Users Guide V6

From GATE collaborative documentation wiki

Introduction

General Concept

Getting started
Defining a geometry
Materials
Setting up the physics
Cut and Variance Reduction Techniques
Source and particle management
Voxelized Source and Phantom
How to run Gate
Visualization
Bibliography

Imaging applications

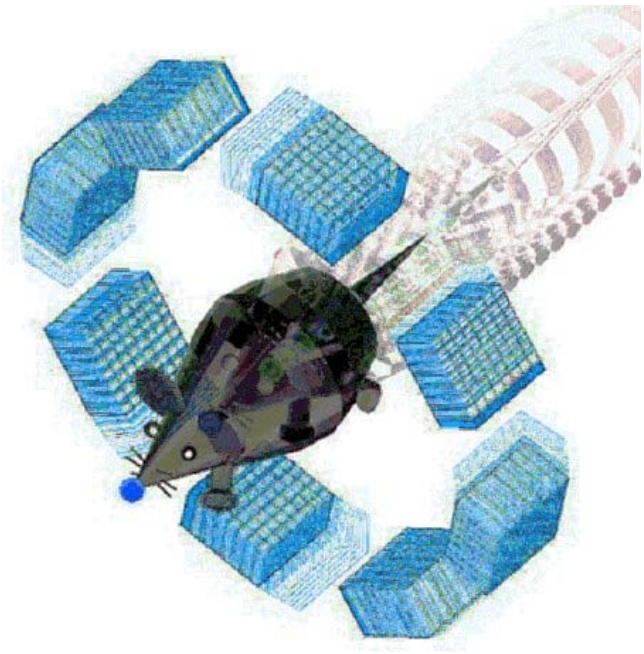
Architecture of the simulation
Defining a system for imaging applications
Sensitive detector concept
Digitizer and detector modeling for imaging applications
Data output management for imaging applications
Generating and tracking optical photons

Radiotherapy applications

Architecture of the simulation
Readout parameters for Radiotherapy applications: Actors
Phase space approach

Users Guide V6:Introduction

From GATE collaborative documentation wiki



Geant4 Application for Emission Tomography: a simulation toolkit for PET and SPECT

OpenGATE Collaboration

http://www.opengatecollaboration.org

Authors

OpenGATE spokesperson: I. Buvat (CNRS-IMNC Paris)

OpenGATE technical coordinator: S. Jan (CEA-SHFJ Orsay)

Editorial board S. Glick (UMASS), S. Kerhoas (CEA Saclay), F. Mayet (CNRS-LPSC Grenoble)

Authors of the first edition: S. Jan, G. Santin, D. Strul, S. Staelens, K. Assié, D. Autret, S. Avner, R. Barbier, M. Bardiès, P. M. Bloomfield, D. Brasse, V. Breton, P. Bruyndonckx, I. Buvat, A. F. Chatziioannou, Y. Choi, Y. H. Chung, C. Comtat, D. Donnarieix, L. Ferrer, S. J. Glick, C. J. Groiselle, D. Guez, P.-F. Honore, S. Kerhoas-Cavata, A. S. Kirov, V. Kohli, M. Koole, M. Krieguer, D. J. van der Laan, F. Lamare, G. Largeron, C. Lartizien, D. Lazaro, M. C. Maas, L. Maigne, F. Mayet, F. Melot, C. Merheb, E. Pennacchio, J. Perez, U. Pietrzyk, F. R. Rannou, M. Rey, D. R. Schaart, C. R. Schmidtlein, L. Simon, T. Y. Song, J.-M. Vieira, D. Visvikis, R. Van de Walle, E. Wieers, C. Morel

Special Thanks: Geant4 Collaboration and LOW energy WG

The GATE mailing list

You are encouraged to participate in the dialog and post your suggestions, questions and answers to colleagues' questions on the gate-users mailing list, the GATE mailing list for users. You can subscribe to the gate-users mailing list, by registering at the GATE web site: http://www.opengatecollaboration.org

If you have a question, it is very likely that it has already been asked and answered, and is now stored in the archives. Please use the search engine to see if your question has already been answered before sending a mail to the GATE-users .

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

Member institutes of the OpenGATE Collaboration (September 2009):

- CNRS Laboratory of Corpuscular Physics @ Clermont-Ferrand (LPC)
- CNRS Centre de Physique des Particules de Marseille (CPPM), Marseille
- CNRS Imaging and Modelling in Neurobiology and Cancerology lab @ Orsay (IMNC), Orsay
- CNRS IRES, Centre Pluridisciplinaire Hubert Curien (CPHC) @ Strasbourg
- CNRS Laboratoire de Physique Subatomique et des technologies associées (SUBATECH), Nantes
- INSERM - CNRS CREATIS lab @ Lyon
- U892 INSERM @ Nantes
- LATIM, U650 INSERM, Brest
- Service Hospitalier Frédéric Joliot (SHFJ), CEA-Orsay
- Healthgrid
- University of Ghent (ELIS)
- Sungkyunkwan University School of Medicine (DMN), Seoul
- Forschungszentrum-Juelich (IME)
- University of California (Crump Institute for Molecular Imaging), Los Angeles
- Memorial Sloan-Kettering Cancer Center (Department of Medical Physics), New York
- University of Athens (IASA)
- Delft University of Technology (IRI)
- John Hopkins University (Division of Medical Imaging Physics), Baltimore
- University of Santiago of Chile (USACH)
- Department of Atomic Physics, Sofia University, Sofia
- Walter Reed Army Medical Center, Washington DC
- University of Pennsylvania School of Medicine

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 [Figure 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.

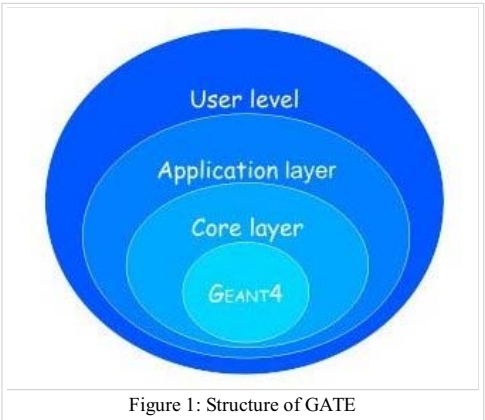


Figure 1: Structure of GATE

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.

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.

Read the full chapters of the Users Guide V6.

Retrieved from "http://wiki.opengatecollaboration.org/index.php/Users_Guide_V6:Introduction"

- This page was last modified on 13 September 2009, at 22:26.

Users Guide V6:Getting started

From GATE collaborative documentation wiki

This paragraph is an overview of the main steps one must go through to perform a simulation using Gate for imaging applications (PET, SPECT and CT). It is presented in the form of a simple example and the user is encouraged to try out the example, in an interactive mode of Gate, 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 a basic understanding of the basic principles of Gate by going through the different steps of a simulation.

General 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
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 **PreInit>** mode

Following the initialization, steps 5-8 are performed in **IDLE>** mode, in which the geometry can no longer be changed. The following paragraph will illustrate these different steps.

General simulation architecture for dosimetry and radiotherapy applications

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
6. define the source(s)
7. start the simulation with the following command lines
8. /gate/application/setTotalNumberOfPrimaries [particle_number]
9. /gate/application/start

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 **PreInit>** 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 (see Users Guide V6:Architecture of the simulation). 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 prompt PreInit:

```
PreInit>/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 paragraphs, the main steps to perform a simulation using Gate are explained. To test this example, the user can run Gate and can execute all the commands proposed in this section, line by line.

First step: Defining a scanner geometry

The first command lines entered at the Gate prompt usually concern the graphical interface. For on-line verification of the geometry being built, a visualization tool needs to be installed, using the following commands.

```
#  V I S U A L I Z 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
```

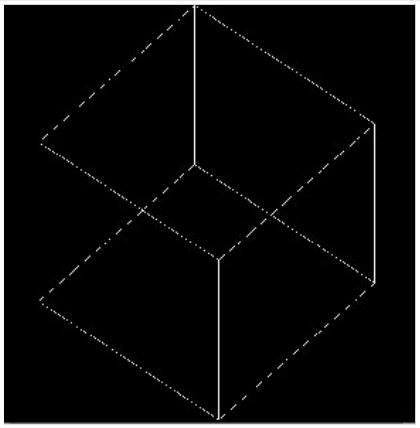


Fig 1.1: World volume.

The different visualization tools and associated commands are discussed in Users Guide V6:Visualization. The visualization being set, 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 base of the tree is represented by the world volume (fig 1.1) which sets the experimental framework of the simulation. All Gate commands related to the construction of the geometry are described in detail in Users Guide V6: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:

```
# 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. Users Guide V6: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 1.2) 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
```

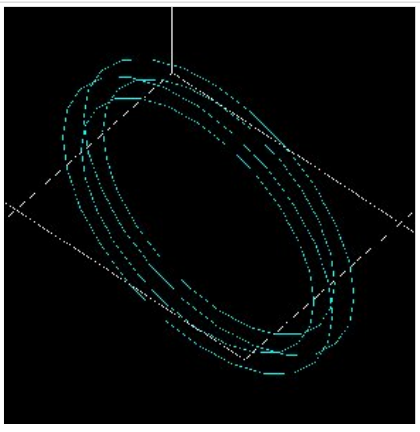


Figure 1.2: Cylindrical scanner

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.

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

```
PreInit> ls /gate/cylindricalPET/vis/
```

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

The following command lines describe this scanner (see Users Guide V6: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
/gate/box1/setMaterial Water
/gate/box1/vis/setColor yellow
/gate/box1/vis/forceWireframe
```

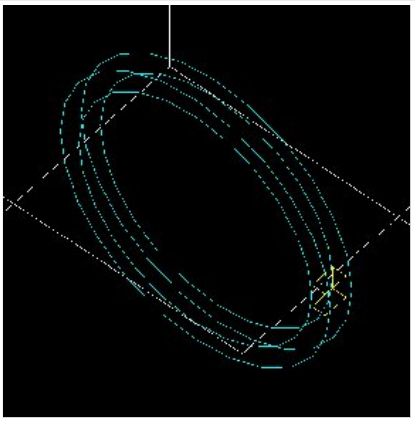


Figure 1.3: first level of the scanner

Once the block is created (figure 1.3), the crystal can be defined as a daughter of the block (figure 1.4)

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)).

To obtain the complete matrix of crystals, the volume box2 needs to be repeated in the Y and Z directions (fig 1.5). To obtain the complete ring detector, the original block is repeated 30 times (fig 1.6).

```
#       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
/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
```

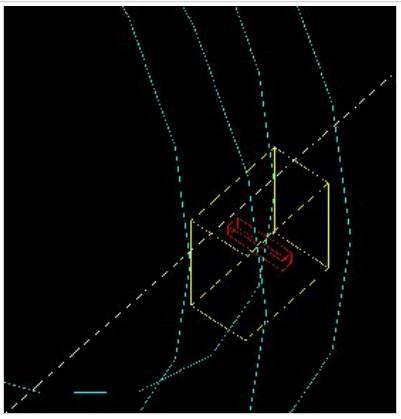


Figure 1.4: crystal, daughter of the block

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 Users Guide V6:Defining a system and Users Guide V6: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 Volumes To a S Y S T E M
/gate/systems/cylindricalPET/rsector/attach box1
/gate/systems/cylindricalPET/module/attach box2 vglue 0.2cm
```

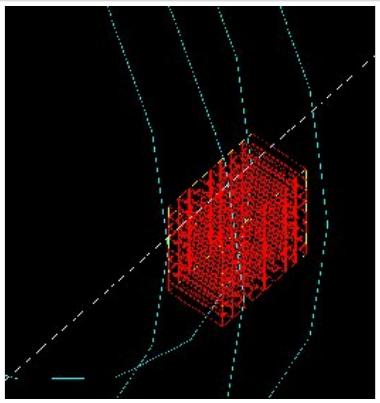


Figure 1.5: matrix of crystals

The names rsector and module are dedicated names and correspond to the first and the second levels of the cylindricalPET system (see Users Guide V6:Defining a system).

In order to save the hits (see Users Guide V6:Digitizer and readout parameters) in the volumes corresponding to the crystals the appropriate command, in this example, is:

```
# Define a S E N S I T I V E Detector
/gate/box2/attachCrystalSD vglue 1cm
```

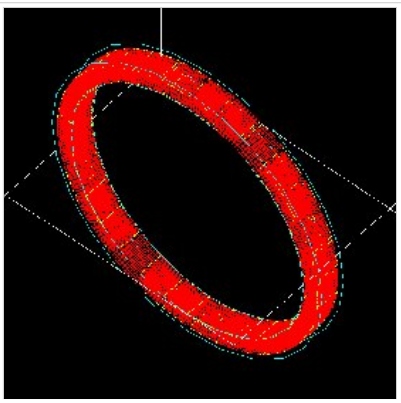


Figure 1.6: complete ring of 30 block detectors

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. Users Guide V6:Defining a geometry describes the movement of volumes in detail.

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
```

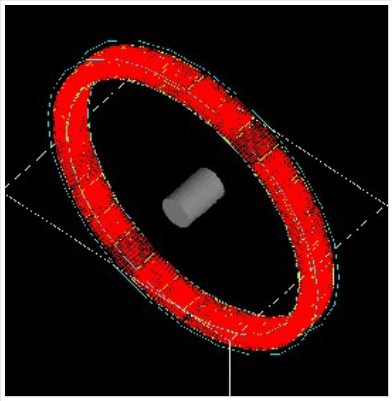


Figure 1.7: cylindrical phantom

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 defined as 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 Users Guide V6:Attaching the sensitive detectors.

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 Users Guide V6:Setting up the physics.

Fourth step: Initialization

When the 3 steps described before are completed, corresponding to the *PreInit* 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 teh 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.

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:

```
# ENERGY BLURRING
/gate/digitizer/Singles/insert blurring
/gate/digitizer/Singles/blurring/setResolution 0.19
/gate/digitizer/Singles/blurring/setEnergyOfReference 511. keV

# ENERGY WINDOW
/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 Users Guide V6:Digitizer and readout parameters.

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 Users Guide V6:Activity, source, voxellized phantoms.

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= examples Sphere or Cylinder
/gate/source/twogamma/gps/shape Cylinder
/gate/source/twogamma/gps/radius 0.5 mm
/gate/source/twogamma/gps/halfz 25 mm

# Confinement option = Only emission points inside the
# confinement volume are accepted
# here NULL means no confinement
/gate/source/twogamma/gps/confine NULL
/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
```

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. Users Guide V6: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 Users Guide V6:Data output .

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 of the simulation are not radioactive source or is they are not defined activity. 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 event 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]
```

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
# 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:

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

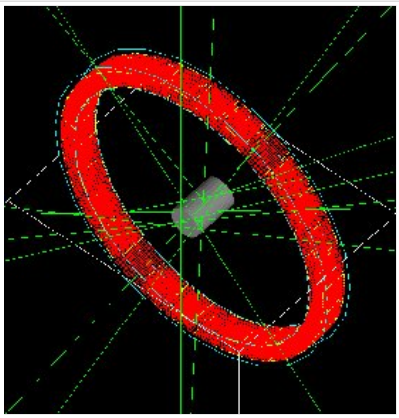


Figure 1.8: Simulation is started

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**.

Slices with variable time

In this approach, each slice has a specific duration. User has to defined 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
```

NOTE: for dosimetry and radiotherapy applications, the general macro set-up should be quiet different - A focus on dedicated example folder is recommanded

Users Guide V6:Defining a geometry

From GATE collaborative documentation wiki

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.

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 centred 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 figure 3.1. 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
```

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

GATE object:        'world'
Is the whole world?         Yes
Number of physical volumes: 1
Total volume:           125000 cm3

        GATE object:        'world'
        Shape:          box
        Length along X: 50 cm
        Length along Y: 50 cm
        Length along Z: 50 cm
        Material:               'Air'
        Total volume:           125000 cm3
        Visibility attributes: visible, Colour=(1,1,1,1), wireframe
Nb of children:         0
```

Figure 3.1: 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.

Creating a volume

Generality - Tree creation

When a volume is created with GATE, it automatically appears in the GATE tree (see Users Guide V6:Getting started). 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 3.1) and shall be referred to using the corresponding abbreviation. The list of units available in GATE can be edited using:

```
/units/list
```

inside the GATE environment (see Users Guide V6:Getting started).

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:

```
/gate/world/daughters/insert 3axes
```

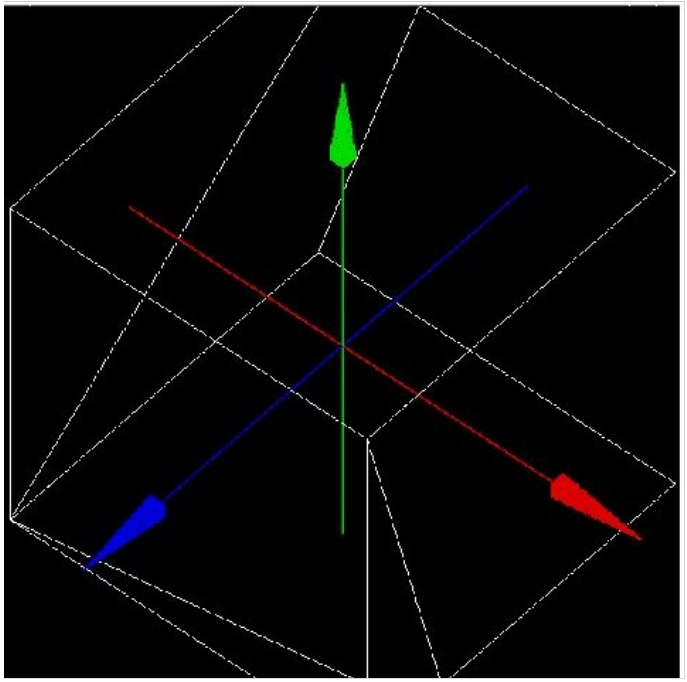


Figure 3.2:Three-axis system defined in GATE. The red, green and blue axes are the X, Y and Z axes respectively

To display an axis with respect to the volume *Name_Volume*, the command

```
/gate/Name_Volume/daughters/insert axe*
```

can be used, where * can be X, Y or Z

The X, Y and Z axes are defined as volumes but they do not meet a fundamental rule of the Geant4 volumes, namely that a children volume must be included in its mother volume. These volumes should therefore not be used during the simulation as this would induce wrong particle transport.

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 `` kilometer3 km3	steradian sr	
centimeter cm	centimeter2 cm2	centimeter3 cm3	degre deg
millimeter mm	millimeter2 mm2	millimeter3 mm3	
micrometer mum			
nanometer nm			
angstrom Ang			
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	megaelectronvolt MeV	
nanosecond ns	meter/min m/min	radian/min rad/min	gigaelectronvolt GeV
picosecond ps	centimeter/min cm/min	degree/s deg/s	teraelectronvolt TeV
	millimeter/min m/min	rotation/s rot/s	petaelectronvolt PeV
	meter/h m/h	degree/min deg/min	joule j
	centimer/h cm/h	degree/min deg/min	
	millimeter/h mm/h	rotation/min rot/min	
	rotation/h rot/h	radian/h rad/h	
		degree/h deg/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 FLUX DENSITY**
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 Users Guide V6:Defining a system) currently defined in GATE: *scanner*, *PETscanner*, *cylindricalPET*, *SPECTHead*, *ecat* and *CPET.*

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

```
/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, ellipsoid, hexagon, polygon, extruded trapezoid and parallelepiped.**

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/daughter_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 (i.e., box for box, sphere for sphere, cylinder for cylinder, cone for cone, ellipso for a tube with an elliptical base, hexagone for hexagon, polycone for polygon, trpd for extruded trapezoid or parallelepiped for parallelepiped). 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 3.2

These commands can be found in the directory

```
/gate/Volume_Name/geometry
```

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 position
setYLength: Set the length of the box along the Y axis	setY1Length: Set half length along Y of the plane at -dz position
setZLength: Set the length of the box along the Z axis	setX2Length: Set half length along X of the plane at +dz position
SPHERE	setY2Length: Set half length along Y of the plane at +dz position
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 box
setPhiStart: Set the start phi angle	setYBoxLength: Set half length along Y of the extruded box
setDeltaPhi: Set the phi angular span (2PI for full sphere)	setZBoxLength: Set half length along Z of the extruded box
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**
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**
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
ELLIPSO	setHeight: Set the height of the hexagon
setLong: Set the long axis length of the ellipse	**WEDGE**
setShort: Set the short axis length of the ellipse	NarrowerXLength: Set the length of the shorter side of the wedge in the X direction.
setHeight: Set the height of the ellipse	XLength: Set the length of the wedge in the X direction
	YLength: Set the length of the wedge in the Y direction
	ZLength: Set the length of the wedge in the Z direction

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 Air. The list of available materials is defined in the GateMaterials.db file. (see Users Guide V6: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 3.3)

Table 3.3: 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

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

A volume named crystal is created as the daughter of a volume the shape of which is defined as a box.

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

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

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

The new volume crystal is filled with NaI.

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

The new volume crystal is colored in yellow.

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

The previous command lists the parameters associated with the crystal volume.

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

The crystal volume is disabled

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 Figure 3.3 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
/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
```

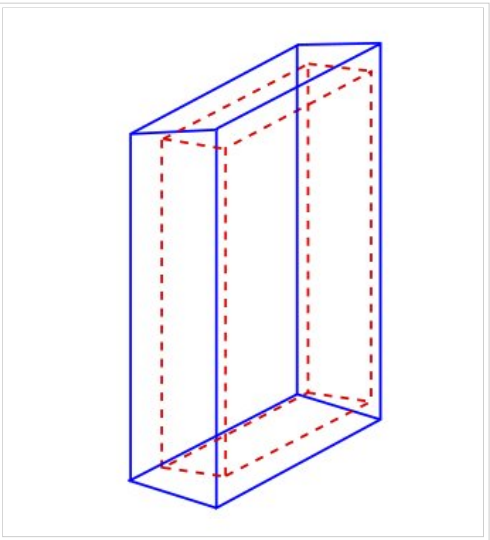


Figure 3.3 Side view of an extruded trapezoid based on a boolean solid. The contours in blue and dashed red represent the contours of the trapezoid and the box respectively

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 Figure 3.4, 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.

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
```

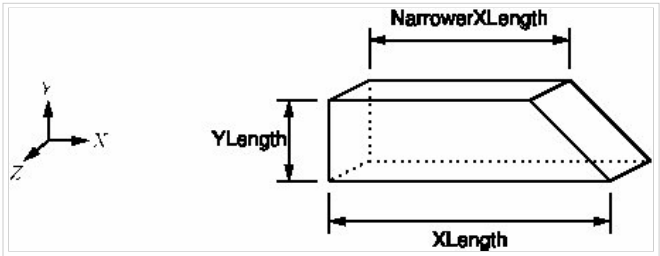


Figure 3.4: When a wedge is inserted, it is oriented as shown in this figure

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 figure 3.5.

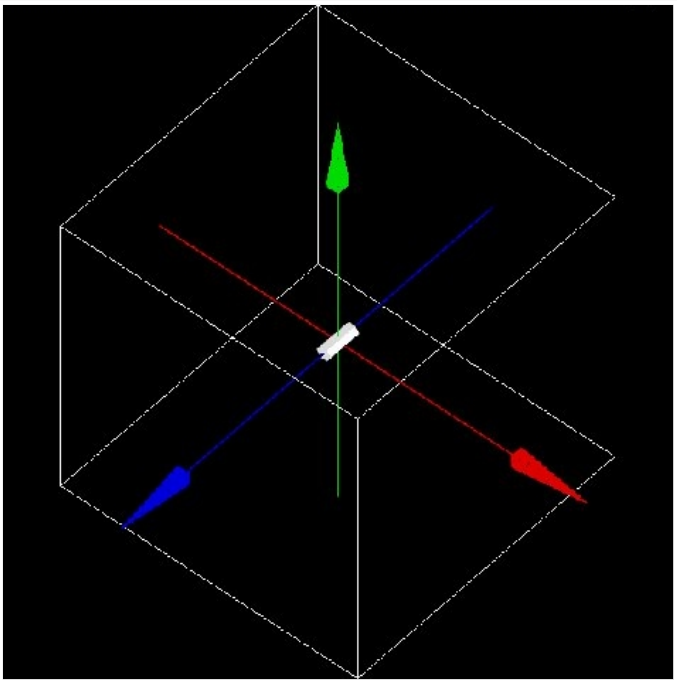


Figure 3.5: Illustration of the application of the linear repeater

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:

```
/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
```

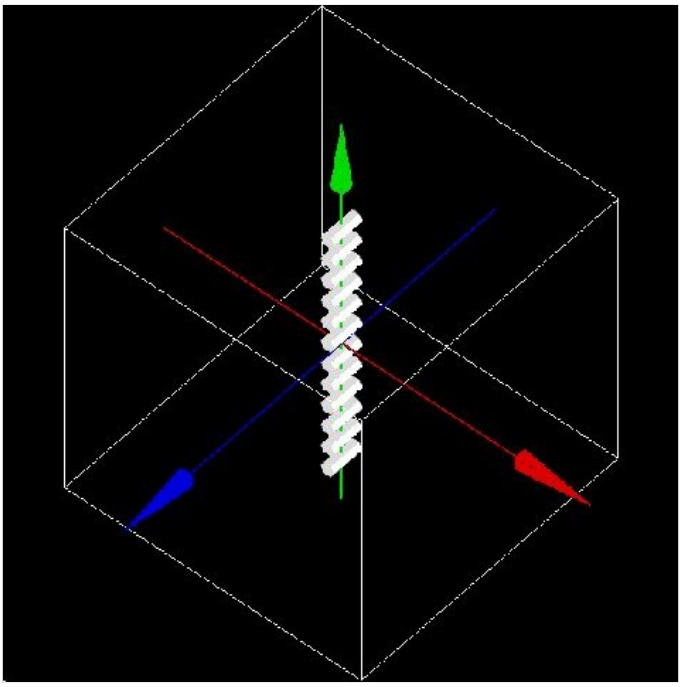


Figure 3.5: Illustration of the application of the linear repeater

When the volume auto-rotation option is enabled, the volume itself is rotated so that its axis remains tangential to the ring (see Figure 3.6). If this option is disabled, all repeated volumes keep the same orientation (see Figure 3.7). 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 figure 3.8.

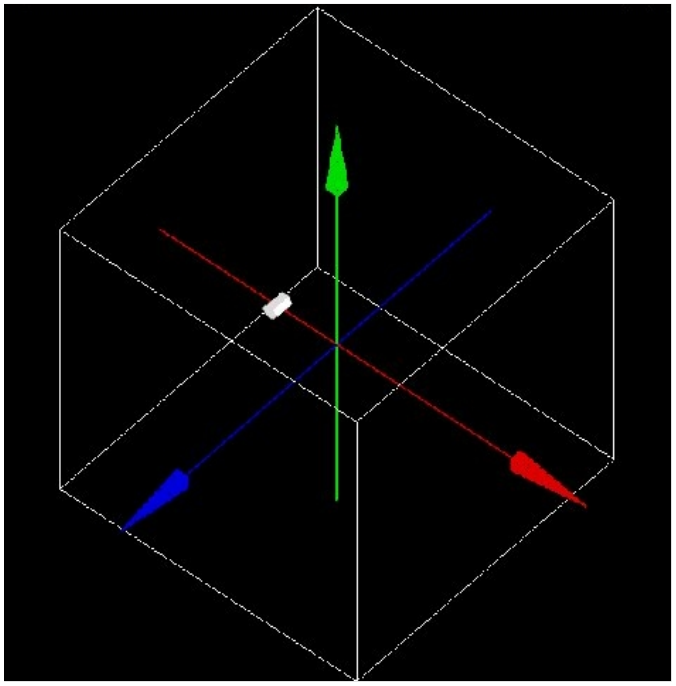


Figure 3.8: Illustration of the application of the ring repeater

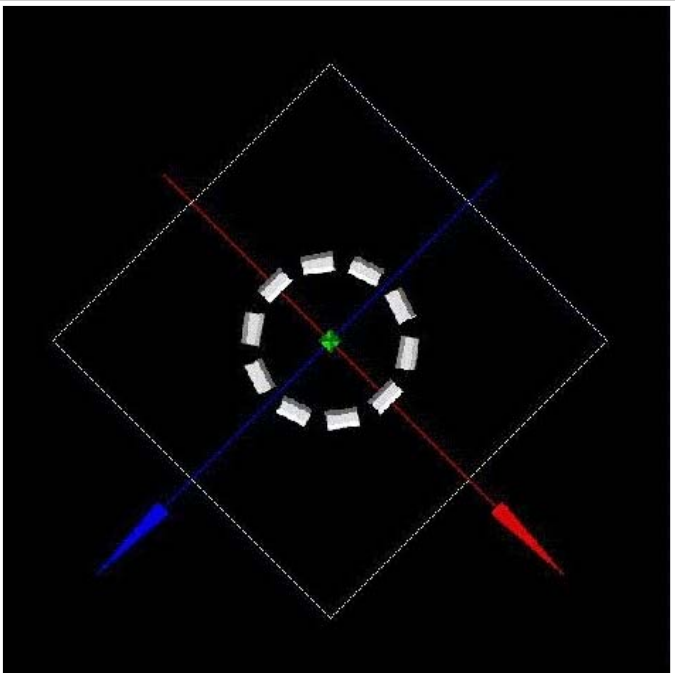


Figure 3.6: Illustration of the application of the auto-rotation option

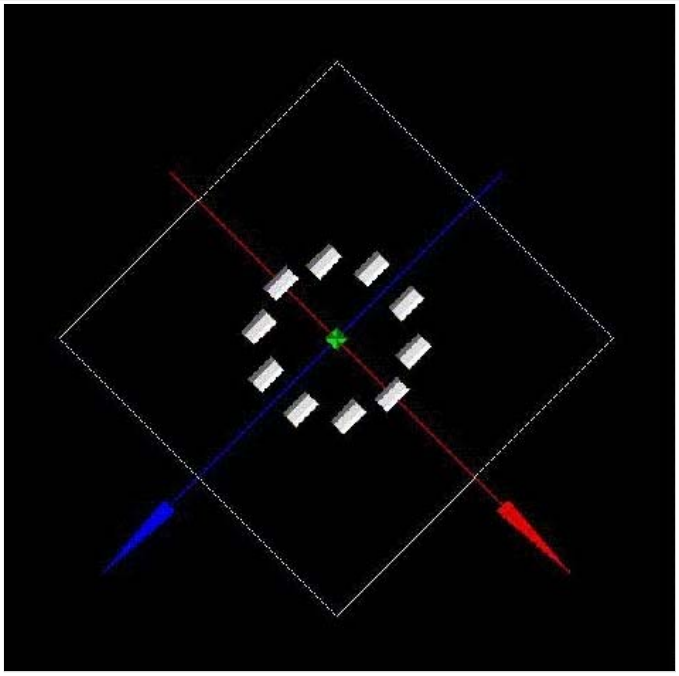


Figure 3.7: Illustration of the application of the ring-repeater when the auto-rotation option is disabled

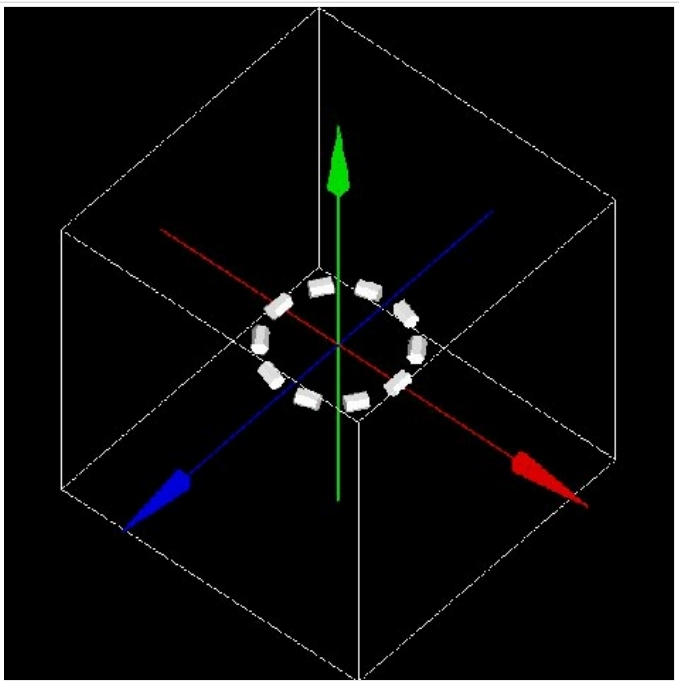


Figure 3.8b: Illustration of the application of the ring repeater

- 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 figure 3.9.

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:

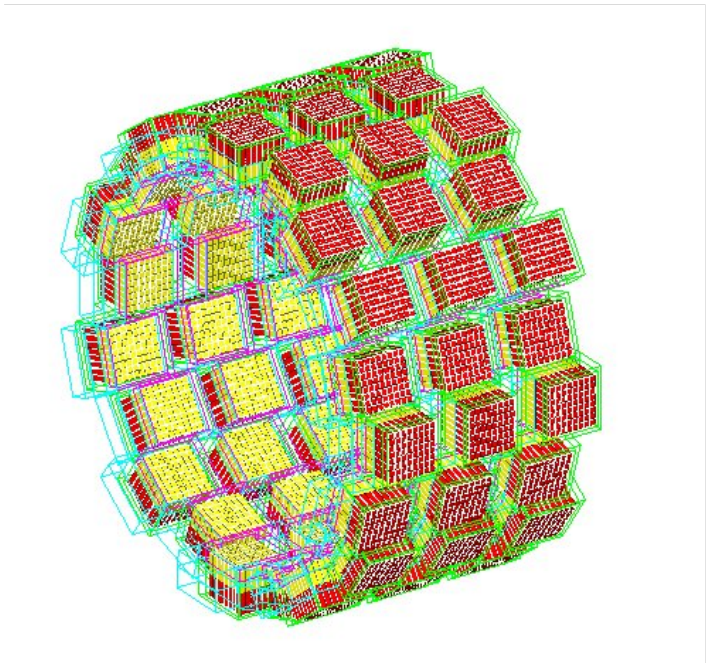


Figure 3.9: Example of a ring repeater with a shift. An array of 3 crystal matrices has been repeated 20 times with a modulo N=2 shift

```
/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 3.10.

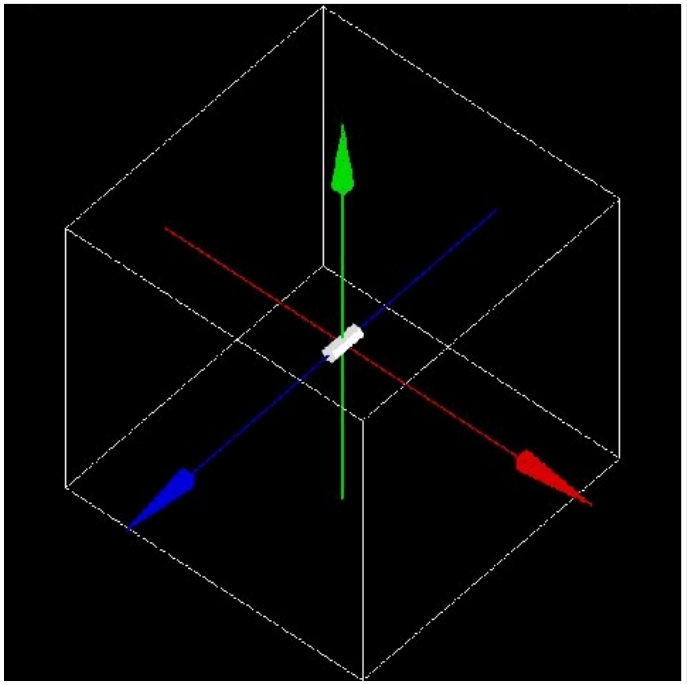


Figure 3.10: Illustration of the application of the cubic array repeater

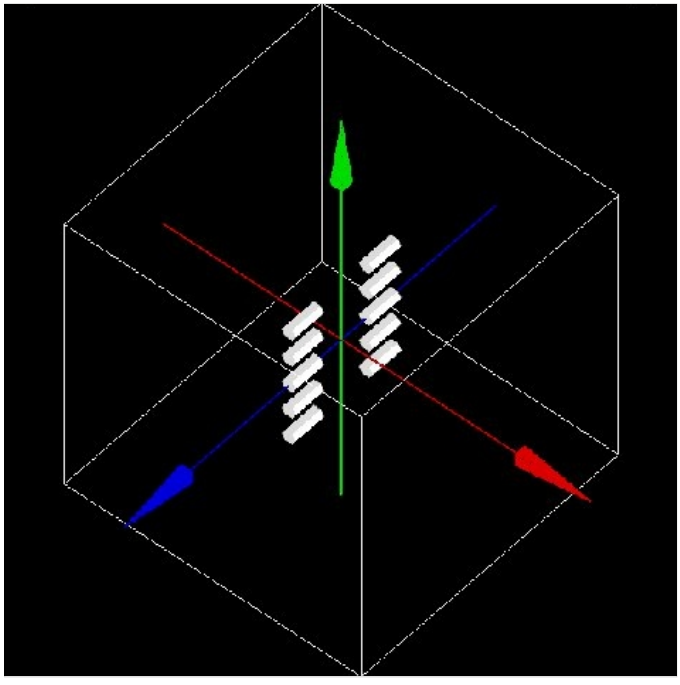


Figure 3.10B: Illustration of the application of the cubic array repeater (after)

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:

```
/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 figure 3.5.

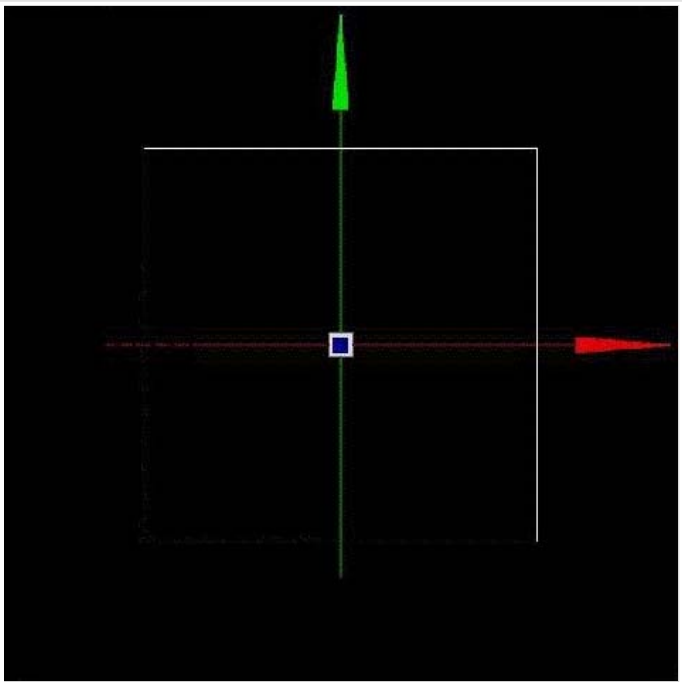


Figure 3.10: Illustration of the application of the cubic array repeater

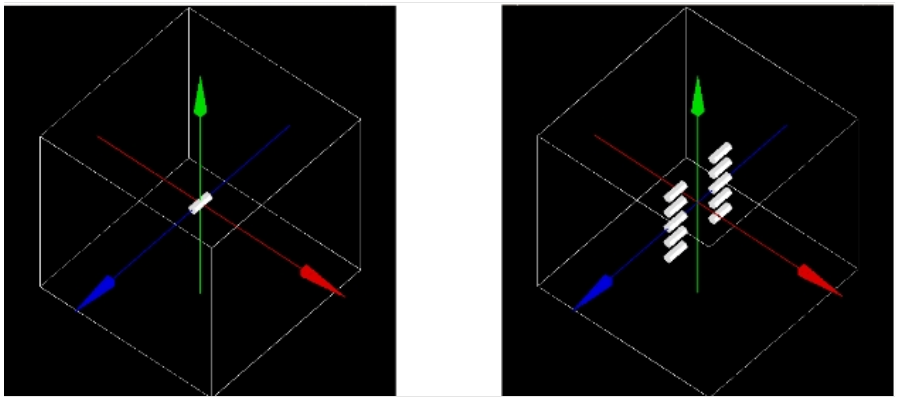


Figure 3.10b: Illustration of the application of the cubic array repeater (after)

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:

```
/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
```

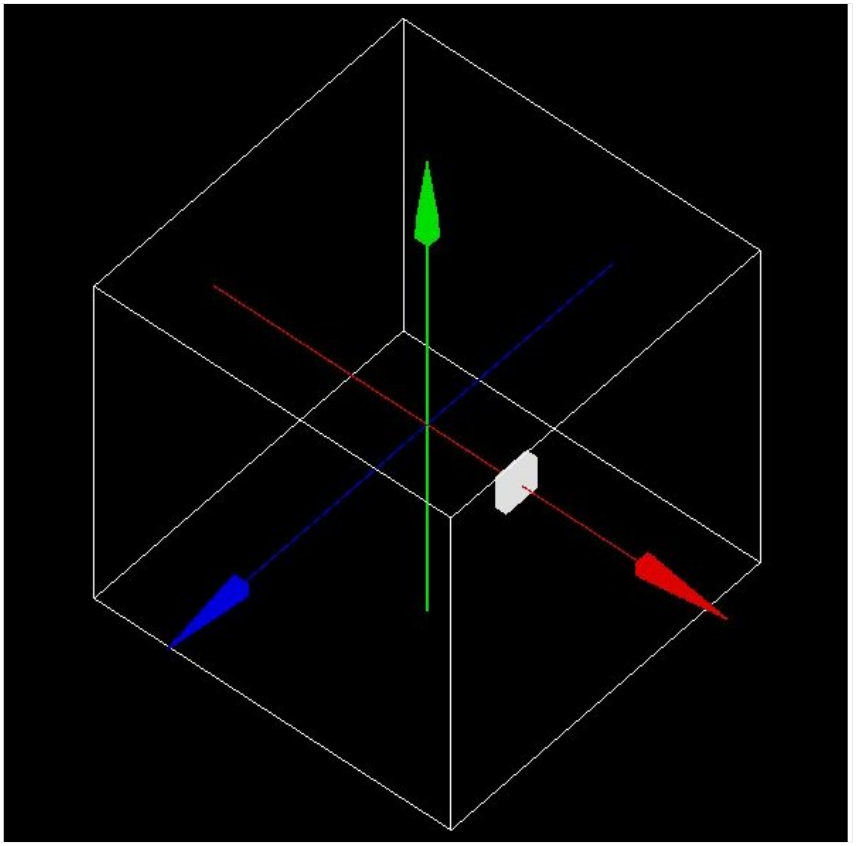


Figure 3.12: Illustration of the application of the sphere repeater

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

Example 3.12

```
/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 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:

```
###### 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
Rotation deg
Translation mm
0       0 1 0       0 0 10
10      0 1 0       0 0 10
15      0 1 0       0 0 20
```

- line with # are ignored
- first word must be Rotation or Translation followed with the unity (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 here

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 figure 3.13.

Rotation

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

```
/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 figure 3.14.

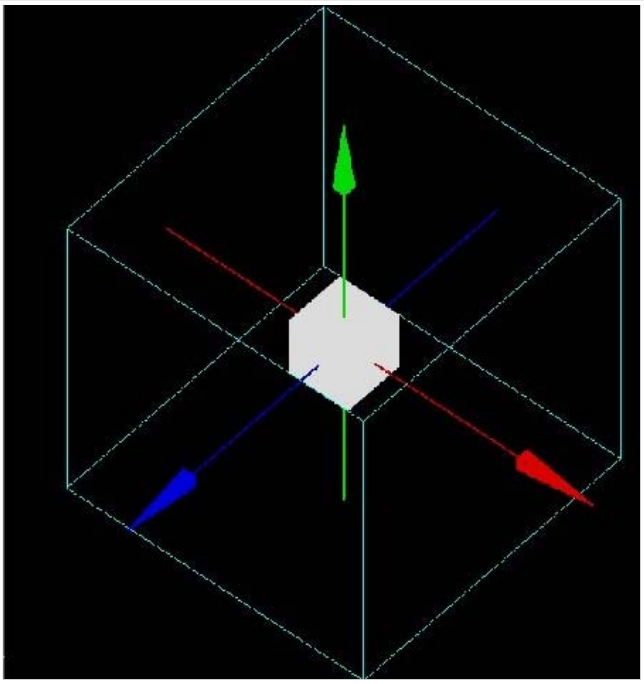


Figure 3.13: Illustration of the translation placement

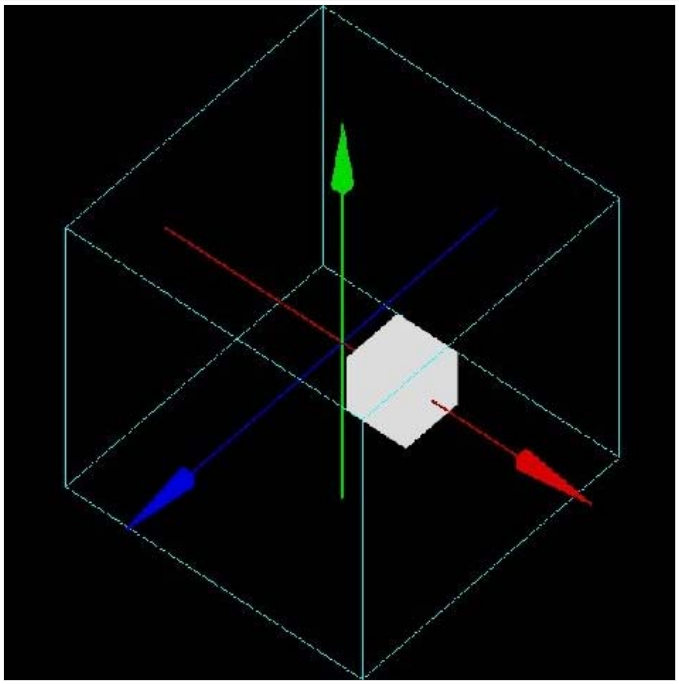


Illustration of the translation placement

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.

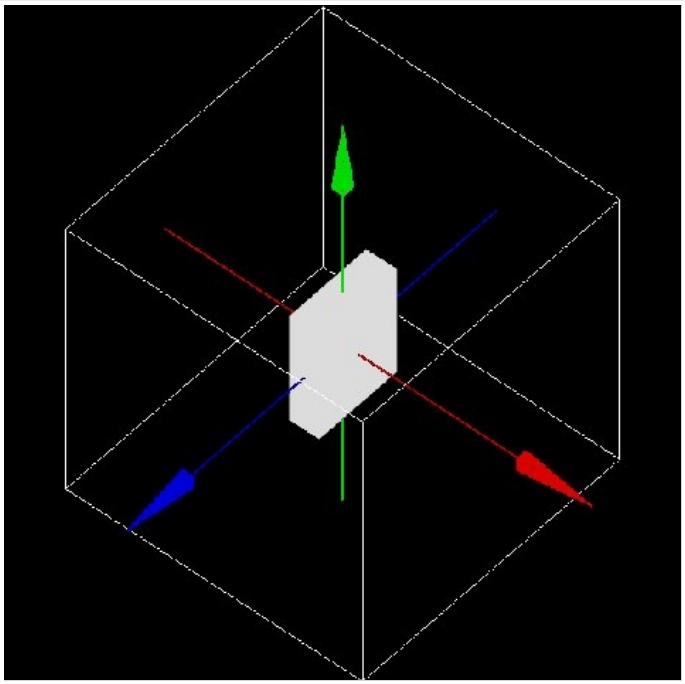


Figure 3.14: Illustration of the rotation placement

Special example: Wedge volume and OPET scanner

The wedge is always created as shown in figure 3.4, 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 Figure 3.15.

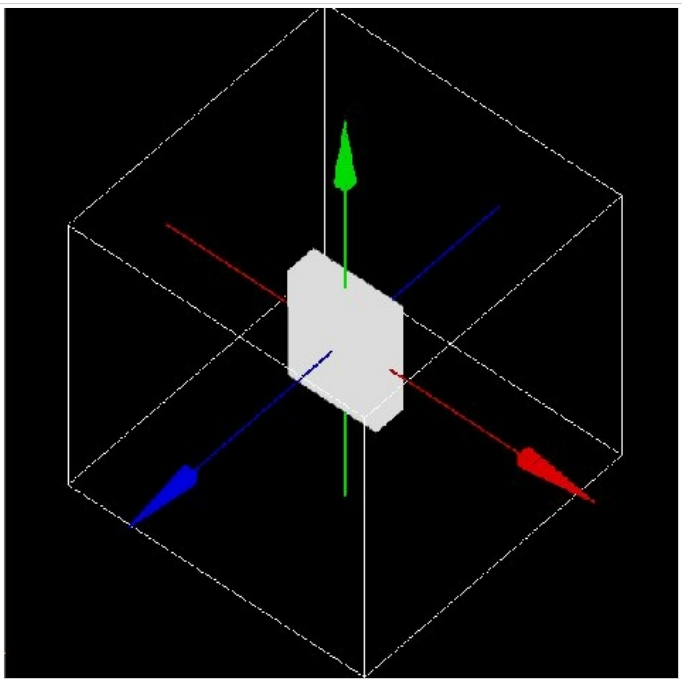


Figure 3.14: Illustration of the rotation placement

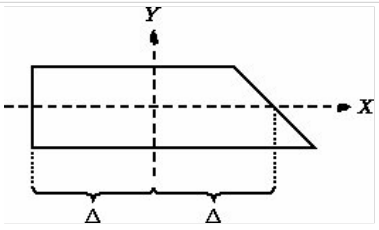


Figure 3.15: Center of wedge

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

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 Figure 3.16.

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
```

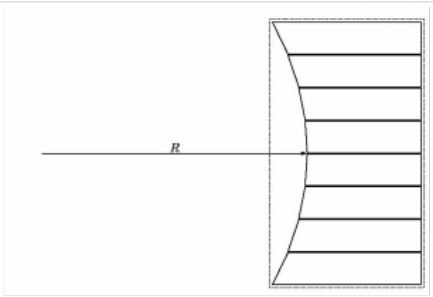


Figure 3.16: A block approximating a true circular geometry

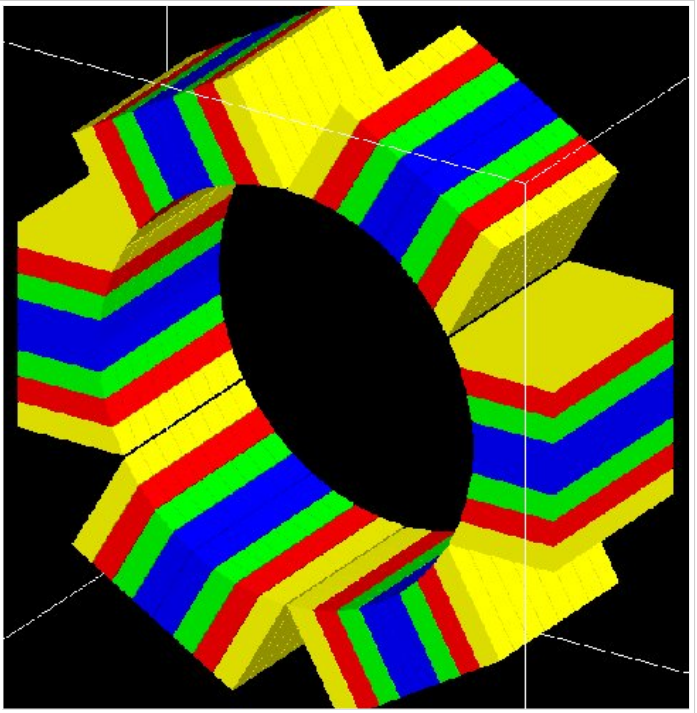


Figure 3.17: The OPET scanner

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
/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.

Figure 4.9 shows the final OPET scanner.

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 GenericRepeater,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 this section for example). At each new run, GATE look into the time-placements list and choose the one that correspond 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 here

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-leafs 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
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.

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 either manually or automatically.

Manual update mode

When one or more modifications are applied to the geometry, the following command line must be used to take them into account and to refresh the display:

```
/gate/geometry/update
```

Auto-update mode

When one or more modifications are applied to the geometry, they will automatically be taken into account and the display will be refreshed using:

```
/gate/geometry/enableAutoUpdate
```


- This page was last modified on 11 January 2010, at 16:12.

Users Guide V6:Materials

From GATE collaborative documentation wiki

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 called 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.

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
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.

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
```


- This page was last modified on 18 March 2010, at 20:44.

Users Guide V6:Setting up the physics

From GATE collaborative documentation wiki

Setting up the physics

The description of all of the physics processes described here are extracted from the Geant4 Physical User's Guide. The collection of processes and options selected and initialized by user is named the physics list.

Currently, the maximum energy used in a medical application is 5 GeV per particle (carbon treatment). Thus, simulation of medical applications requires electromagnetic and hadronic processes below 10 GeV. Each process have a model which generates the final state and a cross-section which represents the probability of interaction (the cross-section is sometimes called dataset).

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)**

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[1].

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/[Process Name]/modelList [Particle]
/gate/physics/[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/[Process Name]/setModel   [Model Name] [Particle]
/gate/physics/[Process Name]/unSetModel [Model Name] [Particle]
/gate/physics/[Process Name]/setDataSet   [DataSet Name] [Particle]
/gate/physics/[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/[Process Name]/[Model Name]/setEmax [Value] [Unit] [Particle] [Option]
/gate/physics/[Process Name]/[Model Name]/setEmin [Value] [Unit] [Particle] [Option]

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

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/IonInelastic/G4BinaryLightIonReaction/setEmin 100 keV
/gate/physics/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
```

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:

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

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/cross-sections are available for electromagnetic processes:

- Standard processes are effective between 1 keV and 100 TeV (Std)
- Low energy processes are effective between 250 eV and 100 GeV (LowE)
- 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.

< ! > User should not used several electromagnetic models in the same physics list (exept for hadron ionisation process and for particles which have only one model).

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
```

or

```
/gate/physics/addProcess LowEnergyPhotoElectric gamma
```

or

```
/gate/physics/addProcess PenelopePhotoElectric gamma
```

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

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

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
```

or

```
/gate/physics/addProcess LowEnergyCompton gamma
```

or

```
/gate/physics/addProcess PenelopeCompton 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
```

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 LowEnergyRayleighScattering gamma
```

or

```
/gate/physics/addProcess PenelopeRayleighScattering gamma
```

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

```
/gate/physics/addProcess PenelopeRayleighScattering
/gate/physics/addProcess PenelopeRayleighScattering Default
/gate/physics/addProcess PenelopeRayleighScattering gamma
```

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
```

or

```
/gate/physics/addProcess LowEnergyGammaConversion
```

or

```
/gate/physics/addProcess PenelopeGammaConversion
```

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

```
/gate/physics/addProcess LowEnergyGammaConversion
/gate/physics/addProcess LowEnergyGammaConversion Default
/gate/physics/addProcess LowEnergyGammaConversion gamma
```

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).

With standard model, 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 low energy and Penelope models have each one a process for electron and positron ionization:

```
/gate/physics/addProcess LowEnergyElectronIonisation e-
```

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

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 have 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 most 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-
```

or

```
/gate/physics/addProcess LowEnergyBremsstrahlung e-
```

or

```
/gate/physics/addProcess PenelopeBremsstrahlung e-
/gate/physics/addProcess PenelopeBremsstrahlung 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 PositronAnnihilationStd e+
```

or

```
/gate/physics/addProcess PenelopePositronAnnihilation e+
```

An improvement of the positron-electron annihilation has been developed to take into account the γγ non-colinearity . The mean value of the angle distribution is $\simeq 0.5^0$.

```
/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. Three 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 for example):

```
/gate/physics/addProcess MultipleScattering e+
/gate/physics/addProcess MultipleScattering e-
/gate/physics/addProcess MultipleScattering proton
/gate/physics/addProcess MultipleScattering alpha
/gate/physics/addProcess MultipleScattering GenericIon
...
```

For all charged particles included in the default particle list:

```
/gate/physics/addProcess MultipleScattering Default
```

The two other processes:

```
/gate/physics/addProcess eMultipleScattering ...
/gate/physics/addProcess hMultipleScattering ...
```

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

< ! > In Geant4 version 9.2, options are available only for standard processes

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] = Δrange/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 UHadronElastic
/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
- LeadingParticleBias (G4Mars5GeV)
- 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, β + , 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 RadioactiveDeacy
```

Users Guide V6:Cut and Variance Reduction Technics

From GATE collaborative documentation wiki

Production threshold, step limiter and special cuts

Production threshold

To avoid infrared divergence, some electromagnetic processes require a threshold below which no secondary 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.

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.

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.

Users Guide V6:Source

From GATE collaborative documentation wiki

Source

The GPS (General Particle Source)

To introduce a source into a GATE simulation, the user has to define the type of source (voxellized, linacBeam, phaseSpace, or GPS) ans 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.

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, the users 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

Many options are available to suit your specific applications:
- Using keywords (e.g. Fluor18, backtoback, linacBeam, phaseSpace, fastI124) to help you to define one or several properties.

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 radio-nuclide; 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
```

In the above example, an ion source of oxygen-15 has been defined with Z=8, A=15, Q=0, E=0 . If 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).

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 concurently 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 explicitely 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 "User", arbitrary point-wise spectrum "Arb", and user-defined energy per nucleon histogram "Epn". Capitalisation 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

```
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
/gate/source/NAME/setForcedHalfLife 6586 s positron for fluorine-18
/gate/source/NAME/gps/particle e+
/gate/source/NAME/gps/energytype Fluor18
/gate/source/NAME/setForcedUnstableFlag true
/gate/source/NAME/setForcedHalfLife 6586 s 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 fastI124
/gate/source/NAME/setType fastI124
/gate/source/NAME/setForcedUnstableFlag true
/gate/source/NAME/setForcedHalfLife 360806 s
```

Table I. Radiation Source Properties

Property	Default values	Ion	particle	backtoback	fastI124
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)*

Figure 7.1: Properties of radioactive sources

Defining the angular distribution of the emission

An emission angle distribution can be defined with the angular span using:

```
/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.
- 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 could be attach to a volume:

```
/gate/source/[Source name]/attachTo   [Volume Name]
```

If the volume move during the simulation, the source takes into account the movement.

Confining a source

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_P
```

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 + '_P'.

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
/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_P
```

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
/gate/source/number1/dump 1
/gate/source/list
```

Visualizing a source

To check that sources are at the right location in the geometry, you can use the following command

```
/gate/source/?/visualize
```

along with a real time viewer (e.g. OpenGL). To visualize a source, Gate will pick up random by a certain number of points within the source and display them on the screen, along with the geometry. The full syntax is:

```
/gate/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 less 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]
```

Phase space source

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, the source calculate the number of event requested. If the event number is higher than the number of particles in file, the source used each particle several times. If particles are re-used, it is possible to rotate the particle position and direction around the z axis of the volume. 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
```

Examples are available here (http://wikireview.opengatecollaboration.org/index.php/GateRT)

Users Guide V6:Voxelized Source and Phantom

From GATE collaborative documentation wiki

Voxelized Source and Phantom

Introduction

GEANT4 offers two convenient methods to describe multiple copies of a volume inside a mother volume: replicas and parameterized volumes. The parameterized volumes method offers several advantages over replicas such as: voxels that can be varied in size, shape and material, voxels not entirely filling the envelope, and visualization attributes.

The parameterized volume method is available in GATE for voxellized phantoms and is preferred over replicas. The current implementation of voxellized phantoms using GEANT4 parameterization supports phantom dose calculations and voxel visualization attributes on a per material basis.

More developments are planned for the future that include the use of a cylindrical envelope (for tight geometries) and a variable voxel size scheme to minimize memory requirements and speed up simulations.

How to use parameterized voxels

To create a parameterized phantom object, use the keyword parameterizedBoxMatrix in the insert command:

```
/gate/world/daughters/name anyname
/gate/world/daughters/insert parameterizedBoxMatrix
```

The parameterizedBoxMatrix supports the imageReader and the interfileReader as well as the range or tabulated translator. As with other objects, placement and moves commands can be specified.

Color attributes

Color attributes may be specified with the tabulated or the range translator. Tables read by translators have a prescribed format. The first line must contain the number of materials (i.e. the number of subsequent lines). The following lines contain either a material number (or range) followed by a material name (for the old format) or, for the new format, a material number (or range), a material name, a visibility boolean (true or false) and color attribute values (red, green, blue, alpha). If the old format is used, the phantom will be of a uniform dark red color.

Example of a tabulated translation table in the new format (range.dat in the example):

```
6
1 Air true 1.0 0.0 0.0 1.0
2 Water true 0.0 1.0 0.0 0.2
3 LSO true 0.0 0.0 1.0 1.0
4 GSO true 1.0 1.0 1.0 0.2
5 Tungsten true 1.0 1.0 0.0 0.2
6 Lucite true 1.0 0.5 0.0 0.2
```

Dose collection

To collect 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 dose in cGy. It has the same dimensions as the phantom. The output module optionally writes a second binary file containing the uncertainty on 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 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
```

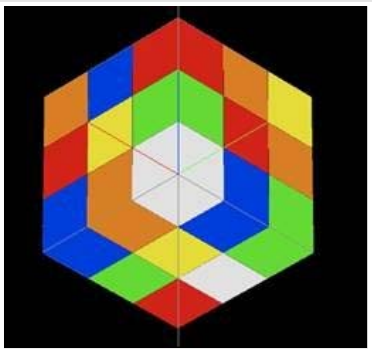


Figure 7.2: A simple voxellized phantom without transparency

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:

$s^2 = E(d^2) - E(d)^2$

$$s^2 = \frac{1}{N^2}[N\sum d_i^2 - (\sum d_i)^2]$$

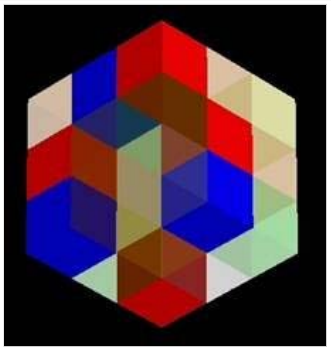


Figure 7.2b: A simple voxellized phantom with transparency

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:

$$\hat{d} = \frac{s}{N} = \frac{s}{\sqrt{N-1}}$$

Compressed phantoms (variable-size voxels)

As phantom resolution increases, the number of voxels can become very large and a significant amount of memory may be required. For most applications however, high resolution is not required everywhere in the phantom but only where necessary to keep smooth boundaries between phantom structures. The compressedMatrix phantom object can be used instead of the parameterizedBoxMatrix to generate a compressed phantom where voxel size is variable. With the compression algorithm, all adjacent voxels of the same material are fused together to form the largest possible rectangular voxel. A compressed phantom uses less memory and also less cpu. To define a variable-size voxellized phantom use the following commands:

```
/gate/world/daughters/name anyname
/gate/world/daughters/insert compressedMatrix
```

It is possible to exclude regions in the phantom from being compressed through the use of an "exclude list" of materials. For example, the following command line:

```
/gate/anyname/compression/excludeList Lung Breast
```

would exclude all regions in the phantom containing Lung or Breast materials from being compressed. The exclusion list, if any, must be placed before the /readFile command of the appropriate reader.

For dosimetry applications, the output dose matrix has the same dimensions as the original uncompressed phantom. Because the number and size of voxels in each region are different in the compressed and uncompressed versions, dose histograms are likely to be different. If precise histograms are needed, then regions of interest should be left uncompressed through the exclusion list mechanism. However, average dose values for a given region should be similar whether it has been compressed or not.

Regular parameterization for voxelized phantoms

To save time when simulating voxelized phantoms, new navigation algorithms are often developed. Here the new one is called the Regular Navigator (developed in Geant4.9.1) and has been implemented in GATE. To optimise the particles transportation in voxelized phantoms including different materials, this navigator includes new features :

Older navigators such as the navigator coming with the parameterizedBoxMatrix determine the next voxel that the particle will enter by searching in the whole list of the voxels contained in the phantom. Now the regular navigator uses a new algorithm that performs this search only for the neighbors of the actual voxel. It therefore highly reduces the time spent on this search, as much as the number of voxels is large. Again, older navigators were doing a new step at each voxel boundary encountered. But now the regular navigator 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.

To summarize: 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 parameterizedBoxMatrix or compressedMatrix. To use this navigator, just replace the name parameterizedBoxMatrix or compressedMatrix by regularMatrix. The associated commands are as follows:

```
/gate/world/daughters/name anyname
/gate/world/daughters/insert regularMatrix
```

Everything else remains the same as for parameterizedBoxMatrix or compressedMatrix (reader, translator, dose collection, attachment).

Speeding-up tracking: Fictitious interaction

In a voxelized phantom fictitious interaction tracking can be activated. This tracking algorithm can be more efficient than the Geant4 tracking, especially if the voxel size is small and if the most attenuating material in the voxelized phantom is not too dense (i.e. bone, but not metal). When this tracking is activated, photons above a certain minimum fictitious energy threshold are not tracked by Geant4, but by a dedicated tracking algorithm. At the border of the voxelized phantom, these photons re-enter Geant4 tracking, and the tracking in the detectors is performed in the standard manner. All other particles in the voxelized phantom (photons below minimum fictitious energy, electrons) are tracked using the regular navigator.

Fictitious interaction tracking requires three obligatory commands. Three other commands are optional. After naming the voxelized phantom with the command (first obligatory command)

```
/gate/world/daughters/name NAME
```

the second obligatory command is

```
/gate/world/daughters/insert fictitiousVoxelMap
```

This command assigns an additional map to the voxelized phantom that handles the fictitious interaction tracking. The underlying voxelized regular phantom is automatically activated for the other particles (photons below the minimum fictitious energy, electrons). The third obligatory should be given in the physics part of the mac file:

```
/gate/physics/addProcess Fictitious
```

This command create a process named Fictitious that will handle gamma particles and all processes related to gamma particles, i.e. photoelectric effect, compton and rayleigh scattering, and gammaconversion. The photoelectric effect and compton interaction will be forced to standard energy processes and rayleigh scattering and gammaconversion will be disabled. Even if these 4 processes are defined in other way before or after the Fictitious process, they will be forced as mentionned.

The minimum energy below which regular tracking is used can be specified manually in the part of the mac file that describes the phantom by using the command

```
/gate/NAME/setFictitiousEnergy 10 keV
```

Here *NAME* is the name of the voxelized phantom specfied by the aforementioned *insert* command. When this energy threshold is not set, the minimum fictitious energy is guessed for the given voxel size, involved processes, and materials. The simulation time can be further reduced by discarding lower energy photons within the phantom. Photons with an energy below the discard energy limit are not simulated. This optional command is

```
/gate/NAME/setGammaDiscardEnergy 300 keV
```

The electron range cut and especially the gamma energy cut (setGammaDiscardEnergy) should be used with care, since they can alter the results.

Example

```
# PHANTOM SECTION
/gate/world/daughters/name voxelphantom
/gate/world/daughters/insert fictitiousVoxelMap
#
/gate/voxelphantom/setFictitiousEnergy 10 keV
/gate/voxelphantom/setGammaDiscardEnergy 300 keV
# Read the file
/gate/voxelphantom/geometry/insertReader interfile
/gate/voxelphantom/interfileReader/insertTranslator range
/gate/voxelphantom/interfileReader/rangeTranslator/readTable range_atn.data
/gate/voxelphantom/interfileReader/rangeTranslator/describe 1
/gate/voxelphantom/interfileReader/readFile interfilename.h33
/gate/voxelphantom/placement/setTranslation 0. 0. 0. mm
# Attach the phantom SD and the output module
/gate/voxelphantom/attachVoxelPhantomSD
/gate/voxelphantom/verbose 2
...

# PHYSICS SECTION
/gate/physics/addProcess Fictitious
#/gate/physics/addProcess PhotoElectric
#/gate/physics/addProcess Compton
#/gate/physics/addProcess GammaConversion
#/gate/physics/addProcess LowEnergyRayleighScattering
```

```
...
```

Alternative usage of voxelized geometry (radiation therapy applications)

Another way to use 3D images is available in GATE. To insert an image you may use the following commands:

```
/gate/world/daughters/name                    patient
/gate/world/daughters/insert                  ImageNestedParametrisedVolume
/gate/patient/geometry/SetImage               data/patient.hdr
/gate/geometry/setMaterialDatabase            data/patient-HUmaterials.db
/gate/patient/geometry/SetHUToMaterialFile    data/patient-HU2mat.txt
```

In this case,three files need to be provided:

- "patient.hdr" is the filename of the image in Analyze (http://eeg.sourceforge.net/ANALYZE75.pdf) file format (hdr/img). **WARNING**: this file format consists in a pair of files with .hdr/.img extension, however it is different from the Interfile image format (which had the same file extension). See for example here (http://www.colin-studholme.net/software/rview/rvmanual/fileform.html) for more details. (We know this is a mess but it will be changed in the future).
- [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 we mainly used CT images. However, any image type can be used (with floating point voxel value).
- "patient-HU2mat.txt" is a simple text file with three columns: HU_start HU_end material_name. It allows to associate a material to each HU voxel value in the image. This text file can be written by hand or generated with the automated method (see next section)

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
/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.
- 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 here (http://wiki.opengatecollaboration.healthgrid.org/?title=GateRT)

Voxellized source

There are two possibilities within GATE to read in voxellized sources. One can chose to read an ASCII file or an InterFile image.

ASCII input

The first line of this file should consist of the number of pixels of the image: nx, ny, nz. This determines the size of one slice and the number of slice the voxelized source should contain. These three dimensions are followed by a sequence of numbers, one number for each voxel. These numbers respresent the activity in each of these voxels and are then later converted to actual activities using a translator that can be a linearTranslator or that can be a rangeTranslator (only the numbers >0 are converted into voxels). The macro-input is detailed hereafter:

```
# Declaration of the fact that a voxelized source will be used.
/gate/source/addSource voxel voxel

# always use the keyword voxel first to declare the type
# Declaration that the voxelized source will be entered
# using image (ASCII) data.
/gate/source/voxel/reader/insert image

# example for a linear translator: this scales all numbers
# directly into activities
/gate/source/voxel/imageReader/translator/insert linear
/gate/source/voxel/imageReader/linearTranslator/setScale 1. Bq

# example for a range translator (can not be used simultaneously)
# here the numbers of the ASCII file are discretized in intervals
# and are then converted to predefined activities
/gate/source/voxel/imageReader/translator/insert range
/gate/source/voxel/imageReader/rangeTranslator/read activityRange.dat
/gate/source/voxel/imageReader/rangeTranslator/describe 1

# The following line allows you to insert the ASCII file
# that contains all necessary numbers for all pixels
/gate/source/voxel/imageReader/readFile image.dat

# 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/voxel/setPosition -12.8 -12.8 0. mm

# 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 511. keV
/gate/source/voxel/gps/angtype iso
/gate/source/voxel/gps/mintheta 90. 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

/gate/source/voxel/dump 1
```

An example of a range translation table from numbers to activities (activityRange.dat in the example) is shown below:

- set the number of subdivisions 6
- define the intervals ex. [200,210] and attach
- a correlated activity: 1. Bq in this example 200 210 1. 211 220 3. 221 230 5. 231 240 10. 241 250 20. 251 255 40.

where you specify the number of subdivisions (6 in this example), followed by their range and the actual activity. If the number in the ASCII file, for a given voxel, is for instance between 221 and 230, 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).

InterFile input

Another more user-friendly possibility is to read in an image stored in InterFile format, where the gray scale of the image is converted into activity values. The rangeTranslator the same kind as the one above. The macro for reading in interFile images as source distributions is detailed below:

```
# Declaration of the fact that a voxellized source will be used.
/gate/source/addSource voxel voxel

# Declaration that the voxellized source will be entered
# using Interfile data.
/gate/source/voxel/reader/insert interfile
```

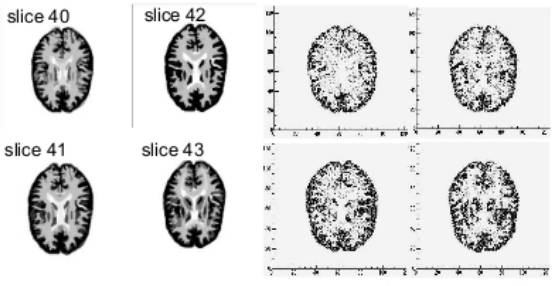


Figure 7.3: emission map from digital Hoffman phantom (left:data - right: translated activity values).

```
# example for a range translator here the numbers of the
# Interfile are discretized in intervals and are then
# converted to predefined activities
/gate/source/voxel/interfileReader/translator/insert range
/gate/source/voxel/interfileReader/... .../rangeTranslator/readTable activityRange.dat
/gate/source/voxel/interfileReader/rangeTranslator/describe 1

# The following line allows you to insert the Interfile datafile
/gate/source/voxel/interfileReader/readFile hof.h33
/gate/source/voxel/interfileReader/verbose 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

# The default position of the voxellized source is in the 1st
# quarter. So the voxellized source has to be shifted over half its
# dimension in the negative direction on each axis
/gate/source/voxel/setPosition -128. -128. 0. mm
/gate/source/voxel/dump 1
```

Using this InterFile reader any digital phantom or patient data, stored in InterFile format, can be read in as emission distribution.

An example of the Hoffman brain phantom, where the gray scales have been translated to activity distributions is shown in Figure 7.3.

GATE also offers the possibility to read in voxellized attenuation geometries following these two strategies. The gray scale is then converted to material definitions in that case using an analogous translator (See the voxellized sources and phantoms examples which are included in the release).

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.

Users Guide V6:How to run Gate

From GATE collaborative documentation wiki

Interactive mode

To start Gate in interactive mode, simply type:

```
$ Gate
```

and the following output (or something similar) will appear on the screen:

```
1       ************************************************
2       Geant4 version $Name:  $
3                                    (3-October-2003)
4                    Copyright : Geant4 Collaboration
5       ************************************************
6       Time set to (s) 0
7       Visualization Manager instantiating...
8       Visualization Manager initialising...
9       Registering graphics systems...
10      You have successfully chosen to use the following graphics systems.
11      Current available graphics systems are:
12        DAWNFILE (DAWNFILE)
13        VRML1FILE (VRML1FILE)
14        VRML2FILE (VRML2FILE)
14        OpenGLImmediateX (OGLIX)
15        OpenGLStoredX (OGLSX)
16      /control/saveHistory
17      /run/verbose 0
18      /event/verbose 1
19      /tracking/verbose 1
20      /gate/timing/setTime 0. s
21      Time set to (s) 0
22      /gate/application/setTimeSlice 1. s
23      PreInit>
```

This output will vary depending on your Gate installation, that is what software was installed and how it was installed. Notice that the numbers on the left do not appear in the actual output. They are shown here just for didactic purposes.

Lines 1-5 indicates the version of the Geant4 software in your installation. Lines 6-9 are initialization messages from Gate. If you installed Gate with visualization functions, then you should see messages like lines those appearing in lines 10-15. Then, Gate runs the file *prerunGate.mac* located in the *petsim* directory. It then outputs the command lines 16-22 found in that file. Finally, and if everything went right, then Gate outputs the interpreter command prompt (line 23). 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       PreInit> ls
2       Command directory path : /
3        Sub-directories :
4          /control/   UI control commands.
5          /units/   Available units.
6          /persistency/   Control commands for Persistency package
7          /geometry/   Geometry control commands.
8          /tracking/   TrackingManager and SteppingManager control commands.
9          /event/   EventManager control commands.
10         /run/   Run control commands.
11         /random/   Random number status control commands.
12         /particle/   Particle control commands.
13         /process/   Process Table control commands.
14         /gate/   Gate detector control.
15         /hits/   Sensitive detectors and Hits
16         /digi/   DigitizerModule
17         /vis/   Visualization commands.
18       Commands :
19      PreInit>
```

When the *Sub-directories* names (lines 4-17) 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       PreInit> ls /control
2       Command directory path : /control/
3       Guidance :
4       UI control commands.
5        Sub-directories :
6        Commands :
7          execute * Execute a macro file.
8          loop * Execute a macro file more than once.
9          foreach * Execute a macro file more than once.
10         suppressAbortion * Suppress the program abortion caused by G4Exception.
11         verbose * Applied command will also be shown on screen.
12         saveHistory * Store command history to a file.
13         stopSavingHistory * Stop saving history file.
14         alias * Set an alias.
15         unalias * Remove an alias.
16         listAlias * List aliases.
17         shell * Execute a (Unix) SHELL command.
18         manual * Display all of sub-directories and commands.
19   createHTML * Generate HTML files for all of sub-directories and commands.
20         maximumStoredHistory * Set maximum number of stored UI commands.
21      PreInit>
```

A * at the end of the *Sub-directories* names means that it is the last level for that subdirectory. In line 7, 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      PreInit> /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:

```
PreInit> exit
Graphics systems deleted.
Visualization Manager deleting...
```

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. Then, we may have the following **Acquisition.mac** macro file:

```
#       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    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, we run Gate, for instance, as follows:

```
$ Gate -a CoincWindow 10 -a lld 350 -a 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 **-a lld 350** is not the same as **-a LLD 350**.
- All aliases in your macro file(s) must be defined when you run Gate. If some are undefined the simulation will fail.

Batch mode

It is possible to run a Gate simulation in *batch* mode, i.e. the mode in which you do nott need to enter the interpreter and run the **/control/execute** and **exit** commands every time.

If you want to run a simulations in *batch* mode, you can do so by redirecting the standard input of Gate with the **<** symbol and the name of the file you want to run. For example,

```
$ Gate -a CoincWindow 10 -a lld 350 -a uld 650 < 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 (although not compulsory) to avoid running visualization commands in batch mode.

Cluster mode

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

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).

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).

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.

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.

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. Figure 13.1 shows the newly created tree in each ROOT file.

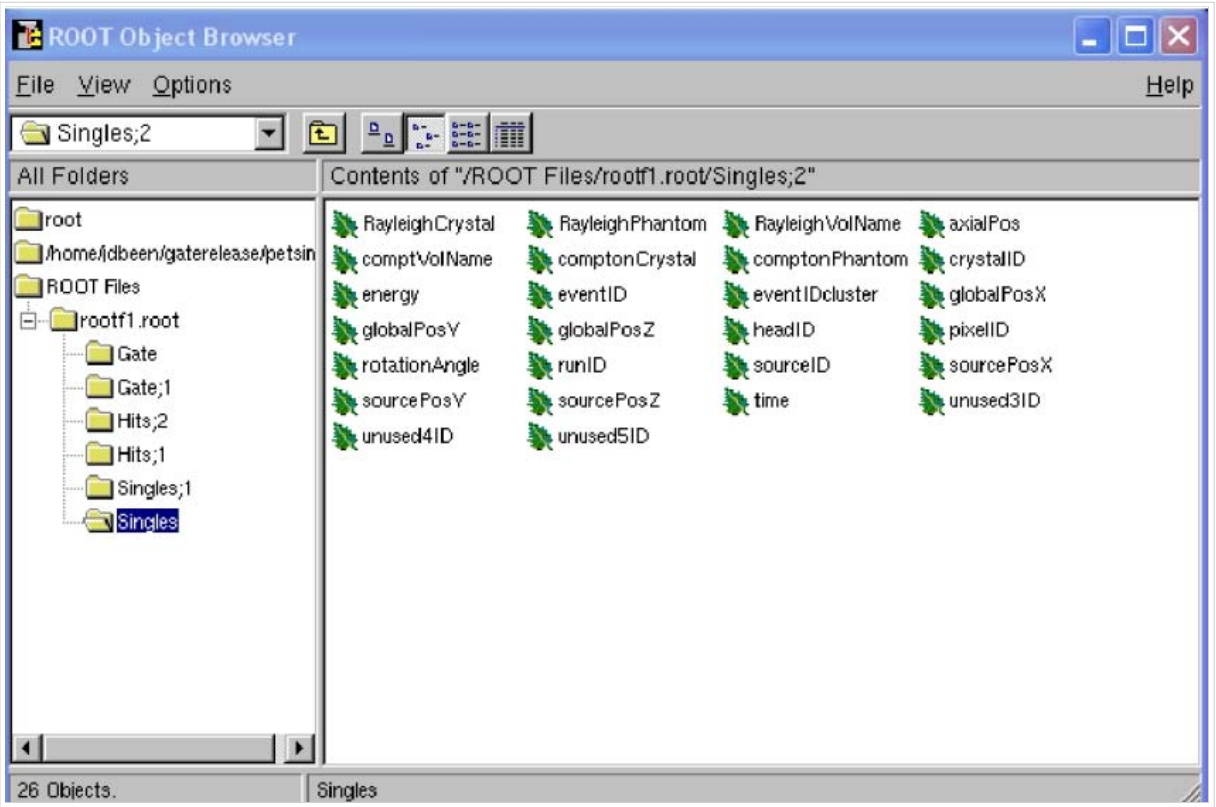


Figure 13.1: Example of ROOT file with added cluster eventIDs

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")
```

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.

How to launch *DigiGate*

GATE offers an operating mode dedicated to digitizer optimization, known as *DigiGate* (see Users Guide V6: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.

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
```

Users Guide V6:Visualization

From GATE collaborative documentation wiki

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.

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.

Command Lines

Basic commands provided by the GEANT4-package can be used to set basic vizualisation options, as shown below.

Visualization with OpenGL

```
# V I E W E R # Opening of the OpenGL Stored-X viewer which is the standard viewer
/vis/open OGLSX
# or opening of an OpenGL Immediate-X viewer used for digital images
/vis/open OGLIX
# define the zoom factor
/vis/viewer/zoom 1.5
# Set the viewing angle
/vis/viewer/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
```

Instead of block as in the example here, objects like crystal, source, scanner can be assigned specific vizualization properties.

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:

```
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.

Users Guide V6:Bibliography

From GATE collaborative documentation wiki

- Buvat I and Castiglioni I 2002 Monte Carlo simulations in SPET and PET **Q. J. Nucl. Med. 46** 48-61
- G3 **Brun R, Bruyant F, Maire M, McPherson A C, Zanarini P 1987 GEANT3** Technical Report **CERN DD/EE/84-1**
- EGS4 **EGS4 website: http://www.slac.stanford.edu/egs/**
- MCNP **MCNP website: http://laws.lanl.gov/x5/MCNP/index.html**
- G4 **Agostinelli S** et al **2003 GEANT4 - a simulation toolkit** Nucl. Instr. Meth. A506 **250-303 GEANT4 website: http://geant4.web.cern.ch/**
- MIC02 **Santin G, Strul D, Lazaro D, Simon L, Krieguer M, Vieira Martins M, Breton V and Morel C 2003 GATE: A GEANT4-based simulation platform for PET and SPECT integrating movement and time management** IEEE Trans. Nucl. Sci. 50 **1516-1521**
- Siena02 **Strul D, Santin G, Lazaro D, Breton V and Morel C 2003 GATE (GEANT4 Application for Tomographic Emission): a PET/SPECT general-purpose simulation platform** Nucl. Phys. B (Proc. Suppl.) 125C **75-79**
- ITBS02 **Assié K** et al **2004 Monte Carlo simulation in PET and SPECT instrumentation using GATE** to appear in Nucl. Instr. Meth.
- GATE **Jan S, Santin G, Strul S, Staelens S, Assié K, Autret D, Avner S, Barbier R, Bardiès M, Bloomfield P, Brasse D, Breton V, Bruyndonckx P, Buvat I, Chatziioannou A F, Choi Y, Chung Y H, Comtat C, Donnarieix D, Ferrer L, Glick S J, Groiselle C, Kerhoas-Cavata S, Kirov A S, Kohli V, Koole M, Krieguer M, van der Laan D J, Lamare F, Largeron G, Lartizien C, Lazaro D, Maas M C, Maigne L, Mayet F, Melot F, Merheb C, Morel C, Nehmeh S, Pennacchio E, Perez J, Pietrzyk U, Rannou F R, Rey M, Schaart D R, Schmidtlein C R, Simon L, Song T Y, Vieira J M, Visvikis D, Van de Walle R, Wieërs E 2004 GATE - GEANT4 Application for Tomographic Emission: a simulation toolkit for PET and SPECT, to be submitted to** Phys. Med. Biol.
- OpenGATE **GATE website: http://www.opengatecollaboration.org**
- Staelens **Staelens S, Strul D, Santin G, Koole M, Vandenberghe S, D'Asseler Y, Lemahieu I and Van de Walle R 2003 Monte Carlo simulations of a scintillation camera using GATE: validation and application modelling** Phys. Med. Biol. 48 **3021-3042**
- Lazaro **Lazaro D, Buvat I, Loudos G, Strul D, Santin G, Giokaris N, Donnarieix D, Maigne L, Spanoudaki V, Styliaris S, Staelens S and Breton V 2003 Monte Carlo simulation of a CsI(Tl) gamma camera dedicated to small animal imaging using GATE** Phys. Med. Biol.
- doxygen doxygen **website: http://www.doxygen.org/index.html**
- RL **R. Lecomte** at al **. Design and Engineering Aspects of a high resolution Positron Tomograph for small animal imaging.** IEEE Trans. Nucl. Sci. **Vol** 41 **, $n0$4, August 1994, 1446-1452**
- BD **Casey M and Nutt R 1986 A multicrystal two dimensional BGO detector block system for positron emission tomography** IEEE Trans. Nucl. Sci. 33 **460-463**
- Biggs **Biggs F, and Lighthill R 1990 Preprint Sandia Laboratory, SAND 87-0070**
- Knoll **Knoll G F 1979 Radiation detection and measurement,** John Wiley \& Sons **, New York**
- ROOT **Brun R, Rademakers F 1997 ROOT - An object oriented data analysis framework** Nucl. Instr. Meth. A389 **81-86 ROOT website : http://root.cern.ch/**
- Interf Interfile **website: http://gamma.wustl.edu/tf/caic/interfile33/**
- FORE **Defrise M** at al **1997 Exact and approximate rebinning algorithms for 3-D PET data** IEEE Trans. Med. Imag. 16 **145-158**
- LMF List Mode Format (LMF) **website: http://www.opengatecollaboration.org**
- STIR Software for Tomographic Image Reconstruction (STIR) **website: http://stir.sourceforge.net**
- geant4physics **Geant4** physics reference manual
- geant4user **Geant4** User's Guide For Application Developers
- levin1996 **A. Levin and C. Moisan.**
- A more physical approach to model the surface treatment of scintillation counters and its implementation into **DETECT**. In **IEEE Nuclear Science Symposium and Medical Imaging Conference Record** , volume 2, pages 702--706. IEEE, 1996.
- nayar1991 **S. K. Nayar, K. Ikeuchi, and T. Kanade. Surface reflection: physical and geometrical perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence** , 13(7):611--633, 1991.

Users Guide V6:Architecture of the simulation

From GATE collaborative documentation wiki

Main rules

GATE simulations are based on the execution of scripted commands (Users Guide V6:Getting started) gathered in macros.

A simulation is generally divided into 7 steps (Fig. 9.1) as follows:

1. Verbosity and Visualization (Users Guide V6:Visualization)
2. Geometry (Users Guide V6:Defining a geometry) (Users Guide V6:Defining a system) (Users Guide V6:Attaching the sensitive detectors)
3. Digitizer (Users Guide V6:Digitizer and readout parameters)
4. Physics (Users Guide V6:Setting up the physics)
5. Sources (Users Guide V6:Activity, source, voxellized phantoms)
6. Outputs (Users Guide V6:Data output)
7. Experiment (Users Guide V6:Digitizer and readout parameters)

The first 4 steps correspond to the PreInit mode of **GEANT4** whereas the last 3 occur after the initialization of the simulation. The first 4 steps are validated by the **GEANT4** command:

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

Once this phase is completed, the sources can be inserted in the setup and the simulation can be run.

Verbosity and Visualization

Verbosity

For each simulation module, one can set a verbosity level between 0 and 2. The higher the verbosity level, the higher the level of information returned by GATE. By default, the verbosity level is set to 0, but if one wants to follow in detail each step of the simulation, it can be set to higher values. As an example, to get the computation time of the simulation written by Gate on the screen at the end of a simulation like this:

```
   User simulation time (sec)   := 13.9
   Real simulation time (sec)   := 15.92
   System simulation time (sec) := 0.03
```

the verbosity level of the output should be set to 2:

```
/gate/output/verbose 2
```

Visualization

There are several tools available for visualization **OpenGL, VRML, DAWN**. They can be activated as a function of the visualization options selected at the GEANT4 configuration step. The on line visualization is a useful tool when developing new geometries. It allows one to visually check the scanner geometry (positions, physical volume overlaps, etc). Once the geometry is checked and one wants to run a complete simulation, it is recommended to disable the on line visualization in order not to overload the CPU.

Geometry

The world

The first volume to be created is the *world*. Each new volume will be inserted in this one, with a given name, and will be a *daughter* of the *world*. The *world* dimensions must be large enough to include the scanner and the phantom.

The system

Next, the *system* type must be chosen, as a function of the scanner to be modelled: *scanner, PETscanner, cylindricalPET, ecat,* CPET *or* SPECTHead. *Each system has a defined number of levels with a specific hierarchical organization (tree geometry) and it is linked with a specific data output format. Two outpout formats are also available, independent of the selected system (ROOT and ASCII).* Once the scanner is built, all the scanner elements should be attached to the system.

The phantom

A *phantom* has to be defined, using materials included in the materials database. The *phantom* can be an analytical or voxelized volume. Each voxel of the volume can be made of a specific material with its own density.

Finally, the sensitive volumes *crystalSD* and *phantomSD* have to be attached. Only the interactions (*hits*) occurring in a volume attached to a sensitive detector will be stored by GATE for the digitization.

Digitizer

The *digitizer* pre-processes the *hits* by sorting, regrouping and adding them to create *singles*. The *singles* are time-stamped and stored in the events history. In PET, the *coincidences* are then sorted out as a function of the coincidence window width. The detection parameters (*temporal resolution, crystal blurring, dead time, threshold, uphold...*) are set at the *digitizer* level.

Physics

The physics part of the simulation defines the simulated physical processes by:

- selecting the appropriate interactions library (*Standard* or *Low Energy package*),
- enabling or disabling the physical processes (*Photoelectric effect, Compton effect, Rayleigh, gamma conversion...*),
- setting cutoffs in energy or range.

Sources

A source is defined by:

- its nature (*particle / ion*),
- its activity (*initial activity, half live...*),
- its geometry (*shape* or *voxelized*),
- its emission angle,
- its movement if necessary.

The source activity can be confined in a specific volume (*e.g* the phantom volume).

Data outputs

The data output formats are of two types in GATE:

- Standard outputs: ASCII, Root
- System dependent outputs: LMF, sinogram, ecat7, Interfile

Standard outputs

ASCII outputs are spread out into 3 files (*Hits*, *Singles,* Coincidences*)*. The Root output is composed of one NTuple (*Gate*) and three TTrees (*Hits*, *Singles*, *Coincidences*) in which the interaction type, position and time information are stored. By default these two types of output are enabled. Each one of these outputs can be enabled or disabled according to the kind of information one is interested in.

Specific outputs

In addition to the two standards outputs, GATE provides system dependent outputs. The LMF output is linked to the *cylindricalPET* system. The Sinogram and the ecat7 outputs are related to the *ecat* system, while the Interfile output relates to the *SPECTHead* system.

All these outputs are characterized by several parameters which have to be correctly set up as explained in Users Guide V6:Data output.

Experiments

The last step of the simulation consists in setting up the experiment. In this part, the duration of the simulated acquisition is set by defining the beginning and the end of the simulation. The overall acquisition time can be subdivided in several time slices of fixed duration. This feature is very useful in GATE, since the geometry is updated only between two time slices. This provides the possibility to take into account the movements of the sources or the detectors by subdividing a run in time slices during the geometry remains unchanged.

Random generator

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.

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
```

Finally, 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.

Example of a PET scanner

The following example describes how to build a PET scanner based on the *cylindricalPET* system.

```
# No verbosity
/control/verbose 0
#
# OpenGL online visualization
/vis/open OGLSX
/vis/viewer/reset
/vis/viewer/set/style surface
/vis/drawVolume
/vis/scene/endOfEventAction accumulate
/vis/viewer/viewpointThetaPhi 30 30
/vis/viewer/zoom 2
#
/tracking/storeTrajectory 1
/gate/geometry/enableAutoUpdate
```

The on line visualization is enabled.

```
#       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* is created. Its dimensions should be large enough to contain all the volumes describing the experiment.

```
#       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 152 mm
/gate/cylindricalPET/geometry/setRmin 130 mm
/gate/cylindricalPET/geometry/setHeight 80 mm
/gate/cylindricalPET/vis/forceWireframe
```

The system is chosen.

```
#       R S E C T O R
/gate/cylindricalPET/daughters/name box1
/gate/cylindricalPET/daughters/insert box
/gate/box1/placement/setTranslation 140 0 0 mm
/gate/box1/geometry/setXLength 20. mm
/gate/box1/geometry/setYLength 19. mm
/gate/box1/geometry/setZLength 76.6 mm
/gate/box1/setMaterial Water
/gate/box1/vis/forceWireframe
#
#       M O D U L E
/gate/box1/daughters/name box2
/gate/box1/daughters/insert box
/gate/box2/geometry/setXLength 20. mm
/gate/box2/geometry/setYLength 19. mm
/gate/box2/geometry/setZLength 19. mm
/gate/box2/setMaterial Water
/gate/box2/vis/forceWireframe
#
#       C R Y S T A L
/gate/box2/daughters/name box3
/gate/box2/daughters/insert box
/gate/box3/geometry/setXLength 20. mm
/gate/box3/geometry/setYLength 2.2 mm
/gate/box3/geometry/setZLength 2.2 mm
/gate/box3/setMaterial Water
/gate/box3/vis/forceWireframe
#
#       L A Y E R LSO
/gate/box3/daughters/name LSO
/gate/box3/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 -5 0 0 mm
/gate/LSO/setMaterial LSO
#
#       L A Y E R LuAP
/gate/box3/daughters/name LuAP
/gate/box3/daughters/insert box
/gate/LuAP/geometry/setXLength  10. mm
/gate/LuAP/geometry/setYLength  2. mm
/gate/LuAP/geometry/setZLength  2. mm
/gate/LuAP/placement/setTranslation 5 0 0 mm
/gate/LuAP/setMaterial LuAP
/gate/LuAP/vis/setColor cyan
#
#       R E P E A T    C R Y S T A L
/gate/box3/repeaters/insert cubicArray
/gate/box3/cubicArray/setRepeatNumberX 1
/gate/box3/cubicArray/setRepeatNumberY 8
/gate/box3/cubicArray/setRepeatNumberZ 8
/gate/box3/cubicArray/setRepeatVector 10. 2.4 2.4 mm
#
#       R E P E A T    M O D U L E
/gate/box2/repeaters/insert cubicArray
/gate/box2/cubicArray/setRepeatNumberZ 4
/gate/box2/cubicArray/setRepeatVector 0. 0. 19.2 mm
#
#       R E P E A T    R S E C T O R
/gate/box1/repeaters/insert ring
/gate/box1/ring/setRepeatNumber 42
#
#       A T T A C H    S Y S T E M
/gate/systems/cylindricalPET/rsector/attach box1
/gate/systems/cylindricalPET/module/attach box2
/gate/systems/cylindricalPET/crystal/attach box3
/gate/systems/cylindricalPET/layer0/attach LSO
/gate/systems/cylindricalPET/layer1/attach LuAP
```

The volumes of the scanner (with names given bu the user, e.g. box1) are connected to the cylindricalPET system (to the predefined names of cylindricalPET system, e.g rsector).

```
#       A T T A C H    L A Y E R  SD
/gate/LSO/attachCrystalSD
/gate/LuAP/attachCrystalSD
#
#       P H A N T O M
/gate/world/daughters/name phantom
/gate/world/daughters/insert cylinder
/gate/phantom/geometry/setRmax 20 mm
/gate/phantom/geometry/setRmin 0. mm
/gate/phantom/geometry/setHeight 100. mm
/gate/phantom/placement/setTranslation 0 0 -40 mm
/gate/phantom/setMaterial Water
/gate/phantom/vis/setColor red
#
#       A T T A C H    P H A N T O M S D
/gate/phantom/attachPhantomSD
```

The scanner system is completely built. It is composed of 42 *rsectors*, each one made of 8 x 8 LSO and LuAP crystals assembled in phoswich.

```
#       M O V E M E N T S
/gate/cylindricalPET/moves/name revolution
/gate/cylindricalPET/moves/insert rotation
/gate/cylindricalPET/revolution/setAxis 0 0 1
/gate/cylindricalPET/revolution/setSpeed 6. deg/s
#
/gate/phantom/moves/name PhantomTranslation
/gate/phantom/moves/insert translation
/gate/phantom/PhantomTranslation/setSpeed 0. 0. 0.1 cm/s
#
# The lines below are just to show how the system moves with time
/gate/timing/setTime 0. s
/gate/timing/setTime 5. s
/gate/timing/setTime 10. s
/gate/timing/setTime 15. s
/gate/timing/setTime 20. s
/gate/timing/setTime 25. s
/gate/timing/setTime 30. s
/gate/timing/setTime 35. s
/gate/timing/setTime 40. s
/gate/timing/setTime 45. s
/gate/timing/setTime 50. s
/gate/timing/setTime 55. s
/gate/timing/setTime 60. s
```

The scanner rotates along the Z axis with a rotation speed of 6 deg/s while the phantom has translation movement with a speed of 0.1 cm/s.

```
#       D I G I T I Z E R
/gate/digitizer/convertor/verbose 0
/gate/digitizer/Singles/insert adder
/gate/digitizer/Singles/adder/verbose 0
#
#       ENERGY BLURRING
/gate/digitizer/Singles/insert crystalblurring
/gate/digitizer/Singles/crystalblurring/setCrystalResolutionMin 0.20
/gate/digitizer/Singles/crystalblurring/setCrystalResolutionMax 0.35
/gate/digitizer/Singles/crystalblurring/setCrystalQE 1.
/gate/digitizer/Singles/crystalblurring/setCrystalEnergyOfReference 511. keV
#
#       READOUT
/gate/digitizer/Singles/insert readout
/gate/digitizer/Singles/readout/setDepth 1
#
#       TEMPORAL RESOLUTION
/gate/digitizer/Singles/insert timeResolution
/gate/digitizer/Singles/timeResolution/setTimeResolution 2. ns
#
#       THRESHOLDER / UPHOLDER
/gate/digitizer/Singles/insert thresholder
/gate/digitizer/Singles/thresholder/setThreshold 250. keV
/gate/digitizer/Singles/insert upholder
/gate/digitizer/Singles/upholder/setUphold 750. keV
/gate/digitizer/Singles/thresholder/verbose 0
#
#       DEAD TIME
/gate/digitizer/Singles/insert deadtime
/gate/digitizer/Singles/deadtime/setDeadTime 250. ns
/gate/digitizer/Singles/deadtime/setMode paralysable
/gate/digitizer/Singles/deadtime/chooseDTVolume box1
#
#       COINCIDENCES SORTER
/gate/digitizer/Coincidences/setWindow 10. ns
/gate/digitizer/Coincidences/minSectorDifference 2
#
#
/gate/systems/cylindricalPET/describe
```

The *digitizer* is setup with a crystal blurring randomly chosen between a minimum and a maximum value. A threshold and an uphold have been respectively set to 250~keV and to 750~keV. The coincidence time window has been set to 10 ns and a coincidence event will be taken into acount only if the difference between the two *rsectors* is greater than or equal to 2. A paralyzable dead-time of 250~ns has been introduced at *rsector* level.

```
#       P H Y S I C S
/gate/physics/gamma/selectCompton lowenergy
/gate/physics/gamma/selectPhotoelectric lowenergy
/gate/physics/gamma/selectRayleigh lowenergy
#
#       INACTIVE COMPTON
#/gate/physics/gamma/selectCompton inactive
#
#       CUT X ,  DELTA AND ELECTRON
/gate/physics/setXRayCut 1 GeV
/gate/physics/setDeltaRayCut 1 GeV
/gate/physics/setElectronCut 1 km
```

At this point, the construction of the detector is over. We can now initialize the simulation.

```
#       I N I T I A L I Z E
/gate/systems/cylindricalPET/verbose 0
/gate/geometry/enableAutoUpdate
/gate/run/initialize
```

An important check can be made just after this initialization in order to test that there is no overlap between volumes from the same "*familly*" (the *mother* volume and her *daughters*) and that a *daughter* volume is inside her *mother* volume. This test is done only once in order to check the geometry.

```
/geometry/test/recursive_test
#
#       V E R B O S I T Y
/control/verbose 0
/grdm/verbose 0
/run/verbose 0
/event/verbose 0
/tracking/verbose 0
/gate/application/verbose 0
/gate/generator/verbose 0
/gate/stacking/verbose 0
/gate/event/verbose 0
/gate/source/verbose 0
#
#       S O U R C E S
/gate/source/addSource twogamma
/gate/source/twogamma/setActivity 1000. becquerel
/gate/source/twogamma/setType backtoback
/gate/source/twogamma/gps/particle gamma
/gate/source/twogamma/gps/energytype Mono
/gate/source/twogamma/gps/monoenergy 0.511 MeV
/gate/source/twogamma/gps/centre 0. 2. 0. cm
/gate/source/twogamma/gps/type Volume
/gate/source/twogamma/gps/shape Sphere
/gate/source/twogamma/gps/radius 0.5 cm
/gate/source/twogamma/gps/shape Cylinder
/gate/source/twogamma/gps/radius 10. mm
/gate/source/twogamma/gps/halfz 50. mmv
/gate/source/twogamma/gps/confine phantom
/gate/source/twogamma/gps/angtype iso
/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/addSource sourceC11
/gate/source/sourceC11/setActivity 10000. becquerel
/gate/source/sourceC11/gps/particle e+
/gate/source/sourceC11/setForcedUnstableFlag true
/gate/source/sourceC11/setForcedHalfLife 1223 s
/gate/source/sourceC11/gps/energytype Carbon11
/gate/source/sourceC11/gps/centre 0. 0. 0. cm
/gate/source/sourceC11/gps/type Volume
/gate/source/sourceC11/gps/shape Cylinder
/gate/source/sourceC11/gps/radius 10. mm
/gate/source/sourceC11/gps/halfz 50. mm
/gate/source/sourceC11/gps/confine phantom
/gate/source/sourceC11/gps/angtype iso
#
/gate/source/list
```

We have defined two sources. The first one called *twogamma* has an activity of 1 kBq and emits two gammas back to back in all directions. The second one, (*sourceC11*), emits beta+ according to the positron energy spectrum of C^{11} decays, with an initial activity of 10~kBq and a halflife of 1223 s. The range of the positron is simulated as well as the *gamma − gamma* acolinearity.

The two sources are confined in the *phantom*.

```
#       O U T P U T
#       ASCII
/gate/output/ascii/disable
/gate/output/ascii/setOutFileHitsFlag 0
/gate/output/ascii/setOutFileSinglesFlag 0
/gate/output/ascii/setOutFileCoincidencesFlag 0
#
#       ROOT
/gate/output/root/enable
/gate/output/root/setFileName root_output
/gate/output/root/setRootNtupleFlag 1
/gate/output/root/setRootHitFlag 0
/gate/output/root/setRootSinglesFlag 1
/gate/output/root/setRootCoincidencesFlag 1
/gate/output/root/setSaveRndmFlag 1
#       LMF
/gate/output/lmf/disable
#
#       S T A R T
/gate/application/setTimeSlice     1.  s
/gate/application/setTimeStart     0.  s
/gate/application/setTimeStop     60.  s
/gate/application/startDAQ
```

The ASCII and LMF outputs are disabled and for the ROOT output,only *Gate* NTuple, *Singles* and *Coincidences* TTrees are stored.

The acquisition duration will last 60 s with slices of 1 s. The geometry will thus be updated every second.

Important hint

You can define the complete simulation in one macro. In order to have a more modular simulation you can divide it into several macros (e.g *vis.mac*, *geometry.mac*, *digi.mac*, *physics.mac*, *sources.mac*, *main.mac*) called from a main macro.

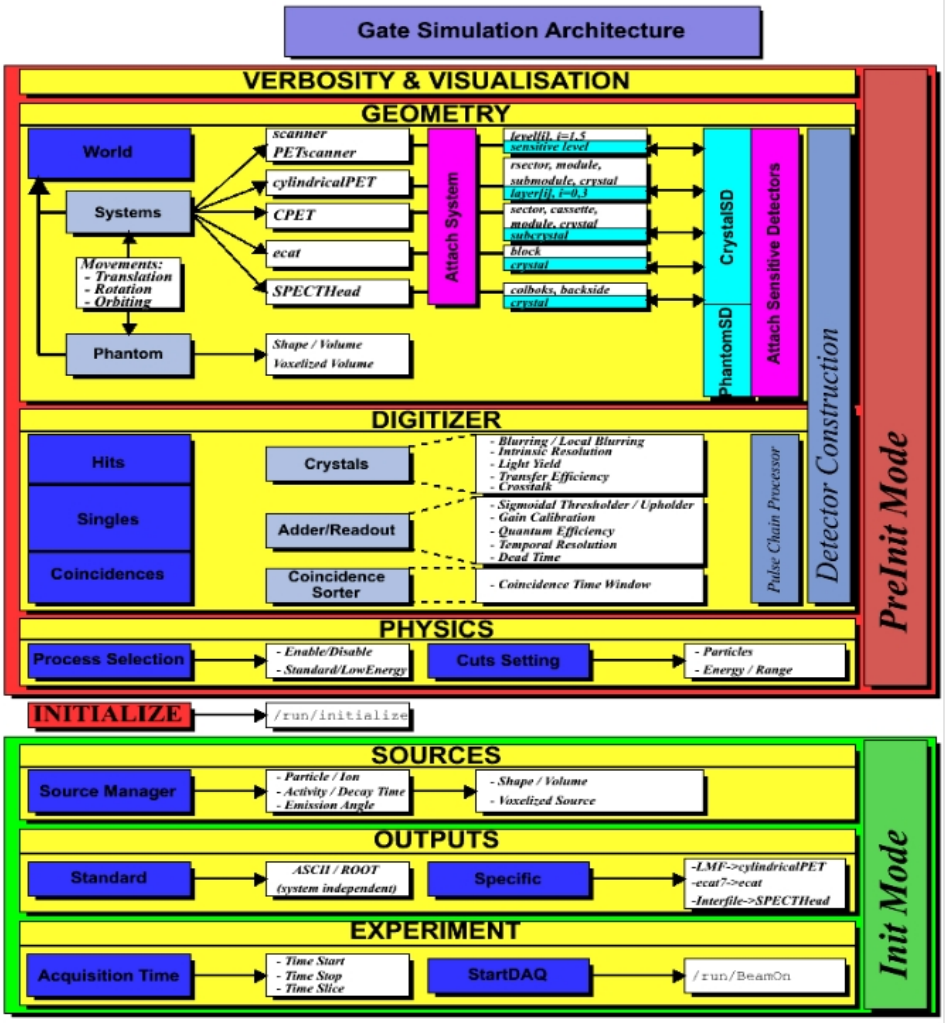


Figure 9.1: GATE simulation architecture

Retrieved from "http://wiki.opengatecollaboration.org/index.php/Users_Guide_V6:Architecture_of_the_simulation"

- This page was last modified on 12 February 2010, at 13:10.

Users Guide V6:Defining a system

From GATE collaborative documentation wiki

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 figure 4.1, composed of volumes called *rsectors*, *modules*, *submodules*, *crystal* and finaly (crystal) *layer*.

Different systems are available in GATE : *scanner*, *SPECTHead*, *cylindricalPET*, *ecat*, *CPET* and *OPET*, which can be used to simulate most of the existing tomographic imaging devices.

Choice of the system

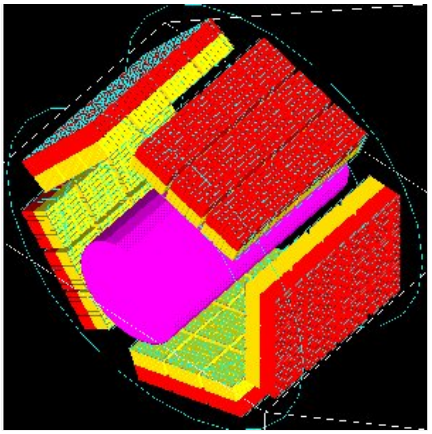


Figure 4.1: Picture of a phantom and a cylindricalPET system composed of 5 rsectors, 4 modules (repeated along Z axis), 3 submodules (repeated along Y axis), 64 crystals (8 x 8) and 2 layers (red and yellow)

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 Users Guide V6:Digitizer and readout parameters) are recorded only for volumes belonging to a defined system (those volumes are declared as *crystalSD*, *SD* for *sensitive detector*, see Users Guide V6: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 4.1. 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 Users Guide V6: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 Users Guide V6:Digitizer and readout parameters), operating at different levels or depths, as shown in Table 4.2. 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 Users Guide V6: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 Users Guide V6:Digitizer and readout parameters).

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 Users Guide V6:Defining a geometry.

- Finally, the specific output corresponding to the system has to be defined for further data analysis (see Users Guide V6:Data output).

Table 4.1: 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 4.2 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	level1 level2 level3 level4 level5	geometry not fixed	Basic output: ASCII or ROOT, coincidences only for PETscanner
CTscanner	module cluster pixel	box box box	Raw Data, ASCII, ROOT
cylindricalPET	rsector module submode crystal layer	box box box box box	Basic Output: ASCII, ROOT and Raw. Specific: LMF
CPET	crystal	cylinder	Basic Output: ASCII, ROOT and Raw
SPECThead	crystal pixel	geometry not fixed	Basic Output: ASCII, ROOT and Raw. Specific: PROJECTIONSET or INTEFILE, no coincidences
ecat	block crystal	box	Basic Output: ASCII, ROOT and Raw. Specific: SINOGRAM or ECAT7
ecatAccel	block crystal	box	Basic Output: AASCII, ROOT and Raw. Specific: SINOGRAM or ECAT7
OPET	rsector module submode crystal layer	box box box box wedge	Basic Output: ASCII, ROOT and Raw. Specific: LMF

Table 4.2: 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 modelized. 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" "level2" "level3" "level4" "level5"	1 2 3 4 5
CTscanner	"module" "cluster_0...2" "pixel_0...2"	1 2 3
cylindricalPET	"rsector" "module" "submodule" "crystal" "layer[i], i=0,3"	1 2 3 4 5
CPET	"crystal"	1
SPECThead	"crystal" "pixel"	1 2
ecat	"block" "crystal"	1 2
ecatAccel	"block" "crystal"	1 2
OPET	"rsector" "module" "submodule" "crystal" "layer[i], i=0,7"	1 2 3 4 5

Different types of systems

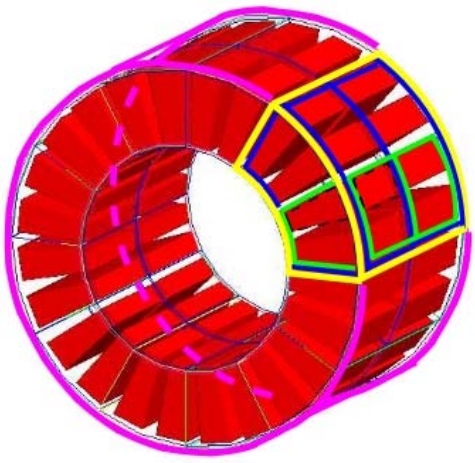


Figure 4.2: Illustration of the scanner system. The different volumes, in particular the sensitive ones, can be of any shape, here cylindrical sector crystals, instead of boxes in other systems. The scanner cylinder is drawn in magenta, whereas one of the sector components : Level1, Level2, Level3, Level4 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 3.2

Figure 4.4 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.

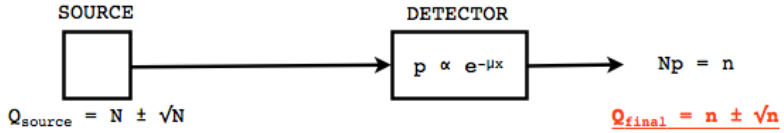
This variance reduction technique (VRT) has been developped with the aim to making the simulation time faster. Here are the successive steps of the implementation:

- **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:

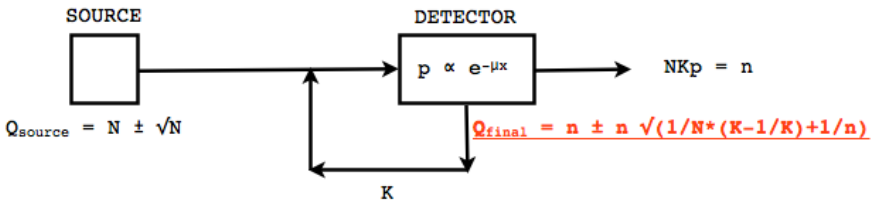
$PATH = MFP * -log(1 - R)$
R being a distribution uniformly random number between 0 and 1

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:

- Without VRT:



- With VRT:



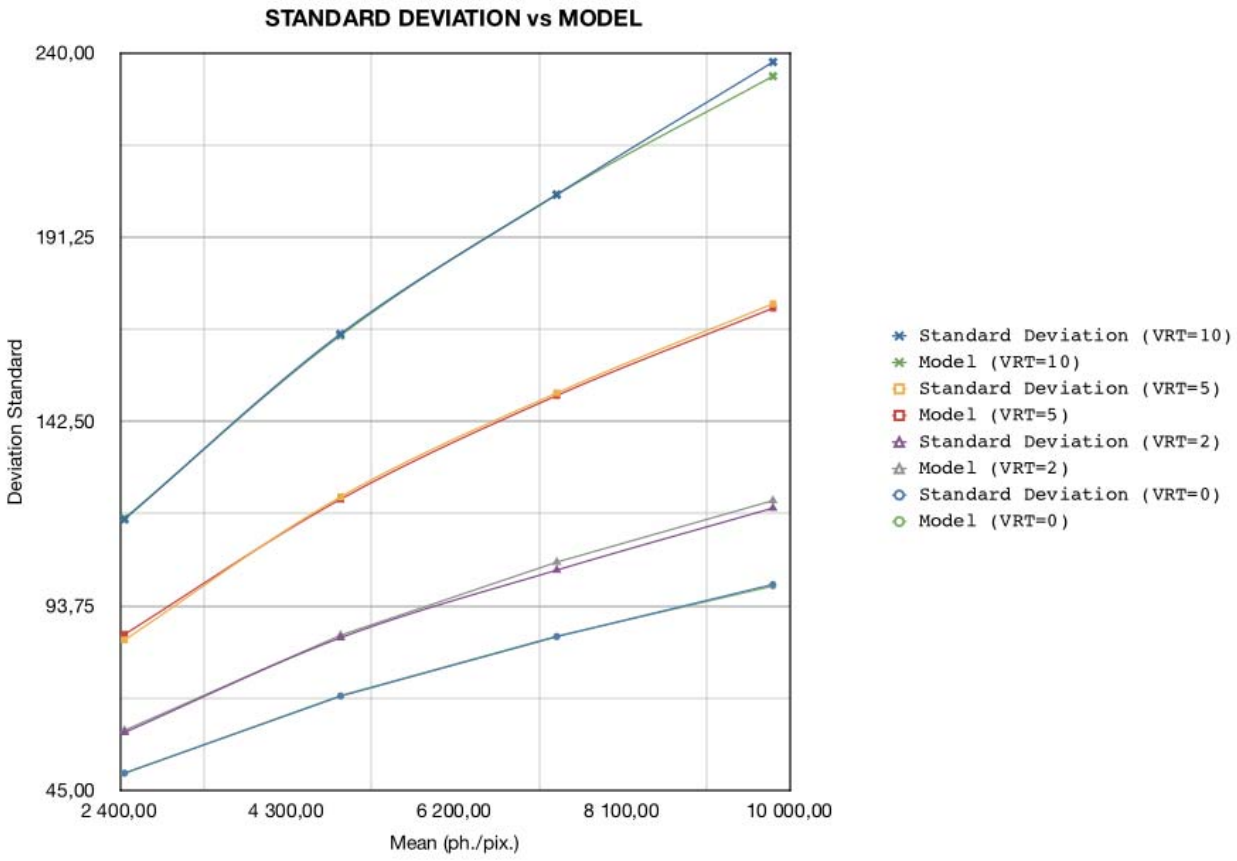
N: mean of the generated particles, and $\sqrt{N}$ its standard deviation.
p: binomial probability of detection of the particle.
n = Np: mean of the number of detected particles, and $\sqrt{n} = \sqrt{Np}$ its standard deviation.

The simulation time decreases linearly with K. But $K \geqslant 10$, 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

The simulation corresponding to the table is:

- A detector with 10,000 pixels (0.5x0.5x0.5 mm^3) 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
/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
/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 Users Guide V6:Data output#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 4.2.

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 **PreInit mode** of Gate (see Users Guide V6: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

# 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 figure 4.3 and Users Guide V6:Defining a geometry for definitions), whereas these components are generally boxes in other systems.

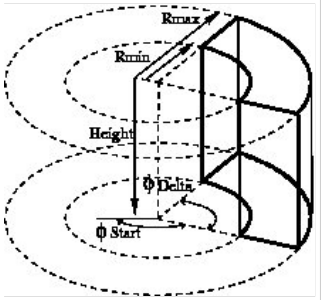


Figure: 4.3: Definition of a CPET sector volume. This system allows one to define sectors with a cylindrical shape instead of sectors with a box shape,like in other PET systems

Use

Described below is an example of code appropriate for modeling a one ring scanner of NaI crystal with a curved shape.

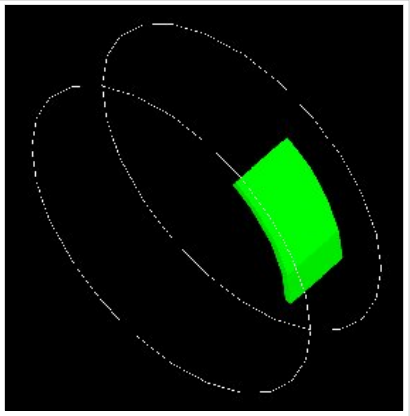


Figure 4.4: One NaI crystal with a curved shape

```
# 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.

```
/gate/systems/CPET/crystal/attach crystal
```

The crystals are set as sensitive detectors (see Users Guide V6:Attaching the sensitive detectors#The crystalSD).

```
/gate/crystal/attachCrystalSD
```

The digitizer part (see Users Guide V6:Digitizer and readout parameters#Digitizer modules) is made of the *adder* module and some blurring module (see Users Guide V6:Digitizer and readout parameters).

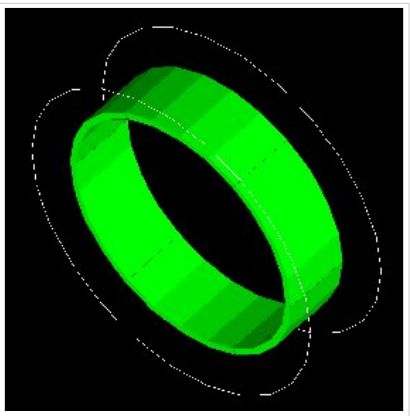


Figure 4.5: After the ring repeater, the scanner consists of 6 NaI crystals

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:

```
$GATEHOME/example_PET_Scanner/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 Users Guide V6:Data output#Sinogram output and Users Guide V6:Data output#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 Figure 4.6).

```
/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
```

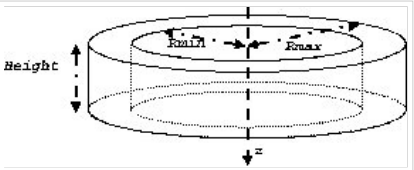


Figure 4.6: Definition of the base

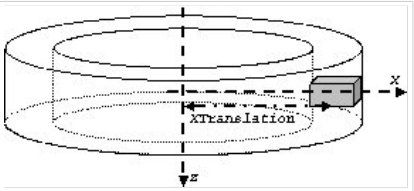


Figure 4.7: Definition of the block

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 Figure 4.7).

```
/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
```

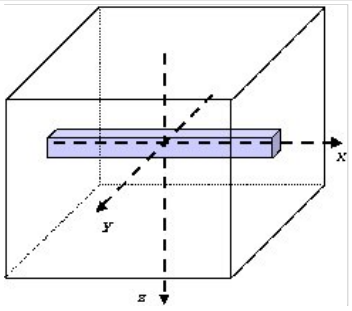


Figure 4.8: Definition of the crystal

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 Figure 4.8).

```
/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 Users Guide V6:Attaching the sensitive detectors#The crystalSD).

```
/gate/crystal/attachCrystalSD
```

The digitizer part (see Users Guide V6:Digitizer and readout parameters#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 [Users Guide V6:Data output#Sinogram output]] and Users Guide V6:Data output#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 Figure 4.6). 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
/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)
/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/
```

Figure 4.9 shows the final OPET scanner.

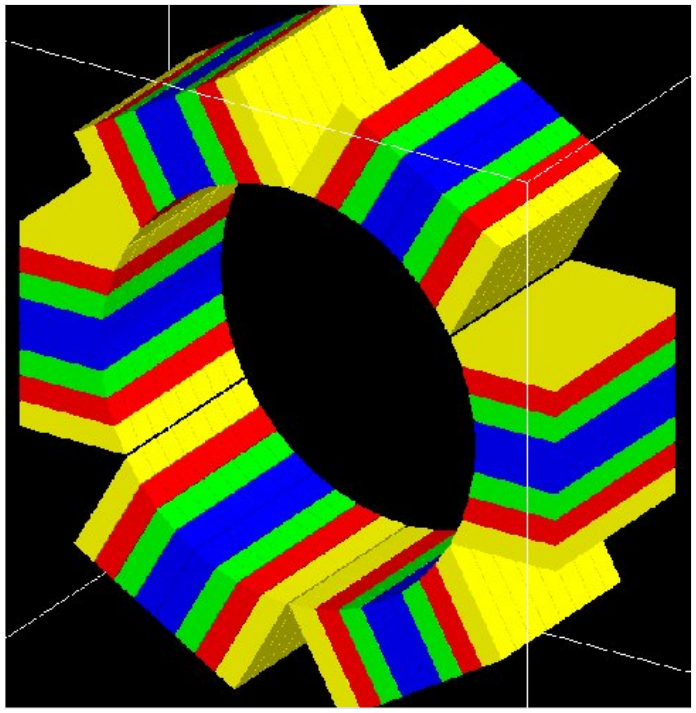


Figure 4.9: The OPET scanner

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 [Users Guide V6:Data output#Interfile output]]. An example macro defining a typical SPECT scanner can be found in:

```
$GATEHOME/exampleSPECT_Scanners/Interfile.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
```

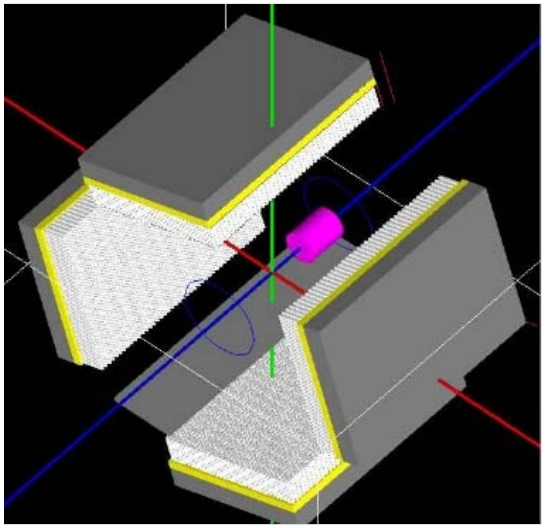


Figure 4.10: Example of a hypothetical four-headed SPECThead system. The detectors are not pixelated in this example

```
# 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
/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
/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
```

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 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
/gate/fanbeam/placement/setRotationAngle -90 deg
/gate/fanbeam/vis/setColor blue
/gate/fanbeam/vis/forceWireframe
```

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
```

Users Guide V6:Attaching the sensitive detectors

From GATE collaborative documentation wiki

General purpose

Once a model has been defined for the scanner through the construction of a system (see Users Guide V6: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. Figure 6.1 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.

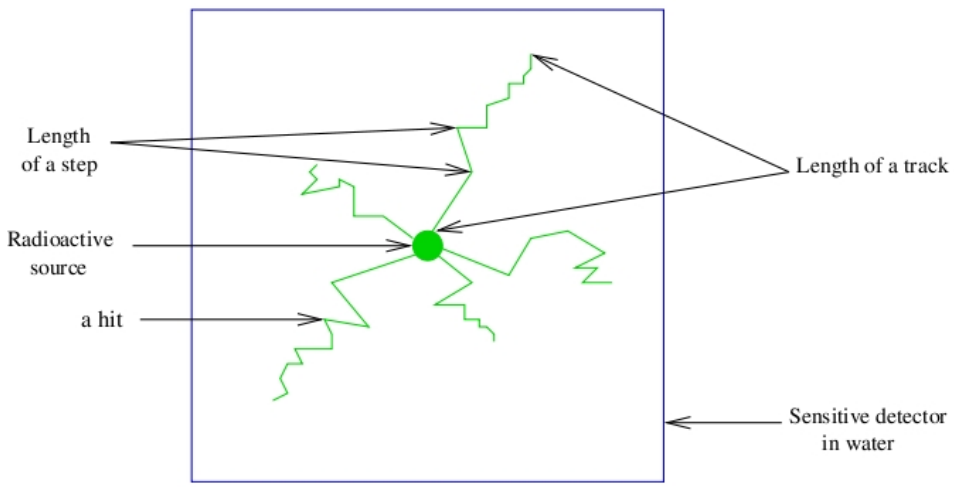


Figure 5.1: Particle interactions in a sensitive detector

It is essential to remember that 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.

Two sensitive detectors are defined in GATE:

- **The crystalSD** is used to record information regarding interactions inside the volumes belonging to a scanner for instance (crystals or collimators).
- **The phantomSD** 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).

A complete definition of the simulation context normally involves performing the two sorts of attachments: some volumes are attached to the *phantomSD*, while others are attached to the *crystalSD*.

The two types of sensitive detector

The crystalSD

Definition and use

The *crystalSD* may be used to record information regarding interactions taking place inside some volumes of the scanner: energy deposition, positions of interaction, origin of the particle (emission vertex), type of interaction (name of the physical processes involved), etc.

Attachment of the crystalSD

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.

```
# A T T A C H S Y S T E M
/systems/SPECThead/crystal/attach crystal
```

Then, the second command attaches the crystalSD to the volume representing the scintillation crystal in the geometry.

```
# A T T A C H S E N S I T I V E D E T E C T O R
/crystal/attachCrystalSD
```

The phantomSD

Definition and use

The *phantomSD* plays a crucial role in GATE simulations, as it is used to detect and tally Compton and Rayleigh interactions taking place in the scanner FOV. 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 relating to the hits:

- The number of Compton and Rayleigh interactions occuring in all the volumes attached to the *phantomSD*: this is stored in the data output variables **nPhantomCompton** and **nPhantomRayleigh**. These pieces of information are also available for the *crystalSD* with the variables **nCrystalCompton** and **nCrystalRayleigh**.
- The last volume attached to the *phantomSD* in which a Compton or a Rayleigh interaction occured: the data output variables including these volume names are **compVolName** and **RayleighVolName**.

Attachment of the phantomSD

- One first needs to define a dummy, air-filled volume covering the whole field-of-view of the scanner.
- Then, all the source volumes should be offsprings (direct or indirect) of this volume.
- Last, all these volumes (FOV and sources) should be attached to the *phantomSD* using the command **attachPhantomSD**.

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 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:

```
# A T T A C H S Y S T E M
/systems/SPECThead/base/attach FOV
/systems/SPECThead/base/attach head
/systems/SPECThead/base/attach body
```

Then, commands attach the phantomSD to the volumes representing the scattering volumes in the geometry:

```
# A T T A C H S E N S I T I V E D E T E C T O R
/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.

```
# ATTACH SYSTEM AND SENSITIVE DETECTOR CRYSTALSD IN ORDER TO RETRIEVE DATA OUTPUTS ON PHANTOMSD
/systems/SPECThead/crystal/attachCrystalSD
/crystal/attachCrystalSD
```

IMPORTANT:

It is impossible to attach two sensitive detectors to the same volume. Thus, to count also the Compton interactions occuring in the scintillating crystal, the variable **nCrystalCompton** has been created: its role is similar to that of the variable nPhantomCompton, e.g it stores the number of Compton interactions in the scintillating crystal.

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 Users Guide V6:Defining a geometry). In order to record the same information concerning the interactions occuring in a voxellized matrix, see Users Guide V6:Activity, source, voxellized phantoms.

Users Guide V6:Digitizer and readout parameters

From GATE collaborative documentation wiki

General Purpose

The purpose of the digitizer module is to simulate the behavior of the scanner detectors and signal processing chain. In this section, the algorithms used to simulate a scanner electronic readout scheme are described. In the case of PET, an overview of the main steps used to produce coincidences from the simulated particle information is briefly described. A more detailed explanation of how to control the behavior of each of these steps is given. Finally, a complete example of a readout macro file is presented.

From particle detection to coincidences in GATE

GATE uses Geant4 to generate particles and transport them through the different materials. This mimics the *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 (referred to in Geant4 simply as a 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 interaction, the type of particle and material, etc. The calculation of 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, and so on. 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 digitizer portion of the GATE architecture. 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).

Once this list is created, a second processing stage can be inserted to sort the *Singles* list for coincidences in case of PET systems. 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 behavior 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. Moreover, in the case of PET analysis, the trigger logic is based on one or more decisions defined by the user that depend upon physically observable quantities such as energy thresholds and coincidence times.

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 for any reason you want to disable the digitizer process and all output (that are already disabled by default), you can use the 2 following commands:

```
/gate/output/analysis/disable
/gate/output/digi/disable
```

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 figure 8.1.

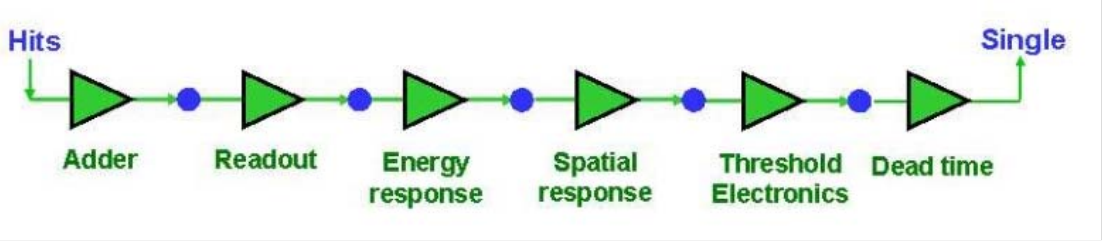


Figure 8.1: The digitizer is organized as a chain of modules that begins with the hit and ends with the single which represents the physical observable seen from the detector.

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

The available modules are explained in the following sections.

Distributions

Definition

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.

Usage

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, the related parameters can be set:

```
/gate/distributions/my_distrib/setMean 350 keV
/gate/distributions/my_distrib/setSigma 30 keV
```

for this example of a Gaussian distribution. All the parameters, for each type of distribution, are summarized in table 8.1

Table: 8.1: 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

Definition

One particle often creates multiple interactions, and consequently multiple *hits*, within a given crystal. For instance, a photon may interact with a single crystal by two Compton scattering events and a photoelectric absorption. 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. Generally, the adder should be the first module of a digitizer chain.

The adder acts on the lowest level in the system hierarchy, as explained in Users Guide V6:Defining a system. As a result:

- A registered system must be used to describe the geometry (also the mother volume name must corresponds to a registered 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.

Command line

The command to use the adder module is

```
/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. This grouping has to be determined by the user through a variable named *depth*. Using this variable, the *pulses* are summed if their volume ID's are same to this level of depth.

Definition

The readout module regroups pulses per block (group of *sensitive detectors*). The user needs to specify the block depth to indicate the depth within the volume hierarchy at which pulses are summed together. The results of this module are the total energy in a block, the position of the pulse with the maximum energy (winner-takes-all).

Command line

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

Figure XXX 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.

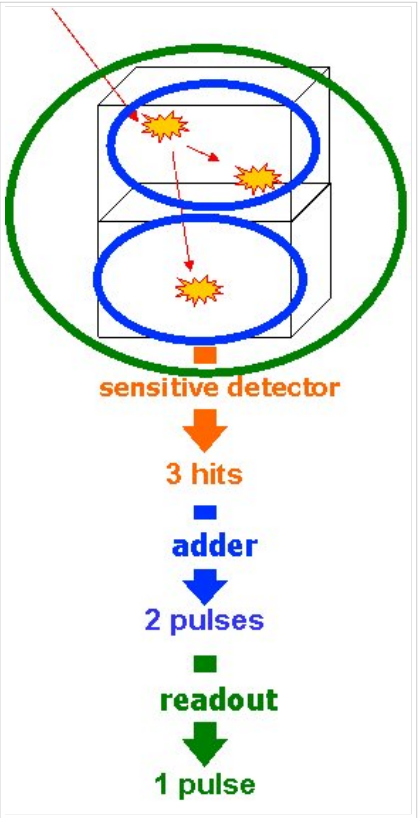


Figure 8.2 Actions of the *it adder* and *it readout* modules

The *setDepth* command line

The importance of this command line is illustrated through the following example from a PET system (see Users Guide V6:Defining a system). In a *cylindricalPET* system, where the first volume level is *rsector*, and the second volume level is *module*, as shown in figure 8.3, 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
```

The energy of this *single* event is the sum of the energy of the pulses inside the white rectangle (*rsector*) of figure 8.3. 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 figure 8.3.

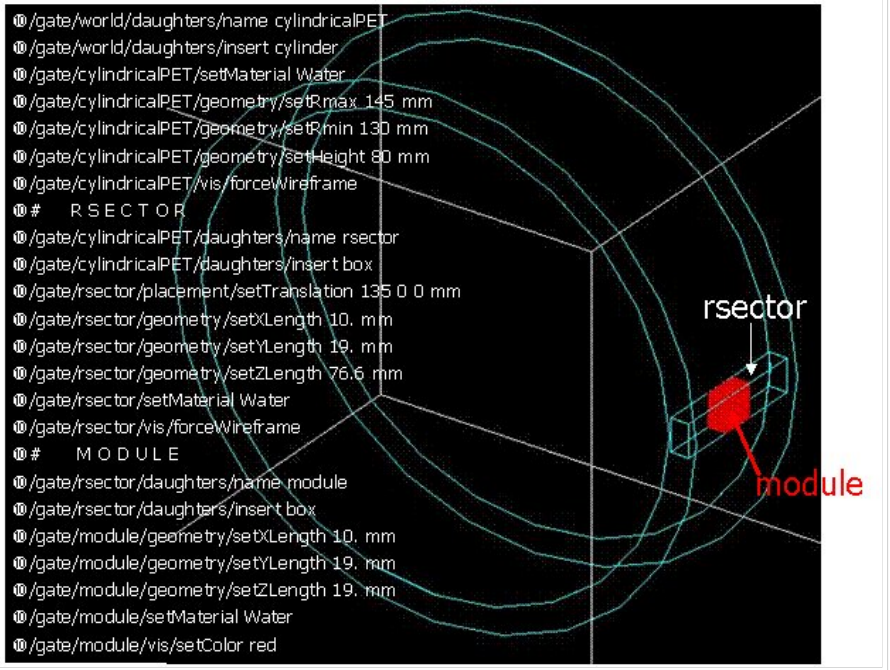


Figure 8.3: Setting the *readout depth* in a CylindricalPET system

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.

Energy blurring

Definition

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. The resolution is a function of energy, R(E), that is proportional to $1/\sqrt{E}$. The energy resolution is set equal to:

$$R = \frac{R_0\sqrt{E_0}}{\sqrt{E}}.$$

Command line

To insert an energy blurring, the commands are as follows (example of a 15% resolution of @ 511 KeV):

```
/gate/digitizer/Singles/blurring
/gate/digitizer/Singles/blurring/setResolution 0.15
/gate/digitizer/Singles/blurring/setEnergyOfReference 511. keV
```

Crystal blurring for a block detector

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.

Command line

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.

Local energy blurring for a detector module with several types of 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).

Command line

To use the *LocalBlurring* module:

- Insert the *LocalBlurring* module,
- choose a valid detector volume name the blurring will be applied to,
- set the resolution for this volume, and set the reference energy for this volume.

For example, 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 !!

Intrinsic resolution blurring with crystals of different compositions

Definition

This blurring pulse-processor simulates a local Gaussian blurring of the energy spectrum (different for different crystals) based on the following model:

$$R = \sqrt{2.35^2 \cdot \frac{1+\bar{\nu}}{\bar{N}_{ph} \cdot \bar{\epsilon} \cdot \bar{p}} + {R_i}^2}$$

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**.

Definition of the light yield

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} .

Definition of the transfer efficiency

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.

Definition of the quantum efficiency

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).

Command lines for modelling the intrinsic resolution

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
/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 numberN_{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.

Command line

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

Definition

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.

BEWARE: this module works only for a chosen volume that is an array repeater!!!

Command line

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.

Thresholder & Upholder

Definition

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.

Command line

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
```

Sigmoidal thresholder

Definition

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.

Command line

```
/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

Definition

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.

Command line

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

For SPECT simulations, the spatial resolution is assumed to follow a Gaussian distribution defined by its width σ.

Command line

```
/gate/digitizer/Singles/insert spblurring
/gate/digitizer/Singles/spblurring/setSpresolution 2.0 mm
/gate/digitizer/Singles/spblurring/verbose 1
```

Spatial blurring for PET

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.

Command line

```
# 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

Definition

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.

Command line

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

Definition

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 lines). Then the distribution of efficiency, for each differentiated volume, is specified via a generic distribution, as described in #Distributions.

Command lines

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 and the memory depth D . 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.

Command line

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

Definition

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 $\left(E_{out} = \sum_{i=0}^{N} E_i\right)$ 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.

Command line

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 figure 8.4.

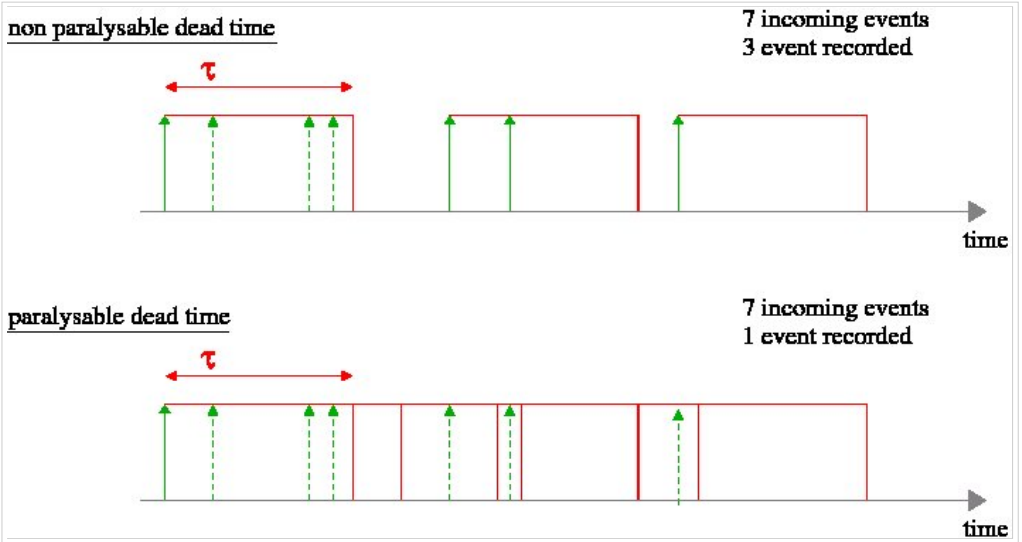


Figure 8.4: For 7 incoming particles and a fixed dead-time τ, the *nonparalysable* electronic readout will accept 3 particles, and the *paralysable* will accept only 1 particle (the dashed arrows represents the removed events, while the solid arrows are the accepted singles)

Definition

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.

Command line

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:

```
# 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 Users Guide V6: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
/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
```

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
```

Coincidence sorter

Definition

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 figure 8.5.

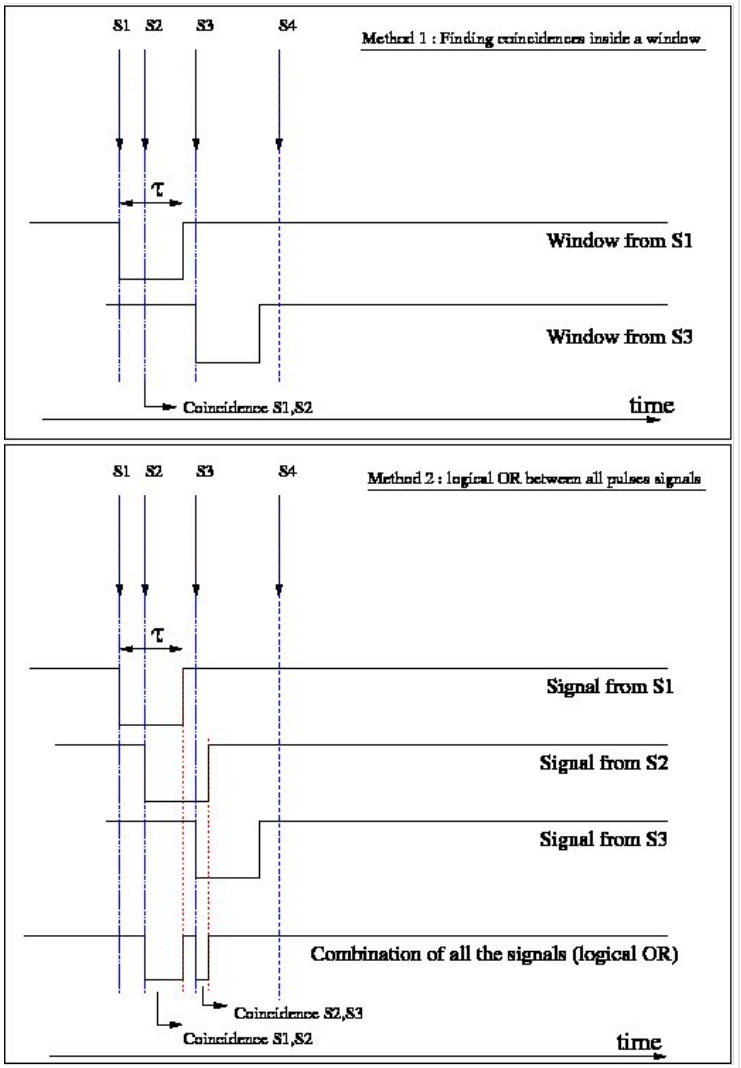


Figure 8.5: Comparison between the two methods of coincidence sorting, for a given succession of singles. In the first one (top), the S2 single does not open its own window since its arrival time is within the window opened by S1. With this method, only one coincidence is created, between S1 and S2. With the second method (bottom), where all singles open their own coincidence window, 2 different coincidences are identified.

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 8.2, and a comparison of the effects of each processing rule for various cases of multiple coincidences is shown in figure 8.6.

Table 8.2: Available multiple policy and associated meaning. When a multiple coincidence involving n *singles* is peocessed, it is first decomposed into a list of $n \cdot (n-1)$ 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.
takeWinnerIfAllAreGoods	If all pairs are goods, take the one with the highest energy.
keepIfOnlyOneGood	If exactly one pair is good, keep all the multicoincidence.
keepIfAnyIsGood	If at least one pair is good, keep all the multicoincidence.
keepIfAllAreGoods	If all pairs are goods, keep all 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.

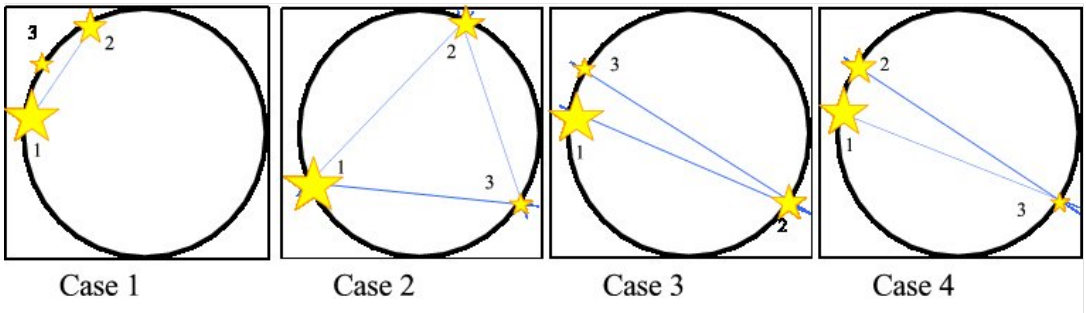


Figure 8.6: Comparison of the behavior of the available multiple processing policies, for various multiple coincidence situations. The stars represent the detected singles. The size of the star, as well as the number next to it, indicate the energy level of the single (ie. single no 1 has more energy 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 associated with Figure 8.6

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)>
```

Finally, the rule to apply in case of multiple coincidences is specified as follows:

```
/gate/digitizer/Coincidences/setMultiplePolicy <policyName>
```

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 figure 8.7.

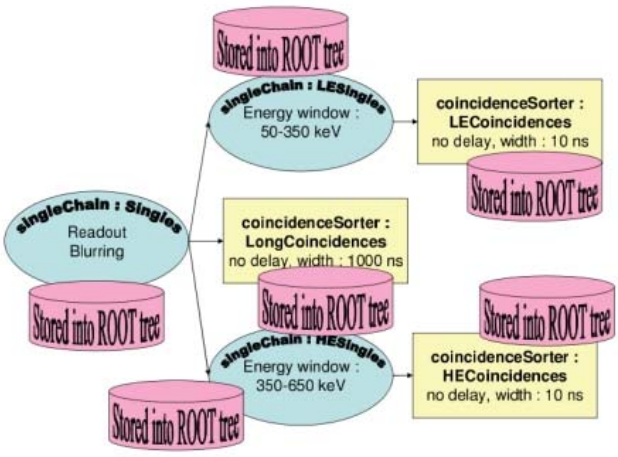


Figure 8.7: Readout scheme produced by the listing from the sections

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.

Definition

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.

Command line

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

Definition

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.

Command line

To insert a multipleKiller, one has to use the syntax:

```
/gate/digitizer/myCoincChain/insert multipleKiller
```

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 Figure 8.8.

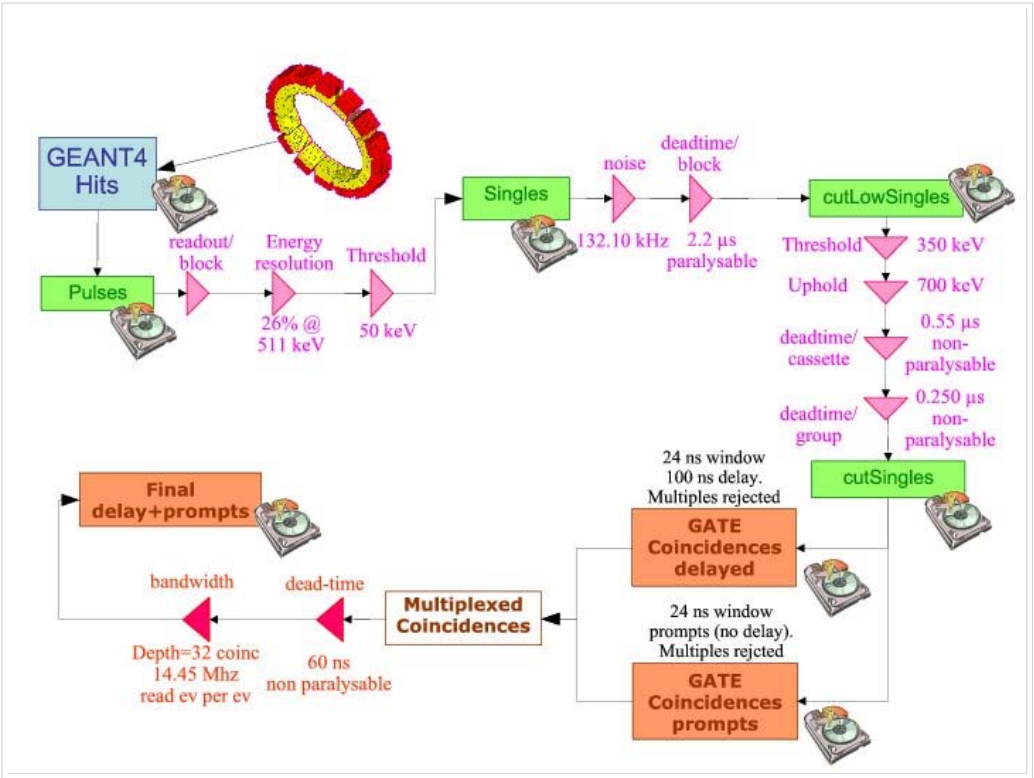


Figure 8.8: Readout scheme produced by the listing below. The disk icons represent the data written to the GATE output files

```
1 # A D D E R
2 /gate/digitizer/Singles/insert adder
3
4 # R E A D O U T
5 /gate/digitizer/Singles/insert readout
6 /gate/digitizer/Singles/readout/setDepth
7
8 # E N E R G Y B L U R R I N G
9 /gate/digitizer/Singles/insert blurring
10 /gate/digitizer/Singles/blurring/setResolution 0.26
11 /gate/digitizer/Singles/blurring/setEnergyOfReference 511. keV
12
13 # L O W E N E R G Y C U T
14 /gate/digitizer/Singles/insert thresholder
15 /gate/digitizer/Singles/thresholder/setThreshold 50. keV
16
17 /gate/digitizer/name cutLowSingles
18 /gate/digitizer/insert singleChain
19 /gate/digitizer/cutLowSingles/setInputName Singles
20
21 # N O I S E
22
23 /gate/distributions/name energy_distrib
24 /gate/distributions/insert Gaussian
25 /gate/distributions/energy_distrib/setMean 450 keV
26 /gate/distributions/energy_distrib/setSigma 30 keV
27
28 /gate/distributions/name dt_distrib
29 /gate/distributions/insert Exponential
30 /gate/distributions/dt_distrib/setLambda 7.57 mus
31
32 /gate/digitizer/cutLowSingles/insert noise
33 /gate/digitizer/cutLowSingles/noise setDeltaTDistributions dt_distrib
34 /gate/digitizer/cutLowSingles/noise setEnergyDistributions energy_distrib
35
36  # D E A D T I M E
37 /gate/digitizer/cutLowSingles/insert deadtime
38 /gate/digitizer/cutLowSingles/deadtime/setDeadTime 2.2 mus
39 /gate/digitizer/cutLowSingles/deadtime/setMode paralysable
40 /gate/digitizer/cutLowSingles/deadtime/chooseDTVolume module
41
42 # H I G H E N E R G Y C U T
43 /gate/digitizer/name cutSingles
44 /gate/digitizer/insert singleChain
45 /gate/digitizer/cutSingles/setInputName cutLowSingles
46 /gate/digitizer/cutSingles/name highThresh
47 /gate/digitizer/cutSingles/insert thresholder
48 /gate/digitizer/cutSingles/highThresh/setThreshold 350. keV
49 /gate/digitizer/cutSingles/insert upholder
50 /gate/digitizer/cutSingles/upholder/setUphold 700. keV
51
52 /gate/digitizer/cutSingles/name deadtime_cassette
53 /gate/digitizer/cutSingles/insert deadtime
54 /gate/digitizer/cutSingles/deadtime_cassette/setDeadTime 0.55 mus
55 /gate/digitizer/cutSingles/deadtime_cassette/setMode nonparalysable
56 /gate/digitizer/cutSingles/deadtime_cassette/chooseDTVolume cassette
57 /gate/digitizer/cutSingles/name deadtime_group
58 /gate/digitizer/cutSingles/insert deadtime
59 /gate/digitizer/cutSingles/deadtime_group/setDeadTime 0.250 mus
60 /gate/digitizer/cutSingles/deadtime_group/setMode nonparalysable
61 /gate/digitizer/cutSingles/deadtime_group/chooseDTVolume group
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).

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.

Angular Response Functions (ARFs) to speed-up single photon (planar and 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 the folder /examples/example_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:

```
# COMPUTE THE ARF TABLES FROM ARF DATA
#
/gate/systems/SPECThead/arf/setARFStage computeTables
```

The digitizer parameters needed for the computation of teh ARF table are defined by:

```
# DIGITIZER PART OF THE ARF SENSITIVE DETECTOR
#
/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 ARFData.txt which is provided in the release folder /examples/example_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
```

Users Guide V6:Data output

From GATE collaborative documentation wiki

Data output is a key point for a software intended to be used for various applications, in various scientific communities. 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. The following chapter describes the various output formats, such as ASCII, Root, Interfile, LMF, and ECAT.

The ASCII output

Introduction

The GateToASCII class of GATE enables the ASCII file output, which is the easiest possible 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 files are not needed for analysis, it is strongly recommended to not enable this output in order to speed up the simulation.

How to enable this output in your macro?

All the output commands

```
/gate/output/...
```

must always be written after the initialization line.

As in most of the output modules of GATE, you can enable ASCII output files for Hits, Singles (at the end of the digitizer chain), Coincidences, but also for Singles after the different steps of the digitizer chain.

As a first step, enable the ASCII output and give an output file name:

```
/gate/output/ascii/enable
/gate/output/ascii/setFileName test
```

In your macro, the different flags should be set to 1:

```
# enable ascii output for hits
/gate/output/ascii/setOutFileHitsFlag 1
#
# enable ascii output for Singles (end of digitizer chain)
/gate/output/ascii/setOutFileSinglesFlag 1
#
# enable ascii output for coincidences
/gate/output/ascii/setOutFileCoincidencesFlag 1
#
# enable ascii output for singles (after a digitizer module)
/gate/output/ascii/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 Users Guide V6:Digitizer and readout parameters.

How to disable this output in your macro?

This output, as for all output is disabled by default, but if it is enabled, some ASCII files will be created, namely: *gateHits.dat, gateSingles.dat, gateRun.dat*. In addition, if coincidences are processed in the simulation, the file *gateCoincidences.dat* will be generated. To disable these ASCII files which can be large, the macro should contain the following lines:

```
/gate/output/ascii/setOutFileHitsFlag 0
/gate/output/ascii/setOutFileSinglesFlag 0
/gate/output/ascii/setOutFileCoincidencesFlag 0
```

Only the file *gateRun.dat* which contain the number of decay per run will then be created.

Description of the ASCII file content

In all files, the units are :

- MeV (energy)
- mm (position)
- s (time)
- deg (angle)

Hits file (gateHits.dat)

Each line is a hit and the columns represent :

- Column 1 : ID of the run (i.e. time-slice)

- Column 2 : ID of the event
- Column 3 : ID of the primary particle whose descendant generated this hit
- Column 4 : ID of the source which emitted the primary particle
- 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.

For cylindricalPET system N=6 :

- Column 5 : ID of volume attached to the "base" level of the system
- Column 6 : ID of volume attached to the "rsector" level of the system
- Column 7 : ID of volume attached to the "module" level of the system
- Column 8 : ID of volume attached to the "submodule" level of the system
- Column 9 : ID of volume attached to the "crystal" level of the system
- Column 10 : ID of volume attached to the "layer" level of the system

For SPECTHead system N=3 :

- Column 5 : ID of volume attached to the "base" level of the system
- Column 6 : ID of volume attached to the "crystal" level of the system
- Column 7 : ID of volume attached to the "pixel" level of the system
- Column N+5 : Time stamp of the hit
- Column N+6 : Energy deposited by the hit
- Column N+7 : Range of particle which has generated the hit
- Column N+8, N+9 ,N+10 : XYZ position of the hit in the world referential
- Column N+11 : Geant4 code of the particle which has generated the hit
- Column N+12 : ID of the particle which has generated the hit
- Column N+13 : ID of the mother of the particle which has generated the hit
- Column N+14 : ID of the photon giving the particle which has generated the hit
- Column N+15 : Number of Compton interactions in phantoms before reaching the detector
- Column N+16 : Number of Rayleigh interactions in phantoms before reaching the detector
- Column N+17 : Name of the process which has generated the hit
- Column N+18 : Name of the last volume where a Compton effect occured
- Column N+19 : Name of the last volume where a Rayleigh effect occured

For the next sections, the system will be set as a cylindricalPET system, so that the number of lines concerning the Volume ID of each level will always be 5.

Singles files (gateSingles.dat)

Each line is a single and the columns are :

- Column 1 : ID of the run (i.e. time-slice)
- Column 2 : ID of the event
- Column 3 : ID of the source
- Column 4, 5, 6 : XYZ position of the annihilation in world referential
- Column 7 to 12 : Volume IDs*(cf. columns 5-10 of sec 11.)
- Column 13 : Time stamp of the single
- Column 14 : Energy deposited by the single
- Column 15 to 17 : XYZ position of the single in the world referential
- Column 18 : Number of Compton interactions in phantoms before reaching the detector
- Column 19 : Number of Compton interactions in detectors before reaching the detector
- Column 20 : Number of Rayleigh interactions in phantoms before reaching the detector
- Column 21 : Number of Rayleigh interactions in detectors before reaching the detector
- Column 22 : Name of the phantom where a Compton effect occursed
- Column 23 : Name of the phantom where a Rayleigh effect occured

Coincidences files (gateCoincidences.dat)

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)
- Column 2 : ID of the event (first single)
- Column 3 : ID of the source (first single)
- Column 4 to 6 : XYZ position of the annihilation in world referential (first single)
- Column 7 : Time stamp (first single)
- Column 8 : Deposited energy (first single)
- Column 9 to 11 : XYZ position in the world referential (first single)
- Column 12 to 17 : volume IDs* (cf. Users Guide V6:Data output#Description of the ASCII file content) (first single)
- Column 18 : Number of Compton interactions in phantoms before reaching the detector (first single)
- Column 19 : Number of Compton interactions in detectors before reaching the detector (first single)
- Column 20 : Number of Rayleigh interactions in phantoms before reaching the detector (first single)
- Column 21 : Number of Rayleigh interactions in detectors before reaching the detector (first single)
- Column 22 : Scanner axial position (first single)
- Column 23 : Scanner angular position (first single)
- Column 24 : ID of the run (i.e. time-slice) (second single)
- Column 25 : ID of the event (second single)
- Column 26 : ID of the source (second single)
- Column 27 to 29 : XYZ position of the annihilation in world referential (second single)
- Column 30 : Time stamp (second single)
- Column 31 : Energy deposited (second single)
- Column 32 to 34 : XYZ position in the world referential (second single)
- Column 35 to 40 : volume IDs.

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)
- Column 42 : Number of Compton interactions in detectors before reaching the detector (second single)
- Column 41 : Number of Rayleigh interactions in phantoms before reaching the detector (second single)
- Column 42 : Number of Rayleigh interactions in detectors before reaching the detector (second single)
- Column 45 : Scanner axial position (second single)
- Column 46 : Scanner angular position (second single)

Selection of the variables in Singles/Coincidences ASCII output

The user can select with a macro command which variables he/she wants in the ASCII file. The mechanism is based on a mask with a mask, i.e. 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. One example is:

```
/gate/output/ascii/setCoincidenceMask 1 0 1 0 1 1
/gate/output/ascii/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 for the ASCII output

When a user defined limit is reached by the Coincidence or Single ASCII 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/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 #The Root output), which contains the same information as the ASCII one, but automatically compressed and ready for analysis.

What is the file gateRun.dat ?

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 Δt 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.

The Root output

How to enable this output in your macro ?

If you need to generate the root output file, this can be done by adding the following lines in the macro :

```
/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.

When launching ROOT with the command:

```
root file.root
root [1] TBrowser t
```

you can easily see the content of your ROOT data file.

How to disable this output in your macro ?

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
```

And if you want to disable the whole ROOT output, just do not call it, or use the following command:

```
/gate/output/root/disable
```

How to analyze of Root outputs

You can either plot the variables directly from the browser, or through a macro file (e.g. called analysis.C). In this case, type :

```
root [0] .x analysis.C
```

You may also use the root class called MakeClass :

```
root [0] Coincidences->MakeClass("test");
```

Please consult the ROOT Homepage: http://root.cern.ch/ for more details.

The ROOT Online plotter

Along with standard output for post-processing (such as root, LMF, ecat), 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
/gate/output/plotter/setPlotHeight 250
/gate/output/plotter/setPlotWidth  300
/gate/output/plotter/addPlot hist Ion_decay_time_s
/gate/output/plotter/addPlot hist Positron_Kinetic_Energy_MeV
/gate/output/plotter/addPlot tree Singles comptonPhantom
/gate/output/plotter/addPlot tree Coincidences energy1
/gate/output/plotter/listPlots
```

The command:

```
addPlot hist NAME_of_the_histo
```

plots an histogram previously defined in GATE, and:

```
addPlot tree NAME_of_the_tree  NAME_of_the_variable
```

plots a variable from one of the GATE trees.

The command *setNColumns* sets the number of display windows to be used.

Figure 10.2 presents an example of online plotter, obtained with the above macro.

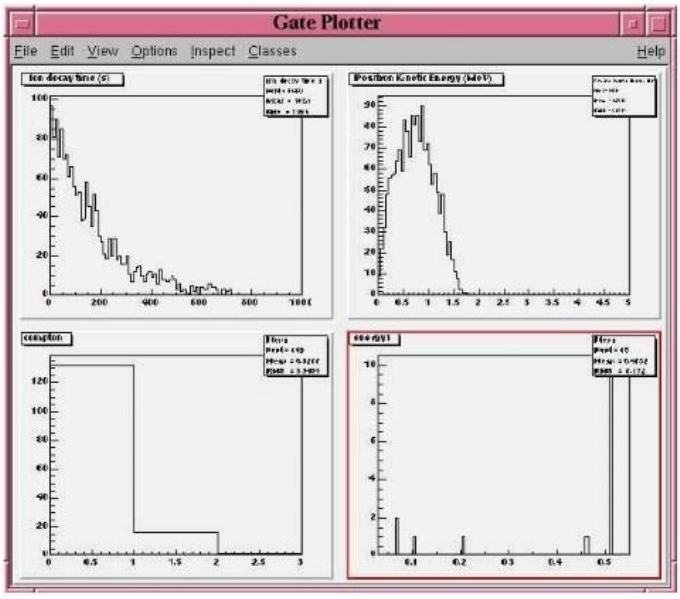


Figure 10.2: The Online Plotter

Interfile output of projection set

Description

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://gamma.wustl.edu/interfile33/interfile33.complete.

When images are acquired in multiple windows (e.g. energy windows, time windows, multiple heads), the images 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
!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 :=
```

Use

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.

Sinogram output

If the ecat system or the ecatAccel system have been selected (see Users Guide V6:Defining a system#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 coincident 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 (Knoxville, TN, U.S.A.), 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;
- an information file in MySinogramFileName.info.

MySinogramFileName.dim contains the minimal information required to read the flat file *MySinogram-FileName.ima'*.

Here is an example with the default setting for the ECAT EXACT HR+ scanner:

```
288 288 1024
-type U16
-dx 1.0
-dy 1.0
-dz 1.0
```

The first line specifies the size of the matrix: 1024 2D sinograms (third coordinate) with 288 radial bins (first coordinate) and 288 azimuthal bins (second coordinate). The second line specifies the format: unsigned short integer. The next three lines specify the size of each bin; there are set arbitrarly to 1.

'*MySinogramFileName.info* describes the ordering of the 2D sinograms in the flat file MySinogram-FileName.ima.

Here is an example with the default setting 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.

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 en- vironment 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 $2 \times N - 1$. 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
```

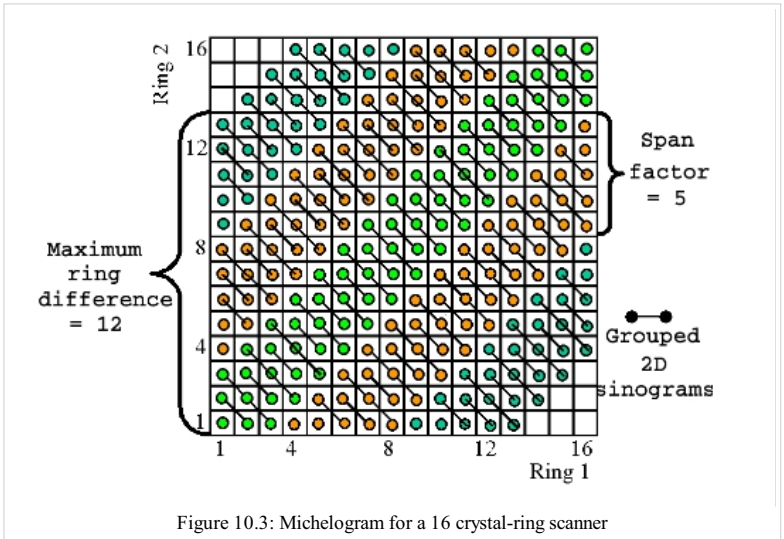


Figure 10.3: Michelogram for a 16 crystal-ring scanner

The *Michelogram* represented in Figure 10.3 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:

```
/gate/application/setTimeSlice 60 s
/gate/application/setTimeStart 0 s
/gate/appication/setTimeStop 300 s
```

5 frames of 60 seconds each will be generated.

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- 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.

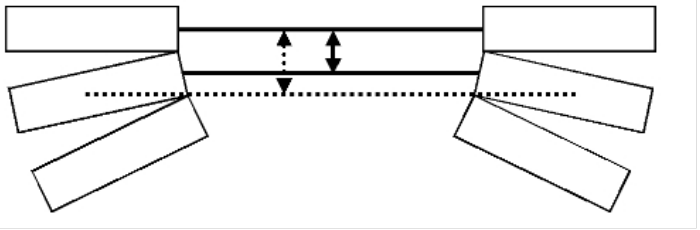


Figure 10.4: Increase of the radial bin size due to the DOI effect.

LMF output

Introduction

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 Users Guide V6: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 deadtimes, 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.

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

Usage

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. Scripting also allows to select the kind of information to be stored. All pieces of information are optional, except time, which makes the ClearPET LMF quite versatile. Table 10.7.2 lists all options and memory requirements that can be stored in the *LMF event record* when using the cylindricalPET system.

The binary output file size depends on its content. It amounts to 11 MB for *1 million* single events stored with their time, energy and detector ID for a small animal PET scanner including about 1500 crystals.

Macros commands (available only once *initialisation* has been performed) used to configure the LMF output are:

```
/gate/output/lmf/enable
```

to enable LMF output.

```
/gate/output/lmf/disable
```

to disable LMF output (but it is disable by default).

```
/gate/output/lmf/setFileName myFirst
```

to set the LMF files name. Here the output files will be myFirst.ccs and myFirst.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.

The following lines must always be included as they appear below with option set to 0:

```
/gate/output/lmf/setSourcePosBool 0
/gate/output/lmf/setNeighbourBool 0
/gate/output/lmf/setNeighbourhoodOrder 0
/gate/output/lmf/setCoincidenceBool 0
```

All information that is not available in real acquisitions is stored in a *GateDigi record* using the command:

```
/gate/output/lmf/setGateDigiBool 1
```

If the option 0 is used, the following commands are ignored

```
/gate/output/lmf/setComptonBool 1
/gate/output/lmf/setComptonDetectorBool 1
```

to store (1) or to not store (0) the number of Compton scattering that occured in a *phantomSD* and 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.

The information regarding the gantry position, translation or rotation speed(s), or the position of the eccentric rotation axis are automatically passed from the macro scripting to the LMF output.

```
/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 submodules 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.

Image CT output

The *imageCT* output is a binary matrix of float numbers that stores the simulated CT image and is produced for each time slice.

The output is enabled by:

```
/gate/output/imageCT/enable
```

The output file name is set by:

```
/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 Users Guide V6:Defining a system#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
```

Raw output

Introduction

The Raw output format should be used in addition to another format. Unlike the other formats available in GATE, it gives direct access to raw images of the source positions of the Singles and/or Coincidences resulting from the digitizer. To store the Coincidences, a coincidence sorter module must be defined in the digitizer (see Users Guide V6:Digitizer and readout parameters#Coincidence sorter) using a PET-like system such as : CPETSystem, CylindricalPETSystem, EcatSystem and EcatAccelSystem (see Users Guide V6:Defining a system). With the SPECTHeadSystem, only the Singles can be stored. For the generic system called ScannerSystem, the Raw output format is currently not available.

The raw images are stored in binary format, and each voxel is represented by a 32 bit float value. This value represents the number of gammas coming from this voxel (for Singles) and back-to-back (for Coincidences). To easily see and analyse the raw images, a software such as *ImageJ* can be used (http://rsb.info.nih.gov/ij/).

Usage

To use the Raw output format, a raw output file name must be set using the following command:

```
/gate/output/raw/setFileName aName
```

Here, 'aName' is the base name and the resulting output files will be *aName*.raw*. If this command is not used, the Raw output format will be automatically

disabled, and other commands related to the raw output format will be ignored.

First, the dimensions of the output images have to be set. To do that, a number of pixels in the X and Y directions and a number of slices (in the Z direction) must be set. Then the dimensions of the voxels along each axis must be set. The following commands illustrate these settings:

```
/gate/output/raw/setNbXPixels 144
/gate/output/raw/setNbYPixels 144
/gate/output/raw/setNbSlices 45
/gate/output/raw/setVoxelSizeX 4. mm
/gate/output/raw/setVoxelSizeY 4. mm
/gate/output/raw/setSliceThickness 4. mm
```

In this example, a stack of 45 images of 144x144 pixels each will be created. Each pixel will be 4 mm x 4 mm, and the slice thickness will be 4 mm. There are no default values for these parameters. If they are not set, a warning will be displayed and the raw output will be disabled. All length units usually available are supported. The volume of interest represented by the stack of images will be automatically centered in the *world* of the simulation.

As mentioned above, the Raw output format can be used to store Singles and/or Coincidences, using the following commands:

```
/gate/output/raw/setSinglesBool 1
/gate/output/raw/setCoincidencesBool 1
```

These are boolean commands where 0 stands for false and 1 for true. In the example above, both Singles and Coincidences will be stored, but in two different files (one with "_S" suffix for Singles and one with "_C" for Coincidences).

When using the SPECTHeadSystem, only the Singles can be stored, and for the other systems, both can be stored but only the Coincidences are stored by default.

The Raw output format makes it possible to store the true and/or scattered events using:

```
/gate/output/raw/setTrueEventsBool 1
/gate/output/raw/setScatterEventsBool 0
```

In the example above, only the true events will be stored. By default, the trues and scattered events are stored. Trues and scattered can be stored separately (in two different files) or not (in a single file) using :

```
/gate/output/raw/setSplitBool 1
```

In this case, the true and scattered events will be stored in two different files (one with "_true" suffix for true events and one with "_scatter" for scattered events), but only if *setTrueEventsBool* and *setScatterEventsBool* are both equal to 1.

Another option can be used to produce different raw files for each run of the simulation:

```
/gate/output/raw/setMultipleRunBool 1
```

In this case, as many sets of raw files as the number of runs set in the simulation will be generated, with "_run1" suffix for the first run, "_run2" for the second, etc ....

Finally, an ascii file can be associated to each run (or only one for all runs if *multipleRunBool* is set to 0). This file summarizes the different parameters chosen for the raw output. It also contains the exact numbers of each type of stored events in the associated raw files (Singles, Coincidences, true, scatter). Moreover, the file indicates the exact numbers and percentages of all the Singles and Coincidences detected in the simulation, including trues, scattered events and randoms, even if the associated source positions are not in the volume of interest defined by the raw images. The command needed to get the ascii files is :

```
/gate/output/raw/setAsciiFileBool 1
```

By default, the ascii files are created. The value should be set to 0 not to create them. As an example, the raw output ascii file produced by the benchmarkPET is :

```
Raw files : benchmarkPET_S.raw,
                  benchmarkPET_C.raw

Type of events stored :
  --> Singles & Coincidences
  --> True & Scatter events in the same file

          --------------
          - Dimensions -
          --------------

Number of pixels in the X dimension : 200
Number of pixels in the Y dimension : 200
Number of slices in the Z dimension : 100

Size of a pixel in the X dimension : 4 mm
Size of a pixel in the Y dimension : 4 mm
Slice thickness in the Z dimension : 4 mm

          --------------
          - Raw images -
          --------------

Singles :
  --> Total number of singles stored     : 6310018
  --> Number of true singles stored      : 3980796
  --> Number of scattered singles stored : 2329222
    --> Number of compton scattered only         : 2297148
      --> Fraction (on scattered) : 98.623 %
    --> Number of rayleigh scattered only        : 16799
      --> Fraction (on scattered) : 0.721228 %
    --> Number of both (comp. & rayl.) scattered : 15275
      --> Fraction (on scattered) : 0.655798 %

Coincidences :
  --> Total number of coincidences stored   : 638408
  --> Number of true coincidences stored    : 312443
  --> Number of scatter coincidences stored : 325965
    --> Number of compton scattered only         : 320165
      --> Fraction (on scattered) : 98.2207 %
    --> Number of rayleigh scattered only        : 2267
      --> Fraction (on scattered) : 0.695473 %
    --> Number of both (comp. & rayl.) scattered : 3533
      --> Fraction (on scattered) : 1.08386 %

          --------------
          - Simulation -
          --------------

Singles : Total number (true,scatter) = 9419776

  --> Number of true singles    : 5901609
    --> Fraction (on total) : 62.6513 %
  --> Number of scatter singles : 3518167
    --> Fraction (on total) : 37.3487 %

  --> Number of compton scattered only         : 3463177
    --> Fraction (on scattered) : 98.437 %
  --> Number of rayleigh scattered only        : 28835
    --> Fraction (on scattered) : 0.819603 %
  --> Number of both (comp. & rayl.) scattered : 26155
    --> Fraction (on scattered) : 0.743427 %

Coincidences : Total number (true,scatter,random) = 706517

  --> Number of true coincidences    : 312629
    --> Fraction (on total) : 44.2493 %
  --> Number of scatter coincidences : 370188
    --> Fraction (on total) : 52.3962 %  |  global scatter fraction : 54.2148 %
  --> Number of random coincidences  : 23700
    --> Fraction (on total) : 3.35448 %

  --> Number of compton scattered only         : 363712
    --> Fraction (on scattered) : 98.2506 %
  --> Number of rayleigh scattered only        : 2270
    --> Fraction (on scattered) : 0.613202 %
  --> Number of both (comp. & rayl.) scattered : 4206
    --> Fraction (on scattered) : 1.13618 %
```

Finally, to set the verbose level for this module, the following command should be used :

```
/gate/output/raw/verbose 3
```

There are six levels of verbosity, from 0 to 5, which give more and more verbose information about the different raw output classes run. By default the verbose level is set to 0 (quiet).

Example

To differentiate the different raw files produced by the simulation, a system of suffixes attached to the base name (given by *setFileName*) is used. Below is an example of the files obtained for a simulation containing 2 runs and based on the following raw output commands :

```
/gate/output/raw/setFileName foobar
/gate/output/raw/setNbXPixels 144
/gate/output/raw/setNbYPixels 144
/gate/output/raw/setNbSlices 45
/gate/output/raw/setVoxelSizeX 4. mm
/gate/output/raw/setVoxelSizeY 4. mm
/gate/output/raw/setSliceThickness 4. mm
/gate/output/raw/setSinglesBool 1
/gate/output/raw/setCoincidencesBool 1
/gate/output/raw/setTrueEventsBool 1
/gate/output/raw/setScatterEventsBool 1
/gate/output/raw/setSplitBool 1
/gate/output/raw/setMultipleRunBool 1
/gate/output/raw/setAsciiFileBool 1
```

The files obtained for such a simulation are :

- foobar_run1_scatter_C.raw *: every scattered Coincidence*

produced during the first run is stored in this file.

- foobar_run1_scatter_S.raw : every scattered Single

produced during the first run is stored in this file.

- foobar_run1_true_C.raw : every true Coincidence

produced during the first run is stored in this file.

- foobar_run1_true_S.raw : every true Single

produced during the first run is stored in this file.

- foobar_run1.txt : the ascii file associated to the first run.
- foobar_run2_scatter_C.raw : every scattered Coincidence

produced during the second run is stored in this file.

- foobar_run2_scatter_S.raw : every scattered Single

produced during the second run is stored in this file.

- foobar_run2_true_C.raw : every true Coincidence

produced during the second run is stored in this file.

- foobar_run2_true_S.raw : every true Single

produced during the second run is stored in this file.

- foobar_run2.txt : the ascii file associated to the second run

Limitations

The random Coincidences cannot be stored.

Users Guide V6:Generating and tracking optical photons

From GATE collaborative documentation wiki

Introduction

Many PET and SPECT designs use scintillators as their detecting medium. For these detectors one may want to investigate the influence of surface finishes and material properties on the detector performance. This chapter explains how Gate can be used to generate and track optical photons in such scintillation detectors in order the investigate, for example, the energy resolution or spatial resolution. To use the optical photon tracking capabilities of Gate, this option has to be switched on first. This can be done using:

```
/gate/physics/optical/enable 1
```

Before discussing how to use the optical photon tracking, it has to be mentioned that there are a few disadvantages to 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 (these will be discussed below), some of which may be difficult to determine.

When optical transport is switched on in Gate, the properties of the materials and surfaces have to be defined. How this is done is discussed in the following two sections. Finally, there is one specific digitizer module for optical photons, which will be discussed in the last section. Much of the information in this chapter is taken from the documentation belonging to Geant4 and the UNIFIED model, discussed in #UNIFIED model, and is described more extensively by Levin (Levin 1996) and Nayar (Nayar 1991).

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** (**Materials.xml** and the later introduced **Surfaces.xml** are, as the extension suggests, XML-files. More information on this file format can be found on the website of the World Wide Web Consortium http://www.w3.org). 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.

Below is an example of how (part of) the *Materials.xml* file could look like:

```
<?xml version="1.0"?>
<materials>
 <material name="LSO">
   <propertiestable>
     <property name="SCINTILLATIONYIELD" value="26000" unit="1/MeV"/>
     <propertyvector name="ABSLENGTH" unit="mm" energyunit="eV">
       <ve energy="6.49" value="0.231"/>
       <ve energy="6.36" value="0.232"/>
       <ve energy="6.23" value="0.321"/>
       <ve energy="6.11" value="0.423"/>
       <ve energy="5.99" energyunit="eV" value="0.483" unit="cm"/>
     </propertyvector>
   </propertiestable>
 </material>
</materials>
```

There are two properties defined for the material LSO: a constant property called **SCINTILLATIONYIELD** and a property vector called **ABSLENGTH**. The property vector contains five elements. The *energyunit* and the *unit* properties specify which unit to use for the energy and value properties respectively for all elements of the vector. However, it is also possible to specify a unit for each vector element separately like in the last element in the example. When the units are omitted, the default units of Geant4 are used.

This example shows only two partially defined properties. There are a lot more properties that can be defined for materials, which will be discussed now. The names of the properties are case sensitive.

Scintillation properties

In order to have scintillation in a material, the first parameter that has to be specified is the **SCINTILLATIONYIELD**, 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 $\mu_N = E$. 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.

$$RESOLUTIONSCAL = \frac{R}{2.35} \cdot \sqrt{E \cdot SCINTILLATIONYIELD},$$

where R is the energy resolution (FWHM) at energy E.

The scintillation follows an exponential decay with two time constants, a fast and a slow one. The ratio between the fast decay and the total decay is given by **YIELDRATIO** and the two time constants by **FASTTIMECONSTANT** and **SLOWTIMECONSTANT**. The emission spectra of both decays are given by the property vectors *FASTCOMPONENT* and *SLOWCOMPONENT* respectively. 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.

All the parameters are summarized in table 12.1

Table 12.1: Material parameters.

Name	Type	Unit
SCINTILLATIONYIELD	const	1/Mev, 1/keV
RESOLUTIONSCALE	const	-
FASTTIMECONSTANT	const	ns, ms, s
SLOWTIMECONSTANT	const	ns, ms, s
YIELDRATIO	const	-
FASTCOMPONENT	vector	-
SLOWCOMPONENT	vector	-
ABSLENGTH	const	m, cm, mm
RINDEX	const	-

Optical properties

To have transport of optical photons inside a material, at least a refractive index should be defined. This can be done in the property vector *RINDEX*. The refractive index can thus be specified as a function of energy. In addition to the refractive index, it is also possible to define an absorption length as a function of optical photon energy. This can be done in the property vector *ABS-LENGTH*. Note that no optical photon will transported in materials for which the refractive index is not defined: all optical photons entering the material will be killed.

Defining surfaces

Surfaces are defined by specifying two volumes. In order to create a surface between two volumes, named **volume1** and **volume2** the following commands should be used:

```
/gate/volume1/surfaces/name   surface1
```

```
/gate/volume1/surfaces/insert volume2
```

Surface parameters.

Name	Type	Unit
SPECULARLOBECONSTANT	vector	-
SPECULARSPIKECONSTANT	vector	-
BACKSCATTERCONSTANT	vector	-
RINDEX	vector	-
REFLECTIVIY	vector	-
EFFICIENCY	vector	-

This creates a surface with the name **surface1** between the two volumes with the names **volume1** and **volume2**. These volumes should have already been defined. The surface between **volume1** and **volume2** is NOT the same surface as that between **volume2** and **volume1**. When there is optical transport in both directions, both surfaces should be created.

Once the surface is created, it should be assigned properties. The surface properties of all the different surfaces are stored in the *Surfaces.xml* file. To load these properties from the file into the newly created surface, the name of the surface properties that should be read must be given to the surface:

```
/gate/volume1/surfaces/surface1/setSurfacename surfacename
```

This will load the surface properties stored under **surfacename** in the Surface.xml file. The entry for this surface could for example look like:

```
<surface name="surfacename" 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="4.08" value="0.95"/>
     <ve energy="1.84" value="0.95"/>
   </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. Finally, 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*. The exact meaning of these values is also discussed in the next section.

The properties table should have the same format as was discussed for the *Materials.xml* file. The possible properties in the property table are shown in table ~\ref{tab_surface_properties}. All of them are vectors and are probabilities and therefore have no unit. These properties are discussed in the next section.

UNIFIED model

The UNIFIED model models the interactions of optical photons at boundaries between two dielectric media. As was already mentioned, this model is discussed extensively in [Levin 1996) and (Nayar 1991). The surface between the two media can be painted on the inside or outside. First the situation in which the surface is not painted will be discussed.

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. The surface normals of the facets are distributed around the average surface normal, following a normal distribution with standard deviation σ_α, which can be specified in degrees in the attribute **sigmaalpha**. 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:

- *SPECULARLOBECONSTANT* gives the probability of specular reflection about the surface normal of the micro facet
- *SPECULARSPIKECONSTANT* gives the probability of specular reflection about the average surface normal
- *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.

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.

Paint

It is possible to cover the surface on the inside or outside with a coating that reflects optical photons using Lambertian reflection. This can be done by setting the attribute *finish* of the surface to *groundbackpainted*, *polishedbackpainted*, *groundfrontpainted* or *polishedfrontpainted*, where *polished' or* ground *have the same meaning as in the previous section.*

When an optical photon reaches the 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.

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.

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*. To use this class, *analysis* has to switched off and *fastanalysis* on:

```
/gate/output/anaysis/disable     1
/gate/output/fastanalysis/enable 1
```

Switching both on has no effect on the results, but only affects the speed of the simulation. There is however a disadvantage to using *fastanalysis*: not all information regarding the singles is kept. The following information is missing:

- SourcePosition (set to -1)
- NPhantomCompton (set to -1)
- NPhantomRayleigh (set to -1)
- ComptonVolumeName (set to "NULL")
- RayleighVolumeName (set to "NULL")
- PhotonID (set to -1)
- PrimaryID (set to -1)
- NCrystalCompton (set to -1)
- NCrystalRayleigh (set to -1)

When this information is needed, the slower (default) *analysis* should be used.

After processing the hits with one of the anaysis routines, the singles should be created from the hits. This is usually done using the *adder*, but for optical photons, a separate adder is defined. The conventional adder only adds hits generated by all particles except optical photons. The *opticaladder* adds all hits generated by optical photons. In this way, it is possible to create two digitizer chains: one containing the usual singles and one containing the singles generated by optical photons.

The *opticaladder* can be added to the digitizer chain using the following command:

```
/gate/digitizer/Singles/insert opticaladder
```

Besides adding only optical photons, there is another important difference with the traditional adder: the *opticaladder* does not add the energies, but counts the number of photons. Therefore, the energy property of the singles contains the number of detected photons and not the deposited energy. Digitizer modules like *threshold* can still be used, but the energy of the threshold should be specified as a number of photons.

Users Guide V6:Architecture of the simulation for radiotherapy

From GATE collaborative documentation wiki

Main rules

GATE simulations are based on the execution of scripted commands (Users Guide V6:Getting started) gathered in macros. We describe here the must important points which are specific to define simulation for radiotherapy applications. The general set-up should be generally divided into 6 steps as follows:

1. Verbosity and Visualization (Users Guide V6:Visualization)
2. Geometry (Users Guide V6:Defining a geometry) - Phantom and material management (Users Guide V6:Voxelized Source and Phantom)
3. Actor and output management (Users Guide V6:Readout parameters for Radiotherapy applications: Actors)
4. Physics (Users Guide V6:Setting up the physics)
5. Sources (Users Guide V6:Activity, source, voxellized phantoms)
6. Experiment - How to run the simulation ? (Users Guide V6: Getting started)

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

The first 4 steps correspond to the PreInit mode of **GEANT4** whereas the last 3 occur after the initialization of the simulation. The first 4 steps are validated by the **GEANT4** command:

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

Once this phase is completed, the sources can be inserted in the setup and the simulation can be run.

Users Guide V6:Readout parameters for Radiotherapy applications: Actors

From GATE collaborative documentation wiki

Readout parameters for Radiotherapy applications: Actor

Actors are tools which allow to interact with the simulation. They can collects information during the simulation, such as energy deposited, 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. There are different type of actors which collect different type of informations, however some commands and behavior are common to all actors. To use selection criteria, it is possible to add filters.

General Purpose

Add an actor

To add an actor, the command is :

```
/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 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 actor, can store some information into a 3D rectangular 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 larger of smaller matrices.

- "attachTo". the main principle is that 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 are 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 occur, 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 stored 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 imagethe 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
```

Actor list

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
```

Dose measurement

This actor 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 enable 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]/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]/normaliseDose   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), (.hdr/.img). The root file works only for 1D distribution.

```
/gate/actor/addActor DoseActor              MyActor
/gate/actor/MyActor/save                    MyOutputFile.hdr
/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
/gate/actor/MyActor/enableUncertaintyEdep   true
/gate/actor/MyActor/enableSquaredEdep       false
/gate/actor/MyActor/enableDose              false
/gate/actor/MyActor/normaliseDose           false
```

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. The script is called with the command:

```
/gate/actor/[Actor Name]/save  MyScript
```

It is possible to save all the others actor before stopping the simulation with the command:

```
/gate/actor/[Actor Name]/saveAllActors  true
```

Example:

```
/gate/actor/addActor  StopOnScriptActor  MyActor
/gate/actor/MyActor/save  MyScript
/gate/actor/MyActor/saveAllActors  true
```

Track length

This actor store 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 four histograms: the initial kinetic energy of each track (energySpectrum), the energy deposition per event (edepHisto), the energy deposition per track (edepTrackHisto), the energy loss per track (eLossHisto). These histograms are stored in a root file. They takes into account the weight of particles. They are three commands to define the boundaries and the binning of the energy spectrum and three commands to define the boundaries and the binning of the energy loss histograms (edepHisto, edepTrackHisto, eLossHisto).

```
/gate/actor/addActor  EnergySpectrumActor  MyActor
/gate/actor/MyActor/save  MyOutputFile.root
/gate/actor/MyActor/energySpectrum/setEmin 0 eV
/gate/actor/MyActor/energySpectrum/setEmax 10 GeV
/gate/actor/MyActor/energySpectrum/setNumberOfBins 200
/gate/actor/MyActor/energyLossHisto/setEmin 0 eV
/gate/actor/MyActor/energyLossHisto/setEmax 15 MeV
/gate/actor/MyActor/energyLossHisto/setNumberOfBins 120
```

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), (.hdr/.img). The root file works only for 1D distribution.

```
/gate/actor/addActor ProductionAndStoppingActor  MyActor
/gate/actor/MyActor/save          MyOutputFile.hdr
/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 recomanded state for 'stepHitType' is 'post'

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 exept alpha, deuton and triton, there is the key word 'GenericIon'.

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. IDs are reset at each event. 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
```

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
/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/setDirection 1 0 0
```

Users Guide V6:Phase space approach

From GATE collaborative documentation wiki

How to create an use a phase space with Gate ?

This actor records informations about particles entering the volume which the actor is attached. They are two file type for the output: root file (.root) and IAEA file (.IAEAphsp and .IAEAheader). Informations recorded are the name of the particle, the kinetic energy, the position along the three axis, the direction along the three axis, the weight. In a IAEA file, the particle are 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 more informations: the name of the volume where the particle was produced and the name of the process which produced the particle. It is posssible to disable some informations in the phase space file:

```
/gate/actor/source/enableEkine              false
/gate/actor/source/enableXPosition          false
/gate/actor/source/enableYPosition          false
/gate/actor/source/enableZPosition          false
/gate/actor/source/enableXDirection         false
/gate/actor/source/enableYDirection         false
/gate/actor/source/enableZDirection         false
/gate/actor/source/enableProductionVolume   false
/gate/actor/source/enableProductionProcess  false
/gate/actor/source/enableParticleName       false
/gate/actor/source/enableWeight             false
```

By default the frame use 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, there is the command:

```
/gate/actor/source/useVolumeFrame
```

```
/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
```