3D PET data set, the data set size is reduced before writing it to disk. Therefore it is not possible to go back from an *ecat7* formatted 3D sinogram to the original 2D sinograms set.

Installation

In order to compile the ecat7 output module of Gate, the ecat library written at the PET Unit of the Catholic University of Louvain-la-Neuve (UCL, Belgium) is required. It can be downloaded from their web site: http://www.topo.ucl.ac.be/ecat_Clib.html

Three files are required: the library file libecat.a and the two header files matrix.h and machine_indep.h.

To compile Gate with the ecat7 library without changing the env_gate.csh and GNUmakefile files, the environment variable ECAT7_HOME has to be defined and set to the name of the home directory where the ecat7 library is installed (for example, /usr/local/ecat7). In this ecat7 home directory, two subdirectories should be created : lib and include.

The header files are put in the ${ECAT7_HOME}/include directory. For each system, a specific subdirectory named after the G4SYSTEM environment variable value should be created in the ${ECAT7_HOME}/lib directory. The corresponding library file libecat.a has to be located in this ${ECAT7_HOME}/lib/${G4SYSTEM} directory. The *matrix.h* file has to be modified to add the declaration of the mh_update() function. The following line can be added in the "high level user functions" part of matrix.h:

```
int mh_update(MatrixFile*);
```

Data reduction

The polar coordinate of a LOR is approximately defined by the crystal-ring index difference between the 2 rings in coincidence. For a scanner with N crystal rings, the total number of polar samples is given by 2 × N 1. Usually, on ecat systems, not all crystal-ring differences are recorded. Only absolute crystal-ring differences up to a given value, referred to as the maximum ring difference, are recorded. In Gate, this maximum ring difference is defined using:

```
/gate/output/ecat7/maxringdiff 22
```

The value of the maximum ring difference should be smaller than N.

A polar mashing is applied to group 2D sinograms with adjacent polar coordinates. The size of this grouping is called the span [reference]. Its minimum value is 3 and it should be an odd integer. The span value can be set using:

```
/gate/output/ecat7/span 9
```

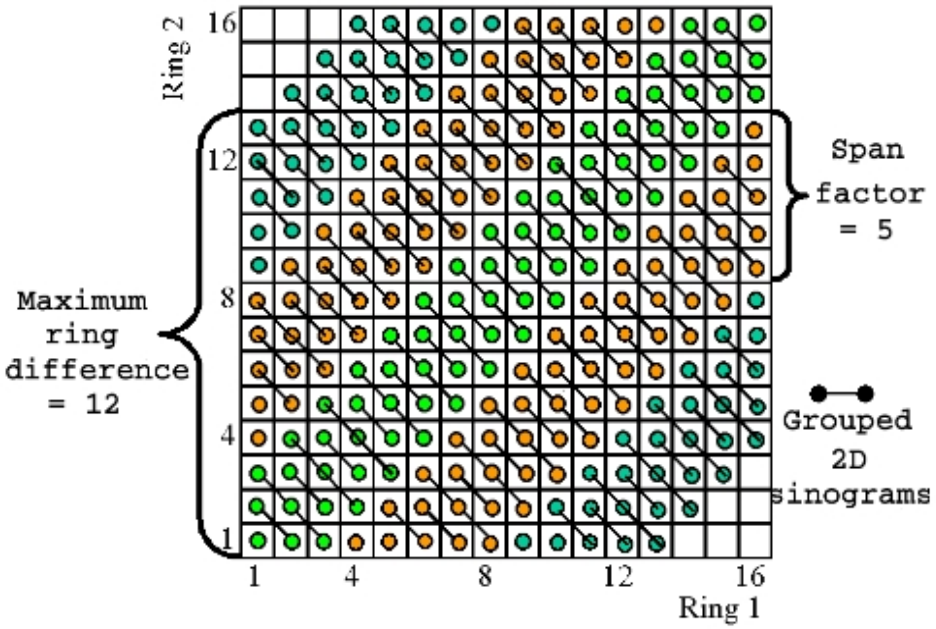


Fig. 3.30: Michelogram for a 16 crystal-ring scanner

The *Michelogram* represented in Fig. 3.30 graphically illustrates mashing in the polar coordinate for a 16 crystal-ring scanner with a maximum ring difference set to 12 and a span factor of 5, resulting to 5 polar samples instead of 31. Each dot represents a 2D sinogram for a given pair of crystal-rings. The grouped 2D sinograms are connected by diagonal lines.

By default, the maximum ring difference is set to $N - 1$ and the span factor to 3. After choosing a maximum ring difference value $MaxRingDiff$, only certain *span* factors are possible as the resulting number of polar samples must be an integer:

$\frac{2 \times MaxRingDiff+1}{span}$

In addition to the polar mashing, the number of azimuthal samples can also be reduced from $N_{azi} = N_{cryst}/2$ to N_{azi}/m where m is the mashing factor. The mashing factor can be set using:

```
/gate/output/ecat7/mashing 2
```

The default mashing value is 1.

Sinogram file

At the end of each run, a new 3D sinogram is written with an incremental frame indexing. For example, with the following configuration, 5 frames of 60 seconds each will be generated:

```
/gate/application/setTimeSlice   60 s
/gate/application/setTimeStart    0 s
/gate/appication/setTimeStop    300 s
```

The *ECAT* code of the scanner model is specified by:

```
/gate/output/ecat7/system 962
```

This information can be needed by some **ecat7** based reconstruction routines.

It should be noted that not all fields of the main-header or sub-header are filled. In particular, the *coincidence_sampling_mode* field of the main-header is always set to *Prompts and Delayed* (1), regardless of the value of the /gate/output/sinogram/TruesOnly tag.

For the scan sub-header, the value of the *prompts* field is correctly filled and the value of the *delayed* field is set to the actual number of random coincidences, and not to the number of delayed coincidences (not simulated).

The radial bin size in the scan sub-header is set to half the value of the crystal transverse sampling and does not take into account the arc and depth-of-interaction (DOI) effects. After arc correction, the radial bin size should be slightly increased to account for the DOI effect. Note that this correction is included in the reconstruction software provided with the *ECAT* scanners.

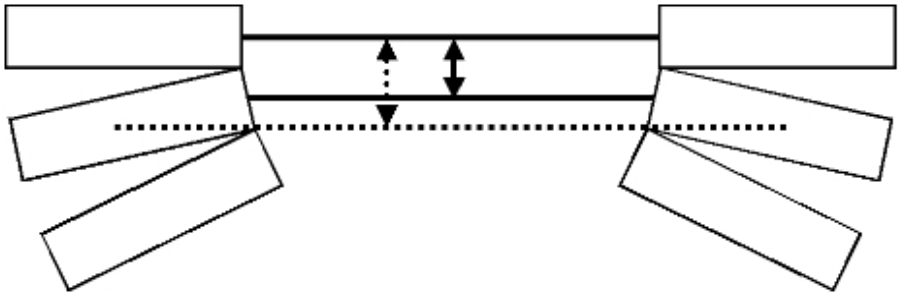


Fig. 3.31: Increase of the radial bin size due to the DOI effect.

3.4.6 LMF output

The Crystal Clear Collaboration has developed a List Mode Format (LMF) to store the data of ClearPET prototypes. Monte Carlo data generated by GATE can also be stored under the same format using the class **GateToLMF**. This format is only available for the cylindricalPET system (see *Defining a system*) and GATE can only store *single* events.

Several tools enabling the reading of this format and the processing of events are implemented in the LMF library. As an example, coincidences can be created from GATE *single* events. It is also possible to apply different dead-times, and even to generate sinograms in the Interfile format as used by the STIR library, which implements several image reconstruction algorithms.

The LMF library and its documentation are available on the OpenGate web site.

Table 3.6: Size of information to be stored in LMF.

Information	Size (bytes/single)	Real machines	GATE
Time	8	YES	YES
Energy	1	YES	YES
detector ID	2	YES	YES
PET's axial position	2	YES	YES
PET's angular position	2	YES	YES
run ID	4	NO	YES
event ID	4	NO	YES
source ID	2	NO	YES
source XYZ Position	6	NO	YES
global XYZ Position	6	NO	YES
number of Compton in phantomSD	1	NO	YES
number of Compton in crystalSD	1	NO	YES

LMF data are composed of two files with the same base-name, but different extensions :

- An ASCII file with a .cch extension contains general information about the scan and about the scanner, like the scan duration, the sizes of the detectors, or the angular rotation speed.
- A binary file with a .ccs extension contains headers, which set the topology of the scanner, followed by fixed size records.

The user can generate these two output files automatically by using the macro scripting. All pieces of information are optional, except time, which makes the ClearPET LMF quite versatile. Table 3.6 lists all options and memory requirements that can be stored in the **LMF event record** when using the cylindricalPET system:

```
/gate/output/lmf/enable    ( or /gate/output/lmf/disable  to disable LMF output (but␣
↪it is disable by default)
/gate/output/lmf/setFileName          myLMFFile   <=== to set the LMF files name.␣
↪Here the output files will be myLMFFile.ccs and myLMFFile.cch
/gate/output/lmf/setDetectorIDBool       1        <=== to store (1) or to not store␣
↪(0) the detector ID
/gate/output/lmf/setEnergyBool           1        <=== to store (1) or to not store␣
↪(0) the energy
/gate/output/lmf/setGantryAxialPosBool   0        <=== to store (1) or to not store␣
↪(0) the axial position
/gate/output/lmf/setGantryAngularPosBool 0        <=== to store (1) or to not store␣
↪(0) the angular position
/gate/output/lmf/setSourcePosBool        0        <===The following lines must␣
↪always be included, with option set to 0
/gate/output/lmf/setNeighbourBool        0
/gate/output/lmf/setNeighbourhoodOrder   0
```

(continues on next page)

(continued from previous page)

```
/gate/output/lmf/setCoincidenceBool       0
/gate/output/lmf/setGateDigiBool          1        <===all information that is not␣
↪available in real acquisitions is stored in a GateDigi record
/gate/output/lmf/setComptonBool           1        <===to store (1) or to not store␣
↪(0) the number of Compton scattering that occured in a phantomSD
/gate/output/lmf/setComptonDetectorBool   1        <===to store (1) or to not store␣
↪(0) the number of Compton scattering that occured in a crystalSD
/gate/output/lmf/setSourceIDBool          0        <=== to store (1) or to not store␣
↪(0) the source ID
/gate/output/lmf/setSourceXYZPosBool      0        <=== to store (1) or to not store␣
↪(0) the source XYZ position
/gate/output/lmf/setGlobalXYZPosBool      0        <=== to store (1) or to not store␣
↪(0) the real XYZ position
/gate/output/lmf/setEventIDBool           1        <=== to store (1) or to not store␣
↪(0) the event ID
/gate/output/lmf/setRunIDBool             1        <=== to store (1) or to not store␣
↪(0) the run ID
```

Limitations

The LMF format was originally designed for the development of small animal PET scanners for which the number of crystals is smaller than for clinical PET scanners. Consequently, the user should carefully read the LMF specifications and make sure that this format allows him to model his scanner design. In particular, the maximum number of sub-volumes in a volume (e.g. the maximum number of sub-modules in a module) is set by the number of bits used to encode the sub-volume ID. The final ID encoding the position of an event has to be stored on 16, 32, or 64 bits only.

3.4.7 Image CT output

The *imageCT* output is a binary matrix of float numbers that stores the number of Singles per pixel and is produced for each time slice:

```
/gate/output/imageCT/enable
/gate/output/imageCT/setFileName     test
```

The output file name is “test_xxx.dat”, where xxx is the corresponding time slice number.

In the case of the fast simulation mode, the number of pixels is set by:

```
/gate/output/imageCT/numPixelX   80
/gate/output/imageCT/numPixelY   80
```

In the case of VRT simulation mode (see *CTscanner*), the VRT K factor is set by:

```
/gate/output/imageCT/vrtFactor   10
```

Finally the random seed can be defined using:

```
/gate/output/imageCT/setStartSeed   676567
```

3.4.8 New unified Tree output (ROOT, numpy and more)

Introduction

The GateToTree class in GATE enables a new unified way for saving Hits, Singles and Coincidences. This new system can be used alongside with current ROOT output system

This class can be used this way, for example if you want to save hits and Singles:

```
/gate/output/tree/enable
/gate/output/tree/addFileName /tmp/p.npy
/gate/output/tree/hits/enable
/gate/output/tree/addCollection Singles
```

Theses commands will create two new files:

```
/tmp/p.hits.npy
/tmp/p.Singles.npy
```

where data are saved (hits in /tmp/p.hits.npy and Singles /tmp/p.Singles.npy)

Because of the extension ".npy", file is a numpy compatible arrat and ca be used directly in python with something like:

```
import numpy
hits = numpy.open("/tmp/p.hits.npy")
```

'hits' is a Numpy structured array

We can add easely add ROOT output:

```
/gate/output/tree/enable
/gate/output/tree/addFileName /tmp/p.npy
/gate/output/tree/addFileName /tmp/p.root
/gate/output/tree/hits/enable
/gate/output/tree/addCollection Singles
```

Important to notice : in order to have same behavior between ROOT, numpy and ascci output, GateToTree do not save several arrays in same file but will create:

```
/tmp/p.hits.root
/tmp/p.Singles.root
```

In GateToTree, one can disable branch to limit size output (instead of mask):

```
/gate/output/tree/hits/enable
/gate/output/tree/hits/branches/trackLocalTime/disable
```

for volumeID[0], volumeID[1], . . . :

```
/gate/output/tree/hits/branches/volumeIDs/disable
```

Also implemented for Singles:

```
/gate/output/tree/addCollection Singles
/gate/output/tree/Singles/branches/comptVolName/disable
```

and Coincidences:

```
/gate/output/tree/addCollection Coincidences
/gate/output/tree/Coincidences/branches/eventID/disable
```

Implemented output format

We take here example of an user which want to save Hits to a file. Output will on a file named "/tmp/p.hits.X" where X depends of the provided extension.

numpy-like format:

```
/gate/output/tree/enable
/gate/output/tree/addFileName /tmp/p.npy #saved to /tmp/p.hits.npy
/gate/output/tree/hits/enable
```

ROOT format:

```
/gate/output/tree/enable
/gate/output/tree/addFileName /tmp/p.root  #saved to /tmp/p.hits.root
/gate/output/tree/hits/enable
```

ASCII format:

```
/gate/output/tree/enable
/gate/output/tree/addFileName /tmp/p.txt #saved to /tmp/p.hits.txt
/gate/output/tree/hits/enable
```

Binary format is not (yet implemented)

Choice of Collection output format

If you want to save only Hits:

```
/gate/output/tree/enable
/gate/output/tree/addFileName /tmp/p.npy
/gate/output/tree/hits/enable
```

If you want to save Hits AND Singles:

```
/gate/output/tree/enable
/gate/output/tree/addFileName /tmp/p.npy
/gate/output/tree/hits/enable               #saved to /tmp/p.hits.npy
/gate/output/tree/addCollection Singles     #saved to /tmp/p.Singles.npy
```

If you want to save Hits AND Singles AND Coincidences:

```
/gate/output/tree/enable
/gate/output/tree/addFileName /tmp/p.npy
/gate/output/tree/hits/enable                   #saved to /tmp/p.hits.npy
/gate/output/tree/addCollection Singles         #saved to /tmp/p.Singles.npy
/gate/output/tree/addCollection Coincidences    #saved to /tmp/p.Coincidences.npy
```

If you want to save only Singles:

```
/gate/output/tree/enable
/gate/output/tree/addFileName /tmp/p.npy
/gate/output/tree/addCollection Singles     #saved to /tmp/p.Singles.npy
```

Selection of the variables to save

In GateToTree, there is a mechanism similar to mask for acscii and binary output in order to select variables to save. However, contrary to mask, the new mechanism is avalaible for Hits, Singles and Coincidences.

For example, for disabling 'trackLocalTime' in hits

```
/gate/output/tree/hits/enable
/gate/output/tree/hits/branches/trackLocalTime/disable
```

Like for mask, the VolumeID variables are enabled/disabled together, as a group:

```
/gate/output/tree/hits/branches/volumeIDs/disable
```

Also, for disabling 'comptVolName' in Singles:

```
/gate/output/tree/addCollection Singles
/gate/output/tree/Singles/branches/comptVolName/disable
```

In hits, variables that can be disabled are:

```
PDGEncoding,
trackID,parentID,
trackLocalTime,
time,
runID,eventID,
sourceID,
primaryID,
posX,posY,posZ,
localPosX,localPosY,localPosZ,
momDirX,momDirY,momDirZ,
edep,
stepLength,trackLength,
rotationAngle,
axialPos,
processName,
comptVolName,RayleighVolName,
volumeID # for disabling volumeID[0],volumeID[1],volumeID[2],volumeID[3],volumeID[4],
↪volumeID[5],volumeID[6],volumeID[7],volumeID[8],volumeID[9],
sourcePosX,sourcePosY,sourcePosZ,
nPhantomCompton,nCrystalCompton,
nPhantomRayleigh,nCrystalRayleigh,
gantryID,rsectorID,moduleID,submoduleID,crystalID,layerID,photonID, #/!\ depend on␣
↪the system type
gammaType,decayType,sourceType # for Extended source
```

In Singles, variables that can be disabled are:

```
runID,eventID,
sourceID,
sourcePosX,sourcePosY,sourcePosZ,
globalPosX,globalPosY,globalPosZ,
gantryID,rsectorID,moduleID,submoduleID,crystalID,layerID, #/!\ depend on the system␣
↪type
time,
energy,
comptonPhantom,comptonCrystal,RayleighPhantom,RayleighCrystal,comptVolName,
↪RayleighVolName,
```

(continues on next page)

(continued from previous page)

```
rotationAngle,axialPos
```

In Coincidences, variables that can be disabled are:

```
runID,
eventID1,eventID2,
sourceID1,sourceID2,
sourcePosX1,sourcePosX2,sourcePosY1,sourcePosY2,sourcePosZ1,sourcePosZ2,
rotationAngle,
axialPos,
globalPosX1,globalPosX2,globalPosY1,globalPosY2,globalPosZ1,globalPosZ2,
time1,time2,
energy1,energy2,
comptVolName1,comptVolName2,
RayleighVolName1,RayleighVolName2,
comptonPhantom1,comptonPhantom2,
comptonCrystal1,comptonCrystal2,
RayleighPhantom1,RayleighPhantom2,
RayleighCrystal1,RayleighCrystal2,
gantryID1,rsectorID1,moduleID1,submoduleID1,crystalID1,layerID1, #/!\ depend on the␣
↪system type
gantryID2,rsectorID2,moduleID2,submoduleID2,crystalID2,layerID2, #/!\ depend on the␣
↪system type
sinogramTheta,
sinogramS
```

Multiple processor chains

To deal with multiple processor chains as explained here (see *Multiple processor chains*) data cas be saved with:

```
/gate/output/tree/enable
/gate/output/tree/addFileName /tmp/p.npy
/gate/output/tree/addCollection Singles #optionnal
/gate/output/tree/addCollection LESingles   #saved to /tmp/p.LESingles.npy
/gate/output/tree/addCollection HESingles   #saved to /tmp/p.HESingles.npy
```

and for disabling variable output:

```
/gate/output/tree/LESingles/branches/comptVolName/disable
```

Multi-system detectors

When "Multi-system detectors" feature is used (see *How to define a multi-system detector*), a new variable appears in Hits, Singles and Coincidences : *systemID". The systemID correspond to the number order of apparation in system definition. For example:

```
/gate/world/daughters/name scanner_lead
/gate/world/daughters/systemType scanner

/gate/world/daughters/name scanner_water
/gate/world/daughters/systemType cylindricalPET
```

Hits which belong to scanner_lead will have systemID equals to 0 and scanner_water to 1.

Concerning componentsID, variables names become "SYSTEMNAME/COMPONENTNAME", for example, here we will have new variables:

```
scanner_lead/level1ID, scanner_lead/level2ID, scanner_lead/level3ID, scanner_lead/
↪level4ID, scanner_lead/level5ID
scanner_water/gantryID, scanner_water/rsectorID, scanner_water/moduleID, scanner_
↪water/submoduleID, scanner_water/crystalID, scanner_water/layerID
```

Additional output summary

The following output, named "summary", will write a txt file at the end of the simulation which indicates the numbers of Hits, Singles, Coincidences.:

```
/gate/output/summary/enable
/gate/output/summary/setFileName output/digit_summary.txt
/gate/output/summary/addCollection Singles
/gate/output/summary/addCollection Coincidences
```

Usually, the 'hits' and 'singles' output lead to very large files, often only needed for debug purpose. We recommend to disable the output of 'hits' and 'Singles' and only keep the 'Coincidences' output. The Summary output can still be used to get the total numbers.

Numpy or Root outputs: what to choose? You will remark that npy outputs are bigger than ROOT ones. You can therefore decide to zip your npy output in order to load it with Python as suggested after: data = np.load('myfile.zip') tab_1 = data['tab_1.npy'] tab_2 = data['tab_2.npy']

Or, you can decide to keep the ROOT files and analyse them using Python and the associated pandas and uproot libraries:

To load a ROOT tree file, read the Hits and/or Singles trees and recover some leaf information:

import uproot import pandas as pd f=uproot.open("myfile.root") rawSingles = f['Singles'].pandas.df().to_records() rawHits = f['Hits'].pandas.df().to_records() posx = rawSingles['globalPosX'] posy = rawSingles['globalPosY'] posz = rawSingles['globalPosZ']

To load a ROOT histogram and plot it with Python, you can do:

import uproot import matplotlib.pyplot as plt f=uproot.open("myfile.root") f.allclasses() h1=f["histo;1"] plt.plot(line.get_xdata(),line.get_ydata())

3.5 Generating and tracking optical photons

Table of Contents

3.5.1 Introduction

To use the optical photon capabilities of GATE, the **GATE_USE_OPTICAL** variable has to be set to **ON** in the configuration process using ccmake.

Before discussing how to use the optical photon tracking, it has to be mentioned that there are a few disadvantages in using optical transport. First, the simulation time will increase dramatically. For example, most scintillators used in PET generate in the order of 10,000 optical photons at 511 keV, which means that approximately 10,000 more particles have to be tracked for each annihilation photon that is detected. Although the tracking of optical photons is relatively fast, a simulation with optical photon tracking can easily be a factor thousand slower than one without. Finally, in order to perform optical simulations, many parameters are needed for the materials and surfaces, some of which may be difficult to determine.

3.5.2 Optical Photon Generation

Example:

```
/gate/source/addSource                          Mysource gps
/gate/source/Mysource/gps/particle              opticalphoton
/gate/source/Mysource/gps/energytype            Mono
/gate/source/Mysource/gps/angtype               iso
```

An optical photon with a wave length of 530nm corresponds to an optical photon of energy=2.34eV [approximation: 1240/E(eV) = wavelength (nm)]:

```
/gate/source/Mysource/gps/monoenergy            2.34 eV
```

An optical photon which is not assigned a polarization at production may not be Rayleigh scattered:

```
/gate/source/Mysource/gps/polarization          1 0 0
```

3.5.3 Optical System

The GATE **OpticalSystem** is appropriate to model Optical Imaging systems. This system is defined in the following section *OpticalSystem*.

3.5.4 Defining material properties

The optical properties of materials are stored in a material property table. In this table each of the properties of a material is identified by a name. There are two kinds of properties. The first are constant properties, these contain only one value. The second are property vectors, these contain properties that depend on the energy of the optical photon. Such a vector is a list of energy-value pairs.

The property tables for the materials used in a simulation are to be stored in a file separate from the material database. This makes it easier to change the properties without having to change the material database. This file should be named **Materials.xml**. When Gate reads in a material from the materials database, it also checks if the *Materials.xml* file contains a property table for this material. If so, this table is read in and coupled to the material.

Scintillation

A scintillator is characterized by its photon emission spectrum. The scintillation follows an exponential decay with two time constants, a fast and a slow one. The relative strength of the fast component **FASTCOMPONENT** as a fraction of total scintillation yield is given by the **YIELDRATIO**. The emission spectra of both decays are given by the property vectors **FASTCOMPONENT** and **SLOWCOMPONENT** and the time constants **FASTTIMECONSTANT** and **SLOWTIMECONSTANT**. These vectors specify the probability that a photon with the given energy is emitted. The sum of each of the vectors should therefore be one.

In order to have scintillation in a material, the first parameter that has to be specified is the **SCINTILLATIONYIELD** (1/Mev, 1/keV), which gives the number of photons that is emitted per amount of energy absorbed, or, more precisely, it gives the *expectation* value of this number, since the real number of emitted photons follows a normal distribution. The variance of this normal distribution is **RESOLUTION-SCALE** times this expectation value. Thus, for example, when a gamma photon deposits E amount of energy in the scintillator, N optical photons are emitted with an expectation value of μ_N = E . SCINTILLATIONYIELD

and a standard deviation of $\sigma_N = RESOLUTIONSCALE \cdot \sqrt{E \cdot SCINTILLATIONYIELD}$

The parameters *RESOLUTIONSCALE* can be calculated from the energy resolution of the scintillator. The energy resolutions specified in the literature may contain contributions of electronic noise. The energy resolution needed to calculate the *RESOLUTIONSCALE* should be the intrinsic energy resolution of the scintillator.

$RESOLUTIONSCALE = \frac{R}{2.35} \cdot \sqrt{E \cdot SCINTILLATIONYIELD}$

where R is the energy resolution (FWHM - Full width at half maximum) at energy E:

```
<material name="LSO">
  <propertiestable>
    <property name="SCINTILLATIONYIELD" value="26000" unit="1/MeV"/>
    <property name="RESOLUTIONSCALE" value="4.41"/>
    <property name="FASTTIMECONSTANT" value="40" unit="ns"/>
    <property name="YIELDRATIO" value="1"/>
    <propertyvector name="FASTCOMPONENT" energyunit="eV">
      <ve energy="2.95167" value="1"/>
    </propertyvector>
    <propertyvector name="ABSLENGTH" unit="m" energyunit="eV">
      <ve energy="1.84" value="50"/>
      <ve energy="4.08" value="50"/>
    </propertyvector>
```

(continues on next page)

(continued from previous page)

```
    <propertyvector name="RINDEX" energyunit="eV">
      <ve energy="1.84" value="1.82"/>
      <ve energy="4.08" value="1.82"/>
    </propertyvector>
  </propertiestable>
</material>
```

Absorption

This process kills the particle. It requires the Material.xml properties filled by the user with the Absorption length *ABSLENGTH* (average distance traveled by a photon before being absorbed by the medium):

```
/gate/physics/addProcess OpticalAbsorption
```

Mie/Rayleigh Scattering

Mie Scattering is an analytical solution of Maxwell's equations for scattering of optical photons by spherical particles. It is significant only when the radius of the scattering object is of order of the wave length.The analytical expressions for Mie Scattering are very complicated. One common approximation (followed by Geant4) made is called **Henyey-Greenstein** (HG). For small size parameter (scattering particle diameter) regime the Mie theory reduces to the Rayleigh approximation:

```
/gate/physics/addProcess OpticalRayleigh
/gate/physics/addProcess OpticalMie
```

For Rayleigh or Mie scattering, we require the final momentum, initial polarization and final polarization to be in the same plane. Mie/Rayleigh processes require material properties to be filled by the user with Mie/Rayleigh scattering length data: **MIEHG/RAYLEIGH**, which is the average distance traveled by a photon before it is Mie/Rayleigh scattered in the medium. In the case of the Mie scattering, the user also needs to provide parameters of the HG approximation: **MIEHG_FORWARD** (forward anisotropy), **MIEHG_BACKWARD** (backward anisotropy), and **MIEHG_FORWARD_RATIO** (ratio between forward and backward angles). Geant4 code allows the forward and backward angles to be treated separately. If your material characteristics provides only one number for the **anisotropy** (= average cosine of the scattering angle), below is an example of how (part of) the Materials.xml file could look like:

```
<material name="Biomimic">
  <propertiestable>
   <propertyvector name="ABSLENGTH" unit="cm" energyunit="eV">
     <ve energy="1.97" value="0.926"/>
     <ve energy="2.34" value="0.847"/>
    </propertyvector>
    <propertyvector name="RINDEX" energyunit="eV">
      <ve energy="1.97" value="1.521"/>
      <ve energy="2.34" value="1.521"/>
    </propertyvector>
 <property name="MIEHG_FORWARD" value="0.62" />
    <property name="MIEHG_BACKWARD" value="0.62" />
    <property name="MIEHG_FORWARD_RATIO" value="1.0" />
    <propertyvector name="MIEHG" unit="cm" energyunit="eV">
      <ve energy="1.97" value="0.04"/>
      <ve energy="2.34" value="0.043"/>
    </propertyvector>
  </propertiestable>
</material>
```

Fluorescence

Fluorescence is a 3 step process: The fluorophore is in an excited state after the absorption of an optical photon provided by an external source (laser, lamp). The life time of the excited state is of order of 1-10ns during which the fluorophore interacts with its environment and ends-up in a relaxed-excited state. The last step is the emission of a fluorescent photon which energy/wave length is smaller(larger) than the one of the excitation optical photon.

Fig. 3.32: Optical Fluorescence

Geant4 simulates the **W** ave **L** ength **S** hifting (WLS) fibers that are used in High Energy Physics experiments. As an example, the CMS hadronic EndCap calorimeter is made of scintillator tiles with WLS fibers embedded. These fibers collect/absorb blue light produced in tiles and re-emit green light so that as much light reaches the PMTs. A new class in Gate has been implemented as a physics builder class that inherits from the G4OpWLS class. The following command line enables the optical photon fluorescence:

```
/gate/physics/addProcess OpticalWLS
```

Gate user needs to provide four parameters/properties to define the fluorescent material: **RINDEX**, **WLSABSLENGTH**, **WLSCOMPONENT** and **WLSTIMECONSTANT**. The **WLSABSLENGTH** defines the fluorescence absorption length which is the average distance travelled by a photon before it is absorbed by the fluorophore. This distance could be very small but probably not set to 0 otherwise the photon will be absorbed immediately upon entering the fluorescent volume and fluorescent photon will appear only from the surface. The **WLSCOMPONENT** describes the emission spectrum of the fluorescent volume by giving the relative strength between different photon energies. Usually these numbers are taken from measurements (i.e. emission spectrum). The **WLSTIMECONSTANT** defines the time delay between the absorption and re-emission.

Simulation of the Fluorescein see

```
We define the refractive index of the fluorophore's environment (water or alcohol):
<material name="Fluorescein">
<propertiestable>
<propertyvector name="RINDEX" energyunit="eV">
<ve energy="1.0" value="1.4"/>
<ve energy="4.13" value="1.4"/>
</propertyvector>
```

The WLS process has an absorption spectrum and an emission spectrum. If these overlap then a WLS photon may in turn be absorpted and emitted again. If you do not want that you need to avoid such overlap. The WLS process does not distinguish between 'original' photons and WLS photons:

```
We describe the fluorescein absorption length taken from measurements or literature␣
↪as function of the photon energy:
<propertyvector name="WLSABSLENGTH" unit="cm" energyunit="eV">
<ve energy="3.19" value="2.81"/>
<ve energy="3.20" value="2.82"/>
<ve energy="3.21" value="2.81"/>
</propertyvector>

We describe the fluorescein Emission spectrum taken from measurements or literature␣
↪as function of the photon energy:
<propertyvector name="WLSCOMPONENT" energyunit="eV">
<ve energy="1.771"  value="0.016"/>
<ve energy="1.850"  value="0.024"/>
<ve energy="1.901"  value="0.040"/>
<ve energy="2.003"  value="0.111"/>
```

(continues on next page)

(continued from previous page)

```
<ve energy="2.073"  value="0.206"/>
<ve energy="2.141"  value="0.325"/>
<ve energy="2.171"  value="0.413"/>
<ve energy="2.210"  value="0.540"/>
<ve energy="2.250"  value="0.683"/>
<ve energy="2.343"  value="0.873"/>
<ve energy="2.384"  value="0.968"/>
<ve energy="2.484"  value="0.817"/>
<ve energy="2.749"  value="0.008"/>
<ve energy="3.099"  value="0.008"/>
</propertyvector>
<property name="WLSTIMECONSTANT" value="1.7" unit="ns"/>
</propertiestable>
</material>
```

Boundary Processes

When a photon arrives at a medium boundary its behavior depends on the nature of the two materials that join at that boundary:

```
/gate/physics/addProcess OpticalBoundary
```

In the case of two dielectric materials, the photon can undergo total internal reflection, refraction or reflection, depending on the photon's wavelength, angle of incidence, and the refractive indices on both sides of the boundary. In the case of an interface between a dielectric and a metal, the photon can be absorbed by the metal or reflected back into the dielectric. When simulating a perfectly smooth surface, the user doesn't have to provide a G4Surface. The only relevant property is the refractive index (RINDEX) of the two materials on either side of the interface. Geant4 will calculate from Snell's Law the probabilities of refraction and reflections.

3.5.5 Defining surfaces

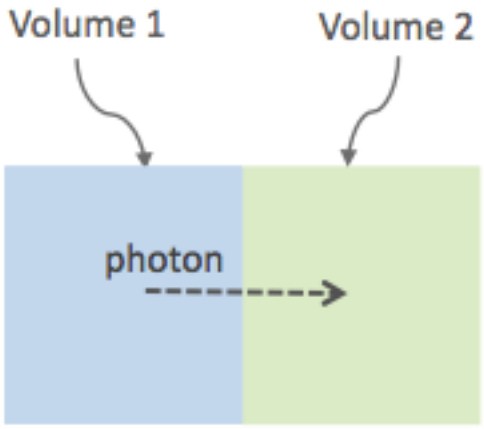


Fig. 3.33: Surface definition

The photon travels through the surface between the two volumes **Volume1** and **Volume2**. To create an optical surface with the name **Surface-From-Volume1-To-Volume2**, the following commands should be used:

```
/gate/**Volume2**/surfaces/name   Surface-From-Volume1-To-Volume2
/gate/**Volume2**/surfaces/insert **Volume1**
```

The surface between **Volume1** and **Volume2** is NOT the same surface as that between Volume2 and Volume1; the surface definition is directional. When there is optical transport in both directions, two surfaces should be created. To load the surface properties stored in the Surfaces.xml file. Surface_name can be any surface defined in the Surfaces.xml file:

```
/gate/Volume2/surfaces/Surface-From-Volume1-To-Volume2/SetSurface Surface_name
```

In Gate, two simulation models that are used at the boundary are available. The recently implemented LUTDavis model (GATE V8.0) and the traditional UNIFIED model (see: source/geometry/src/GateSurface.cc).

LUT Davis Model

Please Note: Necessary modifications in Geant4 are not implemented until Summer 2017. The user can manually modify the Geant4 code. Find detailed instructions here: *Enabling LUT Davis Model*

Available in GATE V8.0 is a model for optical transport called the LUT Davis model [Roncali& Cherry(2013)]. The model is based on measured surface data and allows the user to choose from a list of available surface finishes. Provided are a rough and a polished surface that can be used without reflector, or in combination with a specular reflector (e.g. ESR) or a Lambertian reflector (e.g. Teflon). The specular reflector can be coupled to the crystal with air or optical grease. Teflon tape is wrapped around the crystal with 4 layers.

Table 3.7: Surface names of available LUTs.

	BARE	TEFLON	ESR AIR	ESR GREASE
POLISHED	Polished_LUT	PolishedTeflon_LUT	PolishedESR_LUT	PolishedESRGrease_LUT
ROUGH	Rough_LUT	RoughTeflon_LUT	RoughESR_LUT	RoughESRGrease_LUT

The user can extend the list of finishes with custom measured surface data. In GATE V8.0, this can be achieved by contacting the developers of the LUT Davis model. In future releases, a tool to calculate LUTs will be provided in form of a graphical user interface. In the LUT database, typical roughness parameters obtained from the measurements are provided to characterize the type of surface modelled:

- **ROUGH** Ra=0.48 µm, σ=0.57 µm, Rpv=3.12 µm
- **POLISHED** Ra=20.8 nm, σ=26.2 nm, Rpv=34.7 nm

with Ra = average roughness; σ = rms roughness, Rpv = peak-to-valley ratio.

The desired finish should be defined in Surfaces.xml (file available in https://github.com/OpenGATE/GateContrib/tree/master/imaging/LUTDavisModel):

```
 <surface model="DAVIS" name="RoughTeflon_LUT" type="dielectric_LUTDAVIS" finish=
↪"RoughTeflon_LUT">
  </surface>

The detector surface, called **Detector_LUT**, defines a polished surface coupled to␣
↪a photodetector with optical grease or a glass interface (similar index of␣
↪refraction 1.5). Any surface can be used as a detector surface when the Efficiency␣
↪is set according to the following example:
 <surface model="DAVIS" name="**Detector_LUT**" type="dielectric_LUTDAVIS" finish=
↪"Detector_LUT">
     <propertiestable>
      <propertyvector name="**EFFICIENCY**" energyunit="eV">
```

(continues on next page)

(continued from previous page)

```
        <ve energy="1.84" value="**1**"/>
        <ve energy="4.08" value="**1**"/>
      </propertyvector>
    </propertiestable>
  </surface>
```

Running the simulation produces an output in the terminal confirming that the LUT data is read in correctly. The user should check the presence of these lines in the terminal. For example:

```
===== XML PATH ====: ./Surfaces.xml
===== XML PATH ====: ...
LUT DAVIS - data file: .../Rough_LUT.dat read in!
Reflectivity LUT DAVIS - data file: .../Rough_LUTR.dat read in!
===== XML PATH ====: ./Surfaces.xml
===== XML PATH ====: ...
LUT DAVIS - data file: .../Detector_LUT.dat read in!
Reflectivity LUT DAVIS - data file: .../Detector_LUTR.dat read in!
```

Detection of optical photons

Once the simulation is finished, the optical photon data can be found in the Hits Tree in the ROOT output. The Hits Tree consists of events that ended their path in the geometry defined as the sensitive detector (SD). Thus, photons can either be detected or absorbed in the crystal material when set as SD. The user can identify the optical photons from other particles using the PDGEncoding (0 for optical photons).

Example

The example (https://github.com/OpenGATE/GateContrib/tree/master/imaging/LUTDavisModel) includes a 3 mm x 3 mm x 20 mm scintillation crystal coupled to a 3 mm x 3 mm detector area. The source is positioned at the side of the crystal, irradiating it at 10 mm depth. The set surface is RoughTeflon_LUT in combination with the Detector_LUT as the photo detector surface.

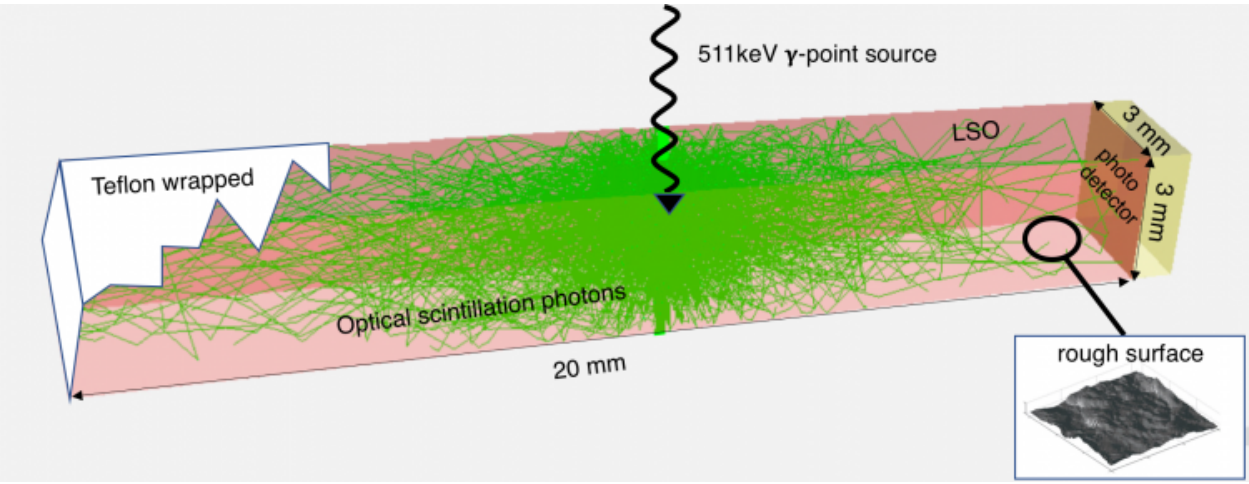


Fig. 3.34: LUT Davis Model

Background

The crystal topography is obtained with atomic force microscopy (AFM). From the AFM data, the probability of reflection (1) and the reflection directions (2) are computationally determined, for incidence angles ranging from 0° to 90°. Each LUT is computed for a given surface and reflector configuration. The reflection probability in the LUT combines two cases: directly reflected photons from the crystal surface and photons that are transmitted to the reflector

surface and later re-enter the crystal. The key operations of the reflection process are the following: The angle between the incident photon (Old Momentum) and the surface normal are calculated. The probability of reflection is extracted from the first LUT. A Bernoulli test determines whether the photon is reflected or transmitted. In case of reflection two angles are drawn from the reflection direction LUT.

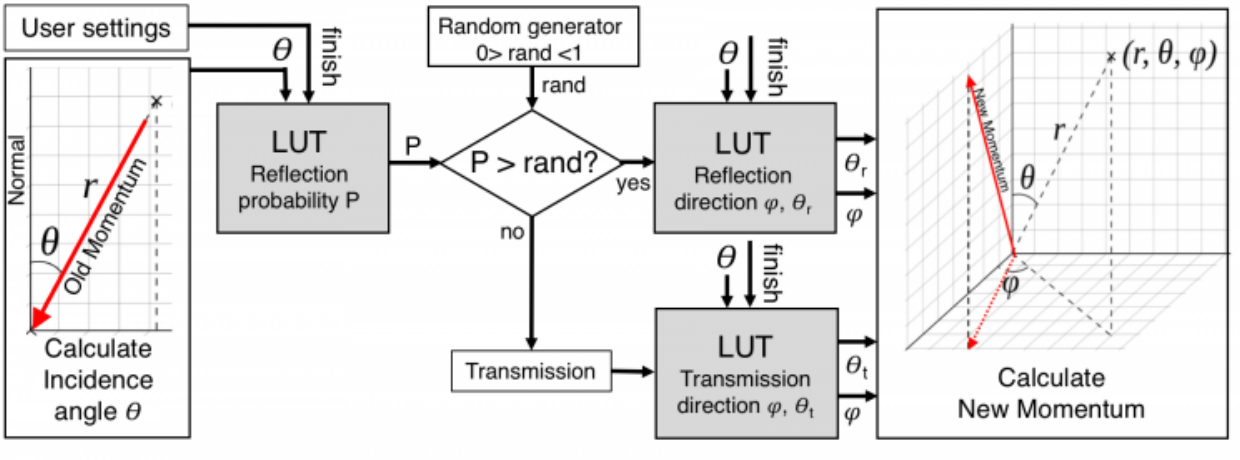


Fig. 3.35: FlowChart LUT Model

Old Momentum to New Momentum. The old momentum is the unit vector that describes the incident photon. The reflected/transmitted photon is the New Momentum described by two angles ϕ, .

UNIFIED Model

The UNIFIED model allows the user to **control the radiant intensity** of the surface: **Specular lobe**, **Specular spike**, **Backscatter spike** (enhanced on very rough surfaces) and **Reflectivity** (Lambertian or diffuse distribution). The sum of the four constants is constrained to unity. In that model, the micro-facet normal vectors follow a Gaussian distribution defined by **sigmaalpha** (σ_α) given in degrees. This parameter defines the standard deviation of the Gaussian distribution of micro-facets around the average surface normal. In the case of a perfectly polished surface, the normal used by the G4BoundaryProcess is the normal to the surface.

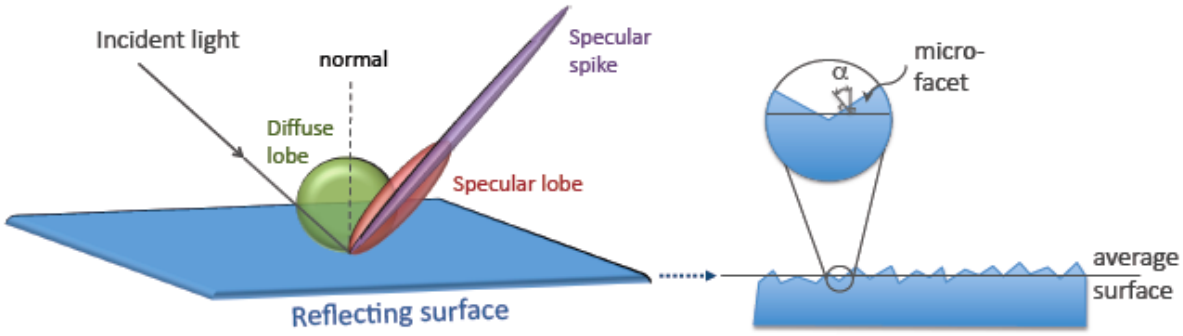


Fig. 3.36: Reflection Types and Microfacets

To load the surface properties stored under **rough_teflon_wrapped** in the Surface.xml file:

```
/gate/**Volume2**/surfaces/Surface-From-Volume1-To-Volume2/SetSurface rough_teflon_
↪wrapped
```

An example of a surface definition looks like:

```
<surface name="rough_teflon_wrapped" type="dielectric_dielectric" sigmaalpha="0.1"␣
↪finish="groundbackpainted">
 <propertiestable>
   <propertyvector name="SPECULARLOBECONSTANT" energyunit="eV">
     <ve energy="4.08" value="1"/>
     <ve energy="1.84" value="1"/>
   </propertyvector>
   <propertyvector name="RINDEX" energyunit="eV">
     <ve energy="4.08" value="1"/>
     <ve energy="1.84" value="1"/>
   </propertyvector>
   <propertyvector name="REFLECTIVITY" energyunit="eV">
     <ve energy="1.84" value="0.95"/>
     <ve energy="4.08" value="0.95"/>
   </propertyvector>
   <propertyvector name="EFFICIENCY" energyunit="eV">
     <ve energy="1.84" value="0"/>
     <ve energy="4.08" value="0"/>
   </propertyvector>
 </propertiestable>
</surface>
```

The attribute *type* can be either *dielectric_dielectric* or *dielectric_metal*, to model either a surface between two dielectrica or between a dielectricum and a metal. The attribute *sigma-alpha* models the surface roughness and is discussed in the next section. The attribute *finish* can have one of the following values: *ground*, *polished*, *ground-back-painted*, *polished-back-painted*, *ground-front-painted* and *polished-front-painted*. It is therefore possible to cover the surface on the inside or outside with a coating that reflects optical photons using **Lambertian reflection**. In case the finish of the surface is *polished*, the surface normal is used to calculate the probability of reflection. In case the finish of the surface is *ground*, the surface is modeled as consisting of small **micro-facets**. When an optical photon reaches a surface, a random angle α is drawn for the micro facet that is hit by the optical photon. Using the angle of incidence of the optical photon with respect to this micro facet and the refractive indices of the two media, the probability of reflection is calculated.

In case the optical photon is reflected, four kinds of reflection are possible. The probabilities of the first three are given by the following three property vectors:

- **SPECULARSPIKECONSTANT** gives the probability of specular reflection about the average surface normal
- **SPECULARLOBECONSTANT** gives the probability of specular reflection about the surface normal of the micro facet
- **BACKSCATTERCONSTANT** gives the probability of reflection in the direction the optical photon came from

LAMBERTIAN (diffuse) reflection occurs when none of the other three types of reflection happens. The probability of Lambertian reflection is thus given by one minus the sum of the other three constants.

Fig. 3.37: Reflections Specular Diffuse Spread

When the photon is refracted, the angle of refraction is calculated from the surface normal (of the average surface for *polished* and of the micro facet for *rough*) and the refractive indices of the two media.

When an optical photon reaches a painted layer, the probability of reflection is given by the property vector **REFLECTIVITY**. In case the paint is on the inside of the surface, the refractive indices of the media are ignored, and when the photon is reflected, it undergoes Lambertian reflection.

When the paint is on the outside of the surface, whether the photon is reflected on the interface between the two media is calculated first, using the method described in the previous section. However, in this case the refractive index given

by the property vector *RINDEX* of the surface is used. When the photon is refracted, it is reflected using Lambertian reflection with a probability *REFLECTIVITY*. It then again has to pass the boundary between the two media. For this, the method described in the previous section is used again and again, until the photon is eventually reflected back into the first medium or is absorbed by the paint.

A **dielectric_dielectric** surface may have a wavelength dependent property **TRANSMITTANCE**. If this is specified for a surface it overwrites the Snell's law's probability. This allows the simulation of anti-reflective coatings.

Detection of optical photons

Optical photons can be detected by using a **dielectric-metal** boundary. In that case, the probability of reflection should be given by the REFLECTIVITY property vector. When the optical photon is reflected, the UNIFIED model is used to determine the reflection angle. When it is absorbed, it is possible to detect it. The property vector EFFICIENCY gives the probability of detecting a photon given its energy and can therefore be considered to give the internal quantum efficiency. Note that many measurements of the quantum efficiency give the external quantum efficiency, which includes the reflection: external quantum efficiency = efficiency*(1-reflectivity).

The hits generated by the detection of the optical photons are generated in the volume from which the optical photons reached the surface. This volume should therefore be a sensitive detector.

3.5.6 Digitizer

The hits generated in the sensitive detector are first processed by *analysis*. Unfortunately *analysis* is quite slow when there are a large number of hits, as is the case when there is optical transport. Therefore, an alternative has been created that is faster ans is therefore called *fastanalysis*:

```
/gate/output/analysis/disable

/gate/output/fastanalysis/enable
```

Switching both on has no effect on the results, but only affects the speed of the simulation. After processing the hits with one of the analysis routines, the singles should be created from the hits. This is usually done using the **opticaladder** which adds all hits generated by optical photons. In this way, it is possible to create a digitizer chain containing the singles generated by optical photons:

```
/gate/digitizer/Singles/insert opticaladder
/gate/digitizer/Singles/insert readout
/gate/digitizer/Singles/readout/setDepth your_detector_readout_level
```

Digitizer modules like **threshold** or **uphold** can be used (see *Thresholder & Upholder*). This is crucial when you do a fluorescence experience for example. If you want to detect only fluorescent photons you need to apply an energy cut (upholder) in order to discard high energy photons (non-fluorescent photons have higher energy than fluorescent photons):

```
/gate/digitizer/Singles/insert upholder
/gate/digitizer/Singles/upholder/setUphold 2.0 eV
/gate/digitizer/Singles/insert thresholder
/gate/digitizer/Singles/thresholder/setThreshold 1.0 eV
```

The **projection** (see *Binary output of projection set*) associated to this digitizer records only photons corresponding to the defined energy window. The projection image is therefore the fluorescence image.

3.5.7 Optical Imaging Simulation Outputs

Root output

When working with optical photons, an additional ROOT tree is created: OpticalData. You can decide to fill this tree or not by using the following command:

```
/gate/output/root/setRootOpticalFlag 0 or 1
```

OpticalData tree is generated with the following information:

```
CrystalLastHitEnergy  CrystalLastHitPos_X CrystalLastHitPos_Y CrystalLastHitPos_Z
Energy and Positions of the photon **last hit** in the Crystal (Detected photon␣
↪position)

PhantomLastHitEnergy  PhantomLastHitPos_X PhantomLastHitPos_Y PhantomLastHitPos_Z
Energy and Positions of the photon **last hit** in the Phantom

NumCrystalWLS
Number of Fluorescence processes per event(photon) in the Crystal

NumPhantomWLS
Number of Fluorescence processes per event(photon) in the Phantom

NumScintillation
Number of Scintillation processes per event(photon) in the Crystal

CrystalProcessName   PhantomProcessName
List of process names that occured in the Crystal or in the Phantom

MomentumDirectionx MomentumDirectiony MomentumDirectionz
Optical photon momentum direction
```

New unified Tree output (ROOT, numpy and more)

One can save optical data in precedent section with new system (see *New unified Tree output (ROOT, numpy and more)*) like this:

```
/gate/output/tree/enable
/gate/output/tree/addFileName p.npy
/gate/output/tree/optical/enable
```

And variables to save can also be disabled, for example:

```
/gate/output/tree/optical/branches/CrystalLastHitEnergy/disable
```

Binary output of projection set

In order to create a projection set (see *Interfile output of projection set*) using the Optical System in GATE, the following lines have to be added to the macro:

```
/gate/output/projection/enable
/gate/output/projection/setFileName        your_name
/gate/output/projection/projectionPlane    XY
/gate/output/projection/pixelSizeX              0.105 cm
/gate/output/projection/pixelSizeY              0.105 cm
```

(continues on next page)

(continued from previous page)

```
/gate/output/projection/pixelNumberX            100
/gate/output/projection/pixelNumberY            100
```

The result of projection set is saved in a binary file (.bin). A header file (.hdr) is also provided with the following information:

```
!INTERFILE :=
!imaging modality := optical imaging
;
!GENERAL DATA :=
data description := GATE simulation
!name of data file := ./OpticalSimulationProjection.bin
;
!GENERAL IMAGE DATA :=
!type of data := OPTICAL
!total number of images := 1
;
!OPTICAL STUDY (general) :=
number of detector heads := 1
;
!number of images divided by number of energy window := 1
projection matrix size [1] := 100
projection matrix size [2] := 100
projection pixel size along X-axis (cm) [1] := 0.105
projection pixel size along Y-axis (cm) [2] := 0.105
!number of projections := 1
!extent of rotation := 360
!time per projection (sec) := 1
;
;GATE GEOMETRY :=
;Optical System x dimension (cm) := 10.5
;Optical System y dimension (cm) := 10.5
;Optical System z dimension (cm) := 2
;Optical System material := Air
;Optical System x translation (cm) := 0
;Optical System y translation (cm) := 0
;Optical System z translation (cm) := 0
;
;Optical System LEVEL 1 element is crystal :=
;Optical System crystal x dimension (cm) := 10.5
;Optical System crystal y dimension (cm) := 10.5
;Optical System crystal z dimension (cm) := 1
;Optical System crystal material := Air
;
;Optical System LEVEL 2 element is pixel :=
;Optical System pixel x dimension (cm) := 2
;Optical System pixel y dimension (cm) := 2
;Optical System pixel z dimension (cm) := 1
;Optical System pixel material := Air
;
!END OF INTERFILE :=
```

3.5.8 Example of an Optical Imaging Simulation

In the GateContrib repository you will find simple examples of a bioluminescence/fluorescence experiment. All macros are located under *imaging/Optical*. In addition, a ROOT macro [*DrawBranches.C*] is available and draws

all branches of the OpticalData tree into a postscript file.

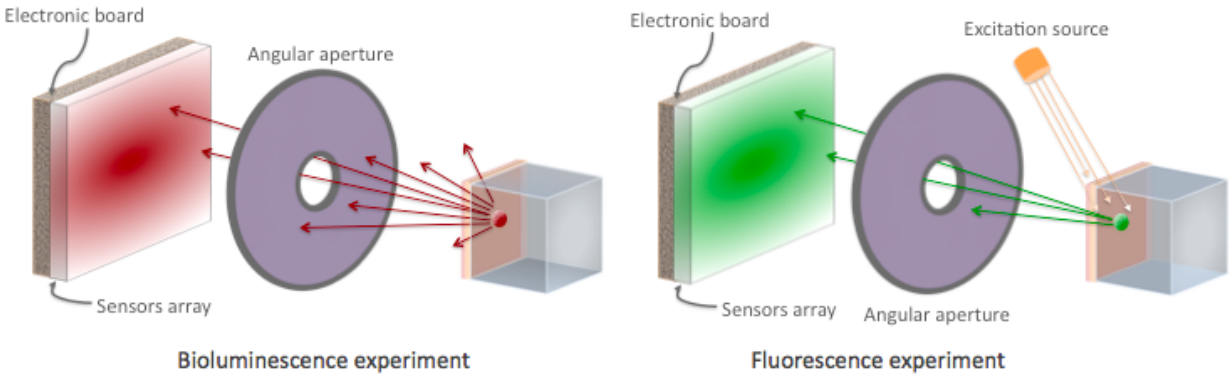


Fig. 3.38: Optical Imaging experiments

The optical imaging system is composed of an array of pixels, an electronic board and an angular aperture that limits the range of angles over which the optical system can accept light. The phantom is composed of a box of water and two layers made of either water, hypodermis or epidermis. In case of a bioluminescence experiment, the tumor is described as a voxelized source of optical photons and is positioned under the inner layer of the phantom. In case of a fluorescence experiment, we assigned the Rhodamine B fluorophore to each voxel of a voxelized tumor and positioned it under the inner layer of the phantom. The fluorophore is excited by two external beam light sources emitting optical photons towards the tumor.

These two experiments are available in *imaging/Optical* through the following macros: bioluminescence.mac and fluorescence.mac. The voxelized source or phantom is available in *imaging/Optical/voxelized-source-phantom* with an attenuation file and an optical-flux file. These macros will generate a root output file with the OpticalData tree enabled and a binary file which corresponds to the GATE ProjectionSet on the XY plane (i.e detection plane). Using the root macros MakeBioluminescencePlots.C and MakeFluorescencePlots.C, you can read the root output file and draw the bioluminescent/fluorescent light that is detected by the optical system. In case of the fluorescence experiment, two plots are drawn: all detected light (any wavelength) and the fluorescent light (wavelength cut). The projection binary file (.bin and .hdr) can be viewed directly using Anatomist or Imagej. In case of the fluorescence experiment, an Upholder (uphold cut) was applied through the digitizer so the binary image illustrates the fluorescent light.

The Materials.xml file is updated with several tissues properties at specific wavelengths (from literature): brain, kidney, epidermis and hypodermis but also with the emission spectra of the Fluorescein and Rhodamine B.

3.5.9 Bibliography

- The NIST XCOM (NIST-XCOM): Photon Cross Sections Database gives attenuation coefficients. The Database Search Form is available directly through this link Database Search Form.
- A Review of the Optical Properties of Biological Tissues, IEEE J. Quantum Electronics, 26, 2166-2185 (1990) (W. F. Cheong, S. A. Prahl, and A. J. Welch). Updated by Wai-Fung Cheong. Further additions by Lihong Wang and Steven L. Jacques. August 6, 1993 - IEEE Journal of Quantum Electronics, Vol. 26, Issue 12, pp. 2166 - 2185
- Optical Absorption of Water with all currently available data (presented in terms of wavelength and absorption coefficient) - Scott Prahl, Oregon Medical Laser Center.
- Optical Brain Imaging in Vivo: Techniques and Applications from Animal to Man. (E. M. C. Hillman) - J. Biomed. Opt. 2007 Sep-Oct;12(5):051402
- Refractive Index Measurement of Acute Rat Brain Tissue Slices using Optical Coherence Tomography (J. Sun, S. J. Lee, L. Wu, M. Sarntinoranont and H. Xie) - Optics Express, Vol. 20, Issue 2, pp. 1084-1095 (2012)

- In Vivo Optical Reflectance Imaging of Spreading Depression Waves in Rat Brain with and without Focal Cerebral Ischemia (S. Chen, Z. Feng, P. Li, S. L. Jacques, S. Zeng and Q. Luo) - J. Biomed. Opt. 2006 May-Jun;11(3):34002
- Brain Refractive Index Measured in Vivo with High-NA Defocus-Corrected Full-Field OCT and Consequences for Two-Photon Microscopy (J. Binding, J. Ben Arous, J-F. Leger, S. Gigan, C. Boccara and L. Bourdieu) - Optics Express, Vol. 19, Issue 6, pp. 4833-4847 (2011)
- Contribution of the Mitochondrial Compartment to the Optical Properties of the Rat Liver: a Theoretical and Practical Approach (B. Beauvoit, T. Kitai and B. Chance) - Biophys. J. 1994 Dec;67(6):2501-10
- Optical Properties of Native and Coagulated human Liver Tissue and Liver Metastases in the Near Infrared Range (C. T. Germer, A. Roggan, J. P. Ritz, C. Isbert, D. Albrecht, G. Muller and H. J. Buhr) - Lasers Surg. Med. 1998;23(4):194-203
- In Vivo Determination of the Optical Properties of Muscle with Time-Resolved Reflectance using a Layered Model (A. Kienle and T. Glanzmann) - Phys. Med. Biol. 1999 Nov;44(11):2689-702
- Optical Properties of Skin, Subcutaneous and Muscle Tissues: a Review (A. N. Bashkatov, E. A. Genina and V. V. Tuchin) - J. Innov. Opt. Health Sci. 04, 9 (2011)
- Determination of the Optical Properties of Rat (Heart) Tissue (A. Singh1, A. E. Karsten, R. M. Smith and G. van Niekerk) - 2010 European Cells & Materials Ltd
- In Vitro Double-Integrating-Sphere Optical Properties of Tissues between 630 and 1064nm (J. F. Beek†, P. Blokland, P. Posthumus, M. Aalders, J. W. Pickering, H. J. C. M. Sterenborg and M. J. C. van Gemert) - Phys. Med. Biol. 42 (11) 1997 2255-61
- Simpson R, Kohl M, Essenpreis M and Cope M 1998 Near-Infrared optical properties of ex vivo human skin and subcutaneous tissues measured using the Monte Carlo inversion technique Phys. Med. Biol. 43 2465-2478
- Baran T M, Wilson J D, Mitra S, Yao J L, Messing E M, Waldman D L and Foster T H 2012 Optical property measurements establish the feasibility of photodynamic therapy as a minimally invasive intervention for tumors of the kidney J. Biomed. Opt. 17 (9) 098002
- Rolfe P 2000 Brain - In vivo near-infrared spectroscopy Annu. Rev. Biomed. Eng. 2 715-754
- Roncali E & Cherry S 2013 - Simulation of light transport in scintillators based on 3D characterization of crystal surfaces. Phys. Med. Biol., Volume 58(7), p. 2185–2198.

3.6 Compton camera imaging simulations: CCMod

Table of Contents

3.6.1 Introduction

The Compton camera imaging system has been designed as an actor (see *Tools to Interact with the Simulation : Actors*) that collects the information of the *Hits* in the different layers of the system. The following commands must be employed to add and attach the actor to a volume that contains the whole system:

```
/gate/actor/addActor  ComptonCameraActor      [Actor Name]
/gate/actor/[Actor Name]/attachTo             [Vol Name]
```

The layers of the Compton camera work as *Sensitive Detectors* storing *Hits* (equivalent to volumes attached to crystalSD in PET/SPECT systems). Therefore the digitizer modules described in *Digitizer and readout parameters* can be applied to *Hits* get *Singles*.

A detailed description of CCMod can be found in the article *CCMod: a GATE module for Compton Camera imaging simulation <https://doi.org/10.1088/1361-6560/ab6529>*

3.6.2 Defining the system

A Compton camera system is typically composed of two types of detectors: the scatterer and the absorber. These terms work as key words within the actor. The behavior of the volumes associated to absorber and scatterer *Sensitive detectors* is equivalent to the crystalSD behavior in PET/SPECT systems. The sensitive detectors are specified with the following commands:

```
/gate/actor/[Actor Name]/absorberSDVolume      [absorber Name]
/gate/actor/[Actor Name]/scattererSDVolume     [scatterer Name]
```

For the absorber one single volume is foreseen whereas multiple scatterer layers can be simulated. At least one volume for the absorber and one volume for the scatterer are expected. The total number of scatterer layers must also be specified using the following command:

```
/gate/actor/[Actor Name]/numberOfTotScatterers [Num of scatterers]
```

When multiple scatterer layers are considered, if they are not created using a repeater (copies), the user must name them following a specific guideline. Once the name of one of the volumes is set, the rest of them must share the same name followed by a number to identify them. The numbers are assigned in increasing order starting from 1. For example, if we have three scatterer layers and we want to name them scatterer the name of those volumes must be scatterer, scatterer1 and scatterer2.

There are no constrains for the geometry.

3.6.3 Digitization

The main purpose of the digitization is to simulate the behavior of the detector response. The same data structures (i. e. *Hits*, *Singles*, *Coincidences*) as in PET/SPECT systems have been employed to be able to share the digitizer modules between the systems and the CCMod. Therefore, the digitizer modules described in *Digitizer and readout parameters* can be directly applied to the Compton camera by inserting the modules using the following command. The key word *layers* instead of *singles* must be employed:

```
/gate/digitizer/layers/insert [Module name]
```

Most of the modules available for systems are global modules; thus, they are applied to all the considered sensitive volumes. However, a Compton camera system is typically composed of two different types of detectors (the scatterer and the absorber). Therefore, it is useful to apply a different digitization chain to each of them. To this end, in addition to the global modules, several local modules have been developed that are applied to specific volumes using the following command:

```
/gate/digitizer/layers/[Module name]/chooseNewVolume [SD volume name]
```

3.6.4 List of additional digitizer modules

Here, there is a list of the additional developed modules. ..

> grid discretization (local module), clustering (local and global modules), ideal adder (local and global modules), DoI modeling (global module), time delay (local module), 3D spatial resolution (local module), multiple single rejection (local module), energy threshold module with different policies for effective energies (local and global modules).

Grid discretization module

This module allows to simulate the readout of strip and pixelated detectors. Since it is a local module, the first thing is to attach it to a specific volume that must be acting as a SD:

```
/gate/digitizer/layers/insert gridDiscretization
/gate/digitizer/layers/gridDiscretization/chooseNewVolume [volName]
```

The number of the strips/pixels must be specified in X and Y directions. In addition, the width of the strips/pixel and an offset can be specified to take into account the insensitive material in the detector layer:

```
/gate/digitizer/layers/gridDiscretization/[volName]/setNumberStripsX [Nx]
/gate/digitizer/layers/gridDiscretization/[volName]/setNumberStripsY [Ny]
/gate/digitizer/layers/gridDiscretization/[volName]/setStripOffsetX   [offSet_x]
/gate/digitizer/layers/gridDiscretization/[volName]/setStripOffsetY [offSet_y]
/gate/digitizer/layers/gridDiscretization/[volName]/setStripWidthX [size_x]
/gate/digitizer/layers/gridDiscretization/[volName]/setStripWidthY [size_y]
```

The *hits* detected in the strips/pixels are merged at the center of the strip/pixel in each spatial direction. When strips are defined in both spatial directions, only the hits in the volume defined by the intersection of two strips are stored; thus, generating pixels.

When the grid discretization module is employed to reproduce the response of strip detectors, it should be generally applied followed by a strip activation energy threshold and a multiple single rejection module to avoid ambiguous strip-intersection identification.

On the other hand, when pixelated crystals are simulated, it can be of interest to apply the readout at the level of blocks composed of several pixels. The number of readout blocks can be set individually in each direction using the following commands:

```
/gate/digitizer/layers/gridDiscretization/[volName]/setNumberReadOutBlocksX  [NBx]
/gate/digitizer/layers/gridDiscretization/[volName]/setNumberReadOutBlocksY  [NBy]
```

The energy in the block corresponds to the sum of the deposited energy and the position to the energy weighted centroid position in the pixels that composed the block.

Clustering module

This module has been designed with monolithic crystals read-out by segmented photodetectors in mind. Both versions the global module and its local counterpart have been developed:

```
/gate/digitizer/layers/insert clustering
```

or for the local counterpart:

```
/gate/digitizer/layers/insert localClustering
/gate/digitizer/layers/localClustering/chooseNewVolume [volName]
```

The hits located within the same volume are regrouped by distance, creating clusters. If a detected *hit* is closer than a specified accepted distance to one of the clusters, it is added to the closest one; otherwise, it generates a new cluster. The *hits* are added summing their deposited energies and computing the energy-weighted centroid position. If two clusters are closer than the accepted distance they are merged following the same criteria. If requested, events with multiple clusters in the same volume can be rejected:

```
/gate/digitizer/layers/clustering/setAcceptedDistance [distance plus units]
/gate/digitizer/layers/clustering/setRejectionMultipleClusters [0/1]
```

or for the local counterpart:

```
/gate/digitizer/layers/localClustering/setAcceptedDistance [distance plus units]
/gate/digitizer/layers/localClustering/setRejectionMultipleClusters [0/1]
```

Ideal adder module

This module has been designed with the aim of recovering the exact Compton kinematics to enable further studies.

The adderCompton module was designed with the same aim. However, it does not work properly when there are several photonic hits with secondary electronic hit associated in the same volume since the module only distinguish between photonic and electronic hits. The adderCompton module is designed so that the energy of the electronic *hits* is added to the last photonic hit in the same volume. Therefore, when there are two photonic hits in the same volume, the energy of all the electronic hits is added to the second photonic hit leaving the first hit in general with an incorrect null energy deposition associated.

In order to develop an adder that allow to recover the exact Compton kinematics also when several primary photonic hits occur in the same volume, extra information such as post-step process, creator process, initial energy of the track, final energy, trackID and parentID was added to the pulses. This module creates a *single* from each primary photon *hit* that undergoes a Compton, Photoelectric or Pair Creation interaction. Additional information, such as the energy of the photon before and after the primary interaction that generates the pulse is included to be able to recover the ideal Compton kinematics, hence its name. The deposited energy value of each pulse corresponds to the sum of the deposited energy of the primary hit and all the secondary hits produced by it. The deposited energy has been validated

using livermore physics list. Both versions the global module and its local counterpart have been developed. They can be employed using the following command:

```
/gate/digitizer/layers/insert adderComptPhotIdeal
```

or:

```
/gate/digitizer/layers/insert adderComptPhotIdealLocal
/gate/digitizer/layers/adderComptPhotIdealLocal/chooseNewVolume [volName]
```

The option to reject those events in which the primary photon undergoes at least one interaction different from Compton or Photoelectric is included in the global module using the following command::

```
/gate/digitizer/layers/insert/rejectEvtOtherProcesses [1/0]
```

In order to get one *single* per volume, the user can apply another module afterwards such as the standard adder to handle multiple interactions.

Energy thresholder module

This module apply an energy threshold for the acceptance of pulses. By default, the threshold is applied to the deposited energy. Both versions the global module and its local counterpart have been developed. They can be added using the following commands.:

```
/gate/digitizer/layers/insert energyThresholder
/gate/digitizer/layers/energyThresholder/[volName]/setThreshold [energy]
```

or:

```
/digitizer/layers/insert localEnergyThresholder
/gate/gate/digitizer/layers/localEnergyThresholder/chooseNewVolume [volName]
/gate/digitizer/layers/localEnergyThresholder/[volName]/setThreshold [energy]
```

This threshold is applied to an effective energy that can be obtained using different criteria. Two options have been implemented namely deposited energy and solid angle weighted energy. In order to explicitly specify that the threshold is applied to the deposited energy, the following command should be employed::

```
/gate/digitizer/layers/energyThresholder/setLaw/depositedEnergy
```

or:

```
/gate/digitizer/layers/localEnergyThresholder/[volName]/setLaw/depositedEnergy
```

For the solid angle weighted energy policy, the effective energy for each pulse is calculated multiplying the deposited energy by a factor that represents the fraction of the solid angle from the pulse position subtended by a virtual pixel centered in the X-Y pulse position at the detector layer readout surface. To this end, the size of the pixel and detector readout surface must be specified. Those characteristics are included using the following commands:

```
/gate/digitizer/layers/energyThresholder/setLaw/solidAngleWeighted
/gate/digitizer/layers/energyThresholder/solidAngleWeighted/setRentangleLengthX [szX]
/gate/digitizer/layers/energyThresholder/solidAngleWeighted/setRentangleLengthY [szY]
/gate/digitizer/layers/energyThresholder/solidAngleWeighted/setZSense4Readout [1/-1]
```

or for the local counterpart:

```
/gate/digitizer/layers/localEnergyThresholder/[volName]/setLaw/solidAngleWeighted
/gate/digitizer/layers/localEnergyThresholder/[volName]/solidAngleWeighted/
↪setRentangleLengthX [szX]
/gate/digitizer/layers/localEnergyThresholder/[volName]/solidAngleWeighted/
↪setRentangleLengthY [szY]
/gate/digitizer/layers/localEnergyThresholder/[volName]/solidAngleWeighted/
↪setZSense4Readout [1/-1]
```

If at least the effective energy of one of the pulses is over the threshold, all the pulses corresponding to the same event registered in the studied sensitive volume are stored, otherwise they are rejected.

The global energy thresholder with the default option (deposited energy law) is equivalent to the already available global thresholder.

DoI modeling

The DoI modeling digitizer is applied using the following command.:

```
/gate/digitizer/layers/insert DoImodel
```

The different considered DoI models can be applied to two readout geometries (Schaart et al. 2009): front surface (entrance surface) readout, in which the photodetector is placed on the crystal surface facing the radiation source, and conventional back-surface (exit surface) readout. To this end, the growth-direction of the DoI must be specified using the command.:

```
/gate/digitizer/layers/DoImodel/setAxis [0 0 1]
```

In the above example the growth-direction of the DoI is set to the growth direction of the Z-axis. The criterion for the DoI growth is set towards the readout surface and thereby the DoI value in that surface corresponds to the thickness of the crystal. The opposite surface of the readout surface is referred to as exterior surface. Therefore, the different uncertainty models implemented can be applied to the different readout configurations.

Two options are available for the DoI modelling: dual layer structure and exponential function for the DoI uncertainty. The dual layer model discretizes the ground-truth DoI into two positions in the crystal. If the position of the pulse is recorded in the half of the crystal closer to the readout surface, the DoI is set to the central section, otherwise it is set to the exterior surface. This model can be selected using the following command:

```
/gate/digitizer/layers/DoImodel/setDoIModel dualLayer
```

The DoI exponential uncertainty is modeled as a negative exponential function in the DoI growth-direction. FWHM value at the exterior surface (maximum uncertainty) and the exponential decay constant must be set as input parameters. This uncertainty model and the necessary parameters can be loaded using the following commands.:

```
/gate/digitizer/layers/DoImodel/setDoIModel DoIBlurrNegExp
/gate/digitizer/layers/DoImodel/DoIBlurrNegExp/setExpInvDecayConst [length]
/gate/digitizer/layers/DoImodel/DoIBlurrNegExp/setCrysEntranceFWHM [length]
```

Local Time delay module

This local module delays the time value of the detected pulses in a specified *Sensitive Detector* volume. It can be useful in a Compton camera system, for instance, to delay the *singles* in the scatterer detector when the absorber gives the coincidence trigger:

```
/gate/digitizer/layers/insert localTimeDelay
/gate/digitizer/layers/localTimeDelay/chooseNewVolume [volName]
/gate/digitizer/layers/localTimeDelay/[volName]/setTimeDelay [time value]
```

Local time resolution

In addition to the global time resolution module described in section *Digitizer and readout parameters* a local version has been included in order to be able to set different time resolutions to the different layers:

```
/gate/digitizer/layers/insert localTimeResolution
/gate/digitizer/layers/localtimeResolution/setTimeResolution [FWHM value]
```

Local 3D spatial resolution

This local module sets independently a Gaussian spatial resolution in each spatial direction. The module is inserted using the following command:

```
/gate/digitizer/layers/insert sp3Dlocalblurring
/gate/digitizer/layers/sp3Dlocalblurring/chooseNewVolume [vol name]
```

and the sigma of the Gaussian function in each direction is set:

```
/gate/digitizer/layers/sp3Dlocalblurring/[vol name]/setSigma [vector (length)]
```

Local Multiple rejection module

This is a local module that allows you to discard multiple pulses. It can be inserted using the following commands.:

```
/gate/digitizer/layers/insert localMultipleRejection
/gate/digitizer/layers/localMultipleRejection/chooseNewVolume [vol]
```

The definition of what is considered multiple pulses must be set. Two options are available: more than one pulse in the same volume name or more than one pulses in the same volumeID. When several identical volumes are needed, for example for several scatterer layers, they are usually created as copies using a repeater. In that case, all volumes share the same name but they have different volumeID. The difference between the rejection based on volume name and volumeID is important in those cases. These options are selected using the following command line.:

```
/gate/digitizer/layers/localMultipleRejection/[vol]/setMultipleDefinition [volumeID/
↪volumeName]
```

Then, the rejection can be set to the whole event or only to those pulses within the same volume name or volumeID where the multiplicity happened.:

```
/gate/digitizer/layers/localMultipleRejection/[vol]/setEventRejection [1/0]
```

3.6.5 Sorter

The sorter developed in GATE for PET systems has been adapted for the CCMod, see coincidence_sorter-label. Same command is employed.:

```
/gate/digitizer/Coincidences/setWindow [time value]
```

An additional option has been included to allow only *singles* in the absorber layer to open its own time window, i. e. absorber coincidence trigger. By default, this option is disabled. In order to enable it the following command must be employed:

```
/gate/digitizer/Coincidences/setTriggerOnlyByAbsorber 1
```

Different coincidence acceptance policies are available for Compton camera: *keepIfMultipleVolumeIDsInvolved*, *keepIfMultipleVolumeNamesInvolved*, *keepAll*. They can be selected using the following command line:

```
/gate/digitizer/Coincidences/setAcceptancePolicy4CC keepAll
```

KeepAll policy accepts all coincidences, no restriction applied.

KeepIfMultipleVolumeIDsInvolved policy accepts *coincidences* with at least two *singles* in different volumeIDs.

KeepIfMultipleVolumeNamesInvolved is the default *coincidence* acceptance policy. *Coincidences* are accepted if at least two of the *singles* within the *coincidence* are recorded in different SD volume names. Volumes created by a repeater have same volume name but different volumeID.

3.6.6 Coincidence processing

The described modules in coincidence_processing-label to process coincidences in PET systems such as dead-time or memory buffer can be in principle applied directly to CCMod using the same commands:

```
/gate/digitizer/name sequenceCoincidence
/gate/digitizer/insert coincidenceChain
/gate/digitizer/sequenceCoincidence/addInputName Coincidences
```

However, since they are designed for PET systems, some of them reject multiple *coincidences* (more than two *singles*).

Coincidence Sequence Reconstruction (CSR) module has been included for CCMod. It is a *coincidence* processor which modifies the order of the *singles* within a *coincidence* to generate a *sequence coincidence*:

```
/gate/digitizer/sequenceCoincidence/insert [name]
```

Different policies have been implemented to order the *singles* within a *coincidence*: randomly, by increasing single time-stamp value (ideal), axial distance to the source (first scatterer then absorber) or deposited energy. Those policies can be selected using the following commands.:

```
/gate/digitizer/sequenceCoincidence/[name]/setSequencePolicy randomly
/gate/digitizer/sequenceCoincidence/[name]/setSequencePolicy singlesTime
/gate/digitizer/sequenceCoincidence/[name]/setSequencePolicy axialDist2Source
/gate/digitizer/sequenceCoincidence/[name]/setSequencePolicy lowestEnergyFirst
```

In addition, a policy based on the so-called revan analyzer from Megalib (Zoglauer et al. 2008), known as Classic Coincidence Sequence Reconstruction (CCSR) has been included. ..

> (It is disabled from the messenger since the the errors in energy and posiiton are not properly included in the pulses)
>
> /gate/digitizer/sequenceCoincidence/[name]/setSequencePolicy revanC_CSR

3.6.7 Data output

Output data is saved using the following command:

```
/gate/actor/[Actor Name]/save   [FileName]
```

Data can be saved in .npy, .root or .txt format. The format is taken from the extension included in the chosen FileName. The information of the *Hits*, *Singles*, *Coincidences* and Coincidence chains can be stored:

```
/gate/actor/[Actor Name]saveHitsTree          [1/0]
/gate/actor/[Actor Name]/saveSinglesTree        [1/0]
/gate/actor/[Actor Name]/saveCoincidencestTree      [1/0]
/gate/actor/[Actor Name]/saveCoincidenceChainsTree  [1/0]
```

For each data format (*Hits*, *Singles*, *Coincidences*, processed coincidence name) a new file is generated with the label of the data included. For examples if the FileName is test.root, then *Singles* are saved in the file called test_singles.root.

Most of the information in the output file can be enabled or disabled by the user. For example, the information of the energy deposition can be disabled using the following command:

```
/gate/actor/[Actor Name]/enableEnergy 0
```

An additional file with electron escape information can be stored:

```
/gate/actor/CC_digi_BB/saveEventInfoTree            [1/0]
```

If this option is enabled and the chosen general FileName is for example *test.root*, a new file *test_eventGlobalInfo.root* is generated. For each electron that goes through a SD volume, a flag that indicates if the electron enters or exits the volume, the SD detector volume name, the energy of the electron, the eventID and the runID are stored.

Optional additional source information

Hits and *Singles* contain information about the source, i.e. energy and particle type (PDGEncoding). When an ion source is employed, instead of the information of the ion, the information associated with one of the particles emitted in the decays can be of interest. An extra option has been included in the actor that allows to specify the parentID of the particle that is going to be considered as *source*. By default, this option is disabled. It can be enabled using the following command:

```
/gate/actor/[Actor Name]/specifysourceParentID 0/1
```

When the option is enabled (it is set to 1), a text file must be included with a column of integers corresponding to the parentIDs of the particles that are going to be considered as primaries:

```
/gate/actor/[Actor Name]/parentIDFileName  [text file name]
```

For example, in the case of a 22Na source, we are interested in the 1274 keV emitted gamma-ray and the annihilation photons that can be identified using a value for the parentID of 2 and 4 respectively (at least using livermore or em opt4 physics list).

3.6.8 Offline processing

Be aware that only .root extension output files can be processed offline. The following executables:

- GateDigit_hits_digitizer
- GateDigit_singles_sorter

- GateDigit_coincidence_processor

perform respectively an offline digitization, an offline sorter and an offline sequence coincidence reconstruction. In order to use these executables during GATE compilation GATE_COMPILE_GATEDIGIT must be set to ON.

3.7 Third-party reconstruction software

Several third-party software can be used to reconstruct images from output of GATE simulations. Here a list (without any particular order).

- STIR
- Castor
- OMEGA
- RTK

list of examples

CHAPTER 4

Radiotherapy and dosimetry applications

4.1 Radiotherapy General Concept

Table of Contents

4.1.1 General concept

The Radiotherapy and Dosimetry reference paper *GATE V6: a major enhancement of the GATE simulation platform enabling modelling of CT and radiotherapy* is available here. If you are interested in Gate simulations in a clinical environment for light ion beam radiotherapy, then you may want to have a look at GateRTion

A **list of radionuclides** is availabe at NUCLEIDE.ORG.

The concept of Actors is very important for the simulation of radiotherapy treatments and dosimetry. *Tools to Interact with the Simulation : Actors* and *MuMapActor* are tools which allow to interact with the simulation. They can collect information during the simulation, such as energy deposit, number of particles created in a given volume, etc. They can also modify the behavior of the simulation. There are different types of actors which collect different types of information, however some commands and behavior are common to all actors. To use selection criteria, it is possible to add filters.

4.1.2 Insert a CT image in the therapy simulation

For a realistic radiotherapy simulation, one wants to use a patient CT image as the voxelized phantom. For that, the following steps must be followed.

Step1: convert Dicom image to Analyze image file format

The vv, the 4D Slicer software can be used for this purpose. It opens the DICOM image and saves it as a header plus raw data. It can also resample the image.

Step2: define HU to materials conversion

The following macro can be used to generate Hounsfield Unit (HU) to material conversion:

```
/gate/HounsfieldMaterialGenerator/SetMaterialTable                          ␣
↪MaterialsTable.txt
/gate/HounsfieldMaterialGenerator/SetDensityTable                           ␣
↪DensitiesTable.txt
/gate/HounsfieldMaterialGenerator/SetDensityTolerance                        0.1 g/cm3
/gate/HounsfieldMaterialGenerator/SetOutputMaterialDatabaseFilename   myimage-
↪HUmaterials.db
/gate/HounsfieldMaterialGenerator/SetOutputHUMaterialFilename            myimage-
↪HUmat.txt
/gate/HounsfieldMaterialGenerator/Generate
```

Step3: insert image as a volume

Example:

```
/gate/world/daughters/name                    patient
/gate/world/daughters/insert                  ImageNestedParametrisedVolume
/gate/patient/geometry/SetImage               myimage.hdr
/gate/geometry/setMaterialDatabase            myimage-HUmaterials.db
/gate/patient/geometry/SetHUToMaterialFile    myimage-HUmat.txt
/gate/patient/placement/setTranslation        0 0 0 mm
```

4.1.3 Dose distribution (Dosimetry)

GATE allows to simulate the dose distribution in a phantom during radiotherapy (therapy with photons) and hadrontherapy (therapy with hadrons : proton, Carbon...). It also allows to perform other radiation simulations such as brachytherapy, dose deposited during x-ray imaging... Fig. 4.1 shows an example of a dose distribution map obtained after radiotherapy treatment on a phantom.

The *Dose measurement (DoseActor)* is the tool that is used to store the **deposited dose in a 3D matrix** which size and resolution can be specified. It can also be used for 1D and 2D dose maps. It can store *dose*, *edep* or *number of hits* and computes associated statistical uncertainty. The **DoseActor** is attached to a volume which can be voxelized or not.

4.2 Beam modelling

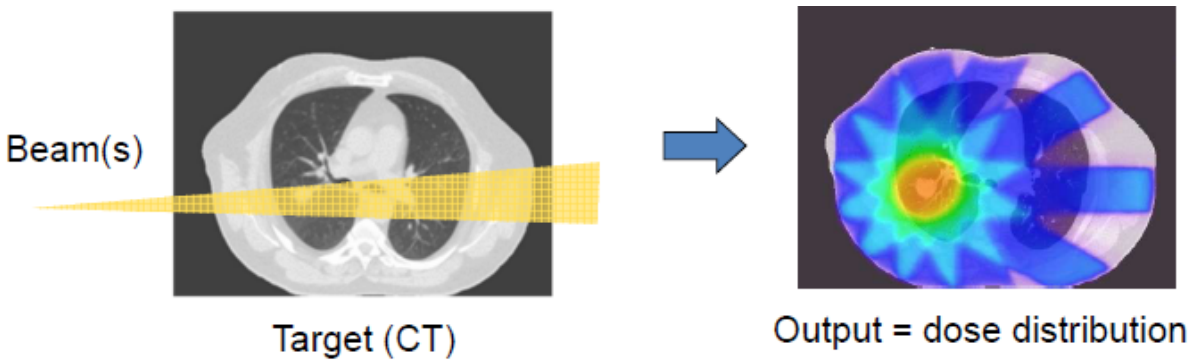


Fig. 4.1: Dose distribution after radiotherapy treatment on a phantom.

Table of Contents

4.2.1 Photon beam : LINAC

A realistic photon beam in a multilayer phantom can be modelled using GATE. The phantom is irradiated with a photon beam, originating from a uniform point source at a certain distance from the phantom surface and collimated at the surface of the phantom using *Kill track*. The energy spectrum of the point source is modelled using the configurations of a VARIAN Clinac for example. The depth dose distribution was calculated along the central axis with a 1D *Dose measurement (DoseActor)*. The production threshold can be set to different values in the world and in the phantom for electrons, positrons and photons.

4.2.2 Hadron therapy application

As an example of a hadron therapy application, we simulate a **C12** scanning **pencil beam** irradiation of an artificial spherical target inside a patient CT image. The treatment plan comprised different pencil beams. Fig. 4.3 illustrates the dose deposition measured using GATE.

4.2.3 In-beam PET modelling

GATE is able to jointly model radiation therapy and emission tomography in order to monitor the deposited dose for a C12 treatment using PET imaging as shown in Fig. 4.4. It models the Carbon beam, the nuclear fragmentation, the β+ emitters, back to back photon and dose monitoring!

A 4D CT patient scan was used to define the numerical phantom. A three **beam treatment** plan was modelled with a total production of 3 beams of 10e9 C12 ions. Each beam was composed of 195 independent spots with 42 different incident energies between 175 and 230 MeV/u. All **hadronic and electromagnetic processes** were included in the simulation. Positron range and annihilation photon pair acollinearity were taken into account. A model of the **Siemens HR+ PET system** was used to simulate a 20 min static acquisition starting immediately after the irradiation. PET data were normalized, corrected for attenuation using an attenuation map derived from the CT, and reconstructed using 3D back projection. The simulation was performed in less than 24 h on a cluster of 1000 Intel Nehalem 2.93 GHz CPUs. Fig. 4.5 shows the reconstructed PET images. It suggests that the C11 activity distribution contains most

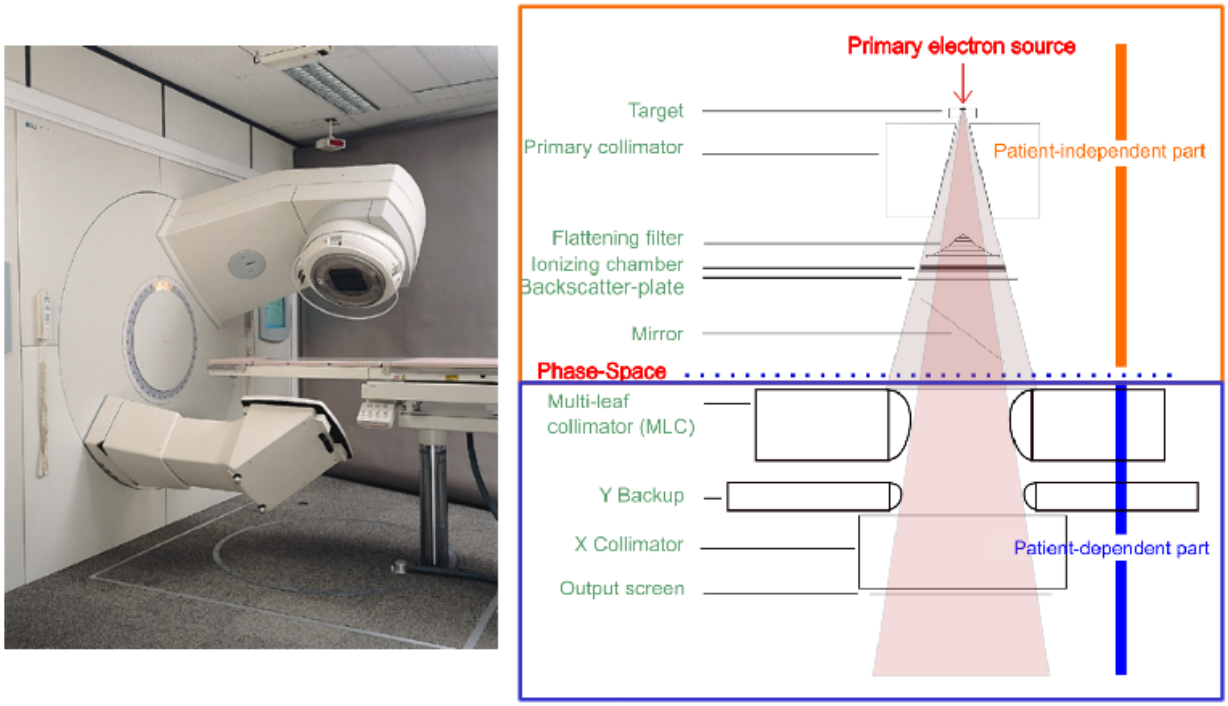


Fig. 4.2: LINAC

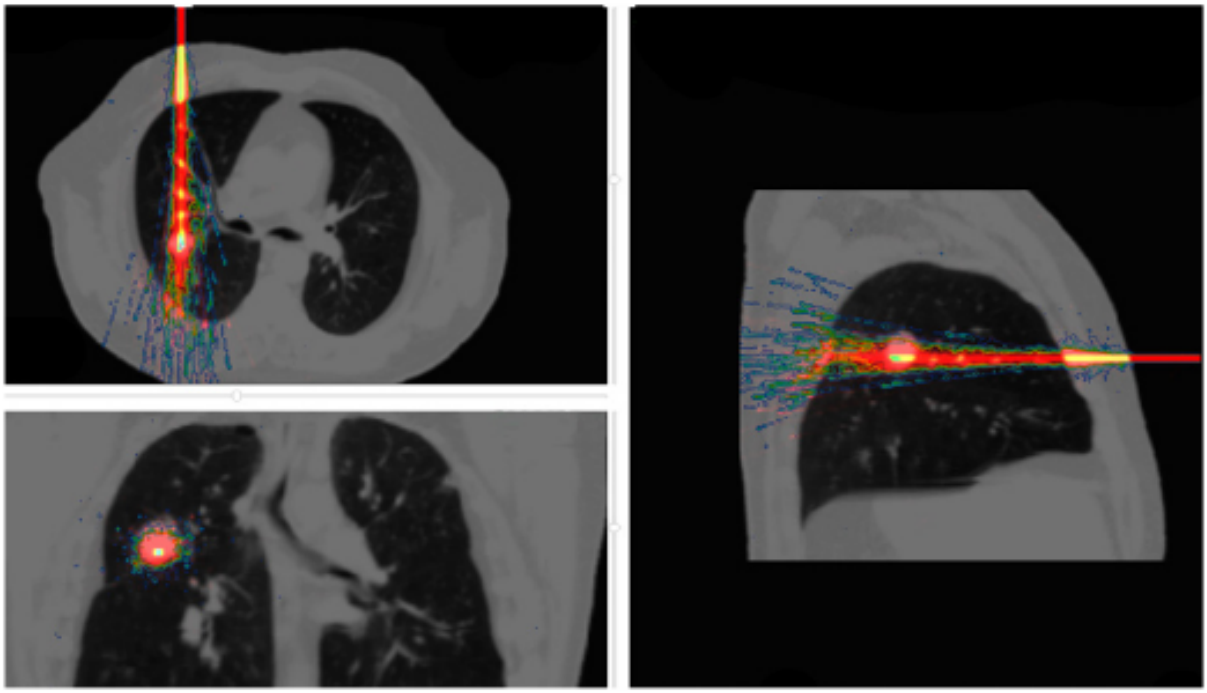


Fig. 4.3: Dose deposited by a carbon ion beam inside a CT image of a thorax. The colour scale is a warm metal scale, with high values (white) corresponding to high-dose deposit and low values (blue) corresponding to low-dose deposit.

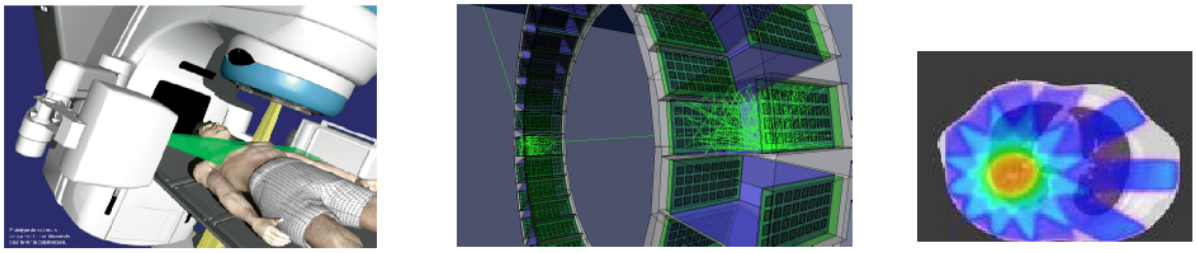


Fig. 4.4: Monitoring of the deposited dose using PET imaging for a C12 treatment.

information regarding the location of the Spread Out Bragg Peak (SOBP), while the O15 activity might be relevant to derive information about the dose to normal tissues.

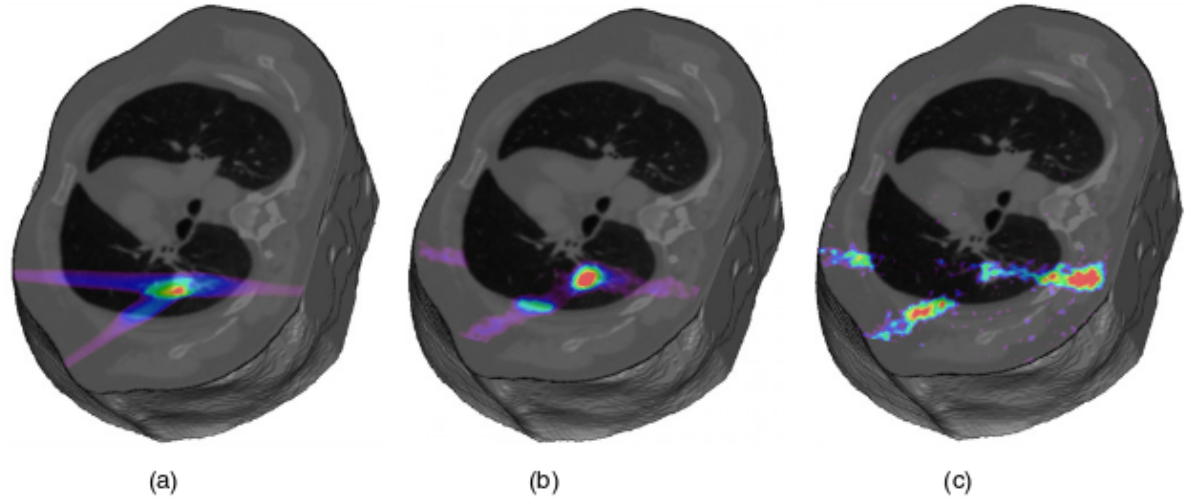


Fig. 4.5: (a)Dose distribution simulated for a C12 irradiation inside a CT image of a thorax. (b) Simulated PET image of the resulting C11 isotope distribution. (c) Simulated PET image of the O15 isotope distribution.

list of exercises and examples

CHAPTER 5

Thermal therapy application

5.1 Nanoparticle mediated hyperthermia

Table of Contents

5.1.1 Pre-requisite

To use the nanoparticle mediated hyperthermia capabilities of GATE, you have to install ITK and turn the CMake flag GATE_USE_ITK to ON in the configuration process using ccmake. Please find detailed instructions here: *Installation Guide V9.0*

5.1.2 Theory

Optical photon deposited energy map

The optical photons emitted by an illumination light (i.e. laser) will be absorbed by the tissues (*OpticalAbsorption process*) or/and by a tissue loaded with a certain concentration of nanoparticles (*NanoAbsorption process*).

In GATE, the optical photon is transported following a step length which is randomly sampled using the mean free path of each physics process associated to the optical photon. The mean free path of the optical photon interaction with the *nanoparticle-infused* medium is L_a. The inverse of the absorption length is referred to as the absorption coefficient (μ_a) and is a function of the density of nanoparticles in the medium (N in m-3) and the nanoparticle absorption cross-section area (C_{abs} in m^2) : $\mu_a = N \times C_{abs}$.

Pennes bioheat model and the analytical solution

The mathematical model that describes the thermal behavior in biological perfused tissues is the Pennes bioheat model :

$\frac{\partial T(x,y,z,t)}{\partial t} = \frac{k}{\rho c}\nabla^2 T(x,y,z,t) + \frac{\rho_b c_b}{\rho c} w_b [T_a(x,y,z,t) - T(x,y,z,t)] + Q(x,y,z,t)$

where Q represents the energy deposition by any external heat source such as the metabolic heat production in the tissue and is considered to be 0.

The first term of the equation describes the transfer of energy between objects in contact (i.e. conduction); the second term accounts for the effects of blood perfusion in tissues. k, ρ and c are the biological tissue thermal conductivity, density and specific heat. Values for blood are given by ρ_b and c_b; w_b is the tissue blood perfusion rate which represents the volume of fluid (i.e. blood) that passes per unit time and per tissue volume. T_a is the temperature of blood in the main arteries and $T(x,y,z,t)$ is the local tissue temperature. Pennes equation is solved analytically via Fourier transformations and convolution theorem. The analytical solution to the Pennes bioheat equation is

$T(x,y,z,t) = [T(x,y,z,0) - T_a] \otimes \frac{1}{(4\pi K_1 t)^{3/2}} e^{x^2+y^2+z^2/4K_1 t} \times e^{-K_2 t} + T_a$

with $K_1 = \frac{k}{\rho c}$ the tissue thermal diffusivity.

The solution of the diffusion equation is equivalent to convolving the initial conditions (3D energy map) with a Gaussian with a standard deviation $\sigma = \sqrt{2tK_1}$. The blood perfusion term appears as an exponential function. The implementation of the heat diffusion in GATE is performed using the *Insight Segmentation and Registration Toolkit* (ITK) which is an open-source, cross-platform system that provides developers with an extensive suite of tools for image analysis.

Hybrid Monte-Carlo and analytical simulation: a dynamic process

During light illumination of a biological tissue, the thermal heat produced by the optical photons deposited energy does not accumulate locally in the tissue; it diffuses in biological tissues during illumination. This dynamic effect has been taken into account in the GATE code. The n seconds light illumination simulation is sampled into p time frame 3D images by setting the simulation parameter *setNumberOfTimeFrames* to p. Each of the p sample images is diffused for a duration of [1, 2, ..., p-1] x n/p seconds. The final voxelized image illustrating the heat distribution in the tissues at the end of the illumination time is obtained by adding all diffused images to the last n/p seconds illumination image. This thermal energy (or heat) map will then diffuse in the biological tissues by setting the simulation parameter *setDiffusionTime* to the value of interest. At a certain point in time after the initial temperature boost induced by nanoparticles, the temperature of the tissues will go back to its initial value due to diffusion. This boundary condition is taken into account in a post processing-step of the GATE simulation.

5.1.3 Illustration of the heat diffusion 3D map obtained by the ThermalActor

In the following example, a cubic phantom made of a material defined with an absorption length L_a and a thermal diffusivity is illuminated by a light source of optical photons with a certain photon flux (i.e. counts per second). The source direction is set perpendicular and positioned towards the phantom surface.

The *ThermalActor* provides the 3D map of the deposited optical photon energy which has diffused during illumination. The actor outputs the following images :

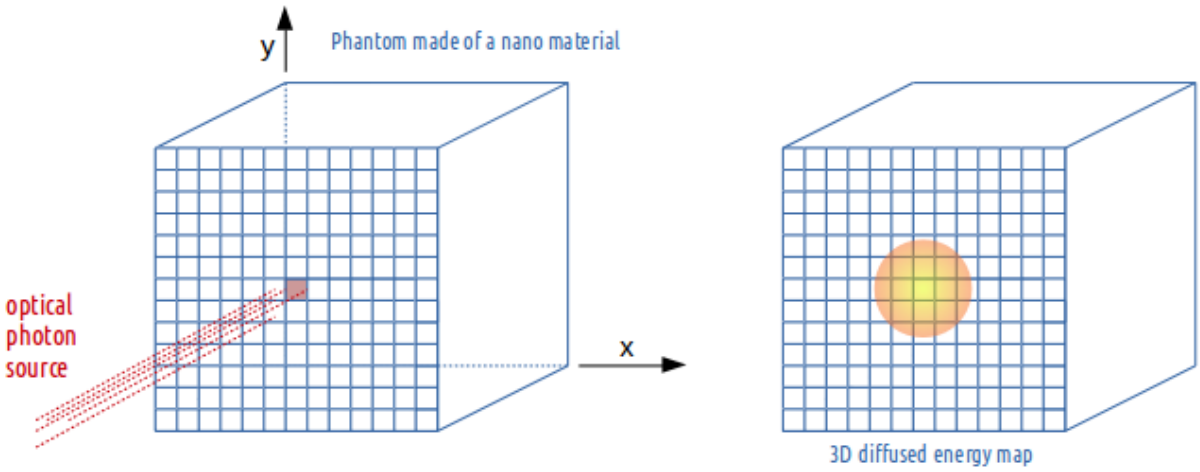


Fig. 5.1: ThermalActor

- FinalAbsorptionMap.img/hdr <= The image at the end of the light illumination (photon energy diffuses during illumination)
- FinalHeatDiffusionMap.img/hdr <= The image after a certain diffusion time following the end of the light illumination

5.1.4 Command lines

Example:

```
/gate/actor/addActor ThermalActor                MyActor
/gate/actor/MyActor/save                         3DMap.hdr
/gate/actor/MyActor/attachTo                     phantom
/gate/actor/MyActor/stepHitType                  random
/gate/actor/MyActor/setPosition                  0. 0. 0. cm
/gate/actor/MyActor/setVoxelSize                 0.5 0.5 0.5 mm

Tissue thermal property :
/gate/actor/MyActor/setThermalDiffusivity        0.32 mm<sup>2</sup>/s
```

Density and heat capacity should just be in the same unit for both blood and tissue. In the following example, the density is in kg/mm3 and the heat capacity in mJ kg-1 C-1:

```
/gate/actor/MyActor/setBloodDensity              1.06E-6
/gate/actor/MyActor/setBloodHeatCapacity         3.6E6
/gate/actor/MyActor/setTissueDensity             1.04E-6
/gate/actor/MyActor/setTissueHeatCapacity        3.65E6
/gate/actor/MyActor/setBloodPerfusionRate        0.004

/gate/actor/MyActor/setDiffusionTime             5 s
/gate/actor/MyActor/setNumberOfTimeFrames        5
```

CHAPTER 6

Parallel computing

6.1 How to use Gate on a Cluster

Table of Contents

To reduce the overall computing time of GATE experiments, a parallel computing platform for running simulations in a cluster of computers was developed which significantly shortens the setup time and provides fast data output handling. To use Gate in cluster mode you need 3 components:

- The job splitter (gjs)
- The file merger (gjm)
- A cluster aware version of Gate

6.1.1 Installation of the job splitter (gjs)

The job splitter can be installed in the same directory as Gate. Two environment variables are already added to the environment file used to compile Gate (but you can customize them):

```
export GC_DOT_GATE_DIR=/somedir/
export GC_GATE_EXE_DIR=/somedir/bin/Linux-g++/
```

The first variable indicates the location of a hidden directory called .Gate. The directory will contain the split macros for each part of the simulation. Even when splitting the same macro several times, a new directory will be created for each instance (with an incremental number). In normal circumstances, one does not need to look into it. In case of an error, it can be used to run only a specific part of a simulation again (See *What about errors?*).

The second environment variable indicates the location of the job splitter executable. As the Gate environment file will be used to compile the job splitter source code, the executable will likely be located in the same directory as the Gate executable.

To install, load the Gate/Geant4 environment variables, go to the job splitter directory (bash example):

```
source env_gate.sh
cd jobsplitter
make
```

By default, the executable will be created in the jobsplitter directory. If the GATEHOME variable is correctly defined, the executable will also be copied in the same directory as the Gate executable (same for the dynamic library).

6.1.2 Installation of the file merger (gjm)

To install, it is the same way, go to the file merger directory and compile (bash example):

```
cd filemerger
make
```

The file merger executable is located in the current directory (and will also be copied in the Gate bin directory as for the gjs program).

6.1.3 Preparing your macro

The cluster software should be able to handle all GATE macros. However, only ROOT is currently supported as an output format for the gjm program. So be aware that other output formats cannot yet be merged with the gjm program and you will have to do this on your own (but it is usually quite simple ~ addition or mean most of the time).

If an isotope with a shorter half life than the acquisition time is simulated, then it may be useful to specify the half life in your macro as follows:

```
/gate/cluster/setTimeSplitHalflife 6600. s
```

This way, the CPU time will be approximately equal for each job.

In planning simulation time, it is important to be aware that Gate simulations currently seem to benefit only from the addition of physical CPUs. A computer with 8 hyper-threaded physical CPU cores (16 logical CPUs) will have the same computational efficiency if 8 processes are run simultaneously as it would with 16 simultaneous processes.

6.1.4 Using the job splitter

To view information regarding general usage, you can run the job splitter executable without any options:

```
+-----------------------------------------+
| gjs -- The GATE cluster job macro spliter |
+-----------------------------------------+

Usage: gjs [-options] your_file.mac

Options (in any order):
-a value alias              : use any alias
-numberofsplits, -n   n     : the number of job splits; default=1
-clusterplatform, -c  name : the cluster platform, name is one of the following:
                              openmosix - condor - openPBS - xgrid
                              This executable is compiled with condor as default

-openPBSscript, os          : template for an openPBS script
                              see the example that comes with the source code (script/
↪openPBS.script)
                              overrules the environment variable below

-condorscript, cs           : template for a condor submit file
                              see the example that comes with the source code (script/
↪condor.script)
-v                          : verbosity 0 1 2 3 - 1 default

Environment variables:
GC_DOT_GATE_DIR  : indicates the .Gate directory for splitted mac files
GC_GATE_EXE_DIR  : indicates the directory with the Gate executable
GC_PBS_SCRIPT : the openPBS template script (!optionnal variable!)

Usage (bash):
  export GC_DOT_GATE_DIR=/home/user/gatedir/
  export GC_GATE_EXE_DIR=/home/user/gatedir/bin/Linux-g++/

Examples:
  gjs -numberofsplits 10 -clusterplatform openmosix macro.mac
  gjs -numberofsplits 10 -clusterplatform openmosix -a /somedir/rootfilename ROOT_
↪FILE macro.mac
  gjs -numberofsplits 10 -clusterplatform openPBS -openPBSscript /somedir/script␣
↪macro.mac
  gjs -numberofsplits 10 -clusterplatform xgrid macro.mac
  gjs -numberofsplits 10  /somedir/script macro.mac
```

The supported platforms are currently: openMosix, openPBS, Condor and Xgrid.

Let's take openMosix as an example:

```
gjs -numberofsplits 5 -clusterplatform openmosix macro.mac
```

The job splitter will subdivide the simulation macro into fully resolved, non-parameterized macros. In this case there are 5 such macros. They are located in the .Gate directory, as specified by the GC_DOT_GATE_DIR environment variable.

A list of all the data output options is given after successful completion, as well as a list of all activated actors. The user is asked to clearly enable each needed output module and to give them an output file name. It is the same for actors. Remember that by default, no output module nor actor is enabled.

If an alias was expected for output files and it was not supplied, then this will be mentioned in the output options list. A standard name will be supplied automatically, as well as appropriate numbering.

The time of each sub-macro is manage using a virtual timeStart and a virtual timeStop calculated by the gjs and used

by the command /gate/application/startDAQCluster. All defined runs and geometry updates will be totally respected. The only inconsistency in the use of gjs is when using the projection output: the virtualStop minus virtualStart time have to be a multiple of timeSlice, otherwise the GateToProjectionSet output will lead to an error.

The .Gate directory will have a subdirectory called as the macro name, that contains the following files:

```
macro1.mac
macro2.mac
macro3.mac
macro4.mac
macro5.mac
macro.split
```

The 5 macros are listed as well as well as the .split file that contains information about the splitted simulation and that will be used to merge the data after the simulation (using the gjm program). The current directory, from which the jobsplitter was called, now contains the cluster submit file. In order to run the split simulation on the cluster, one only needs to execute or call this file with a certain program (depending on the cluster platform used).

The .Gate directory supports automatic numbering. If the same macro is used repeatedly, then the subsequent directories will be numbered using an incremental number.

6.1.5 Using the file merger

The file merger have to be run giving the split file as input. To view information on general usage, just run the file merger executable without any options:

```
+-----------------------------------------+
| gjm -- The GATE cluster job output merger |
+-----------------------------------------+

Usage: gjm [-options] your_file.split

You may give the name of the split file created by gjs (see inside the .Gate␣
↪directory).
!! This merger is only designed to ROOT output. !!

Options:
-outDir path              : where to save the output files default is PWD
-v                        : verbosity 0 1 2 3 - 1 default
-f                        : forced output - an existing output file will be␣
↪overwritten
-cleanonly                : do only a the cleanup step i.e. no merging
                            erase work directory in .Gate and the files from the␣
↪parallel jobs
-cleanonlyTest            : just tells you what will be erased by the -cleanonly
-clean                    : merge and then do the cleanup automatically
-fastMerge                : correct the output in each file, to be used with a TChain␣
↪(only for Root output)

Environment variable:
GC_DOT_GATE_DIR : points to the .Gate directory
```

To merge the output files into a single file, just supply the split file to the file merger. The output file could be used as a usual single CPU output file:

```
gjm macro.split

Combining: ./rootf1.root ./rootf2.root ./rootf3.root ./rootf4.root ./rootf5.root $->$␣
↪./rootf.root
```

In case a single output file is not required, it is possible to use the option **fastMerge**. This way, the eventIDs in the ouput files are corrected locally. Fig. 6.1 shows the newly created tree in each ROOT file.

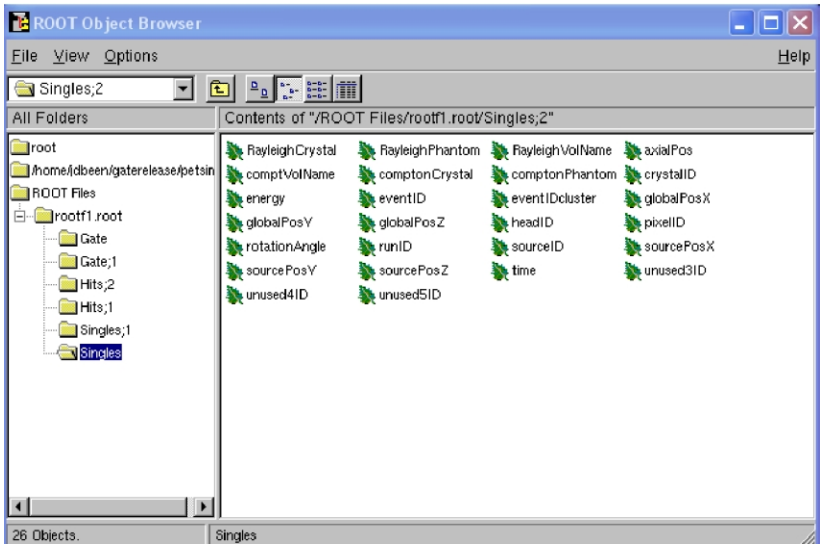


Fig. 6.1: Example of ROOT file with added cluster eventIDs

A ROOT chain, which is a list of files containing the same tree, is then required to link the output files together for analysis. A chain for the Singles could be made as follows (in a file called chain.c):

```
\{
gROOT->Reset();
TChain chain("Singles");
chain.Add("rootf1.root");
chain.Add("rootf2.root");
chain.Add("rootf3.root");
chain.Add("rootf4.root");
chain.Add("rootf5.root");
\}
```

Once all files are added to the chain, one can use the chain as a regular Ttree, and the normal ROOT prompt is returned:

```
 $root chain.c

FreeType Engine v2.1.3 used to render TrueType fonts.
Compiled for linux with thread support.
CINT/ROOT C/C++ Interpreter version 5.15.94, June 30 2003
Type ? for help. Commands must be C++ statements.
Enclose multiple statements between \{ \}.
root [0]
Processing chain.c...
root [1]
root [1] Singles->Draw("energy")
```

6.1.6 Alternative to the file merger

Root files can also be merged by using the **hadd** utility on the command line:

```
hadd result.root file1.root file2.root ... filen.root
```

6.1.7 What about errors?

If something went wrong during a simulation and a ROOT file is corrupted or incomplete, then this will be detected by the file merger. There are two options. First, one can restart only the specific part of the simulation that went wrong. This can be easily done, as the ROOT files are numbered and one can edit the submit file so it only launches that specific part. Alternatively, one can find the macro file that was used to start that part of the simulation in the .Gate directory and start the simulation directly with the macro file and its corresponding seed file.

The second option is to edit the split file, located in the .Gate directory. Once the reference to the corrupted root file is removed from it, it is possible to merge the files again. At this point, the eventIDs will not be valid anymore.

6.2 How to use Gate on a GPU

!! The GPU module has been discontinued !!

This experiment ends and the code have been removed. You still can find some details here: https://github.com/OpenGATE/Gate/pull/322. Part of the experience have been transfered to the GGEMS code: https://ggems.fr

CHAPTER 7

GateTools

The GateTools repository contains a list of python command line tools to facilitate Gate simulations running and analysis. See associated readme for information.

CHAPTER 8

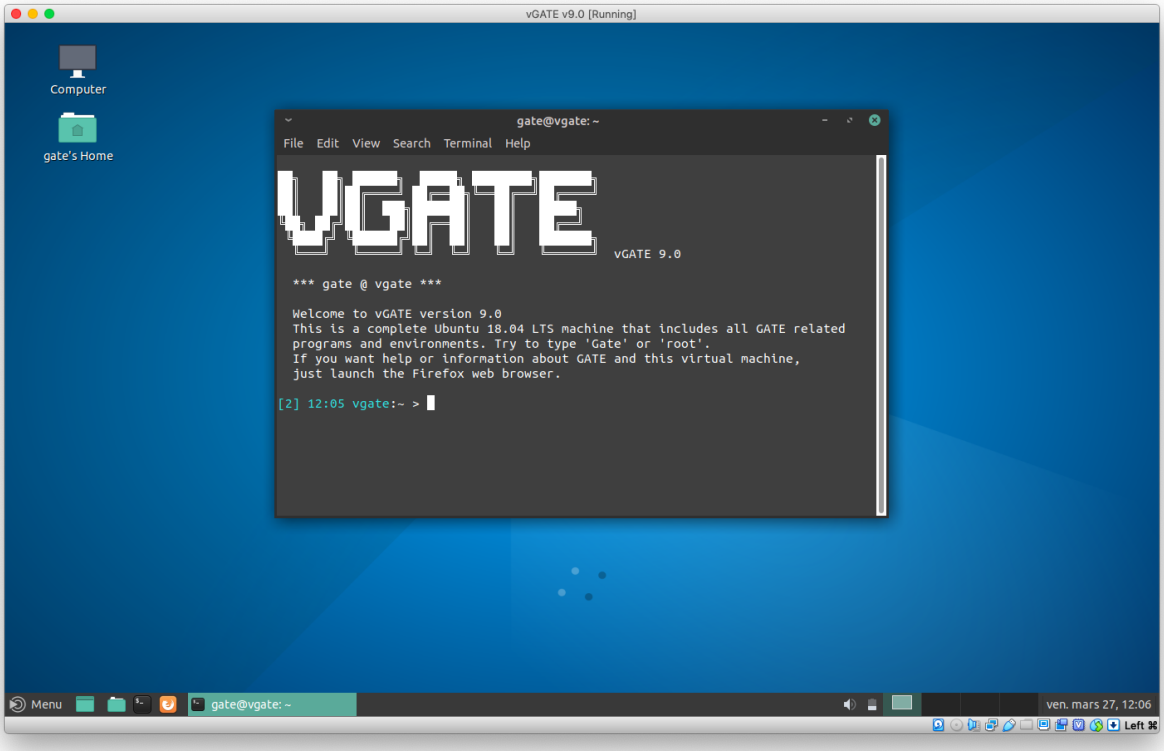
vGate (virtual Gate)

Table of Contents

8.1 Generalities

8.1.1 What is vGate?

vGate stands for Virtual Gate. It is a complete virtual machine running an Ubuntu 64 bits operating system and made using the free software Virtual Box. This virtual machine can be run on any host machine (Linux, Windows, MacOS, ...) provided the Virtual Box program is installed and ready for use. Note: also install the VirtualBox Extension Pack that provides support for USB 2.0 and USB 3.0 devices, and will allow you to create a shared folder between your computer and vGate.



With vGate you can launch your first GATE simulation in just a few steps! No need to install anything, no need to configure anything, and no time spent to understand compilation and related stuff. A full Linux environment is totally set up to be able to use GATE just by launching a simple command: “Gate”.

The following software is installed on this machine:

- Ubuntu LTS 18.04 on Virtual Box (40GB virtual HD)
- GATE 9.0
- Geant4 10.06.1
- GateContrib: a user-oriented public repository of Gate (macros, examples and user contributions)
- Gate tools
- Gate exercises
- Root 6.14.00
- libtorch cxx11 (cpu) 1.4.0
- ITK 4.13.1 with Module_RTK=ON (v2.0.0)
- VTK v7.1.0
- vV 1.4
- ImageJ (Fiji) 1.52d
- Visual studio code
- Jupyter Notebook, Jupyter Lab

- Python3 libraries: numpy, matplotlib, scipy, pydicom, pandas, SimpleITK, uproot

8.1.2 How to get vGATE now?

Go to the OpenGate collaboration website. You will then be able to download the virtual machine under the "Download/vGATE" menu.

Be aware that the file you will download a pretty big (about 10 Gbytes), so if several users are downloading the file at the same time, your download speed will be limited and you will have to be patient.

8.1.3 How to use vGATE now?

As the vGate machine has been built using the Virtual Box software, you will have to install this software on your host machine first. And since the version of Virtual Box used to build vGate was the release 6.0.18, you have to install at least this version to be able to run the virtual machine.

Once Virtual Box is installed, here are the steps to get your virtual machine working:

- Launch Virtual Box and in the File menu, select Import Appliance. The Appliance Import wizard is displayed in a new window.
- Click on the small yellow folder icon 'Choose', browse to the location containing the *.ovf or *.ova file of the virtual machine you want to import, and click Open. The Appliance Import Settings step will display.
- Click Import. The Appliance Import Wizard is closed and after a few moments, the imported virtual machine will be listed in Oracle VM VirtualBox Manager.
- After the import, select the imported virtual machine and in the toolbar click the Settings button. Review the virtual machine settings to make sure that the virtual machine has the hardware it needs to operate. You can adjust the number of CPUs and the RAM you want to give to the VM.
- Once you have reviewed the settings, select the imported virtual machine and in the toolbar click the Start button.

That's it!

Login credentials are (qwerty keyboard):

- user: gate
- password: virtual
- ('gate' is sudo)

8.1.4 How can I find and launch GATE in vGate?

Everything is already configured in the virtual machine to be able to launch GATE without any difficulty. If you want to know how the machine has been configured, you can find all information inside the virtual machine.

Once you start the virtual machine, you can launch the web browser Firefox. Firefox is directly showing the documentation pages (in HTML) that are inside the virtual machine. So please refer to this documentation.

To launch Gate simply open a terminal and type:

```
Gate
```

Any additional questions can be posted on the gate-users mailing-list.

8.2 Miscellaneous

8.2.1 How to get my keyboard properly working?

As the keyboard type is automatically detected during the Ubuntu installation, it is for the moment adapted for the person how build the virtual machine! It could be annoying.

So if you want your keyboard to work properly, proceed as follows:

- Go into the "System" menu, then in "Preferences" and finally in "Keyboard".
- Go in the "Layout" tab and choose the appropriate layout corresponding to your keyboard.

It should work now.

8.2.2 How to get the network working in my virtual machine?

The default settings should just work fine.

There is several ways to get a network connection in the virtual machine. This strictly depends on the characteristics of the network of the host machine (public network, private network, dhcp server policy, dynamic IP, static IP, . . .). So ask your network administrator or yourself if your are the administrator.

Once you get this information, then you can read the Virtual Box documentation concerning the network section, or at least see the proposed solutions in the machine settings menu. As they say: "In most cases, this default setup will work fine for you."!

8.2.3 How to transfer files from the virtual machine to my actual machine?

There are several solutions:

- Configure a shared directory between the host and the guest machine. This is explained in the Virtual Box documentation, so please read this documentation section on Folder Sharing.
- In case of a connection on a network including machines that you own, you can establish a NFS (Network File System) to be able to mount an existing filesystem of another machine in your virtual machine. Again you can read documentation on that by searching for NFS (documentation for Ubuntu).
- If you have an internet connection, you can use FTP access (using FileZilla for example) on an external FTP server on which you have access.
- At least you can send your files via email!

8.2.4 How to minimize the size of the VDI (Virtual Disk Image)?

First you have to force a *fsck* (FileSystem Check) of your guest system to have all data arranged at the beginning of the virtual disk. To do that you have to create an empty file named "forcefsck" at the root level (/), using:

```
sudo touch /forcefsck
```

Then you can reboot the virtual machine and the *fsck* will be forced at the boot time. Depending on the space used in your disk, it can take some time.

Once the machine is rebooted, we have to fill all remaining free space with 0 (zero) value. To do this, just run the following command until there is no free space at all:

```
sudo dd if=/dev/zero of=/dd_zero_file
```

It can take a while because it will create a file with the size of the total free space before you run the command.

Be aware that the size of the VDI of the virtual machine in your host machine will grow too ! (but not necessarily linearly)'

It will grow to the maximum allowed size of the dynamic VDI (default is 20Gbytes).

So check your free space.

Once it is done, just remove the created "dd_zero_file" file and shutdown the virtual machine and also the Virtual Box program. Then in your host system, just open a terminal, go in the directory where your VDI file is, and use the following command to finally compress your VDI file:

```
sudo VBoxManage modifyvdi /absolute/path/to/your/image.vdi compact
```

It will also take a while, but after that, your VDI file will be smaller than initially.

8.2.5 How to update the maximum allowed size of my VDI to a bigger one?

To do that, the trick is to do as if you wanted to add a new physical hard drive disk (HDD) to your computer. Every step will be the same except that instead of adding a real HDD, we will add a virtual HDD.

Here are the steps to have more space into your virtual machine:

- The first step is to create a new virtual disk image (VDI). To do that go in Virtual Box in the "File" menu and click on "Virtual Media Manager". Click on "New" to create the VDI, choose a dynamic disk, give it a name, a size, and click on "Finish".
- Then shutdown your machine if it is running, and go into the "Settings" menu. Go into the "Storage" section and click on the "Add Hard Disk" icon. And add your new VDI that you have just created (automatically done in most cases).
- Now turn your virtual machine on. And open a terminal.
- Type the following command:

  ```
  ls -l /dev/sd*
  ```

  You will see your new device that appears under a name *sdX*, where *X* will be the next letter in alphabetical order after the last disk you inserted in your system. So if it is the first time you do that, your disk will be *sdb*.
- The next step is to create a partition in this new disk. We will use the *fdisk* program. So type the following command:

  ```
  sudo fdisk /dev/sdX (where X is the appropriate letter of your disk)
  ```
- Then in the *fdisk* menu, you can type **m** to get the list of commands. In our case, type **n** to create a new partition, select 'primary partition' as number 1. Then let the default values to get a full partition on the whole disk.
- Once it is done, type **w** to write the partition table. The program *fdisk* will exit on finish.
- Now you have to format your new partition. This partition appears in *dev/* as *sdX1*. To do that, use the following command:

  ```
  sudo mkfs.ext4 /dev/sdX1 (again where X is the appropriate letter of your disk)
  ```
- Your disk is ready for use, you just have to mount it somewhere to use it. For example if we want to have this disk in */mnt/* (usual way to do) with the name *my_new_disk*, proceed as follows:

```
sudo mkdir /mnt/my_new_disk (to create the directory where the disk will be␣
↪mounted)
sudo mount /mnt/sdX1 /mnt/my_new_disk/ (to mount the disk in the directory)
```

- It is done! You can access and use your new disk in */mnt/my_new_disk*. You can type the command *df* to see your new disk is here.
- Also if you want your new disk to be automatically mounted each time you reboot your machine, you have to add an entry in the file */etc/fstab*. **Be careful as this file is very sensitive to mistakes, your system can be hard to repair if you modify existing lines or introduce mistakes in it!**
- But here is the line to add in this file to have an automated mount of your disk:

```
/dev/sdX1   /mnt/my_new_disk  ext4   defaults   0   3
```

- Of course do not forget to replace the *sdX1* by the appropriate name of your partition, and also for *my_new_disk* is you choose to give it another name. *ext4* is the type of the file system used here.
- On next reboot your disk will be automatically mounted.

CHAPTER 9

GATE using Docker

Table of Contents

9.1 GATE 9.0 on docker

A docker image for gate version 9.0 is available here: Click here to download GATE 9.0 on docker

9.2 Example to install GATE with Docker on Amazon Web Services (AWS) (Amazon Linux machine):

Example:

```
# First: create and launch a Linux Virtual Machine on AWS
# Second: register your local machine ssh public key to AWS
# connect with ssh to your new Amazon Server VM on AWS (replace "IPv4" with the␣
↪corresponding address)
ssh ec2-user@ec2-"IPv4".eu-west-3.compute.amazonaws.com
# install docker
sudo yum update -y
sudo yum install docker
sudo service docker start
sudo usermod -a -G docker ec2-user
#logout
```

(continues on next page)

(continued from previous page)

```
exit
# log back in
ssh ec2-user@ec2-"IPv4".eu-west-3.compute.amazonaws.com
docker info
docker run -it opengatecollaboration/gate:8.2
Gate
```

9.3 Example to install GATE with Docker on Amazon Web Services (AWS) (Ubuntu Linux machine):

Example:

```
# First: create and launch a Linux Virtual Machine on AWS
# Second: register your local machine ssh public key to AWS
# connect with ssh to your new Ubuntu Server VM on AWS (replace "IPv4" with the␣
↪corresponding address)
ssh ubuntu@ec2-"IPv4".eu-west-3.compute.amazonaws.com
# install docker
sudo apt update
sudo apt install -y docker.io
# to run docker without sudo
sudo usermod -a -G docker ubuntu # and then log out and back in
# launch a docker container with GATE
docker run -it opengatecollaboration/gate:8.2
Gate
```

CHAPTER 10

Indices and tables

- genindex
- modindex
- search