GATE - Geant4 Application for Tomographic Emission:
a simulation toolkit for PET and SPECT

OpenGATE Collaboration
http://www.opengatecollaboration.org

Version 5.0.0
10 June 2009
Authors

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

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


Special Thanks : Geant4 Collaboration

The GATE mailing list :
You are encouraged to participate in the dialog and post your suggestion or even implementation on the gate-users mailing list, the GATE mailing list for users.
You can subscribe to the gate-users mailing list, by registering at the GATE web site :
http://www.opengatecollaboration.org
If you have a question, it is very likely that it has been asked, answered, and 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 talk.
Preface

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 [1]. On the other hand, accurate and versatile simulation codes such as GEANT3 [2], EGS4 [3], MCNP [4], and recently GEANT4 [5] 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 Tomographic Emission [6, 7, 8, 9], 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 [10] with the objective to provide the academic community with a free software, general-purpose, GEANT4-based simulation platform for emission tomography. The collaboration comprised of 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 [11, 12]. Particular attention was paid to provide meaningful documentation with the simulation software package, including installation and user’s guides, online source code documentation by doxygen [13], 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 listed by order of joining the collaboration:

- Ecole Polytechnique Fédérale de Lausanne (LPHE-IPEP)
- University of Clermont-Ferrand (LPC)
- University of Ghent (ELIS)
- CHU Pitié-Salpêtrière (U494 INSERM), Paris
- Vrije Universiteit Brussel (IIHE)
- Centre d’Exploration et de Recherche Médicales par Emission de Positons (CERMEP), Lyon
- Service Hospitalier Frédéric Joliot (SHFJ), CEA-Orsay
- CHU Nantes (U601 INSERM)
- Sungkyunkwan University School of Medicine (DMN), Seoul
- University Claude Bernard (IPNL), Lyon
- University Louis Pasteur (IRES), Strasbourg
- University Joseph Fourier (LPSC), Grenoble
- Forschungszentrum-Juelich (IME)
- University of Massachusetts Medical School (Division of Nuclear Medicine), Worcester, MA
- CHU Morvan (LATIM, U650 INSERM), Brest
- University of California (Crump Institute for Molecular Imaging), Los Angeles
- University of Toronto (CAMH)
- IRPHU, CEA-Saclay
- Memorial Sloan-Kettering Cancer Center (Department of Medical Physics), New York
- University of Athens (IASA)
- Delft University of Technology (IRI)
Overview

GATE combines the advantages of the GEANT4 simulation toolkit well-validated physics models, sophisticated geometry description, and powerful visualization and 3D rendering tools with original features specific to emission tomography. It consists of several hundred C++ classes. Mechanisms used to manage time, geometry, and radioactive sources form a core layer of C++ classes close to the GEANT4 kernel [Fig. 1]. 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](image)

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.
# Contents

<table>
<thead>
<tr>
<th>Preface</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview</td>
<td>vii</td>
</tr>
</tbody>
</table>

## 1 Getting started

1.1 General simulation architecture .................................................. 5
1.2 The user interface: a macro language ............................................. 6
1.3 First step: Defining a scanner geometry ......................................... 7
1.4 Second step: Defining a phantom geometry ...................................... 10
1.5 Third step: Setting-up the physics processes ................................... 11
1.6 Fourth step: Initialization ........................................................... 12
1.7 Fifth step: Setting-up the digitizer ............................................... 13
1.8 Sixth step: Setting-up the source ............................................... 14
1.9 Seventh step: Defining data outputs ............................................. 14
1.10 Eighth step: Starting an acquisition ........................................... 15

## 2 Visualization

2.1 Introduction ................................................................. 17
2.2 Important Hints ............................................................. 17
2.3 Command Lines ...................................................................... 17

## 3 Define a geometry

3.1 The world .............................................................................. 21
3.2 Creating a volume .................................................................. 22
3.3 Repeating a volume ............................................................... 31
3.4 Placing a volume .................................................................... 39
3.5 Moving a volume ..................................................................... 43
3.6 Updating the geometry ............................................................ 46

## 4 Define a system

4.1 Definition ............................................................................. 49
4.2 Choice of the system ............................................................. 50
4.3 How to connect the geometry to a system ...................................... 51
4.4 Different types of systems ....................................................... 54
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Attach the sensitive detectors</td>
<td>81</td>
</tr>
<tr>
<td>5.1</td>
<td>General purpose</td>
<td>81</td>
</tr>
<tr>
<td>5.2</td>
<td>Two types of sensitive detector</td>
<td>82</td>
</tr>
<tr>
<td>6</td>
<td>Set up the physics</td>
<td>85</td>
</tr>
<tr>
<td>6.1</td>
<td>Physics process</td>
<td>85</td>
</tr>
<tr>
<td>6.2</td>
<td>Setting the cut</td>
<td>90</td>
</tr>
<tr>
<td>6.3</td>
<td>Initialize the physics</td>
<td>91</td>
</tr>
<tr>
<td>7</td>
<td>Activity, source, voxellized phantom</td>
<td>93</td>
</tr>
<tr>
<td>7.1</td>
<td>The GPS (General Particle Source)</td>
<td>93</td>
</tr>
<tr>
<td>7.2</td>
<td>Create a source</td>
<td>93</td>
</tr>
<tr>
<td>7.3</td>
<td>Voxellized phantom</td>
<td>101</td>
</tr>
<tr>
<td>7.4</td>
<td>Real time motion management for voxellized source and phantom</td>
<td>110</td>
</tr>
<tr>
<td>8</td>
<td>Digitizer and readout parameters</td>
<td>115</td>
</tr>
<tr>
<td>8.1</td>
<td>General Purpose</td>
<td>115</td>
</tr>
<tr>
<td>8.2</td>
<td>Digitizer modules</td>
<td>117</td>
</tr>
<tr>
<td>8.3</td>
<td>Multiple processor chains</td>
<td>134</td>
</tr>
<tr>
<td>8.4</td>
<td>Coincidence sorter</td>
<td>135</td>
</tr>
<tr>
<td>8.5</td>
<td>Multiple coincidence sorters</td>
<td>137</td>
</tr>
<tr>
<td>8.6</td>
<td>Coincidence treatment and filtering</td>
<td>141</td>
</tr>
<tr>
<td>8.7</td>
<td>A complete example</td>
<td>142</td>
</tr>
<tr>
<td>8.8</td>
<td>Digitizer optimization</td>
<td>145</td>
</tr>
<tr>
<td>8.9</td>
<td>Angular Response Functions (ARFs) to speed-up simulations</td>
<td>147</td>
</tr>
<tr>
<td>9</td>
<td>Architecture of the simulation</td>
<td>151</td>
</tr>
<tr>
<td>9.1</td>
<td>Principal rules</td>
<td>151</td>
</tr>
<tr>
<td>9.2</td>
<td>Random generator</td>
<td>154</td>
</tr>
<tr>
<td>9.3</td>
<td>Example of a PET scanner</td>
<td>155</td>
</tr>
<tr>
<td>10</td>
<td>Data output</td>
<td>163</td>
</tr>
<tr>
<td>10.1</td>
<td>The ASCII output</td>
<td>163</td>
</tr>
<tr>
<td>10.2</td>
<td>The Root output</td>
<td>168</td>
</tr>
<tr>
<td>10.3</td>
<td>The Online plotter</td>
<td>169</td>
</tr>
<tr>
<td>10.4</td>
<td>Interfile output</td>
<td>170</td>
</tr>
<tr>
<td>10.5</td>
<td>Sinogram output</td>
<td>173</td>
</tr>
<tr>
<td>10.6</td>
<td>ECAT7 output</td>
<td>175</td>
</tr>
<tr>
<td>10.7</td>
<td>LMF output</td>
<td>177</td>
</tr>
<tr>
<td>10.8</td>
<td>imageCT output</td>
<td>180</td>
</tr>
<tr>
<td>10.9</td>
<td>Raw output</td>
<td>180</td>
</tr>
<tr>
<td>11</td>
<td>Materials</td>
<td>187</td>
</tr>
<tr>
<td>11.1</td>
<td>The Gate material database</td>
<td>187</td>
</tr>
<tr>
<td>11.2</td>
<td>Modifying the Gate material database</td>
<td>188</td>
</tr>
</tbody>
</table>
12 Generating and tracking optical photons

12.1 Introduction ................................................................. 193
12.2 Defining material properties ........................................... 193
12.3 Defining surfaces ......................................................... 195
12.4 Digitizer ................................................................. 198

13 How to run Gate

13.1 Interactive mode ............................................................ 201
13.2 Running parameterized macros ......................................... 203
13.3 Batch mode .............................................................. 204
13.4 Cluster mode ............................................................ 205
13.5 How to launch $DigiGate$ .................................................. 209
13.6 How to separate the phantom and detector tracking - Phase space approach ... 210
Chapter 1

Getting started

This paragraph is an overview of the main steps one must go through to perform a simulation using Gate. It is presented in the form of a simple example and the user is encouraged to try out the example, in an interactive mode of Gate, while reading this chapter. A more detailed description of the different steps is given in the following chapters of this user’s guide.

The use of Gate does not require any C++ programming. This is due 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 chapter 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.

1.1 General simulation architecture

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.
1.2 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 launched, 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 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 state all commands describing the different sections of a simulation in the right order. Usually these sections 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 cut down 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 chapter ??). Cutting down 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 launch Gate and can execute all the commands proposed in this chapter, line by line.
1.3 First step: Defining a scanner geometry

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

```plaintext
# VISUALIZATION
/vis/open OGLSX
/vis/viewer/reset
/vis/viewer/viewpointThetaPhi 60 60
/vis/viewer/zoom 1
/vis/viewer/style surface
/vis/drawVolume
/tracking/storeTrajectory 1
/vis/scene/EndOfEventAction accumulate
/vis/viewer/update
/gate/geometry/enableAutoUpdate
```

Figure 1.1: World volume

The different visualization tools and their related commands are discussed in more detail in chapter 2.

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 ??) which fixes the experimental framework of the simulation. All Gate commands related to the construction of the geometry are described in detail in Chapter 3.

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 in this chapter simulates a system that fits into a box of $40 \times 40 \times 40$ cm$^3$. Thus, the world volume may be defined as follows.

```plaintext
# WORLD
/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.

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

The name vol_name of the first daughter of the world has a specific meaning and name. It fixes the type of scanner to be simulated. Chapter 4 gives the specifics of each type of scanner, also called system (3 types for PET and 2 types for SPECT). 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.

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
1.3 First step: Defining a scanner geometry

![Figure 1.2: Cylindrical scanner](image1)

of 86 mm. The length of the cylinder is fixed to 18 mm. The last command line forces the visualization to be in wireframe.

It is noted that at any time, the user can list all the possible commands. For example, the command line for the visualization commands is:

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

Let’s assume that the scanner is made of 30 blocks (box1), each block containing $8 \times 8$ LSO crystals (box2). The following command lines construct this scanner (see chapter 3 to find a detailed explanation of these commands).

First of all, the geometry of each block needs to be defined as the daughter of the system (here cylindricalPET system).

![Figure 1.3: first level of the scanner](image2)

Once the block is created (fig 1.3), the crystal can be defined as a daughter of the block (fig 1.4). The zoom command line in the script allows the user to zoom the geometry and the panTo command is used to translate the viewer window in 60 mm in horizontal and 40 mm in vertical directions (the default is the origin of the world (0,0)).

In order 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).
1.3 First step: Defining a scanner geometry

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 the chapter 4 and 5 respectively.

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

```
# 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
```

The names rsector and module are fixed names and correspond to the first and the second
1.4 Second step: Defining a phantom geometry

The volume to be imaged 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 media of the phantom. In order to retrieve informations 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
```

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. Chapter 3 describes the movement of volumes in detail.

### Figure 1.6: complete ring of 30 block detectors

Levels of the cylindrical PET system (see chapter 4).

In order to save the hits (see chapter 8) 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
```

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. Chapter 3 describes the movement of volumes in detail.

1.4 Second step: Defining a phantom geometry

The volume to be imaged 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 media of the phantom. In order to retrieve informations 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
```
1.5 Third step: Setting-up the physics processes

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 discussed more fully in chapter 5.

1.5 Third step: Setting-up the physics processes

After the description of the volumes and the corresponding sensitive detectors, one needs to specify which interaction processes are to be included in the simulation. Three steps are expected to be defined by the user:

- specify type(s) of particle to be transported
- specify the physics processes to be taken into account
- set the production cuts

Gate uses the GEANT4 models for physical processes. The user has to choose among these processes for each particle. For photons, the following processes can be modeled; Photo-electric effect Compton and Rayleigh scattering and pair production. For electrons, the following processes can be modeled; ionization, Moller scattering and Bremsstrahlung. For the electron-positron annihilation, the gamma pair non-collinearity is modeled.

For each type of interaction, the user can choose between two models or ignore the interaction completely;
1.6 Fourth step: Initialization

- standard: use standard model (transport of photons and electrons down to 10 keV, Raleigh scattering not modeled)

- lowenergy: use low-energy model (transport of photons and electrons down to 250 eV, Raleigh scattering included)

- inactive: do not simulate the interaction

The default Gate physics is;

- for gamma particle, low-energy models used for all the processes
- for electron particles the standard models are used.

The following is an arbitrary example of a physics list:

```
/gate/physics/gamma/selectPhotoElectric lowenergy
/gate/physics/gamma/selectCompton lowenergy
/gate/physics/gamma/selectGammaConversion standard
/gate/physics/gamma/selectRayleigh inactive
```

Gate allows to set three cuts;

- range cut for the electrons
- Energy cut for X-rays
- Energy cut for delta rays

For the most accurate results, one should use low cuts, or even no cuts at all (a low cut always exist for instance to avoid infrared divergence in the delta ray production, see GEANT4 documentation) so that all secondary particles are produced and tracked. In this case the physics is very accurate, but the simulation is slow. For fast simulation, one should use large cut values in order to minimize the time spent on the tracking of secondary particles.

```
# High cuts for fast simulation
/gate/physics/setElectronCut 1000. km
/gate/physics/setXRayCut 1. GeV
/gate/physics/setDeltaRayCut 1. GeV
```

The details of the interactions processes and cuts available in Gate are described in chap. 6.

1.6 Fourth step: Initialization

When the preceding steps, corresponding to the PreInit mode of GEANT4, are set, the simulation should be initialized using the following command;

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

The principal effect of this initialization is that the cross section tables are computed. After this step, the physics list can not be modified any more and new volumes can not be inserted into the geometry.
1.7 Fifth step: Setting-up the digitizer

The basic output of Gate is a *hit* collection in which informations such as the position, the time and the energy of each hit are stored. The history of a particle is thus saved 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 chain. 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
- 1st 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, a new module can be added.

```
# 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, a resolution of $\sigma_E = 19\%$ at 511 KeV is set. 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.

```
# COINCIDENCE SORTER
/gate/digitizer/Coincidences/setWindow 10. ns
```

Additional *digitizer* modules are available in Gate and can be found in chapter 8.
1.8 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 as well as 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 reproduce realistic acquisitions. For a complete description of all available functions, see chap. 7.

In the current example, the source is a line source with an activity of 1 MBq. 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 noncollinearity of its two gammas can also be taken into account).

```
# SOURCE
/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
```

1.9 Seventh step: Defining data outputs

By default, the data output formats for all the systems used by Gate are ASCII and ROOT as described in the following command lines
1.10 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, using the commands `setTimeStart` and `setTimeStop`. In addition, Gate needs a time slice parameter (`setTimeSlice`) which defines the 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.

```
gate/application/setTimeSlice 1. s
/gate/application/setTimeStart 0. s
/gate/application/setTimeStop 1. s
# START the ACQUISITION
/gate/application/startDAQ
```

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

\[
N_{\text{run}} = \frac{\text{setTimeStop}-\text{setTimeStart}}{\text{setTimeSlice}}
\]
1.10 Eighth step: Starting an acquisition

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`. 
Chapter 2

Visualization

2.1 Introduction

The visualization options in GATE provide the same functionality as provided in GEANT4. Most options in GEANT4 to visualize detector geometry, particle trajectories, tracking steps, etc. are available also in GATE. The graphics systems that can be selected in GATE are: DAWNFILE, VRMLFILE (version 1 and 2) and OpenGL in stored and immediate mode, with OpenGL required as an external library. Most of the libraries are available in free implementations.

2.2 Important Hints

When loading digital images in combination with OpenGL, instead of the frequently used Stored-X viewer, the OpenGL Immediate-X viewer is recommended.

Concerning DAWN and VRMLVIEW it should be noted, that complicated geometries may take very long to get rendered, like a huge number of crystals in a cylindrical PET system. To decrease the file size and to speed up the visualization the following option may be used:

/gate/crystal/vis/setVisible 0

Now the individual crystals are not rendered, instead they are shown as a wireframe.

2.3 Command Lines

Basic command lines as they are used in most macro files. These commands are provided by the GEANT4-package.

2.3.1 Visualization with OpenGL

# V I E W E R # We open an OpenGL Stored-X viewer which is the #standard viewer

/vis/open OGLSX

# or we open an OpenGL Immediate-X viewer used for digital images
2.3 Command Lines

/vis/open OGLIX

# define 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 they are accumulated. Detector remains or is redrawn.
/vis/scene/endOfEventAction accumulate

The following commands implement additional options for application 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 also body; cylindrical; crystal; ecat; module; scource; scanner can be used in macros.

2.3.2 Visualization with DAWN

Instead of real-time visualization based on OpenGL, storing images in a file (mostly eps) for further processing is often wanted. DAWN offers many of these features.

The package can be downloaded from the Internet and installed following the instruction given therein.

http://geant4.kek.jp/~tanaka/src/dawn_3_85e.taz

To use DAWN and DAWNFILE in your macro you have to add the following line, while omitting the statement for OpenGL.
2.3 Command Lines

/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

Note, that specific lines have to be added to the environment of your shell in order to have access to DAWN inside GATE (for example 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

2.3.3 Visualization with VRML

Sometimes it may be helpful to check a geometry setup by interactively manipulating the visualized scene. These features are offered by using 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/

Note, that you have to add a line to your environment:

setenv G4VRMLFILE_VIEWER vrmlview

For using this option in Gate, the following line has to be added to a macro and the corresponding line for OpenGl omitted. During processing in GATE, a file is written with the extension wrl.

/vis/open VRML2FILE

Again, the appropriate environment, as shown for the c-shell, has to be setup:
2.3 Command Lines

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
Chapter 3

Define a geometry

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

3.1 The world

3.1.1 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.

3.1.2 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.

3.1.3 Description and modification

The world volume has been defined with 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
```

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

3.2.1 Generality - Tree creation

When a volume is created with GATE, it automatically appears in the GATE tree (see chapter 1). 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 Sections 3.2.4, 3.3, 3.4 and 3.5.
3.2 Creating a volume

3.2.2 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:

\[ \text{/units/list} \]

inside the GATE environment (see Chapter 1).

3.2.3 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:

\[ \text{/gate/world/daughters/insert 3axes} \]

Figure 3.2 shows the three-axis system.

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

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

\[ \text{/gate/Name_Volume/daughters/insert axe*} \]

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.
### 3.2 Creating 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;

### Table 3.1: List of units available in GATE and corresponding abbreviations

<table>
<thead>
<tr>
<th>LENGTH</th>
<th>SURFACE</th>
<th>VOLUME</th>
<th>ANGLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>parsec pc</td>
<td></td>
<td></td>
<td>radian rad</td>
</tr>
<tr>
<td>kilometer km</td>
<td>kilometer² km²</td>
<td>kilometer³ km³</td>
<td>milliradian mrad</td>
</tr>
<tr>
<td>meter m</td>
<td>meter² m²</td>
<td>meter³ m³</td>
<td>steradian sr</td>
</tr>
<tr>
<td>centimeter cm</td>
<td>centimeter² cm²</td>
<td>centimeter³ cm³</td>
<td>degree deg</td>
</tr>
<tr>
<td>millimeter mm</td>
<td>millimeter² mm²</td>
<td>millimeter³ mm³</td>
<td></td>
</tr>
<tr>
<td>micrometer mum</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>nanometer nm</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>angstrom Ang</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fermi fm</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TIME</th>
<th>SPEED</th>
<th>ANGULAR SPEED</th>
<th>ENERGY</th>
</tr>
</thead>
<tbody>
<tr>
<td>second s</td>
<td>m/s</td>
<td>radian/s rad/s</td>
<td>electronvolt eV</td>
</tr>
<tr>
<td>millisecond ms</td>
<td>centimeter/s cm/s</td>
<td>degree/s deg/s</td>
<td>kiloelectronvolt keV</td>
</tr>
<tr>
<td>microsecond μs</td>
<td>millimeter/s mm/s</td>
<td>degree/s deg/s</td>
<td>megaelectronvolt MeV</td>
</tr>
<tr>
<td>nanosecond ns</td>
<td>meter/min m/min</td>
<td>radian/min rad/min</td>
<td>gigaelectronvolt GeV</td>
</tr>
<tr>
<td>picosecond ps</td>
<td>centimeter/min cm/min</td>
<td>degree/min deg/min</td>
<td>teraelectronvolt TeV</td>
</tr>
<tr>
<td>millimeter/min</td>
<td>m/min</td>
<td>rotation/s rot/s</td>
<td>petaelectronvolt PeV</td>
</tr>
<tr>
<td>meter/h</td>
<td>m/h</td>
<td>radian/min rad/min</td>
<td>joule J</td>
</tr>
<tr>
<td>centimeter/h cm/h</td>
<td>degree/min deg/min</td>
<td></td>
<td></td>
</tr>
<tr>
<td>millimeter/h mm/h</td>
<td>rotation/min rot/min</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rotation/h rot/h</td>
<td>radian/h rad/h</td>
<td></td>
<td></td>
</tr>
<tr>
<td>degree/h deg/h</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ACTIVITY - DOSE</th>
<th>AMOUNT OF SUBSTANCE</th>
<th>MASS</th>
<th>VOLUMIC MASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>becquerel Bq</td>
<td>mole mol</td>
<td>milligram mg</td>
<td>g/cm³ g/cm³</td>
</tr>
<tr>
<td>curie Ci</td>
<td>gram g</td>
<td>mg/cm³</td>
<td></td>
</tr>
<tr>
<td>gray Gy</td>
<td>kilogram kg</td>
<td>kg/m³</td>
<td>kg/m³</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ELECTRIC CHARGE</th>
<th>ELECTRIC CURRENT</th>
<th>ELECTRIC POTENTIAL</th>
<th>MAGNETIC FLUX - MAGNETIC FLUX DENSITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>eplus e+</td>
<td>ampere A</td>
<td>volt V</td>
<td>weber Wb</td>
</tr>
<tr>
<td>coulomb C</td>
<td>milliamperes mA</td>
<td>kilovolt kV</td>
<td>tesla T</td>
</tr>
<tr>
<td>microampere μA</td>
<td></td>
<td>megavolt MV</td>
<td>gauss G</td>
</tr>
<tr>
<td>nanoampere nA</td>
<td></td>
<td>kilogauss kG</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TEMPERATURE</th>
<th>FORCE - PRESSURE</th>
<th>POWER</th>
<th>FREQUENCY</th>
</tr>
</thead>
<tbody>
<tr>
<td>kelvin K</td>
<td>newton N</td>
<td>watt W</td>
<td>hertz Hz</td>
</tr>
<tr>
<td>pascal Pa</td>
<td>kilohertz kHz</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bar</td>
<td>megahertz MHz</td>
<td></td>
<td></td>
</tr>
<tr>
<td>atmosphere atm</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**3.2.4 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;
3.2 Creating a volume

- 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:

\[
\text{/gate/mother\_Volume\_Name/daughters/name Volume\_Name}
\]

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 6 GATE systems (see chapter 4) 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

\[
\text{/gate/Volume\_Name/}
\]

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:

\[
\text{/gate/world/daughters/info}
\]

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

\[
\text{/gate/daughter\_Volume\_Name/daughters/insert Volume\_shape}
\]

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, ellipsio 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:

\[
\text{/gate/Volume\_Name/daughters/list}
\]

- Example

\[
\text{/gate/world/daughters/name Phantom}
\]

\[
\text{/gate/world/daughters/insert box}
\]

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

<table>
<thead>
<tr>
<th>BOX</th>
<th>TRPD</th>
</tr>
</thead>
<tbody>
<tr>
<td>setXLength</td>
<td>Set the length of the box along the X axis</td>
</tr>
<tr>
<td>setYLength</td>
<td>Set the length of the box along the Y axis</td>
</tr>
<tr>
<td>setZLength</td>
<td>Set the length of the box along the Z axis</td>
</tr>
</tbody>
</table>

SPHERE

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 | setZBoxLength | Set half length along Z of the extruded box |
setPhiStart | Set the start phi angle | setXBoxLength | Set half length along X of the extruded box |
setDeltaPhi | Set the phi angular span (2PI for full sphere) | setYBoxLength | Set half length along Y 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

setRmin | Set the internal radius of the cylinder (0 for full cylinder) | setZBoxPos | Set center position Z of the box |
setRmax | Set the external radius of the cylinder | setBoxPos | Set Dx dimension of the parallelepiped |
setHeight | Set the height of the cylinder | setBoxLength | Set Dz dimension of the parallelepiped |
setPhiStart | Set the start phi angle | setProfile | Set vectors of z, rInner, rOuter positions |
setDeltaPhi | Set the phi angular span (2PI for full cylinder) | setAlpha | Set Alpha angle |

CONE

setRmin1 | Set the internal radius of one side of the cone (0 for full cone) | setTheta | Set Theta angle |
setRmax1 | Set the external radius of one side of the cone | POLYCONEX | Set the start phi angle |
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 | Set the radius of the hexagon |
setDeltaPhi | Set the phi angular span (2PI for full cone) | setRadius | Set the radius of the hexagon |

ELLIPSO

setLong | Set the long axis length of the ellipse | WEDGE | Set the length of the shorter side of the wedge in the X direction |
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 |

Table 3.2: Commands of the sub-tree geometry for different shapes
3.2 Creating a volume

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

```plaintext
/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 shall 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 Chapter 11). The following command fills the volume *Volume_Name* with a material called *Material*:

```plaintext
/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>
<thead>
<tr>
<th>Command</th>
<th>Action</th>
<th>Argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>setColor</td>
<td>Selects the color for the current volume</td>
<td>white, gray, black, red, green, blue, cyan, magenta and yellow</td>
</tr>
<tr>
<td>setVisible</td>
<td>Shows or hides the current volume</td>
<td></td>
</tr>
<tr>
<td>setDaughtersInvisible</td>
<td>Shows or hides the current volume daughters</td>
<td></td>
</tr>
<tr>
<td>setLineStyle</td>
<td>Sets the current volume line-style</td>
<td>dashed, dotted and unbroken</td>
</tr>
<tr>
<td>setLineWidth</td>
<td>Sets the current volume line-width</td>
<td></td>
</tr>
<tr>
<td>forceSolid</td>
<td>Forces solid display for the current volume</td>
<td></td>
</tr>
<tr>
<td>forceWireframe</td>
<td>Forces wireframe display for the current volume</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.3: List of commands of the GATE sub-tree geometry

These commands can be found in the tree

```
/gate/Volume_Name/vis.
```
3.2 Creating a volume

- 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**

1) **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 which its shape 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.
3.2 Creating a volume

The new volume crystal is filled with NaI.

The new volume crystal is colored in yellow.

with this previous command, the parameters associated with the crystal volume are listed.

The crystal volume is disabled.

2) 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
3.2 Creating a volume

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.

![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.](image)

3) how to build a "Wedge" volume

Gate provides the class *GateTrapCreator* to create and insert trapezoidal volumes into the geometry. In order to create a trapezoid, the user needs to specify eleven parameters (besides its name and material) which makes it rather difficult to use.

In order to have "slanted" crystals, we have derived from *G4Trap* a new class called *GateWedgeCreator* which 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, if you want to insert a wedge crystal as a daughter of a module, you would write:

```
/gate/module/daughters/name wedge0
/gate/module/daughters-insert wedge
```
3.3 Repeating a volume

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

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

/gate/Name_Volume/repeaters/info

3.3.1 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

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

The "true" option yields the centering of 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](image)

3.3.2 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 shall 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
```
3.3 Repeating a volume

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).

\[
\frac{\text{AngularSpan} - \text{FirstAngle}}{\text{RepeatNumber} - 1} = \text{Angular Pitch}
\]

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

/gate/Name_Volume/ring/setModuloNumber M

When the volume auto-rotation option is enabled, this means that 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 \( M_{\text{ModuloNumber}} \) elements, the shift values are defined as \( Z_{\text{shift}_i} \equiv Z_{\text{shift}_j} \) where:

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

To set a shift and to fix 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(s) value(s) 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 & al." and is fully described in the LMF output, in particular in the ASCII header file entry:

"z shift sector j mod M_{\text{ModuloNumber}}: Z_{\text{shift}_j} units".

Here \( j^1 \) stands for the \( n^{th} \) object being shifted each \( M_{\text{ModuloNumber}} \) 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

\(^1\)actually \( j \) is numbered starting here at 0
3.3 Repeating a volume

starting at 0 degree and finishing at 90 degree (a quarter of ring) will not be supported by the LMF output. See also related class GateToLMF for LMF output.

- Example 1

![Figure 3.6: Illustration of the application of the auto-rotation option](image)

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](image)

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

/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
3.3 Repeating a volume

The hole volume is repeated 10 times around the Y axis. The application of this ring repeater is illustrated in figure 3.8.

- Example 2

![Figure 3.8: Illustration of the application of the ring repeater](image)

The rsector volume is repeated 20 times along a ring. The sequence length is 2, with the first and the second volume shifted by \(-3500 \, \mu m\) and \(3500 \, \mu m\) respectively.

The rsector volume could also include itself several volumes, each of them being duplicated all together, which is illustrated in figure 3.9.

### 3.3.3 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 \(N_x, N_y\) and \(N_z\) the volume \(Name\_Volume\) has to be repeated along the X, Y and Z axes respectively, use:

```
/gate/hole/cubicArray/setRepeatNumberX Nx
/gate/hole/cubicArray/setRepeatNumberY Ny
/gate/hole/cubicArray/setRepeatNumberZ Nz
```
3.3 Repeating a volume

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.

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 (see paragraph 3.3.1). 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.
3.3 Repeating a volume

3.3.4 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 enables 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
```
3.3 Repeating a volume

Figure 3.11: Illustration of the application of the cubic array repeater

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

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

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

### 3.3.5 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 Ecatt 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 the following command:

```
/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
```
3.4 Placing a volume

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

**Example 3.12:**

```plaintext
/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 in 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 of 20 deg. The sphere defined here as a 25 cm radius.

### 3.4 Placing a volume

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

```plaintext
/placement/
```

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

#### 3.4.1 Translation

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

```plaintext
/gate/Name_Volume/placement/setTranslation x. 0. 0. cm
```
3.4 Placing a volume

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:
\[
\text{/gate/Name_Volume/placement/setPhiOfTranslation N deg}
\]

To set the Theta angle (with regard to the Z axis) of the translation vector, use:
\[
\text{/gate/Name_Volume/placement/setThetaOfTranslation N deg}
\]

To set the magnitude of the translation vector, use:
\[
\text{/gate/Name_Volume/placement/setMagOfTranslation xx cm}
\]

- Example

\[
\text{/gate/Phantom/placement/setTranslation 1. 0. 0. cm}
\]
\[
\text{/gate/Phantom/placement/setMagOfTranslation 10. cm}
\]

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.

![Figure 3.13: Illustration of the translation placement](image)

3.4.2 Rotation

To rotate the Name_Volume volume by N degrees around the X axis, the commands are:
\[
\text{/gate/Name_Volume/placement/setRotationAxis X 0 0}
\]
\[
\text{/gate/Name_Volume/placement/setRotationAngle N deg}
\]
\[
\text{/gate/Name_Volume/placement/setAxis 0 1 0}
\]

The default rotation axis is the Z axis.

- Example

\[
\text{/gate/Phantom/placement/setRotationAxis 0 1 0}
\]
\[
\text{/gate/Phantom/placement/setRotationAngle 90 deg}
\]

The Phantom volume is rotated by 90 degrees around the Y axis. The application of this rotation placement is illustrated in figure 3.14.
3.4 Placing a volume

3.4.3 Alignment

Using the alignment command, a volume having an axis of symmetry (cylinder, ellipsoids, cone and hexagon) 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.

3.4.4 Special example: Wedge volume and OPET scanner

The wedge is always created as shown in the figure 3.4, that is with the slanted plane oriented towards the positive X direction. If one needs to have 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

\[
\frac{\text{setYLength}}{2}, \quad \frac{\text{setZLength}}{2}
\]
3.4 Placing a volume

![Diagram of a wedge with coordinates Y and X, and a center marked with a triangle.](image)

**Figure 3.15:** Center of wedge.

![Diagram of a block approximating a true circular geometry with radius R and a series of 8 crystals with varying lengths.](image)

**Figure 3.16:** A block approximating a true circular geometry.

respectively. But for the $X$ direction the center is located such that

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

where $2\Delta$ is the length of the wedge across the middle of the $Y$ direction, as shown in Figure 3.15.

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 crystal with varying 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
```

Then, a box that will contain the first wedge crystal is located inside the module.
3.5 Moving a volume

Figure 3.17: The OPET scanner.

```
/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
```

Notice that it is necessary to locate each crystals 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.

### 3.5 Moving a volume

The GEANT geometry architecture requires the geometry to be static during a simulation. However, the typical duration of a single event (e.g. ps for the particle transport, μs for scintillation, or ms for the response of the electronics) is very short when compared to most of the geometrical changes to be
modelled (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 (see time management in chapter?). 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.

### 3.5.1 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.

### 3.5.2 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.
3.5 Moving a volume

3.5.3 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.

3.5.4 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 \( dM(t) = A \times \sin(2 \pi f t + \phi) \) 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
/gate/Name_Volume/osc-trans/setAmplitude x. 0. 0. cm
```

to set the amplitude vector of the oscillating translation,

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

to set the frequency of the oscillating translation,

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

to set the period of the oscillating translation,
3.6 Updating the geometry

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

to set the phase at t=0 of the oscillating translation.

- 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

The movement that is modeled is defined by \( dM(t) = 10 \times \sin(100 \pi + 90) \).

3.5.5 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
/gate/Name_Volume/eccent-rot/setShiftXYZ x y z cm

to set the shift on the X-Y-Z directions,

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

to set the orbiting angular speed.

Remark: This particular move is closely related to LMF definition since the move parameters (shift 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 rotated around the Z axis during the simulation with a speed of 10 degrees per second.

3.6 Updating the geometry

Updating the geometry is needed to take into account any change in the geometry. It also refreshes the display window. The geometry can be updated either manually or automatically.
3.6 Updating the geometry

3.6.1 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`

3.6.2 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`
Chapter 4

Define a system

4.1 Definition

A System is a key-concept of GATE. It provides a template of a predefined geometry to simulate a scanner. Families of geometry like for example PET or SPECT and sharing the same general geometrical characteristics can be so derived from these templates.

Each system can be described by components organized in a tree level structure, each component having its own specific role or ordering.

As for an example of 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, one family of PET scanners as shown in figure 4.1 can be composed of such volumes called “rsectors”, “modules”, “submodules”, “crystal” and finally (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 imagery devices.

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)
4.2 Choice of the system

It is possible to use GATE without using a system, but in that case, no informations on 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 chap. 8) are recorded only for volumes that belong to a defined system (those volumes are declared as crystalSD, SD for “sensitive detector”, see chap. 5). Once the user is only testing a scanner geometry, the use of a predefined system is not necessary. But since the user wants to save the physical informations from the particle tracks inside the detector, the geometry has to be connected to a system. This chapter will explain the elements and rules of this connection.

4.2.1 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 active material which compose 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 level has to be fixed and must conform to the specification listed in table 4.1. The numbering of different sensitive volumes is completely fixed 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 bit reserved to the numbering of crystals\(^1\).

Other constraints could come from the specific output format being used. (see Chap.10 for further details)

4.2.2 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 checked in order to define its main components. This concerns not only the single channel simulation, with effects like thresholder response, but also the crosstalk in between different channels including the electronic or the optical crosstalk among components in a same level.

For PET scanner, one should simulate the coincidence between two channel, based on the preceding “single component” simulations.

In GATE, it is possible to introduce all these signal processing through digitizers (see chap. 8), which are based on the same idea of hierarchy level inside a system, also referenced as the “depth value” in table 4.2. The depth value is used here to tag a group of similar components sitting at a certain level, which could be the scintillator block level (crystal, with depth=5), the matrix of crystal (submodule or module), or a group of crystal matrix(ces) depth=1. Depth values given here refers to the cylindricalPET system.

In order to simulate the electronic circuitry in relation to the system, one should consider the following procedure:

- Regroup the detector’s electronic components in different levels

\(^1\)See page 180 for an example of such limitations
4.3 How to connect the geometry to a system

- Draw up the list of the signal processing to be used on groups (see “adder”, “readout”, “dead time” in chap. 8).

- Combine the signals coming from different volume, with for example: readout module for the signals summation of a volume (page ??), the crosstalk and/or the coincidence between signals (pages 127 and ??).

**NOTE**:
- One or several crosstalk processing can be applied to components of different level (e.g. crosstalk between crystals, then crosstalk between modules). This involve components in the same level.
- For PET scanners, a test is done to valid a coincidence on the number difference of the uppermost level components (as defined as depth = 1 in table 4.2). This test is done to reject accidental coincidence between adjacent logic structure. When the user built its geometry, this logic organisation should correspond to the first level of a system in order to use this coincidence rejection (see chapter 8).

### 4.3 How to connect the geometry to a system

The connection between the geometry and a system is done in several stages:

1. The geometrical structure need to be first defined, keeping in mind that it must fulfill some constraints, as described before.

2. 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 your geometrical structure (table 4.2):

   **The general macro line is**:
   /gate/systems/SystemName/Level/attach UserVolumeName

   **where**:
   - **SystemName** is the reserved name of the system (one of the entry in column 1),
   - **Level** is the reserved name of the level (see column 2),
   - **UserVolumeName** is the name the user gave to a volume, according to the conventions of Chap. 3.

3. Finally, the specific output corresponding to the system has to be defined to analyse the data if necessary (see chap. 10).
4.3 How to connect the geometry to a system

<table>
<thead>
<tr>
<th>System</th>
<th>Components and its shape</th>
<th>Available outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>scanner</td>
<td>level1, level2, level3, level4, level5, geometry not fixed</td>
<td>Basic output : Ascii or ROOT, coincidences only for PETscanner</td>
</tr>
<tr>
<td>CTScanner</td>
<td>block, pixel</td>
<td>Binary CT image; Basic output : Ascii or ROOT</td>
</tr>
<tr>
<td>cylindricalPET</td>
<td>rsector, module, submode, crystal, layer</td>
<td>Basic output : Ascii, ROOT and Raw. Specific: LMF</td>
</tr>
<tr>
<td>CPET</td>
<td>crystal</td>
<td>Basic output : Ascii, ROOT and Raw.</td>
</tr>
<tr>
<td>SPECThead</td>
<td>crystal, pixel</td>
<td>Basic output : Ascii, ROOT and Raw. Specific: PROJECTIONSET or INTERFILE, no coincidences</td>
</tr>
<tr>
<td>ecat</td>
<td>block, crystal</td>
<td>Basic output : Ascii, ROOT and Raw. Specific: SINOGRAM or ECAT7</td>
</tr>
<tr>
<td>ecatAccel</td>
<td>block, crystal</td>
<td>Basic output : Ascii, ROOT and Raw. Specific: SINOGRAM or ECAT7</td>
</tr>
<tr>
<td>OPET</td>
<td>rsector, module, submode, crystal, layer</td>
<td>Basic output : Ascii, ROOT and Raw. Specific: LMF</td>
</tr>
</tbody>
</table>

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.
### 4.3 How to connect the geometry to a system

<table>
<thead>
<tr>
<th>System</th>
<th>attach keyword argument</th>
<th>Depth for readout segmentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>scanner</td>
<td>“level1”</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>“level2”</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>“level3”</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>“level4”</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>“level5”</td>
<td>5</td>
</tr>
<tr>
<td>CTscanner</td>
<td>“block”</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>“pixel”</td>
<td>2</td>
</tr>
<tr>
<td>cylindricalPET</td>
<td>“rsector”</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>“module”</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>“submodule”</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>“crystal”</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>“layer[i], i=0,3a”</td>
<td>5</td>
</tr>
<tr>
<td>CPET</td>
<td>“crystal”</td>
<td>1</td>
</tr>
<tr>
<td>SPECTHead</td>
<td>“crystal”</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>“pixel”</td>
<td>2</td>
</tr>
<tr>
<td>ecat</td>
<td>“block”</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>“crystal”</td>
<td>2</td>
</tr>
<tr>
<td>ecatAccel</td>
<td>“block”</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>“crystal”</td>
<td>2</td>
</tr>
<tr>
<td>OPET</td>
<td>“rsector”</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>“module”</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>“submodule”</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>“crystal”</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>“layer[i], i=0,7a”</td>
<td>5</td>
</tr>
</tbody>
</table>

**Special volumes attachment (see Chap.5)**

<table>
<thead>
<tr>
<th>System</th>
<th>Attachment</th>
</tr>
</thead>
<tbody>
<tr>
<td>All systems</td>
<td>“attachCrystalSD”</td>
</tr>
<tr>
<td>Sensitive detector volume</td>
<td>“attachCrystalSD”</td>
</tr>
<tr>
<td>Sensitive phantom volume</td>
<td>“attachPhantomSD”</td>
</tr>
</tbody>
</table>

*up to 4 layers, named “layer0”, ..., “layer3” can be attached to cylindricalPET*

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.
4.4 Different types of systems

Figure 4.2: Illustration of the scanner system. The different volumes, in particular the sensitive one, can be of any shape, here cylindrical sectors crystals, instead of box in other systems. Scanner cylinder is draw in magenta, whereas one of the sector hierarchy: Level1, Level2, Level3, Level4 is overdrawn in yellow, blue, green, red, respectively. “Detector” volume of cylindrical sector shapes are shown in plain red.

4.4.1 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 choose in table 3.2 on page 26. The figure 4.4 illustrate what kind of detector one can simulate with this system without geometry constraints. On the other hand there is no specific output format and the informations on hits are only available in ROOT or ASCII format.
4.4 Different types of systems

4.4.2 CTscanner

Description

The CT scanner system permits to simulate a simple CT scanner. It has two possible levels: the \textit{block} component, that can be linearly repeated along the \textit{Y} axis and the \textit{pixel} component that can "cubicArray" repeated inside the block. It is possible to ask for a binary output (\textit{imageCT} output) that stores the simulated CT image and it is produced for each time slice. The image format is a "float" matrix of dimensions given by the total number of pixels in the \textit{X} and the \textit{Y} directions, the content corresponding to the number of counts per pixel per acquisition (time slice). Two run modes are proposed to the user:

- **Complete simulation:** Both the \textit{blocks} and the \textit{pixels} are user defined. All volumes are created by Geant4 and the digitalisation can be made at the pixel level (\textit{level 2}).

- **Fast simulation:** Only the block level is defined. Geant4 creates one volume corresponding to the CT block (only one block being possible) and the digitalisation is made by the output module. The number of pixels per block are given through the output module Messenger (see 10.8). This mode is faster since only one Geant4 volume is simulated but obviously a rather approximated scanner response can be guaranteed.

Use

\textbf{example 1 : complete CT simulation}

```plaintext
# CT for small animal imaging
/gate/geometry/enableAutoUpdate
/gate/world/daughters/name CTscanner
/gate/world/daughters/insert box
/gate/CTscanner/placement/setTranslation 0. 0. 20. mm
/gate/CTscanner/geometry/setXLength 100. mm
/gate/CTscanner/geometry/setYLength 100. mm
/gate/CTscanner/geometry/setZLength 4 mm
/gate/CTscanner/setMaterial Air
/gate/CTscanner/vis/forceWireframe
/gate/CTscanner/vis/setColor white
/gate/geometry/update

### BLOCK

/gate/CTscanner/daughters/name block
/gate/CTscanner/daughters/insert box
/gate/block/placement/setTranslation 0 0 0. mm
/gate/block/geometry/setXLength 80 mm
/gate/block/geometry/setYLength 8. mm
/gate/block/geometry/setZLength 0.700 mm
/gate/block/setMaterial Air
```

- 55 -
4.4 Different types of systems

/gate/block/vis/forceWireframe
/gate/block/vis/setColor gray
/gate/geometry/update

### P I X E L

/gate/block/daughters/name pixel
/gate/block/daughters/insert box
/gate/pixel/geometry/setXLength 1.00 mm
/gate/pixel/geometry/setYLength 1.00 mm
/gate/pixel/geometry/setZLength 0.700 mm
/gate/pixel/setMaterial Silicon
/gate/pixel/vis/forceWireframe
/gate/pixel/vis/setColor red
/gate/geometry/update

### R E P E A T   B L O C K

/gate/block/repeaters/insert linear
/gate/block/linear/setRepeatNumber 10
/gate/block/linear/setRepeatVector 0.8 0 mm

### R E P E A T   P I X E L

/gate/pixel/repeaters/insert cubicArray
/gate/pixel/cubicArray/setRepeatNumberX 80
/gate/pixel/cubicArray/setRepeatNumberY 8
/gate/pixel/cubicArray/setRepeatNumberZ 1
/gate/pixel/cubicArray/setRepeatVector 1.0 1.0 0 mm
/gate/pixel/cubicArray/autoCenter true
/gate/geometry/update

### A T T A C H   S Y S T E M

/gate/systems/CTscanner/block/attach block
/gate/systems/CTscanner/pixel/attach pixel

### A T T A C H   L A Y E R   S D

/gate/pixel/attachCrystalSD

**example 2:** *Fast CT simulation*

############################################################
# CT for small animal imaging #
############################################################

- 56 -
4.4 Different types of systems

4.4.3 cylindricalPET

Description

“cylindricalPET” is a PET system that allows to describe most of the small animal PET scanners. The main specificity of cylindricalPET is the possibility to store output data in List Mode Format (LMF) of the Crystal Clear Collaboration. The way to store GATE output in LMF is fully described in this manual (see part 10.7). You will find an example of macro with a cylindricalPET definition in the macro listed
4.4 Different types of systems

bellow. A CylindricalPET is based on a cylindrical geometry. It is composed of 5 hierarchic levels. Hierarchy, from mother to daughter is:

- **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 the phoswich system) radially arranged box(es) inside crystal. Do not use a repeater for layers, but build them one by one in your macro. layer must be set as a sensible detector with the macro command: /attachCrystalSD

Material of layer(s) must be the material of your detector Ex.: LSO or BGO + GSO for a double layer phoswich system. Materials of other levels (crystals, submodules, modules, rsectors, cylindricalPET) can be anything else. Keep in mind that GATE set as material of a point of world, the material of the more internal volume (here the material of the more internal volume is layer material).

**IMPORTANT** : Visualization should help you to build this geometry with no overlap. GATE performs a test which is able to detect overlapping, but with a limited precision. This test is done at the end *PreInit mode* of Gate (see Chap. 1):

```
/run/initialize
/geometry/test/recursive_test
```

Users should take care in particular of case where a volume is not bigger than its mother volume.

**Use**

You will find hereafter an example with comments (comments lines start with a “#” symbol). This construction of a cylindricalPET geometry must be at the beginning of your macro, before initializations.

```
# WORLD
/gate/world/geometry/setXLength 40 cm
/gate/world/geometry/setYLength 40. cm
/gate/world/geometry/setZLength 40. cm

# MOUSE
/gate/world/daughters/name mouse
/gate/world/daughters/insert cylinder
```

² Please note that the words in bold characters are reserved; see also keywords in table 4.2
4.4 Different types of systems

/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

# CYLINDRICAL
/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

# RSECTOR
/gate/cylindricalPET/daughters/name rsector
/gate/cylindricalPET/daughters/insert box
/gate/rsector/placement/setTranslation 135 0 0 mm
/gate/rsector/geometry/setXLength 10. mm
/gate/rsector/geometry/setYLength 19. mm
/gate/rsector/geometry/setZLength 76.6 mm
/gate/rsector/setMaterial Water
/gate/rsector/vis/forceWireframe

# MODULE
/gate/rsector/daughters/name module
/gate/rsector/daughters/insert box
/gate/module/geometry/setXLength 10. mm
/gate/module/geometry/setYLength 19. mm
/gate/module/geometry/setZLength 19. mm
/gate/module/setMaterial Water
/gate/module/vis/forceWireframe
/gate/module/vis/setColor gray

# CRYSTAL
/gate/module/daughters/name crystal
/gate/module/daughters/insert box
/gate/crystal/geometry/setXLength 10. mm
/gate/crystal/geometry/setYLength 2.2 mm
/gate/crystal/geometry/setZLength 2.2 mm
/gate/crystal/setMaterial Water
/gate/crystal/vis/forceWireframe
/gate/crystal/vis/setColor magenta

# LAYER
/gate/crystal/daughters/name LSO
/gate/crystal/daughters/insert box
4.4 Different types of systems

/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

# REPEAT 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 10. 2.4 2.4 mm

# REPEAT MODULE
/gate/module/repeaters/insert cubicArray
/gate/module/cubicArray/setRepeatNumberZ 4
/gate/module/cubicArray/setRepeatVector 0. 0. 19.2 mm

# REPEAT RSECTOR
/gate/rsector/repeaters/insert ring
/gate/rsector/ring/setRepeatNumber 42

# ATTACH SYSTEM
/gate/systems/cylindricalPET/rsector/attach rsector
/gate/systems/cylindricalPET/module/attach module
/gate/systems/cylindricalPET/crystal/attach crystal
/gate/systems/cylindricalPET/layer0/attach LSO

# ATTACH LAYER SD
/gate/LSO/attachCrystalSD
/gate/mouse/attachPhantomSD
4.4.4 CPET

This system was conceived for the simulation of a CPET-like scanner (C-PET Plus, Philips Medical Systems, Best, the Netherlands), with one ring of NaI crystal with a curve shape. For this scanner, one level after the system level is enough to describe the volume hierarchy.

description

This system has the particularity to have cylindrical shaped sector component, e.g. based on the “cylinder” shape (see fig.4.3 and Chap.3, page 26 for definition.), whereas those components are generally boxes in other systems.

![Figure 4.3: Definition of a CPET sector volume. This system allows one to define cylindrical shape sector instead of box shape sectors, like in other PET systems](image_url)

Use

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

Description

```plaintext
# 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

# FISRT 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
```
4.4 Different types of systems

/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

Figure 4.4: One NaI crystal with a curved shape

Figure 4.5: After the ring repeater, 6 NaI crystals are describing the scanner.

Attachment. Attach the object crystal to its corresponding component in the CPET system.

/gate/systems/CPET/crystal/attach crystal
Sensitive detector. Set the crystals as sensitive detectors (see section 5.2.1, p.82).

/gate/crystal/attachCrystalSD

The digitizer part (see section 8.2) is made of the *adder* module and some blurring module (see Chapter 8).
4.4 Different types of systems

4.4.5 ecat

Description

The ecat system is a simplified version of cylindricalPET and was named ecat because it is aimed to simulate the ECAT scanner family from CPS Innovations (Knoxville, TN, U.S.A.). It is based on the block detector principle [15]: an array of crystals, typically 8 × 8, read by a few photomultipliers, typically 4. The blocks are organized on an annular geometry to form multi-ring detectors.

You will find an example of macro with an ecat definition in the file:

$GATEHOME/example_PET_Scanner/PET_Ecat_System.mac

The ecat system has only three hierarchical levels: one global 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, allowing the generated data to be written in a sinogram format. These are called the sinogram and the ecat7 output modules and are discussed in sections 10.5 and 10.6.

Use

Described below is an example code for modeling a four block-rings scanner.

Description of the base. It has to be named after the selected system (ecat here) and is declared as a volume daughter of the world. It has a ring shape and should comprise entirely 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.
4.4 Different types of systems

**Description of the block.** Set the size and the position of the first block within the base *ecat*. It is a rectangular parallelepiped and should comprise entirely 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 and not at one edge of the base (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 4.7: Definition of the block.](image)

**Description of the crystal.** 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 in the center of the block and not at one of its edges (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
```

**Description of the crystal array.** Set the size and the sampling of the crystal array within one block. The crystal array will be 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
```
Description of the block-rings. 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, \textit{i.e.} a 32 crystal-ring scanner, with 576 crystals per crystal-ring.

Attachment. Attach the objects \textit{block} and \textit{crystal} to their corresponding component in the ecat system.

```
/gate/systems/ecat/block/attach block
/gate/systems/ecat/crystal/attach crystal
```

Sensitive detector. Set the crystals as sensitive detectors (see section 5.2.1, p.82).

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

The digitizer part (see section 8.2) remains the same as for the cylindricalPET system.
4.4.6 ecatAccel

Description

A new system, called ecatAccel, was introduced to model the new PET scanner family ECAT ACCEL from CPS Innovations, Knoxville, TN, U.S.A. The ecatAccel system differ 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 section 10.5 and 10.6 for further information on sinogram and ecat7 outputs for the ecatAccel system.

Use

Described below is an example code for modeling the ACCEL PET scanner belonging to the BIOGRAPH-LSO (SIEMENS - CTI) PET-CT scanner.

Description of the base. It has to be named after the selected system (ecatAccel here) and is declared as a volume daughter of the world. As for the ecat system, it has a ring shape and should comprise entirely all detectors (see Figure 4.6). For the BIOGRAPH, it can be described as follows:

```
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
```

Description of the block. Set the size and the position of the first block within the base ecatAccel. As for the ecat system, it is a rectangular parallelepiped and should comprise entirely 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 and not at one edge of the base.

```
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
```

Description of the crystal. 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 in the center of the block and not at one of its edges.

```
gate/block/daughters/name crystal
gate/block/daughters/insert box
```
4.4 Different types of systems

/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

Description of the crystal array. Set the size and the sampling of the crystal array within one 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

Description of the block-rings. Set the number of blocks per block-ring (command setRepeatNumberWithTheta) and the number of block-rings (command setRepeatNumberWithPhi). The angle between two adjacent blocks in a block-ring should be set with the command setThetaAngle and the angle between two adjacent blocks belonging to two neighbouring rings in the axial direction should be set with the command setPhiAngle. Multiple block-ring systems 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/setThetaAngle 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.

Attachment. Attach the objects block and crystal to their corresponding component 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.

4.4.7 OPET

Description

A new system, called OPET, was introduced to model the new PET prototype.
4.4 Different types of systems

Use

Described below is an example code for modeling the OPET PET scanner.

```python
# R S E C T O R (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 mom 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
```

- 69 -
4.4 Different types of systems

/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
4.4 Different types of systems

# 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 crystal4
/gate/module/daughters/insert box

- 71 -
4.4 Different types of systems

/gate/crystal4/geometry/setXLength 10 mm
/gate/crystal4/geometry/setYLength 2.1620 mm
/gate/crystal4/geometry/setZLength 2.1620 mm
/gate/crystal4/placement/setTranslation 0. 1.1145 0. mm
/gate/crystal4/setMaterial Air
/gate/crystal4/vis/setColor black
/gate/crystal4/vis/setVisible false

# L A Y E R (Put the LSO in the small box)
/gate/crystal4/daughters/name LSO4
/gate/crystal4/daughters/insert wedge
/gate/LSO4/geometry/setXLength 7.773 mm
/gate/LSO4/geometry/setNarrowerXLength 7.637 mm
/gate/LSO4/geometry/setYLength 2.1620 mm
/gate/LSO4/geometry/setZLength 2.1620 mm
/gate/LSO4/placement/setRotationAxis 0 0 1
/gate/LSO4/placement/setRotationAngle 180 deg
/gate/LSO4/placement/setTranslation 1.1475 0. 0. mm
/gate/LSO4/setMaterial BGO
/gate/LSO4/vis/setColor blue

#---------------------------------------------
# Daughter crystal1 inside mom crystal
/gate/module/daughters/name crystal5
/gate/module/daughters/insert box
/gate/crystal5/geometry/setXLength 10 mm
/gate/crystal5/geometry/setYLength 2.1620 mm
/gate/crystal5/geometry/setZLength 2.1620 mm
/gate/crystal5/placement/setTranslation 0. 3.3435 0. mm
/gate/crystal5/setMaterial Air
/gate/crystal5/vis/setColor black
/gate/crystal5/vis/setVisible false

# L A Y E R (Put the LSO in the small box)
/gate/crystal5/daughters/name LSO5
/gate/crystal5/daughters/insert wedge
/gate/LSO5/geometry/setXLength 8.193 mm
/gate/LSO5/geometry/setNarrowerXLength 7.773 mm
/gate/LSO5/geometry/setYLength 2.1620 mm
/gate/LSO5/geometry/setZLength 2.1620 mm
/gate/LSO5/placement/setRotationAxis 0 0 1
/gate/LSO5/placement/setRotationAngle 180 deg
/gate/LSO5/placement/setTranslation 1.0085 0. 0. mm
/gate/LSO5/setMaterial BGO
/gate/LSO5/vis/setColor green

- 72 -
### Different types of systems

#### Daughter crystal 1 inside mom crystal
- /gate/module/daughters/name crystal6
- /gate/module/daughters/insert box
- /gate/crystal6/geometry/setXLength 10 mm
- /gate/crystal6/geometry/setYLength 2.1620 mm
- /gate/crystal6/geometry/setZLength 2.1620 mm
- /gate/crystal6/placement/setTranslation 0.55725 0. mm
- /gate/crystal6/setMaterial Air
- /gate/crystal6/vis/forceWireframe
- /gate/crystal6/vis/setColor black
- /gate/crystal6/vis/setVisible false

#### L A Y E R (Put the LSO in the small box)
- /gate/crystal6/daughters/name LSO6
- /gate/crystal6/daughters/insert wedge
- /gate/LSO6/geometry/setXLength 8.921 mm
- /gate/LSO6/geometry/setNarrowerXLength 8.193 mm
- /gate/LSO6/geometry/setYLength 2.1620 mm
- /gate/LSO6/geometry/setZLength 2.1620 mm
- /gate/LSO6/placement/setRotationAxis 0 0 1
- /gate/LSO6/placement/setRotationAngle 180 deg
- /gate/LSO6/placement/setTranslation 0.7215 0. 0. mm
- /gate/LSO6/setMaterial BGO
- /gate/LSO6/vis/setColor red

### Daughter crystal 2 inside mom crystal
- /gate/module/daughters/name crystal7
- /gate/module/daughters/insert box
- /gate/crystal7/geometry/setXLength 10 mm
- /gate/crystal7/geometry/setYLength 2.1620 mm
- /gate/crystal7/geometry/setZLength 2.1620 mm
- /gate/crystal7/placement/setTranslation 0.78015 0. mm
- /gate/crystal7/setMaterial Air
- /gate/crystal7/vis/forceWireframe
- /gate/crystal7/vis/setColor black
- /gate/crystal7/vis/setVisible false

#### L A Y E R (Put the LSO in the small box)
- /gate/crystal7/daughters/name LSO7
- /gate/crystal7/daughters/insert wedge
- /gate/LSO7/geometry/setXLength 10 mm
- /gate/LSO7/geometry/setNarrowerXLength 9.07 mm
- /gate/LSO7/geometry/setYLength 2.1620 mm
4.4 Different types of systems

/gate/LSO7/geometry/setZLength 2.1620 mm
/gate/LSO7/placement/setTranslation 0.2698 0. 0. mm
/gate/LSO7/placement/setRotationAxis 0 0 1
/gate/LSO7/placement/setRotationAngle 180 deg
/gate/LSO7/setMaterial BGO
/gate/LSO7/vis/setColor yellow

# -------------

# Repeat 8 time the level2 to get 8 rings (Z direction)
/gate/module/repeaters/insert linear
/gate/module/linear/setRepeatNumber 8
/gate/module/linear/setRepeatVector 0. 0. 2.2275 mm

/gate/rsector/repeaters/insert ring
/gate/rsector/ring/setRepeatNumber 6

#/gate/OPET/placement/setRotationAxis 0 0 1
# /gate/OPET/placement/setRotationAngle 30 deg

# ATTACH SYSTEM

/gate/systems/OPET/rsector/attach rsector
/gate/systems/OPET/module/attach module
/gate/systems/OPET/layer0/attach LSO0
/gate/systems/OPET/layer1/attach LSO1
/gate/systems/OPET/layer2/attach LSO2
/gate/systems/OPET/layer3/attach LSO3
/gate/systems/OPET/layer4/attach LSO4
/gate/systems/OPET/layer5/attach LSO5
/gate/systems/OPET/layer6/attach LSO6
/gate/systems/OPET/layer7/attach LSO7

# ATTACH LAYER SD : definition of your sensitive detector
/gate/LSO0/attachCrystalSD
/gate/LSO1/attachCrystalSD
/gate/LSO2/attachCrystalSD
/gate/LSO3/attachCrystalSD
/gate/LSO4/attachCrystalSD
/gate/LSO5/attachCrystalSD
/gate/LSO6/attachCrystalSD
/gate/LSO7/attachCrystalSD

Figure 4.9 shows the final OPET scanner.
4.4 Different types of systems

4.4.8 SPECTHead

Description

*SPECTHead* is a SPECT system that enables users 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 section 10.4. 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 reserved word.

- **crystal** which is coupled to the main detector block.

- **pixel** which can be used if one is modeling a pixelated detector.

If a uniform detector block is being used, than *crystal* should meet the material definition of the detector. If the detector is pixelated then the *pixel* material definition should be chosen correctly while the crystal material can be anything non-specific.

Use

You will find hereafter a part of the SPECT benchmark macro, which was distributed with the GATE software. It details the SPECTHead system definitions, which should be given as input before the initialization.

```plaintext
#- World
#- Define the 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 4.9: The OPET scanner.
4.4 Different types of systems

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

```plaintext
#- Create the SPECT system, which will yield
#- an Interfile output of the projection data
/gate/world/daughters/name 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
```
4.4 Different types of systems

#- Define some visualisation options
/gate/SPECThead/vis/forceWireframe

#- Collimator
#- Create a full volume defining the shape of
#- the collimator (typical for SPECT)
/gate/SPECThead/daughters/name 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 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 these holes in a linear
/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

- 77 -
4.4 Different types of systems

Parameterized

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 by this approach in less than a second.

Example code for 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
```
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
Chapter 5

Attach the sensitive detectors

5.1 General purpose

Once a model has been defined for the scanner through the construction of a system (see Chapter 4), 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 on interactions of a particle in the matter (“hits”) using information from steps along a 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 store various information associated to a step object, the information can be: the energy deposition of a step, geometrical information, position and time of a step, etc.

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 the information regarding the interactions occurring in non-sensitive volumes is lost.

Two sensitive detectors are defined in GATE:

- **The crystalSD** allows to record information on interactions inside the volumes belonging to a scanner for instance, crystals or collimators.
5.2 Two types of sensitive detector

- The **phantomSD** may be used to record information on Compton and Rayleigh interactions taking place in the volumes before the detection in the scanner system (e.g., in a case of a SPECT camera: the examination table, the phantom, and the collimator are some of the volumes where it can be important to get information about Compton and Rayleigh interactions).

A complete definition of the simulation context normally involves performing both series of attachments: one series of volumes is attached to the **phantomSD**, another series of volumes is attached to the **crystalSD**.

5.2 Two types of sensitive detector

5.2.1 The **crystalSD**

Definition and use

The **crystalSD** may be used to record information on 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.

Here 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 SPECT-head system.

```
# ATTACH SYSTEM
/gate/systems/SPECThead/crystal/attach crystal
```

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

```
# ATTACH SENSITIVE DETECTOR
/gate/crystal/attachCrystalSD
```

5.2.2 The **phantomSD**

Definition and use

The **phantomSD** plays a crucial role in GATE simulations, as it is used to detect and count Compton and Rayleigh interactions taking place in the scanner FOV. This information is then used to estimate whether a photon reaching a detector is a direct or a Compton-scattered photon. Thus, in PET, the **phantomSD**
5.2 Two types of sensitive detector

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’s possible to retrieve two types of information considering the hits:

- The number of Compton and Rayleigh interactions occurring in all the volumes attached to the phantomSD; this information is stored in the data output variables nPhantomCompton and nPhantomRayleigh.
  Note: This information is 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 occurred; the data output variables used are named 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 offspring (direct or indirect) of this volume.

- Last, all these volumes (FOV and sources) should be attached to the phantomSD using the command attachPhantomSD.

IMPORTANT: In order to retrieve data output informations on hits occurring in the phantomSD (nPhantomCompton and compVolName), a crystalSD has to be defined in addition in the simulation. Otherwise, data output variables will be created but 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.

Note that 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 systems:

First commands are used to attach the scattering volumes to the detection level “base” of the SPECTHead system.

```bash
# ATTACH SYSTEM
/gate/systems/SPECTHead/base/attach FOV
/gate/systems/SPECTHead/base/attach head
/gate/systems/SPECTHead/base/attach body
```

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

```bash
# ATTACH SENSITIVE DETECTOR
/gate/FOV/attachPhantomSD
```
5.2 Two types of sensitive detector

/gate/head/attachPhantomSD
/gate/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
/gate/systems/SPECThead/crystal/attachCrystalSD
/gate/crystal/attachCrystalSD

IMPORTANT: It’s impossible to attach two sensitive detectors to the same volume. Thus, in order to be able to count also the Compton interactions occurring 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 voxelized matrix: Previous commands to attach sensitive detectors are used for the volumes created using the geometry commands of GATE (see Chapter 3). In order to record the same information concerning the interactions occurring in a voxelized matrix: see Chapter 7.
Chapter 6

Set up the physics

6.1 Physics process

The description of all of the physics processes described here are extracted from the Geant4 Physical User’s Guide.

6.1.1 Process model

With Geant4, two types of packages are available to simulate electromagnetic processes:

- Standard Energy Electromagnetic Processes (SEP)
- Low Energy Electromagnetic Processes (LEP)

With the SEP, it is possible to simulate photoelectric and Compton diffusion interactions with an energy higher than 10 keV. Concerning the LEP package, the Geant4 physical tutorial gives some comments and details:

The low energy processes of Geant4 represent electromagnetic interactions at lower energies than those covered by the equivalent Geant4 standard electromagnetic processes. The current implementation of low energy processes is valid for energies down to 250 eV (and can be used up to approximately 100 GeV), unless differently specified. It covers elements with atomic number between 1 and 99. All processes involve two distinct phases:

- the calculation and use of total cross sections, and
- the generation of the final state.

Both phases are based on the theoretical models and on exploitation of evaluated 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;
6.1 Physics process

- binding energy values based on data of Scofield.

Evaluated data sets are produced through the process of critical comparison, selection, renormalization and averaging of the available experimental data, normally complemented by model calculations. These libraries provide the following data relevant for the simulation of Geant4 low energy processes:

- total cross-sections for photoelectric effect, Compton scattering, Rayleigh scattering, pair production and bremsstrahlung,
- subshell integrated cross sections for photoelectric effect and ionization,
- energy spectra of the secondaries for electron processes,
- scattering functions for the Compton effect,
- form factors for Rayleigh scattering,
- binding energies for electrons for all subshells,
- transition probabilities between subshells for fluorescence and the Auger effect, and
- stopping power tables.

The energy range covered by the data libraries extends from 100 GeV down to 1 eV for Rayleigh and Compton effects, down to the lowest binding energy for each element for photoelectric effect and ionization, and down to 10 eV for bremsstrahlung. The final state products of the processes are generated by sampling relevant physical quantities, such as energies and angular distributions of secondaries, from distributions derived from theoretical models and evaluated data. The energy dependence of the parameters which characterize the distributions is taken into account either by direct interpolation of the data available in the libraries, or by interpolation of values obtained from fits to the data. When generating the final state, an atom of the material in which the interaction occurs is randomly selected and atomic de-excitation is simulated. Secondaries which would be produced with energies below their user defined production threshold are not created and their energy is deposited locally.

6.1.2 Process list

Photoelectric effect

Cross Section and Mean Free Path

The photoelectric effect is the ejection of an electron from a material after a photon has been absorbed by that material. It is simulated by using a parameterized photon absorption cross section to determine the energy of the ejected electron, and the K-shell angular distribution to sample the direction of the electron. The parameterization of the photoabsorption cross section proposed by Biggs et al. [16] was used:

\[
\sigma(Z, E_\gamma) = a(Z, E_\gamma) E_\gamma + b(Z, E_\gamma) E_\gamma^2 + c(Z, E_\gamma) E_\gamma^3 + d(Z, E_\gamma) E_\gamma^4
\]

Using the least-squares method, a separate fit of each of the coefficients a, b, c, d to the experimental data was performed in several energy intervals. As a rule, the boundaries of these intervals were equal to the corresponding photoabsorption edges. In a given material the mean free path, \( \lambda \), for a photon to interact via the photoelectric effect is given by:

\[
\lambda(E_\gamma) = \left( \sum_i n_{ati} \sigma(Z_i, E_\gamma) \right)^{-1}
\]
6.1 Physics process

Where $n_{ati}$ is the number of atoms per volume of the $i^{th}$ element of the material. The cross section and mean free path are discontinuous and must be computed `on the fly` from the 2 precedent formulas.

**Final State**

The binding energies of the shells depend on the atomic number $Z$ of the material. In compound materials the $i^{th}$ element is chosen randomly according to the probability:

$$Prob(Z_i, E_\gamma) = \frac{n_{ati}\sigma(Z_i, E_\gamma)}{\sum_i[n_{ati}\sigma_i(E_\gamma)]}$$

A quantum can be absorbed if where the shell energies are taken from G4AtomicShells data: the closest available atomic shell is chosen. In the current implementation the relaxation of the atom is not simulated, but instead is counted as a local energy deposit.

**Compton effect**

**Cross Section per Atom and Mean Free Path**

When simulating the Compton scattering of a photon from an atomic electron, an empirical cross section formula is used, which reproduces the cross section data down to 10 keV. The values of the parameters can be found within the method which computes the cross section per atom. A fit of the parameters was made to over 511 data points chosen from the intervals $1 \leq Z \leq 100$ and $E_{\gamma} \in [10keV, 100GeV]$. In a given material the mean free path, $\lambda$, for a photon to interact via Compton scattering is given by :

$$\lambda(E_\gamma) = \left(\sum_i n_{ati}\sigma_i(Z_i, E_\gamma)\right)^{-1}$$

Where $n_{ati}$ is the number of atoms per volume of the $i^{th}$ element of the material.

**Final State**

The quantum mechanical Klein-Nishina differential cross section per atom is :

$$\frac{d\sigma}{d\epsilon} = \pi r_e^2 \frac{m_e c^2}{E_0} Z \left[\frac{1}{\epsilon} + \epsilon\right]\left[1 - \frac{\epsilon \sin^2 \theta}{1 + \epsilon^2}\right]$$

where :

- $r_e$ = classical electron radius
- $m_e$ = electron mass
- $E_0$ = energy of the incident photon
- $E_1$ = energy of the scattered photon
- $\epsilon = E_1/E_0$

Assuming an elastic collision, the scattering angle $\theta$ is defined by the Compton formula :

$$E_1 = E_0 \frac{m_e c^2}{m_e c^2 + E_0(1 - \cos \theta)}$$
6.1 Physics process

Rayleigh effect

Total Cross Section
The total cross section for the Rayleigh scattering process is determined from the data as described in low energy process section.

Sampling of the Final State
The coherent scattered photon angle is sampled according to the distribution obtained from the product of the Rayleigh formula \((1 + \cos^2 \theta) \sin \theta\) and the square of Hubbel's form factor \(F_F^2(q)\)

\[ \Phi(E, \theta) = [1 + \cos^2 \theta] \sin \theta \cdot F_F^2(q) \]

where \(q = 2E \cdot \sin(\theta/2)\) is the momentum transfer.

Form factors introduce a dependency on the initial energy \(E\) of the photon that is not taken into account in the Rayleigh formula. At low energies, form factors are isotropic and do not affect angular distribution, while at high energies they are forward peaked.

The sampling procedure is as follows:

- \(\cos \theta\) is chosen from a uniform distribution between -1 and 1
- the form factor \(F_F\) is extracted from the data table for the considered element, using logarithmic data interpolation, for \(q = 2E \cdot \sin(\theta/2)\)
- if the value obtained for \(\Phi(E, \theta)\) is larger than a random number uniformly distributed between 0 and \(Z^2\), the procedure is repeated from step 1, otherwise \(\theta\) is taken as the photon scattering angle with respect to its incident direction.
- the azimuthal direction of the scattered photon randomly is chosen.

Electron and positron physics

In GATE, the SEP is used for modeling the transport of electrons and positrons. The most important processes are:

- Multiple Scattering;
- Ionization;
- Bremsstrahlung;
- \((e^+e^-)\) annihilation.

Multiple Scattering
GEANT4 uses a new multiple scattering (MSC) model to simulate the multiple scattering of charged particles in matter. This model does not use the Molière formalism, but is based on the more complete Lewis theory. The model simulates the scattering of the particle after a given step, and also computes the path length correction and the lateral displacement.

MSC simulation algorithms can be classified as either "detailed" or "condensed". In the detailed algorithms, all the collisions/interactions experienced by the particle are simulated. This simulation can be considered as exact; it gives the same results as the solution of the transport equation. However, it can be used only if the number of collisions is not too large, a condition fulfilled only for special geometries (such as thin foils), or low enough kinetic energies. For larger kinetic energies the average number of collisions is very large and the detailed simulation becomes very inefficient. High energy simulation
codes use condensed simulation algorithms, in which the global effects of the collisions are simulated at
the end of a track segment. The global effects generally computed in these codes are the net displace-
ment, energy loss, and change of direction of the charged particle. These quantities are computed from
the multiple scattering theories used in the codes. The accuracy of the condensed simulations is limited
by the approximations of the multiple scattering theories.

Most particle physics simulation codes use the multiple scattering theories of Molière, Goudsmit and
Saunderson and Lewis. The theories of Molière and Goudsmit-Saunderson give only the angular dis-
tribution after a step, while the Lewis theory computes the moments of the spatial distribution as well.
None of these MSC theories gives the probability distribution of the spatial displacement. Therefore
each of the MSC simulation codes incorporates its own algorithm to determine the spatial displacement
of the charged particle after a given step. These algorithms are not exact, of course, and are responsible
for most of the uncertainties in the MSC codes. Therefore the simulation results can depend on the value
of the step length and generally one has to select the value of the step length carefully.

A new class of MSC simulation, the "mixed" simulation algorithms, appeared in the literature recently.
The mixed algorithm simulates the "hard" collisions one by one and uses a MSC theory to treat the ef-
teffects of the "soft" collisions at the end of a given step. Such algorithms can prevent the number of steps
from becoming too large and also reduce the dependence on the step length.

The MSC model used in GEANT4 belongs to the class of condensed simulations. 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.

**Ionization**

The G4eIonisation class calculates the continuous and discrete energy losses of electrons and positrons
due to ionization in a material. Above a given threshold energy the energy loss is simulated by the ex-
PLICIT production of delta rays by Möller scattering \((e^- e^-)\), or Bhabha scattering \((e^+ e^-)\). Below the
threshold, soft electrons ejected are simulated as a continuous energy loss of the incident \(e^\pm\).

**Bremsstrahlung**

The class G4eBremsstrahlung calculates the energy loss of electrons and positrons due to the radiation
of photons in the field of a nucleus. 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 conti-
uous energy loss. The equation used for this simulation are the very similar to those used in GEANT3.
The only important difference is that in GEANT4 both the Landau-Pomeranchuk-Migdal effect and the
dielectric supression of bremsstrahlung have been implemented. Only the dielectric suppression is im-
plemented in GEANT3, where it is referred to as the Migdal correction.

**Positron and electron annihilation**

For the positron physics, this is also the Standard Energy Package which is implemented. Concerning
the positron/electron annihilation, the class GatePositronAnnihilation simulates the in-flight annihilation
of a positron with an atomic electron. 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. The \(\gamma\gamma\) non-colinearity is taken into account. The mean value of the angle distribution is \(\simeq 0.5^0\).

### 6.1.3 Changing a process selection

The physical processes which specified for electron and positron transport are hard coded in the software
and can not be altered. User can define the photon processes list and for each process, the user can select
the specific energy model with the following commands:
6.2 Setting the cut

- **standard**: use standard model (SEP);
- **lowenergy**: use low-energy model (LEP);
- **inactive**: do not simulate the interaction.

### 6.1.4 Example of process selection

An example of a script command lines to define a complete photon physic list:

```
/gate/physics/gamma/selectPhotoelectric lowenergy
/gate/physics/gamma/selectCompton standard
/gate/physics/gamma/selectRayleigh inactive
```

### 6.2 Setting the cut

The cut functions in GATE define some thresholds for the production of secondary particles. We have 2 sorts of cuts, the cut in range and the cut in energy.

Concerning the cut in range, this is the definition of the minimal distance for allowing production. This cut is not applied close to the volume boundaries and it is not applied for some particles.

The cut in energy defines the minimum energy for allowing production. This cut is always applied for low energy processes.

#### 6.2.1 Cut for the electrons, X-ray and Delta-ray

GATE allows the user to set three types of cuts that are applied on electron, X-ray and $\delta$ray (low-energy electrons which are produced by some atomic interactions).

The cut which is applied on electron is a cut in range and we can define it with the following command line:

```
/gate/physics/setElectronCut 30. m
```

In this example, if the electron range is less than 30 meters, the particle is not generated.

For X-ray and $\delta$ray, energy cuts can be applied as:

```
/gate/physics/setXRayCut 1. GeV
/gate/physics/setDeltaRayCut 1. GeV
```

In this example, if the X-ray or $\delta$ray energy is less than 1 GeV, particles are not generated.

#### 6.2.2 Example of a complete physic list

```
/gate/physics/gamma/selectPhotoelectric lowenergy
/gate/physics/gamma/selectCompton standard
/gate/physics/gamma/selectRayleigh inactive
/gate/physics/setElectronCut 30. m
/gate/physics/setXRayCut 1. GeV
/gate/physics/setDeltaRayCut 1. GeV
```

- 90 -
6.3 Initialize the physics

6.3.1 Why?

The initialization must be done after the geometry detector description and processes definition. This initialization build the geometry and the cross section tables for each process.

6.3.2 Command line

The command line to initialize the physics is:

/run/initialize
Chapter 7

Activity, source, voxellized phantom

7.1 The GPS (General Particle Source)

To introduce a source into a GATE simulation, the user needs first to define the properties of the activity distribution, then the geometry. GATE can model complex activity distributions. For each new event, the source manager randomly decides, based on total source activities, which source decays. It’s then the chosen source that takes care of generating one or more primary particles. The Geant4 “General Particle Source” is used to shoot particles of a given type to a given direction with a given kinetic.

7.2 Create a source

Simulations can use multiple sources. Each source is independent. It has a user-defined name and its own command tree:

/gate/source/NAME

where "NAME" defines the arbitrarily user-defined name of the source.

7.2.1 Adding a source

The first step is to add the source. To add a source, the user simply types the following GATE command:

/gate/source/addSource NAME

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.

7.2.2 Defining a type of source

There is a number of options available to you to suit your specific applications: you can define the particle type explicitly by name (ion, $e^-$, $e^+$, gamma, etc) with all its properties, or use helper keywords (e.g. Fluor18, backtoback) to help define one or more of those properties. Finally, there are also specialized sources (fastI124) that take care of most property definitions.
7.2 Create a source

Define the type

**Ion source**  The ion 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 radioactive decay and 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 it’s 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 to simulate 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 example above, more properties of fluorine-18 have been added: half-life and energy distribution, through the helper keyword Fluor18 (see below). Note that 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 reflect this.

**Helper keywords**

These keywords are defined to help in the definition of particle properties. The first three define an energy distribution for positrons from common beta emitters, and the last one defines a special two-particle source.

**Fluor18**  This keyword is to be used on the `../gps/energytype` command to define the energy spectrum of positrons from fluorine-18. Note that the keyword only defines the energy spectrum, you still have to specify the particle ($e^+$) and the half-life.

**Oxygen15**  This keyword is to be used on the `../gps/energytype` command to define the energy spectrum of positrons from oxygen-15. Note that the keyword only defines the energy spectrum, you still have to specify the particle ($e^+$) and the half-life.
7.2 Create a source

**Carbon11**  This keyword is to be used on the .../gps/energytype command to define the energy spectrum of positrons from carbon-11. Note that the keyword only defines the energy spectrum, you still have to specify the particle \( (e^+) \) and the half-life.

**backtoback**  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 to select the emission angles.

To use the back-to-back source:

```
/gate/source/NAME/setType backtoback
```

Note that there is no radioactive decay simulated when using the backtoback type and that you still have to define the particle (gamma), energy type (Mono) and value (0.511 MeV).

**Specialized source**

**FastI124**  This specialized source implements a simplified decay scheme of the non-pure beta emitter iodine-124 in which: positrons are emitted (but not neutrinos), there is no nuclear recoil, gammas are emitted if their emission probability is \( > 1 \% \), and no atomic de-excitation occurs (no x-rays, Auger electrons). These simplifications allow for an increase in speed with respect to the ion source while retaining important features of iodine-124, i.e. gammas may be emitted concurrently with positrons to possibly create "dirty" coincidences. Since decay is simulated, branching ratios are respected hence no activity compensation is necessary. To use the fastI124 source:

```
/gate/source/NAME/setType fastI124
```

The source takes care of particle definitions (gamma, positron) and energy distribution so there is no need to specify a particle or mention energy.

**Defining 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).

### 7.2.3 Defining energy

**Energy distribution**

If the type of energy distribution is not taken care of by the source (e.g. fastI124), then it has to be explicitly defined. This can be achieved either by using a pre-defined spectrum (see helper keywords above) or by using built-in distributions.
7.2 Create a source

**Built-in distributions** Candidates for built-in energy distributions are: mono-energetic "Mono", linear "Lin", powerlaw "Pow", exponential "Exp", Gaussian "Gauss", bremsstrahlung "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 monoeenergetic distributions (like backtoback 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
```

different fastI124
```
/gate/source/NAME/setType fastI124
```

7.2 Create a source

### 7.2.4 Define the distribution of the emission

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

```
/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

### 7.2.5 Define the shape of the source

When creating a 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 chose. 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
```
7.2 Create a source

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, one gives the radius (“radius”), for an annulus one gives the inner (“radius0”) and outer radii (“radius”), and for an ellipse, a square, or a rectangle one gives the half-lengths in x (“halfx”) and y (“halfy”). A sphere can be defined simply by specifying the radius (“radius”). Ellipsoids are defined giving their half-lengths in x (“halfx”), y (“halfy”), and z (“halfz”). Cylinders are defined such that the axis is parallel to the z-axis, the user is therefore required to give the radius (“radius”) and the z half-length (“halfz”). Parallelepipeds are defined by giving x (“halfx”), y (“halfy”), and z (“halfz”) half-length, plus the angles alpha (“paralp”), theta (“parthe”), and phi (“parphi”).

7.2.6 Define the placement of the source

The position of the source distribution is then defined by the following command:

```
/gate/source/NAME/gps/centre 1. 0. 0. cm
```

In that example, the centre of the source distribution is 1 cm off-center along the x-axis.

7.2.7 Confine a source: movement of a source

In order to be able to define sources in movement, the source distribution has 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: GATEnname + ’_P’.

One should note that the confinement slows down the simulation, the confinement volume should have an intersection with the GPS shape, and the confinement volume should 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:

```plaintext
# 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
```
7.2 Create a source

/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

7.2.8 Example: two gamma

The following example shows 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
# There is no confinement: the spatial distribution of emission points
# is specified by the shape defined above
/gate/source/twogamma/gps/confine NULL
# 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
7.2 Create a 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

7.2.9 Define a cold source

In order to be able to define a cold area 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 which defined the cold area:

/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

Finally you will describe your source with the Forbid command to associate the cold volume

/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

7.2.10 Visualizing a source

To confirm that sources are at the right location in the geometry, you can use the
7.3 Voxellized phantom

7.3.1 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 now 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.

7.3.2 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.
7.3 Voxellized phantom

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
```
7.3 Voxellized phantom

Figure 7.2: A simple voxellized phantom a) without transparency, b) with transparency.

```plaintext
/gate/CCD/imageReader/readFile ccd300Phantom.dat

# Place the phantom and rotate it so that it is in the XZ plane
/gate/CCD/placement/setTranslation 0 -82.269 0 mm
/gate/CCD/placement/setRotationAxis 1 0 0
/gate/CCD/placement/setRotationAngle 90 deg

# Attach the phantom SD and the output module
/gate/CCD/attachVoxelPhantomSD
/gate/CCD/addOutput doseOutput
/gate/output/doseOutput/saveUncertainty true
/gate/output/doseOutput/setFileName ccdDose.bin

Comments

Depending on the phantom dimensions, the use of a parameterizedBoxMatrix may increase memory usage by up to a factor of 2 and increase CPU time by 5-50%.

If you plan to collect only dose in the phantom, it is suggested that you disable other types of output, for example:

``/gate/output/ascii/disable``

Dose calculations

The relative uncertainty on dose is calculated on a per voxel basis. Let \{d_i\} i = 1,\ldots,N be the sequence of energy deposits in a given voxel, we can calculate the following quantities:

mean energy deposit:

$$\bar{d} = E(d) = \frac{1}{N} \sum_{i=1}^{N} d_i$$
7.3 Voxellized phantom

Sample variance:

\[ s^2 = E(d^2) - E(d)^2 \]

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

Population variance estimator:

\[ s^2 = \frac{N}{N-1} s^2 \]

Standard deviation:

\[ s = s \sqrt{\frac{N}{N-1}} \]

Standard error of the mean:

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

**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.
7.3 Voxellized phantom

7.3.3 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).

7.3.4 Tracking speed-up: 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
```
the second obligatory command is

```
/gate/world/daughters/insert fictitiousVoxelMap
```
7.3 Voxellized phantom

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/gamma/selectFictitious volumeTrace

This command introduces a master process for gammas which handles the fictitious interactions. The real physical processes are invoked by the usual commands. At present, fictitious interaction tracking can be used together with standard processes (Compton, photoelectric) as well as low-energy Compton and Rayleigh. The other gamma processes (i.e. GammaConversion) should be set inactive.

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 specified 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 electron range cut in the phantom can be set by using the command
/gate/NAME/setElectronCut 50 cm

In this way, a different electron range cut in the phantom than in the rest of the simulation can be defined. If the electron range cut is for example chosen very large (e.g. 50 cm), electrons are suppressed within the voxelized phantom. This yields additional acceleration. 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/voxelphantom
/gate/world/daughters/insert fictitiousVoxelMap
#/gate/voxelphantom/setFictitiousEnergy 10 keV
/gate/voxelphantom/setElectronCut 50 cm
#/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 interfilenam.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/gamma/selectFictitious volumeTrace
/gate/physics/gamma/selectGammaConversion inactive
/gate/physics/gamma/selectRayleigh inactive
/gate/physics/gamma/selectPhotoelectric standard
/gate/physics/gamma/selectCompton standard
/gate/physics/setXRayCut 1. GeV
/gate/physics/setDeltaRayCut 1. GeV
/gate/physics/verbose 0
/gate/physics/setElectronCut 0.1 mm

[...]

7.3.5 Voxellized source

There are two possibilities within GATE to read in voxellized sources. One can chose to read in 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 voxellized source should contain. These three dimensions are followed by a sequence of numbers, one number for each voxel. These numbers represent 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 voxellized source will be used.
/gate/source/addSource voxel voxel
# always use the keyword voxel first to declare the type
# Declaration that the voxellized 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 deafult position of the voxellized source is in the 1^{st}
7.3 Voxellized phantom

# quarter. So the voxellized 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 for 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 voxellized 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
7.3 Voxellized phantom

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

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).
7.4 Real time motion management for voxellized source and phantom

Gate offers some commands to define in the simulation set-up, Time Activity Curves (TACs) in a voxellized phantom. With this approach, users could combine and manage organ motions and tracer kinetics. In the following section, examples will illustrate how to use this capability and how to modify the global simulation script to perform these options.

7.4.1 Management of the voxelized phantom motions

The strategy which is recommended follows the 2 next items:

- Generate N frames for the phantom corresponding to the global acquisition time - Example: 50 frames for an acquisition of 5s - each frame represents 100 ms;
- The name structure of your 50 frames should be define with BaseFileName_atn_frameNumber for the attenuation map - Example if BaseFileName is NCAT: NCAT_atn_1.i33, NCAT_atn_2.i33 ... NCAT_atn_N.i33 should be define in your directory;
- For emission map, the approach will be the same - Example if BaseFileName is NCAT: NCAT_act_1.i33, NCAT_act_2.i33 ... NCAT_act_N.i33 should be define in your directory.

Don’t forget that the image input format is always the Interfile format as explain in previous section of this guide. In consequence, the binary matrix is associated to a header file - NCAT_header.h33 according to our example - which looks like:

```
!matrix size [1] := 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
```

– Important remark regarding this section –
REMOVED THE COMMAND

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

Which is generally used in the beginning of main macro

Assuming that the time phantom sequences are correctly defined, you can built the attenuation matrix as it is done in the next example:

```
# M A T R I X

# Choose a phantom volume name
#
/gate/world/daughters/name Ncat
```
# Use a voxelized navigator
#
/gate/world/daughters/insert compressedMatrix

# Call the reader for interfile format
#
/gate/Ncat/geometry/insertReader interfile

# Insert the "RTVPhantom" module
#
/gate/RTPhantom/insert RTVPhantom

# Attachment to the phantom volume name
#
/gate/RTVPhantom/AttachTo Ncat

# Define the base Name of the image sequences
# and the path directory where data are stored
#
/gate/RTVPhantom/setBaseFileName NCAT

# The name of the header file which is associated
# to the binary images
#
/gate/RTVPhantom/setHeaderFileName NCAT_header.h33

# Number of frames (images)
# Time sampling per frame (image)
#
/gate/RTVPhantom/SetNumberOfFrames 50
/gate/RTVPhantom/SetTimePerFrame 0.1 s

# Following commands are the same which are used in
# the classical case of voxelized phantom using
#
/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

## 7.4.2 Management of the Time activity Curves description

We can use the same set of images to define the activity distribution. The global approach is quiet similar to the voxelized source mechanism, including some modifications which are explained in the next command line examples:
7.4 Real time motion management for voxellized source and phantom

# V O X E L   S O U R C E

# voxel source
#
/gate/source/addSource voxel voxel

/gate/RTVPhantom/AttachToSource voxel
/gate/source/voxel/reader/insert interfile
/gate/source/voxel/interfileReader/translator/insert range

# TIME ACTIVITY option
#
/gate/source/voxel/interfileReader/rangeTranslator/readTable activityRange.dat
/gate/source/voxel/interfileReader/SetTimeActivityTablesFrom TimeActivity_Tables.dat
/gate/source/voxel/interfileReader/SetTimeSampling 0.1 s

/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

The activityRange.dat is a text file where the user defines the label or pixel range value included activity. In this ascii file example, we consider that the activity is define for voxel values 46 and 261. With no background activity in the voxel value 46 and 10 Bq/voxel for voxel value 261.

2
46 46 0
261 261 10

The TimeActivity_Tables.dat will define the TACs files which are associated to the voxel values defined in the previous data file.

2
46 46 lesion.dat
261 261 liver.dat

Lesion.dat and liver.dat are 2 text files which contain the TACS for these organs, as follow:

11
0.0 0.01
### 7.4 Real time motion management for voxellized source and phantom

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>Activity (Bq/voxel)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>100</td>
</tr>
<tr>
<td>0.2</td>
<td>200</td>
</tr>
<tr>
<td>0.3</td>
<td>300</td>
</tr>
<tr>
<td>0.4</td>
<td>500</td>
</tr>
<tr>
<td>0.5</td>
<td>800</td>
</tr>
<tr>
<td>0.6</td>
<td>1000</td>
</tr>
<tr>
<td>0.7</td>
<td>700</td>
</tr>
<tr>
<td>0.8</td>
<td>400</td>
</tr>
<tr>
<td>0.9</td>
<td>200</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
</tr>
</tbody>
</table>

Where first column is the time in second and the second one is the activity in Bq/voxel at time t.
Chapter 8

Digitizer and readout parameters

8.1 General Purpose

The purpose of the digitizer module is to simulate the behavior of the scanner’s detectors and signal processing chain. In this section, the algorithms used to simulate a scanner’s electronic readout scheme are described. To introduce them, an overview of the main steps used to produce coincidences from the simulated particle trajectories information is briefly described below. Following this, 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.

8.1.1 From particle detection to coincidences in GATE

GATE makes use of 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 composed as follows:

1. A particle is generated, with its parameters, such as initial type, time, momentum, and energy.

2. 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 [5].

3. If a step occurs within a volume corresponding to a sensitive detector (Chapter 5), 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.

4. 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.
5. 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 are 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 done 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 sets of Singles, that are detected within a given time interval (the so called 'coincident events').

Finally, the coincidences data can be filtered-out (if so required) to mimic any possible data loss which could occurs in the coincidence logical circuit or during the data transportation. As for the singles data, the treatment is done 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.

### 8.1.2 Role of the digitizer

As mentioned above, the information contained in the hit does not correspond to what is accessible 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. In order 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 hits 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 is explained in the next sections.

8.2 Digitizer modules

The digitization consists of a series of signal processors. The output at each step along the series is defined as a pulse. At the end of the chain, the output pulses are named singles. These singles realistically simulate the physical observables of a detector's response to a particle interacting with it. An example is shown figure 8.1.

![Diagram](image)

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 observables seen from the detector.

To specify a new signal-processing module (i.e. add a new processing unit in the readout scheme), in the macro file one have to use the following command template:

```
/gate/digitizer/insert MODULE
```

Where MODULE is the name of the digitizer module. Note that, the order of the module declaration should make sense. The data flow follows the same order than the modules declaration in the macro. In a typical scanner the following sequence works well, however, it is not mandatory (the module names will be explained in the rest of the chapter):

- insert adder before readout
- insert readout before thresholder/upholder
- insert blurring before thresholder/upholder

The available modules are explained in the following sections.


8.2 Digitizer modules

8.2.1 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 into GATE, under this term.

Five type of distributions 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 in parameter. This method is more adapted for large number 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 ones described above, that is Flat, Gaussian, Exponential, Manual or File.

Once the distribution is created, the related parameters can be set:

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

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

8.2.2 Adder

Definition

One particle often creates multiple interaction, 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.
<table>
<thead>
<tr>
<th>Parameter name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>FLAT DISTRIBUTION</strong></td>
<td></td>
</tr>
<tr>
<td>setMin</td>
<td>set the low edge of the range where the function is not null (default is 0).</td>
</tr>
<tr>
<td>setMax</td>
<td>set the high edge of the range where the function is not null (default is 1).</td>
</tr>
<tr>
<td>setAmplitude</td>
<td>set the value taken by the function within the non null range (default is 1).</td>
</tr>
<tr>
<td><strong>GAUSSIAN DISTRIBUTION</strong></td>
<td></td>
</tr>
<tr>
<td>setMean</td>
<td>set the mean value of the distribution (default is 0).</td>
</tr>
<tr>
<td>setSigma</td>
<td>set the standard deviation of the distribution (default is 1).</td>
</tr>
<tr>
<td>setAmplitude</td>
<td>set the amplitude of the distribution (default is 1).</td>
</tr>
<tr>
<td><strong>EXPONENTIAL DISTRIBUTION</strong></td>
<td></td>
</tr>
<tr>
<td>setLambda</td>
<td>set the power of the distribution (default is 1).</td>
</tr>
<tr>
<td>setAmplitude</td>
<td>set the amplitude of the distribution (default is 1).</td>
</tr>
<tr>
<td><strong>MANUAL DISTRIBUTION</strong></td>
<td></td>
</tr>
<tr>
<td>setUnitX</td>
<td>set the unit for the x axis.</td>
</tr>
<tr>
<td>setUnitY</td>
<td>set the unit for the y axis.</td>
</tr>
<tr>
<td>insertPoint</td>
<td>insert a new point, giving a pair of (x,y) values.</td>
</tr>
<tr>
<td>addPoint</td>
<td>add a new point, giving its y value, and auto incrementing the x value.</td>
</tr>
<tr>
<td>autoXstart</td>
<td>in case of auto incremental x value, set the first x value to use.</td>
</tr>
<tr>
<td><strong>FILE DISTRIBUTION</strong></td>
<td></td>
</tr>
<tr>
<td>setUnitX</td>
<td>set the unit for the x axis.</td>
</tr>
<tr>
<td>setUnitY</td>
<td>set the unit for the y axis.</td>
</tr>
<tr>
<td>autoX</td>
<td>specify if the x values are read from file or if they’re auto-incremented.</td>
</tr>
<tr>
<td>autoXstart</td>
<td>in case of auto incremental x value, set the first x value to use.</td>
</tr>
<tr>
<td>setFileName</td>
<td>the name of the ASCII file where the data have to be read.</td>
</tr>
<tr>
<td>setColumnX</td>
<td>which column of the ASCII file contains the x axis data.</td>
</tr>
<tr>
<td>setColumnY</td>
<td>which column of the ASCII file contains the y axis data.</td>
</tr>
<tr>
<td>read</td>
<td>do read the file (should be called after specifying all the other parameters).</td>
</tr>
</tbody>
</table>

Table 8.1: Summary of the parameters for each distribution type
8.2 Digitizer modules

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 the chapter 4). This consideration implies two requirements in order to use the adder module:

1. A registered system must be used to describe the geometry (also the mother volume’s name must correspond to a registered system name)

2. The lowest level of this system must be attached to the detector volume and it 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 crystals 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 first hit’s time

Command line

The command to use the adder module is

/gate/digitizer/Singles/insert adder

8.2.3 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 pulse with maximum energy (winner-takes-all).
8.2 Digitizer modules

The importance of this command line is illustrated through the following example from a PET system (see Chapter 4). In a cylindricalPET system, where the first volume level is rsector, the second volume level is module, as is shown in figure 8.3, the readout depth depends upon how the electronic readout functions.

If one PMT reads the four modules in axial direction, declare the depth 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), declare the depth with the command,

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

Figure 8.2: Actions of the adder and readout modules.
8.2 Digitizer modules

In this case, the energy of the single event is the sum of the energies of the pulses inside the red box (module) of the 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’s response and should include the behaviors of the photo-detector, electronics, and acquisition system.

8.2.4 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, $E_0$. By examining a Poisson process, it is seen that the resolution is a function of energy, $R(E)$, that is proportional to $1/\sqrt{E}$. The resolution is equal to:

$$ R = \frac{R_0 \sqrt{E_0}}{\sqrt{E}}. \quad (8.1) $$

To define Gaussian distribution determined by its mean, $m$, and its standard deviation, $\sigma$. The resolution of a Gaussian distribution is given by:

$$ R = 2 \sqrt{2 \ln 2} \frac{\sigma}{m} \approx 2.35 \frac{\sigma}{m}. \quad (8.2) $$
8.2 Digitizer modules

Command line

Using the blurring module is as follows:

- Insert the blurring module,
- set the resolution,
- and set the reference energy.

For example, to apply a resolution of 15% @ 511 keV, use the following commands:

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

8.2.5 Crystal Blurring for a block detector

This type of blurring is used for the scanners where all the detectors are 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. In addition, in order to model the efficiency of the system a coefficient (between 0 and 1) can be set.

Command line

As an example, to set 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% use 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 parameter, crystal quantum efficiency\(^1\) (setCrystalQE), is fixed and represents the probability to detect the event by the photo-detector.

8.2.6 Local Blurring for a detector module with several types of crystals

The LocalBlurring module is very similar to the Blurring module, but in this case, different 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).

\(^1\)This parameter represents the effect of the transfer efficiency of the crystal and the quantum efficiency of the photo-detector.
8.2 Digitizer modules

Command line

To apply the *LocalBlurring* module, use the following:

- 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, use the following commands:

```plaintext
/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 !!

8.2.7 Intrinsic resolution blurring with crystals of different composition

For a phoswich detector, in order to more precisely simulate the energy resolution, a better model than the linear law of equation 8.1 is required.

Definition

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

\[
R = \sqrt{2.35^2 \cdot \frac{1 + \tilde{\nu}}{N_{ph} \cdot \tilde{\epsilon} \cdot \tilde{p}} + R_i^2}
\]  

(8.3)

where \(N_{ph} = LY \cdot E\) and \(LY, \tilde{p}\) and \(\tilde{\epsilon}\), are Light Yield, Transfer, and Quantum Efficiencies for each crystal. \(\tilde{\nu}\) is the relative variance of the gain of a Photo Multiplier Tube (PMT) or an Avalanche Photo Diode (APD); it is hard-coded 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*. After defining these quantities, the command lines for multi-crystal type blurring are given below.

Definition of the Light Yield

The *Light Yield* pulse-processor simulates the crystal’s light yield. Each crystal must be given the correct light yield. This module converts the pulse’s 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’s 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 photodetector, which can be a Photo Multiplier Tube (PMT), or an Avalanche Photo Diode (APD).

Command lines for using intrinsic resolution

To introduce the command lines, 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 ellipsis). The *digitizer* command lines are:

```bash
# LSO layer
/gate/crystal/daughters/name LSOlayer
....
# BGO layer
/gate/crystal/daughters/name LuYAPlayer
....
# ATTACH SYSTEM
....
/gate/systems/cylindricalPET/crystal/attach crystal
/gate/systems/cylindricalPET/layer0/attach LSOlayer
/gate/systems/cylindricalPET/layer1/attach LuYAPlayer
# ATTACH CRYSTAL SD
/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 8.3, the parameters discussed above must be defined for each layer (i.e. Light Yield, Transfer, Intrinsic Resolution, and the Quantum Efficiency).

```bash
# 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
```

```bash
# DEFINE LIGHT YIELD FOR EACH LAYER
```

A complete example of a phoswich module is found in the PET benchmark in the CVS repository of Gate
8.2 Digitizer modules

/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 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’s files.

To set multiple quantum efficiencies using files (fileName1, fileName2, and ... for each of the different modules), use the following commands:

/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 volume crystal is a daughter of a volume module which is an array of 8 × 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 these files for each modules.

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 phoswich module, it’s important to base the threshold on this number and not on the real energy. In this situation, to apply a threshold at this step of the digitizer
8.2 Digitizer modules

In this case, the GATE program knows that these modules have been used, and it will apply threshold based upon the number $N_{\text{pe}}$ rather than energy. The threshold, set with this sigmoidal function in energy unit by the user, is translated into number $N_{\text{pe}}$ with the lower light yield of the phoswich 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 $N_{\text{phe}} \rightarrow E_{\text{nergy}}$ (allows energy "recalibration"). 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.

### 8.2.8 Crosstalk

**Definition**

The *crosstalk* module of the pulse-processor is for simulating the optical and/or an 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 functions only for a chosen volume that is an array repeater !!!

**Command line**

To insert a crosstalk module that distributes 10% of input pulse energy for the adjacent crystals and 5% to the corner crystals, use the following commands:

```
/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, within each neighbor of the crystal that received the pulse, a pulse is created that has 10% (5% for each corner crystals) of initial energy of the pulse.
8.2 Digitizer modules

8.2.9 Thresholder & Upholder

Definition

The Thresholder/Upholder modules allow the user to apply an energy window cut to remove 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 scatter reduction, image reconstruction, and count rate performance.

Command line

In a typical PET scanner, the energy selection for the photo-peak is done by the following commands. Using a low threshold of 0 \( keV \) allows the user to see all the events, and is often useful for debugging a simulation.

\[
\text{/gate/digitizer/Singles/insert thresholder} \\
\text{/gate/digitizer/Singles/thresholder/setThreshold 250. keV} \\
\text{/gate/digitizer/Singles/insert upholder} \\
\text{/gate/digitizer/Singles/upholder/setUphold 750. keV}
\]

8.2.10 Sigmoidal thresholder

Definition

The Sigmoidal thresholder represents a pulse processing class that models a threshold discriminator based on a sigmoidal function\(^3\),

\[
\sigma(E) = \frac{1}{1 + \exp\left(\alpha \frac{E - E_0}{E_0}\right)},
\]

where the parameter \( \alpha \) 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 \( \text{(setThreshold)} \), the percentage of acceptance for this threshold \( \text{(setThresholdPerCent)} \), and the \( \alpha \) parameter \( \text{(setThresholdAlpha)} \). 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. If, on the other hand, this criteria is not met the input pulse is discarded.

Command line

\[
\text{/gate/digitizer/Singles/insert sigmoidalThresholder} \\
\text{/gate/digitizer/Singles/sigmoidalThresholder/setThreshold 250 keV} \\
\text{/gate/digitizer/Singles/sigmoidalThresholder/setThresholdAlpha 60.} \\
\text{/gate/digitizer/Singles/sigmoidalThresholder/setThresholdPerCent 0.95}
\]

\(^3\)A sigmoidal function is an S-shaped function of the form, \( \sigma(x) = \frac{1}{1+\exp(-ax)} \), that acts as an exponential ramp from 0 to 1.
8.2.11 Temporal 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 rather than 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`

8.2.12 Spatial blurring for SPECT

For SPECT analysis, the spatial resolution is assumed to follow a Gaussian distribution determined from its width $\sigma$.

Command line

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

8.2.13 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 this crystal is based on the crystal with the highest energy deposited. Without additional spatial blurring of the crystal, simulation results will always have a smaller 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
```

8.2.14 Noise

Definition

Different sources of background noise could exist in a PET/SPECT architecture. For example, the electronic can carry out its own noise, or some crystals used for the detection, such as LSO, contains radioactive nucleus, which can participate to the background detection count rate. Within GATE, the noise module is aimed to add such background events, in a totally generic way, so that any kind of source
of noise can be simulated. To achieve this, the energy, as well as the inter-event time interval are chosen randomly, for each event, into user defined distributions, by using the mechanism described in the chapter 8.2.1.

**Command line**

In the following example, one introduce a noise source, whose energy is distributed according to a Gaussian law, and whose time distribution follows a Poisson process. To do this, one first define the two necessary distributions. Let note that, since the noise description uses the distribution of the time interval between consecutive events, one has to define an exponential distribution.$^4$

```plaintext
/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 attributed for those noise events.

### 8.2.15 Local efficiency

**Definition**

The different crystals, or group of crystals, composing a PET/SPECT can be characterized by their own efficiency. GATE offers a way to describe such efficiency per crystal or volume that the user has to define.

In order to define the efficiency distribution in the scanner, one can specify which level of the volume hierarchy of the system are differentiated (see following examples). Then the distribution of efficiency, for each differentiated volumes, is specified via a generic distribution, as described in the section 8.2.1.

**Command line**

In the following examples, one assume 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).

```plaintext
/gate/distributions/name block_eff_distrib
/gate/distributions/insert File
```

$^4$If the probability of detecting $k$ events in a time interval of $t$ is distributed along a Poisson law $P_1(k, t) = e^{-\lambda t} \left(\frac{\lambda t}{k!}\right)^k$, then the probability density of having a time interval in the range $[t; t + dt]$ between two consecutive events is given by $dP_2(t) = \lambda e^{-\lambda t} dt$. 

- 130 -
8.2 Digitizer modules

In the second example, one specify 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`.

Finally, in the last example, each crystal have its own efficiency, described in the file `eff_per_crystal.dat` containing $8 \times 64$ elements.

8.2.16 Memory buffers and bandwidth

To mimics the effect of limited transfert rate, a module allows to simulate 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 $\nu$ and the memory depth $D$. Moreover, two reading methods can be modelized, 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 limit data rate is then limited to $\nu$, while in the second method, the limit data rate is equal to $D \cdot \nu$. When the size limit is reached, any new pulse is rejected, until the next reading clock tick arrival which will free a part of the buffer. In such a case, a non null buffer depth allows to face to a local rise of the input data flow.
8.2 Digitizer modules

Command line

To specify a buffer, read at 10 MHz, with a buffer depth of 64 events, in a mode where all the 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 singles chain that the user has defined. The size of the buffer represent the number of elements, in this example 64 Singles, that the user can store in a buffer. To read the buffer in an event by event basic, one should replace the last line by setMode = 0

8.2.17 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 combination in term 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 a output energy defined by the sum of the energies ($E_{\text{out}} = \sum_{i=0}^{N} E_i$) and a time set to the last time of the last pulse participating to the pile-up $t_{\text{out}} = t_N$.

Since multiple events are regrouped into a unique one with the pile-up effect, one should consider it as the removal of the events occurring during a given time length which can be seen as a dead time effect. Moreover, since the pile-up end time are always updated with the last single occurring, the effect is more or less represented by a paralysable dead-time.

Command line

To declare a pile-up corresponding to a signal formation time of 100 ns, and made in a a module corresponding of the crystal regroupments represented 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

8.2.18 Dead time

Because of physical reasons, or due to the shaping time of signals or for any other reasons, each detection of a single event can lead to hide the subsequent singles, occurring on the same electronic module, and for a certain amount of time, depending on the characteristics of the detectors used as well as the readout electronics. This effect of dead time is modelized in GATE and is presented below. Two models of the dead-time behavior of the counting system have been implemented in the digitizer: paralysable and
**nonparalysable** response. These models represent the idealized behavior and can be implemented *event by event* during a simulation. The detailed workings of these models can be found in reference [17]. The fundamental assumptions made in these two models are illustrated in figure 8.4.

![Diagram](image)

Figure 8.4: For 7 incoming particles and a fixed dead-time $\tau$, the **nonparalysable** electronic readout will accept 3 particles, and the **paralysable** will accept only 1 particle (the dashed arrows represent the removed events, while the plain ones are the accepted singles).

**Definition**

The *dead time* module is applied to a specific volume within the Sensitive Detector system’s hierarchy. All events that take place within this volume level will trigger a dead-time detector response. This action of the digitizer simulates the time during which this detector, treating a particle, will not be able to treat 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 treated. 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**

For instance, to apply a dead-time to the `volume_name` (which has to be previously attached to a level of the system), use the following commands:

```
# 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
```
8.3 Multiple processor chains

The use of multiple processor chains allows maximum flexibility in the design of the digitizer and data output system. 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

The components of the standard processor chain must have the following commands for a standard PET (with BGO crystals) processor chain:

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

Then, to add the blurring filter to the "Single" branch:
8.4 Coincidence sorter

8.4.1 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. In the second method, all singles are opening 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 concrete case is sketched in the figure 8.5.

Finally, to exclude coincidence coming from the same particle that scattered from a block to an adjacent block, a test is done on the proximity of the two blocks forming the coincidence event. 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 so required.

Delayed coincidences

Each Single emitted from a given source particle is stored with an event ID number, which uniquely identify 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.
8.4 Coincidence sorter

Figure 8.5: Comparison between the two methods of coincidences finding, for a given succession of singles. In the first one (in the upper side), the S2 single don’t 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. In the other side, with the second method, where all the singles are opening their coincidence window, one find, with the same single order, 2 different coincidences.
8.5 Multiple coincidence sorters

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. In order to cover a large range of applications, GATE allows to choose between 9 different rules to apply in such a case. The list of rules along with their explanation is given in the table 8.2, and a comparison of the effects of each treatment rule for various cases of multiple coincidences is shown in the figure 8.6.

8.4.2 Command line

In order 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), add the additional command line,

```
/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, with 100 ns time shift, is to be simulated, one should specify the offset value with the command:

```
/gate/digitizer/Coincidences/setOffset 100. ns
```

To specify the depth of the system’s hierarchy under which the coincidences have to be made, one could use the following command:

```
/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>
```

8.5 Multiple coincidence sorters

Multiple coincidence sorters can be used in GATE. To create a coincidence sorter, the sorter must be named and a location specified for the input data. In the example below, three new coincidence sorters are created:
### Table 8.2: Table of available multiple policy, along with a description of their functionalities.

When a coincidence with \( n \) pairs is treated, it is first decomposed into a list of \( \frac{n(n-1)}{2} \) sub-pairs, which will be analyzed individually. In this table, the term "good" means that a sub-pair of singles are in coincidence and are separated by a number of blocks greater or equal to the `minSectorDifference` parameter of the coincidence sorter. The prefix "take" means that a one or more sub-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 last 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 the section 8.6.4). Finally, the "kill" prefix will obviously mean that all the event will be cleared, and will not produce any coincidence.

<table>
<thead>
<tr>
<th>Policy name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>takeAllGoods</td>
<td>Each good sub-pairs is taken.</td>
</tr>
<tr>
<td>takeWinnerOfGoods</td>
<td>Only the good sub-pair with the highest energy is taken.</td>
</tr>
<tr>
<td>takeWinnerIfIsGood</td>
<td>If the sub-pair with the highest energy is good, take it, otherwise, kill the event.</td>
</tr>
<tr>
<td>takeWinnerIfAllAreGoods</td>
<td>If all sub-pairs are goods, take the one with the highest energy (case a a veto logic implemented to suppress events with adjacent singles).</td>
</tr>
<tr>
<td>keepIfOnlyOneGood</td>
<td>If exactly one sub-pair is good, keep all the multicoincidence.</td>
</tr>
<tr>
<td>keepIfAnyIsGood</td>
<td>If at least one sub-pair is good, keep all the multicoincidence.</td>
</tr>
<tr>
<td>keepIfAllAreGoods</td>
<td>If all sub-pairs are goods, keep all the multicoincidence.</td>
</tr>
<tr>
<td>killAllIfMultipleGoods</td>
<td>If more than one sub-pairs is good, the event is seen as a real &quot;multiple&quot; and thus, all the event is killed.</td>
</tr>
<tr>
<td>killAll</td>
<td>No multiple coincidences are accepted, no matter how many good sub-pairs are present.</td>
</tr>
</tbody>
</table>
8.5 Multiple coincidence sorters

Figure 8.6: Comparison of the behavior of all the available multiple treatment policy, for various multiple coincidence situation. In the figure, the stars represents the singles detected in the event. The size of the star, as well as the number next to it, indicate the energy level of the single (ie. the single no 1 has more energy than the single no 2, which is itself more energetics than the single no 3). The lines represents the possibles good coincidences (ie. with a sector difference higher or equal to the minSectorDifference of the coincidence sorter). In the table, a minus(-) sign indicate that the event is killed (ie. no coincidence is formed). The ★ sign indicate that all the singles are kept into a unique multicoi ncidence, which will not be written to disk, but which could participate to data loss via dead time or bandwidth occupancy. In the other cases, the list of pairs which will be written to disk (unless being removed thereafter, by a possible filter applied to the coincidences) is indicated.
8.5 Multiple coincidence sorters

- 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
```

The scheme corresponding to this example is shown in the figure 8.7

![Diagram of coincidence sorters and readout scheme](image)

Figure 8.7: Readout scheme produced by the listing from the sections 8.3 and 8.5.
8.6 Coincidence treatment and filtering

8.6.1 Coincidence pulse processors

Once the coincidences are formed, some treatment should be applied on to reproduce sources of count loss that may occur because of the acquisition limitations. For instance, dead time can be associated to the coincidence unit. Another count loss may be due to the limited bandwidth of wires or buffer capacities of the I/O interfaces. In the following, the way to modelize such effects within GATE is explained. Moreover, in the case of scanners using delayed coincidence line, data coming from the two kind of coincidences (ie. prompts and delayed coincidences) can be formed by a unique coincidence processor, and thus, each coincidence type can affect the count rate of the other one. For instance, a prompt coincidences can produce dead time which will hide a consecutive delayed coincidence. Another possibility is that the prompts coincidence events saturate the bandwidth, so that the randoms events are partially hidden.

Modelisation of such effects imply to regroup the two different coincidences data into a unique one, which is treated by a unique filter.

Definition

A coincidence pulse processor is a structure which contains the list of coincidence sources onto which apply the set of filters, along with the list of filters themselves. Let note that the order of the list of coincidence may play a role on the repartition of the data loss between the prompt and the delay lines. For example, if the line of prompt coincidences has priority on the line of the delayed coincidences, then the events of the latter have more risk to be rejected by a possible buffer overflow than those of the first. This kind of effects can be suppressed by specifying than, inside an event, all the coincidences arriving with the same time flag re merged in a random order.

Command line

To implement a new coincidence pulse processor, merging two coincidence line into it, and apply a module XXX followed by another module YYY on the total data flow, one should use the following command, 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 treated with a random order, one should use the command:

```
/gate/digitizer/myCoincChain/usePriority false
```
8.6.2 Coincidence dead time

The dead time for coincidences is running in the same way that those 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 exists) is not yet implemented. Apart of this limitation, the command lines for coincidence dead time are identical to the one for singles dead time, and described in the section 8.2.18, page 132. Additionally in the case where more than one coincidence can occurs for a unique GEANT4 event (if more than one coincidence line are added to the coincidence pulse processor, or if multiple coincidences are treated as many sub-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

8.6.3 Coincidence buffers

Once again, the buffer module which is designed for coincidences uses exactly the same command lines and functionalities as the one made for single pulse lists, and described in the chapter 8.2.16, page 131.

8.6.4 Multiple coincidence supression

Definition

If the multiple coincidences are kept and not splitted into sub-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 so required, this supression could be done at any former coincidence treatment step, by inserting the multipleKiller module at the required level. This module has no parameter and just kill the multicoincidence events. Let not that multiple coincidences split into many sub pairs are not affected by this modules, and cannot be distinguished from the normal "simples" coincidences.

Command line

To insert a multipleKiller, one just have to use the syntax:

/gate/digitizer/myCoincChain/insert multipleKiller

8.7 A complete example

In this example, a complete digitizer section of a GATE macro file is examined line by line. The readout scheme produced by this macro, which is commented on below, is shown in figure 8.8.

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 3
8.7 A complete example

7  #       ENERGY BLURRING
8  /gate/digitizer/Singles/insert blurring
9  /gate/digitizer/Singles/blurring/setResolution 0.26
10 /gate/digitizer/Singles/blurring/setEnergyOfReference 511. keV
11 
12  #       LOW ENERGY CUT
13 /gate/digitizer/Singles/insert thresholder
14 /gate/digitizer/Singles/thresholder/setThreshold 50. keV
15 
16 /gate/digitizer/name cutLowSingles
17 /gate/digitizer/insert singleChain
18 /gate/digitizer/cutLowSingles/setInputName Singles
19 
20  #       NOISE
21 
22 /gate/distributions/name energy_distrib
23 /gate/distributions/insert Gaussian
24 /gate/distributions/energy_distrib/setMean 450 keV
25 /gate/distributions/energy_distrib/setSigma 30 keV
26 
27 /gate/distributions/name dt_distrib
28 /gate/distributions/insert Exponential
29 /gate/distributions/dt_distrib/setLambda 7.57 mus
30 
31 /gate/digitizer/cutLowSingles/insert noise
32 /gate/digitizer/cutLowSingles/noise setDeltaTDistributions dt_distrib
33 /gate/digitizer/cutLowSingles/noise setEnergyDistributions energy_distrib
34 
35  #       DEAD TIME
36 /gate/digitizer/cutLowSingles/insert deadtime
37 /gate/digitizer/cutLowSingles/deadtime/setDeadTime 2.2 mus
38 /gate/digitizer/cutLowSingles/deadtime/setMode paralysable
39 /gate/digitizer/cutLowSingles/deadtime/chooseDTVolume module
40 
41  #       HIGH ENERGY CUT
42 /gate/digitizer/name cutSingles
43 /gate/digitizer/insert singleChain
44 /gate/digitizer/cutSingles/setInputName cutLowSingles
45 /gate/digitizer/cutSingles/name highThresh
46 /gate/digitizer/cutSingles/insert thresholder
47 /gate/digitizer/cutSingles/highThresh/setThreshold 350. keV
48 /gate/digitizer/cutSingles/cut Singles/insert upholder
49 /gate/digitizer/cutSingles/upholder/setUphold 700. keV
50 
51 /gate/digitizer/cutSingles/name deadtime_cassette
52 /gate/digitizer/cutSingles/insert deadtime
53 /gate/digitizer/cutSingles/deadtime_cassette/setDeadTime 0.55 mus
54 
55 - 143 -
8.7 A complete example

Lines 1 to 15 The branch names "Singles" contains the result of applying, in order, the adder, readout, blurring, and a 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 µs. These two distributions are used to define a noise by attributing the Gaussian distribution to the energy level of the background levels, and the exponential as the distribution of time interval between two consecutive background events (lines 32-34).
Lines 37 to 40 A paralysable (line 39) dead time of 2.2 μs 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 application), composed of those events which pass through the 350 keV/700 keV threshold/ uphold window (lines 46-50). In addition, the events counted in this branch must pass the two dead-times 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" set ("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 A treatment will be done on the regroupment of the delayed and of the prompts coincidence lines (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’s inserted in the before the prompt line, and the priority is used (line 81). First, a non-paralysable dead time of 60 ns is applied on the delayed+prompts coincidences (lines 82-85). If more than one coincidences occurs inside a given event, the dead time can kill all of them or no one 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 on the delayed+prompt sum (lines 86-89).

8.8 Digitizer optimization

In GATE standard operation mode, primary particles are generated by the source manager, and then propagated through the attenuating geometry before generating hits in the detectors, which feed into the digitizer chain. While this operation mode is suited for production 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 many 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 are unnecessary.

To suit this specific situation, GATE offers an operation mode dedicated to digitizer optimization, known as DigiGATE. In this mode, hits are no longer generated: rather, 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 explained further in chapter 13.
8.8 Digitizer optimization

Figure 8.8: Readout scheme produced by the listing in section 8.7. The disk icons represents the data written to the GATE output files.
8.9 Angular Response Functions (ARFs) to speed-up simulations

The ARF is a function of the incident photon’s 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’s direction vector and the detection plane in an energy window of interest. The use of ARF tables is divided in three steps.

8.9.1 ARF Tables Data computation

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 to tell GATE to compute the ARF data is:

```
/gate/systems/SPECThead/arf/setARFStage generateData
```

The ARF data are stored in ROOT format from the GATE command output:

```
/gate/output/arf/root/setARFDataFileName testARFdata
```

By default the maximum size of a ROOT file is 1.8 Gb. Once the file has reached this size, ROOT automatically closes it and open 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 summarizes the commands listed before.

8.9.2 Computation of the ARF tables from previous stage

Data have been stored to ROOT files, we may compute the tables and store them to a binary file:

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

We define the digitizer parameters for their computation:

```
# DIGITIZER PART OF THE ARF SENSITIVIE DETECTOR
#/gate/systems/SPECThead/ARFTables/setEnergyDepositionThreshold 328. keV
#/gate/systems/SPECThead/ARFTables/setEnergyDepositionUpperLimit 400. keV
#/gate/systems/SPECThead/ARFTables/setEnergyResolution 0.10
#/gate/systems/SPECThead/ARFTables/setEnergyOfReference 140. keV
```

Here we shot photons with $364.5 \text{ keV}$ as kinetic energy. We choose an energy resolution of 10% and we define 328 $\text{keV}$ and 400 $\text{keV}$ as the energy window boundaries. The energy of reference is chosen to be 140 $\text{keV}$ which allows to define the FWHM of the response of the detector:
Angular Response Functions (ARFs) to speed-up simulations

\[ fwhm = 0.10 \times \sqrt{140 \times E_{dep}} \]

where \( E_{dep} \) is the photon deposited energy. If we don’t want to consider photons which deposit less than 130 keV we may use following commands:

```
# if we are not interested in photons which deposited less than 130 keV
# we put
#/gate/systems/SPECTHead/setEnergyDepositionThreshold 130. keV
```

The ARF tables depend strongly on the distance from the detector to the source we used in the previous stage. We have to enter this parameter to get proper tables. The detector plane is set to be the half-middle plan of the detector part of the SPECT head. In the given example, we set the translation of the SPECT head to be 34.5 cm along the X direction (radial axis), the detector is 2 cm wide along X and its center is 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 AT HALF DIMENSION OF THE CRYSTAL RESPECTIVELY
# TO THE SPECTHEAD FRAME
# here it is 34.5 cm + 1.5 cm
#/gate/systems/SPECTHead/ARFTables/setDistanceFromSourceToDetector 36 cm
```

Now we compute the tables from a text file which contains informations 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 datas for each energy window
# Inident 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 the first one are stored inside 20 files whose base file name is test1head which were generated in the previous stage. It means the ARF data are stored in test1head.root, test1head_1.root ... test1head_19.root. Finally we store the computed tables to a binary file:
8.9 Angular Response Functions (ARFs) to speed-up simulations

/gate/systems/SPECThead/ARFTables/list
   # SAVE ARF TABLES TO A BINARY FILE FOR PRODUCTION USE
/gate/systems/SPECThead/ARFTables/saveARFTablesToBinaryFile ARFSPECTBench.bin

8.9.3 Use of the 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
Chapter 9

Architecture of the simulation

9.1 Principal rules

GATE simulations are based on the execution of scripted commands (see Chapter 1) gathered in macros. A simulation is generally divided into 7 steps (Fig. 9.1) as follows:

1. Verbosity and Visualization (see Chapter 2)
2. Geometry (see Chapters 3, 4, 5)
3. Digitizer (see Chapter 8)
4. Physics (see Chapter 6)
5. Sources (see Chapter 7)
6. Outputs (see Chapter 10)
7. Experiment (see Chapter 8)

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:

/run/initialize

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

9.1.1 Verbosity and Visualization

Verbosity:

For each simulation module, one can set a verbosity level between 0 and 2. The higher the verbosity level is, the higher the level of information returned by GATE will be. 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, in order to have at the end of a simulation the computation time of the simulation written by Gate on the screen as follow:
9.1 Principal rules

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 to visually check the scanner geometry (positions, physical volume overlaps…). 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.

9.1.2 Geometry

The world:

The first volume you have to create 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 you will have to choose the system type: scanner, PETscanner, cylindricalPET, ecat, CPET or SPECTHead. Each system has a defined number of levels with a hierarchical organization (tree geometry) and it is linked with a specific data output format. You also have with provisions two formats of data independent of the selected system (ROOT and ASCII). Once the scanner is built you have to attach all the scanner elements to the system.

The phantom:

You can define a phantom. Its material must be referred in the materials database. The phantom can be an analytical or voxelised volume. Each voxel of the volume can be made of a specific material with its own density.

Finally, you have to attach the sensitive volumes crystalSD and phantomSD. The interactions (hits) occurring in these volumes will be stored by GATE for the digitization.

9.1.3 Digitizer

The digitizer pre-processes the hits by sorting, regrouping and adding them in order to build singles. The singles are time-stamped and stored in the events history. 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.
9.1 Principal rules

9.1.4 Physics
The physics part of the simulation is dedicated to the definition of the simulated physical processes list 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.

9.1.5 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).

9.1.6 Data outputs
The data output formats are of two types in GATE:

1. Standard outputs: ASCII, Root
2. System dependent outputs: LMF, sinogram and 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 informations are stored. By default these two types of outputs are enabled. Each one of these outputs can be enabled or disabled according to the kind of informations one is interested to get.

**Specific outputs**
In addition to the two standards outputs, GATE can provide as well 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 to the *SPECTHead* system.

All these outputs are characterized by several parameters which have to be correctly set up. (see Chapter 10).
9.2 Random generator

9.1.7 Experiments

The last step of the simulation consists in setting up the experiment. In this part you have to fix the duration of the simulated acquisition 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 characterized by the same geometry.

9.2 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 to switch to another one the following command must be used:

```
/gate/random/setEngineName aName
```

where aName can be: Ranlux64, JamesRandom, or MersenneTwister.

Then another important thing is the choice of the generator’s seed. 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 launched. To randomly generate the seed, the CPU time (i.e. 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 launched a new seed is used.

- The 'manual' option. In this case it is the user that manually sets 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
```

In this case it is better to put off the automatic initialization from the file endOfRun.rndm by using the command:

```
/gate/output/root/setSaveRndmFlag 0
```
Finally, as usual the level of verbosity of the random engine can be chosen. It consists into printing
the random engine status, thus depending on the type of generator used. The command associated to the
verbosity is the following:

/gate/random/verbose 1

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

9.3 Example of a PET scanner

The following example describes how to build a PET scanner based on the \textit{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 \textit{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/geometry/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
The volumes of the scanner (with user’s name, e.g. box1) are connected to the cylindricalPET system (to the predefined names of cylindricalPET system, e.g rsector).

The scanner system is completely built. It is composed of 42 rsectors, each one made of 8 × 8 LSO and LuAP crystals assembled in phoswich.
9.3 Example of a PET scanner

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.

The lines below are just to show how the system moves with time:

```
# 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.
9.3 Example of a PET scanner

The digitizer is setup with a crystal blurring randomly chose 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 account only if the difference between the two rsectors is greater or equal than 2. We have introduced a paralyzable dead-time of 250 ns at rsector level.

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

An important check can be made just after this initialization in order to test that there are 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.
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 positrons is simulated as well as the $\gamma$-$\gamma$ acolinearity. The two sources are confined in the `phantom`.

# O U T P U T

- 160 -
# ASCII
/gate/output/ascii/setOutFileHitsFlag 0
/gate/output/ascii/setOutFileSinglesFlag 0
/gate/output/ascii/setOutFileCoincidencesFlag 0
# ROOT
/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

# START
/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. So, the geometry will be updated every second.

Notice
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
Chapter 10

Data output

Data output is a key point for a software intended to be used for various applications, in various scientific communities. It has been chosen to have several types of output format, which can be enabled or disabled if needed.
The following chapter describes the various output formats, such as ASCII, Root, Interfile, LMF, ECAT.

10.1 The ASCII output

10.1.1 Introduction

The GateToASCII class of GATE allows to obtain the ASCII file output, which are the easiest possible output. It allows you to treat your raw data with your own tools. In 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 disable this output in order to speed up the simulation.

10.1.2 How to enable this output in your macro ?

All the output commands (/gate/output/...) must always be 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 the Singles after the different steps of the digitizer chain. In your macro, set to 1 the different flags (resp.):

# 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 the Chapter 8.
10.1 The ASCII output

10.1.3 How to disable this output in your macro?

To disable completely the ASCII files:

```
/gate/output/ascii/disable
```

If the macro does not contain this line, by default some ASCII files will be created which are: `gate-Hits.dat`, `gateSingles.dat`, `gateRun.dat`. In addition, if coincidences are treated 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 be created.

10.1.4 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 are:
- 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: the following N columns represent 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
10.1 The ASCII output

* 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 come
* 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 occurs
* Column N+19 : Name of the last volume where a Rayleigh effect occurs

For the next sections, the system will be fixed at a cylindricalPET system, so that the number of lines concerning the Volume ID of each level will be always five.

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. 10.1.4)
- 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 occurs
- Column 23 : Name of the phantom where a Rayleigh effect occurs

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 : Energy deposited (first single)
- Column 9 to 11 : XYZ position in the world referential (first single)
10.1 The ASCII output

- Column 12 to 17: volume IDs* (cf. columns 5-10 of sec. 10.1.4) (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, (cf. columns 5-10 of sec. 10.1.4) (second single).

- 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)

10.1.5 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 10.1.4. 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.
10.1.6 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 fixed to 2000000000 bytes. One can change the number of bytes with a command like

\[ /\text{gate/output/ascii/setOutFileSizeLimit} \ 30000 \]

If the value is < 10000, no file swapping is made (to avoid creating thousands of files by mistake).

For example, if one doesn’t have a limit in the Operating System, he/she can put the number to 0, and there will be only one large (large) file in the end.

In case of high statistics applications, one might consider enabling only the ROOT output (see section 10.2), which contains the same information as the ASCII one, but automatically compressed (and, in addition, ready for analysis).

10.1.7 What is the file gateRun.dat?

This file is the list of the number of decays generated at 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 slightly different from the expected number \( N = A \times \Delta t \) where \( A \) is the activity and \( \Delta t \) is the time of the acquisition, due to 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.

10.1.8 What is the file voxels.dat?

When you build a voxelised source or a voxelized phantom, by default, GATE writes the voxel density values in a file (voxels.dat), to allow a cross-check of what you have loaded. In your macro, you can disable this option:

\[ /\text{gate/output/ascii/setOutFileVoxelFlag} \ 0 \]

The format of the file voxels.dat is:

* line 1 : \( N_x \ N_y \ N_z \) : number of voxels in the three directions,
* line 2 : \( d_x \ d_y \ d_z \) : size of the voxel in millimeters,
* line 3 to the end: voxel density values.

The order of the voxels is chosen as follow : increase the index along x, then y, then z.

For example for a \( 2 \times 2 \times 2 \) matrix the densities are written in the following order:

* \( d(0,0,0) \ d(1,0,0) \)
* \( d(0,1,0) \ d(1,1,0) \)
* \( d(0,0,1) \ d(1,0,1) \)
* \( d(0,1,1) \ d(1,1,1) \)
10.2 The Root output

10.2.1 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 line in the macro:

```
/gate/output/root/setFileName FILE_NAME
```

which will provide you with a FILE_NAME.root file.

If this previous command is not in the macro, and if the `/gate/output/root` is not disabled, the default name will be `gate.root`. Please note that a file `gateVoxels.root` is also created when using a voxelized phantom in your macro (see sec. 7.3).

By default, this root file will contain: 4 histograms, 1 Ntuple (Gate), and 2 Trees for SPECT systems (Hits and Singles) or 3 Trees for PET systems (Coincidences, Hits and Singles) in which several variables are stored.

The informations contained in the 4 histograms and in the Ntuple concern the PET simulations. They are present in the root output after a SPECT simulation but are empty.

The 4 histograms contain the distributions of:

- `Acolinea_Angle_Distribution_deg`: the angle in degree between the two gamma of annihilation of the $\beta^+$. 
- `Positron_Kinetic_Energy_MeV`: the energy of the $\beta^+$ 
- `Ion_decay_time_s`: the time of the decay 
- `Positron_annihl_distance_mm`: the range of the $\beta^+$

In the Ntuple Gate, four similar variables are available in order to look at correlations amongst them.

When launching ROOT with the command:

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

you should see the contain of the root file (see. fig.10.1).

Figure 10.1: The ROOT Object Browser when opening the GATE output file, containing: 4 histograms and 4 trees (GATE, COINCIDENCES, HITS and SINGLES) in which several variables are stored.
10.2.2 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:

```plaintext
/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.

```plaintext
/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
```

10.2.3 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:

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

You may also use the root class called MakeClass. Example:

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


10.3 The Online plotter

Along with standard output for post-treatment (such as root, LMF, ecat), GATE provides a very convenient tool called the online plotter, which allows to have 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 do not want to save a root file which can be a large one.

It can be easily used with the following macro:

```plaintext
/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
```
10.4 Interfile output

/addgate/output/plotter/addPlot tree Singles comptonPhantom
/addgate/output/plotter/addPlot tree Coincidences energy1
/addgate/output/plotter/listPlots

with the commands of the previous macro being:

addPlot hist NAME_of_the_histo

to plot an histogram previously defined in GATE.

and:

addPlot tree NAME_of_the_tree  NAME_of_the_variable

to plot a variable from one of the GATE trees.

The commands setNColumns allows to choose 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](image)

10.4 Interfile output

10.4.1 Description

The Interfile Projection Set is designed to mimic an acquisition protocol for a multiple headed rotating gamma camera. The total description of the Interfilev3.3 format can be found on the Interfile website [19].

Especially important is to understand how the storage of the projection data is done. When image data relate to multiple windows etc. (e.g. energy windows, time windows, multiple heads) the images shall be nested according to the order in which the corresponding keys are defined. Thus if multiple energy windows are used, all the 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: gate.hdr and gate.sin. The header file contains all the information
about the acquisition while the *gate.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 := gate.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
!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
```
10.4 Interfile output

!number of projections := 32
!matrix size [1] := 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 :=

10.4.2 Use

In order to achieve such an InterFile Projection Output Set, the following lines have to be added to the executed macro:

# PROJECTION
/gate/output/projection/projectionPlane YZ
/gate/output/projection/pixelSizeY 1. mm
/gate/output/projection/pixelSizeX 1. mm
/gate/output/projection/pixelNumberY 16
10.5 Sinogram output

If the ecat system or the ecatAccel system have been selected (see section 4.4.5), the sinogram output module will automatically be enabled, unless specified.

For the ecat system:

```
/gate/output/sinogram/disable
```

For the ecatAccel system:

```
/gate/output/sinoAccel/disable
```

This module stores the coincident events 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 by the command:

For the ecat system:

```
/gate/output/sinogram.RadialBins 256
```

For the ecatAccel system:

```
/gate/output/sinoAccel.RadialBins 256
```

For a system with crystals per crystal-ring, the default value is equal to . The number of radial bins should be smaller or equal to . The number of azimuthal bins is fixed and equal to . The default for the ECAT EXACT HR+ (576 crystals per crystal-ring) corresponds to a 2D sinogram size. 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 enable:

For the ecat system:
10.5 Sinogram output

/gate/output/sinogram/RawOutputEnable

For the ecatAccel system:

/gate/output/sinoAccel/RawOutputEnable

The name of the file is specified by:

For the ecat system:

/gate/output/sinogram/ setFileName MySinogramFileName

For the ecatAccel system:

/gate/output/sinoAccel/ setFileName MySinogramFileName

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 MySinogramFileName.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; they are set arbitrarily to unity.

MySinogramFileName.info describes the ordering of the 2D sinograms in the flat file MySinogramFileName.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 $ring_1$ and $ring_2$. Instead of indexing the 2D sinograms by $ring_1$ and $ring_2$, they are indexed by the ring difference $ring_2 − ring_1$ and the axial position $ring_2 + ring_1$:

```
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.
10.6 ECAT7 output

If, and only if, both the ecat system and the sinogram output module have been selected, the ecat7 output module will automatically be enabled, unless specified

/gate/output/ecat7/disable

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.

10.6.1 Installation

In order to compile the ecat7 output module of Gate, the ecat library written at the PET Unit of the Catholic University of Louvain-la-Neuve (UCL, Belgium) is required. It can be downloaded from their web site:

http://www.topo.ucl.ac.be/ecat_Clib.html

Three files are required: the library file libecat.a and the two header files matrix.h and machine_indep.h. To compile Gate with the ecat7 library without changing the env_gate.csh and GNUmakefile files, the environment variable ECAT7_HOME has to be defined and set to the name of the home directory where the ecat7 library is installed (for example, /usr/local/ecat7). In this ecat7 home directory, two subdirectories should be created: lib and include. The header files are put in the $[ECAT7_HOME]/include directory. For each system, a specific subdirectory named after the G4SYSTEM environment variable value should be created in the $[ECAT7_HOME]/lib directory. The corresponding library file libecat.a has to be located in this $[ECAT7_HOME]/lib/$[G4SYSTEM] directory. The matrix.h file has to be modified to add the declaration of the mh_update() function. The following line can be added in the "high level user functions" part of matrix.h:

int mh_update(MatrixFile*);

10.6.2 Data reduction

The polar coordinate of a LOR is approximately defined by the crystal-ring index difference between both rings in coincidence. For a $N_R$ crystal-rings scanner, the total number of polar samples is given by $2 \times N_R - 1$. Usually, on ecat systems, not all crystal-ring differences are recorded; only absolute crystal-ring differences up to a given value, referred as the maximum ring difference, are recorded:

/gate/output/ecat7/maxringdiff 22

The value of the maximum ring difference should be smaller than $N_R$.

A polar mashing is applied to group 2D sinograms with adjacent polar coordinates. The size of this grouping is called the span [20]. Its minimum value is 3 and it should be an odd integer.

/gate/output/ecat7/span 9
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 $N_R - 1$ and the span factor to 3. It should be noted that 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:

$$\frac{2 \times MaxRingDiff + 1}{span}$$  \hspace{1cm} (10.1)

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

```
/gate/output/ecat7/mashing 2
```

The default mashing value is 1.

### 10.6.3 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/application/setTimeStop 300 s
```
5 frames of 60 second each will be generated. The name of the sinogram file is specified by

```
/gate/output/ecat7/setFileName MySinogramFile
```

and 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 the half-value of the crystal transverse sampling and does not take into account the arc and depth-of-interaction (D.O.I) effects. After arc correction, the radial bin size should be slightly increased to account for the D.O.I. effect (see Figure 10.4). 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 D.O.I. effect.](image)

### 10.7 LMF output

#### 10.7.1 Introduction

The Crystal Clear Collaboration has developed a “List Mode Format” (LMF) to store the data of Clear-PET 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 Chap.4) and GATE can only store `single` events.

Several tools that allow to read this format and to process events are implemented in the LMF library [21]. As an example, coincidences can be associated from GATE single events. It is also possible to apply different deadtimes, and eventually to generate sinograms in interfile format as used by the STIR library [22], which implements several image reconstruction algorithms (this latter is available only with the STIR library).

The LMF library and its documentation are available on the OpenGate web site.
### 10.7.2 Usage

LMF data are composed of two files with the same base-name, but different extensions:

- A 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 fix 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 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 comprising about 1500 crystals.

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

- `/gate/output/lmf/enable`
to enable LMF output

- `/gate/output/lmf/disable`
to disable LMF output

- `/gate/output/lmf/setFileName myFirst`
to set the LMF files name. Here the output files will be myFirst.ccs and myFirst.cch

<table>
<thead>
<tr>
<th>Information</th>
<th>Size (bytes/single)</th>
<th>Real machines</th>
<th>GATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>8</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>Energy</td>
<td>1</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>detector ID</td>
<td>2</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>PET’s axial position</td>
<td>2</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>PET’s angular position</td>
<td>2</td>
<td>YES</td>
<td>YES</td>
</tr>
<tr>
<td>run ID</td>
<td>4</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>event ID</td>
<td>4</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>source ID</td>
<td>2</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>source XYZ Position</td>
<td>6</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>global XYZ Position</td>
<td>6</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>number of Compton in phantomSD</td>
<td>1</td>
<td>NO</td>
<td>YES</td>
</tr>
<tr>
<td>number of Compton in crystalSD</td>
<td>1</td>
<td>NO</td>
<td>YES</td>
</tr>
</tbody>
</table>

Figure 10.5: Size of informations to be stored in LMF.
10.7 LMF output

/gate/output/lfm/setDetectorIDBool 1
to store (1) or to not store (0) the detector ID

/gate/output/lfm/setEnergyBool 1
to store (1) or to not store (0) the energy

/gate/output/lfm/setGantryAxialPosBool 0
to store (1) or to not store (0) the axial position

/gate/output/lfm/setGantryAngularPosBool 0
to store (1) or to not store (0) the angular position

Note that the following lines must always to be used as they appear below with option 0:

/gate/output/lfm/setSourcePosBool 0
/gate/output/lfm/setNeighbourBool 0
/gate/output/lfm/setNeighbourhoodOrder 0
/gate/output/lfm/setCoincidenceBool 0

All information that is not available in real world is stored in a GateDigi record adjoining the event record by using the command:

/gate/output/lfm/setGateDigiBool 1

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

/gate/output/lfm/setComptonBool 1
/gate/output/lfm/setComptonDetectorBool 1

to store (1) or to not store (0) the number of Compton scattering that occurs in (resp.) a phantomSD and a crystalSD

/gate/output/lfm/setSourceIDBool 0
to store (1) or to not store (0) the source ID

/gate/output/lfm/setSourceXYZPosBool 0
to store (1) or to not store (0) the source XYZ position

/gate/output/lfm/setGlobalXYZPosBool 0
to store (1) or to not store (0) the real XYZ position

The information on 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/lfm/setEventIDBool 1

to store (1) or to not store (0) the event ID

/gate/output/lfm/setRunIDBool 1

to store (1) or to not store (0) the run ID
10.7.3 Limitation

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 fixed by the number of bits used to encode the sub-volume ID. All together, the final ID encoding the position of an event has to be stored on 16, 32, or 64 bits only.

10.8 imageCT output

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

```
/gate/output/imageCT/RawOutputEnable true
```

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 (see 4.4.2), the number of pixels is set by

```
/gate/output/imageCT/numPixelX 80
/gate/output/imageCT/numPixelY 80
```

Finally the random seed can be defined by the command

```
/gate/output/imageCT/setStartSeed 676567
```

10.9 Raw output

10.9.1 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 Section 8.4) using a PET-like system such as : CPETSystem, CylindricalPETSystem, EcatSystem and EcatAccelSystem (see Chap.4). With the SPECTHeadSystem, only the Singles can be stored. And 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 bits 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/).
10.9.2 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 on the X and Y axes and a number of slices (on the Z axis) 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 will be created. Each pixel will be 4 mm x 4 mm, and the slice thickness will be 4 mm too. 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, 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 others 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 equal to 1.

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

/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 true, scatter and random, 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 %
10.9 Raw output

--- 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 informations about the different raw output classes run. By default the verbose level is set to 0 (quiet).

### 10.9.3 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

### 10.9.4 Limitation

The random Coincidences cannot be stored.
Chapter 11

Materials

11.1 The Gate material database

The primary method for defining the properties of the materials used in Gate is by a materials database. The database is located in the directory “.../petsim/” in the file GateMaterials.db. 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 GateMaterials.db file 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.

11.1.1 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.

It should be noted that 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.
11.2 Modifying the Gate material database

11.1.2 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 CH₄ 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 ng/cm³.

11.2 Modifying the Gate material database

As mentioned in the previous section the Gate material database is located in the file GateMaterials.db. This file must be present in every directory in which Gate is run unless Gate is specifically compiled with the database’s location specified. The relevant file is

```c
include/GateMaterialDatabase.hh
```

and the modification is “define DEFAULT_GATEMATERIALDB "the location of GateMaterials.db"”. The database file is designed to be easily modified for user dependent applications. It must contain all the element and material definitions associated with the system being modeled.

11.2.1 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 project 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
```
11.2 Modifying the Gate material database

Note the format in the above example. 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 follows the symbolic name given by \( Z = \) atomic number with a semi-colon and by \( A = \) molar mass units for the molar mass and it’s units.

11.2.2 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 done 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 begin by opening the `GateMaterials.db` file with a text editor. This time the new entry is made below the header named `[Materials]`. All material definitions required by the project must be included below this second header. Materials definitions span several lines. The first line specifies their name, density, number of constituents, and an optional parameter describing the materials state (solid, liquid, or gas). The second and subsequent lines specify the individual constituents 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. It should be noted that in every case the elements being used in a material definition must be previously defined as elements.

Elements as materials

Substances comprised 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), it’s density, it’s number of constituents (which is one in this case), and optionally it’s 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 one). 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
```

- 189 -
11.2 Modifying the Gate material database

On the first line the density (with units) is defined by \( d = \text{material density units} \) and is separated by a semi-colon from the number of constituents in the material defined by \( n = \text{number of elements} \). 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 \(+el: \text{name= name of the element} ; n= \text{number of atoms}\). Notice that if the name of the element and the material are the same the elements name can be defined by \(+el: \text{name=auto}\) command.

Compounds as materials

Substances comprised of a chemical compounds are defined based upon the elements that comprise them and their chemical formula. The first line is identical to the first line of a pure substances 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 \(+el: \text{name= name of the element} ; n= \text{number of atoms}\). For example:

Compounds as materials example GateMaterials.db:

```plaintext
[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

Substances comprised of a mixture of elements are defined by indicating the mass fraction of the elements that makeup 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 \(+el: \text{name= name of element} ; f= \text{mass fraction}\). In the case of material mixtures the sum of the mass fractions should be one. For example:

Mixtures as materials example GateMaterials.db:

```plaintext
[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
```
11.2 Modifying the Gate material database

+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=Nickle ; 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 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. Note that when specifying materials rather than elements the +mat:: name= identifier must be used.
Chapter 12

Generating and tracking optical photons

12.1 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 to investigate for example the energy resolution or spatial resolution. In order to use the optical photon tracking capabilities of GATE it has to switched on first. This can be done using the following command:

/gate/physics/optical/enable 1

Before discussing how the 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 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 in order to have generation and tracking of optical photons. How this is done is discussed in the following two sections. Finally, there is one new 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 [24, 23], and the UNIFIED model, discussed in section 12.3.1, is described more extensively by Levin [25] and Nayar [26].

12.2 Defining material properties

The optical properties of materials are stored in a material properties table. In this table each of the properties of a material is identified by a name. There are two different 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 stores in a file separate from the materials database. This is done to make it more easy to change them without having to change the
materials database. This file should be named Materials.xml\(^1\). When GATE reads in a material from the materials database, it also checks if the Materials.xml file contains a properties table for this material. If the file contains one, this table is read in and coupled to the material.

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

```xml
<?xml version="1.0"?>
<materials>
  <material name="LSO">
    <propertystable>
      <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>
    </propertystable>
  </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 elements 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. Note that the names of the properties are case sensitive.

12.2.1 Scintillation properties

In order to have scintillation in a materials, 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 precise, it gives the expectation value of this number, since the real number of photons emitted follows a normal distribution. The variance of this normal distribution is RESOLUTIONSCALE 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

\[
\mu_N = E \cdot \text{SCINTILLATIONYIELD}
\]

and a standard deviation of

\[
\sigma_N = \text{RESOLUTIONSCALE} \cdot \sqrt{E \cdot \text{SCINTILLATIONYIELD}}.
\]

\(^1\)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 Wold Wide Web Consortium (http://www.w3.org)
12.3 Defining surfaces

Table 12.1: Material parameters.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCINTILLATIONYIELD</td>
<td>const</td>
<td>1/Mev, 1/keV</td>
</tr>
<tr>
<td>RESOLUTIONSCALE</td>
<td>const</td>
<td>-</td>
</tr>
<tr>
<td>FASTTIMECONSTANT</td>
<td>const</td>
<td>ns, ms, s</td>
</tr>
<tr>
<td>SLOWTIMECONSTANT</td>
<td>const</td>
<td>ns, ms, s</td>
</tr>
<tr>
<td>YIELDRATIO</td>
<td>const</td>
<td>-</td>
</tr>
<tr>
<td>FASTCOMPONENT</td>
<td>vector</td>
<td>-</td>
</tr>
<tr>
<td>SLOWCOMPONENT</td>
<td>vector</td>
<td>-</td>
</tr>
<tr>
<td>ABSLENGTH</td>
<td>const</td>
<td>m, cm, mm</td>
</tr>
<tr>
<td>RINDEX</td>
<td>const</td>
<td>-</td>
</tr>
</tbody>
</table>

The parameters RESOLUTIONSCALE can be calculated from the energy resolution of the scintillator:\(^2\):

\[
\text{RESOLUTIONSCALE} = \frac{R}{2.35} \cdot \sqrt{\frac{E \cdot \text{SCINTILLATIONYIELD}}{\mathrm{const}}},
\]

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

The scintillation is follows an exponential decay with two time constants, a fast and a slow one. The ratio between the fast and 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.

12.2.2 Optical properties

In order to have transport of optical photons inside a material the least that has to be defined is the refractive index. This can be done in the property vector RINDEX. The refractive index can thus be specified as a function of energy. Besides 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 ABSLENGTH. Note that no optical photons will transported in materials for which the refractive index is not defined; all optical photons entering the material will be killed.

12.3 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 issued:

```
/gate/volume1/surfaces/name surface1
/gate/volume1/surfaces/insert volume2
```

\(^2\)Note that the energy resolutions specified in literature may contain contributions of for example electronic noise. The energy resolution needed to calculate the RESOLUTIONSCALE should be the intrinsic energy resolution of the scintillator.
12.3 Defining surfaces

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPECULARLOBECONSTANT</td>
<td>vector</td>
<td>-</td>
</tr>
<tr>
<td>SPECULARSPIKECONSTANT</td>
<td>vector</td>
<td>-</td>
</tr>
<tr>
<td>BACKSCATTERCONSTANT</td>
<td>vector</td>
<td>-</td>
</tr>
<tr>
<td>RINDEX</td>
<td>vector</td>
<td>-</td>
</tr>
<tr>
<td>REFLECTIVITY</td>
<td>vector</td>
<td>-</td>
</tr>
<tr>
<td>EFFICIENCY</td>
<td>vector</td>
<td>-</td>
</tr>
</tbody>
</table>

This creates a surface with the name ‘surface1’ between the two volumes with the names ‘volume1’ and ‘volume2’. These volumes should already exists. It must be noted that 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.

The surface is now created, but it does not yet have any properties. The surface properties of all the different surfaces are stored in the Surfaces.xml file. In order 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:

```xml
<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 dielectrics 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, groundbackpainted, polished-backpainted, groundfrontpainted and polishedfrontpainted. 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 properties table are shown in table 12.2. All of them are vectors and are probabilities and therefore without unit. These properties are discussed in the next section.

12.3.1 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 [25] and [26]. The surface between the two media can be painted on the inside or outside. First the situation will be discussed in which the surface is not painted.

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 \( \sigma_\alpha \), which can be specified in degrees in the attribute `sigmaalpha`. When an optical photon reaches a surface, a random angle \( \alpha \) 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 different 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 happen. The probability of Lambertian reflection is thus given by one minus 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 `polished-frontpainted`, where `polished-` or `ground-` have the same meaning as in the previous section.

When an optical photon reaches the paint 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, first is calculated whether or not the photon is reflected on the interface between the two media using the method described in the previous section. However, in this case not the refractive index of the second volume is used, but that given by the property vector `RINDEX` of the surface. 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 of 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.

### 12.3.2 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 \textsc{reflectivity} property vector. When the optical photon is reflected the \textsc{unified} model is used to determine the reflection angle; when it is absorbed it is possible to detect it. The property vector \textsc{efficiency} gives the probability of detecting a photon given its energy and can therefore be considered to give the internal quantum efficiency\(^3\).

The hits generated by the detection of the optical photons are generated in the volume from which the optical photon reached the surface. This volume should therefore be made sensitive detector.

### 12.4 Digitizer

The hits generated in the sensitive detector are first processed by \textsc{analysis}. Unfortunately \textsc{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 and is therefore called \textsc{fastanalysis}. In order to use this class \textsc{analysis} has to be switched off and \textsc{fastanalysis} on:

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

Switching both on has no effect on the results only on the speed of the simulation. There is however a disadvantage to using \textsc{fastanalysis}: not all information of the singles is set. 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)

\(^3\)Note that many measurements of the quantum efficiency give the external quantum efficiency, which includes the reflection: external quantum efficiency = efficiency\(^*\)(1-reflectivity)
When this information is needed, the slower analysis should be used (as is the default). After processing the hits with one of the analysis routines, the singles should be created from the hits. Normally this is done using the adder, but for optical photons a separate adder is defined. The normal 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 traditional 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 only adding optical photons there is one more 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 photons detected and not the energy deposited. Digitizer modules like threshold can still be used, but the energy of the threshold should be specified in a number of photons.
12.4 Digitizer
Chapter 13

How to run Gate

13.1 Interactive mode

To start Gate in interactive mode, simply type:

$ Gate

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

1 **********************************************
2 Geant4 version $Name: $ (3-October-2003)
3 Copyright : Geant4 Collaboration
4 **********************************************
5 Time set to (s) 0
6 Visualization Manager instantiating...
7 Visualization Manager initialising...
8 Registering graphics systems...
9 You have successfully chosen to use the following graphics systems.
10 Current available graphics systems are:
11 DAWNFILE (DAWNFILE)
12 VRML1FILE (VRML1FILE)
13 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>
13.1 Interactive mode

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 and lines 6-9 are initialization messages from Gate. If you installed Gate with visualization support, then you should see messages like lines 10-15. Then, Gate runs the file prerunGate.mac located in the petsim directory and it outputs the command lines 16-22 found in that file. Finally, and if everything went right, then Gate outputs the interpreter’s 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.
```

- 202 -
13.2 Running parameterized macros

It is very common for users to run several simulations that differ in 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 these cases, the user does not need to write a complete set of macros for each simulation scenario. Instead, He or She can write parameterized macros. The actual values of the parameters are specified at command line when starting up 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:

```plaintext
# DIGITIZER
1 /gate/digitizer/Singles/insert adder
2 /gate/digitizer/Singles/insert readout
3 /gate/digitizer/Singles/readout/setDepth 1
```
13.3 Batch mode

It is possible to run a Gate simulation in “batch” mode, i.e. the mode in which you don’t need to enter the interpreter and run /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 simulation 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 necessary, to avoid running visualization commands in batch mode.
13.4 Cluster mode

In order to reduce the overall computing time of GATE experiments, a parallel computing platform for running such simulations in a cluster of computers was developed which significantly shortens the setup time and provides fast data output handling. In order to use GATE in cluster mode you need 3 components:

- The job splitter
- The file merger
- A cluster aware version of Gate

13.4.1 Installation of the job splitter

The job splitter can be installed in the same directory as GATE. Two environment variables need to be added to the environment file used to compile GATE:

- 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 splitted macros for each simulation. Even when splitting the same macro several times, a new directory will be created for each instance. 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 section 13.4.6). 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 and unpack the job splitter source code into its own directory (bash example):

$ source environment_variables
$ tar -zxvf jobsplitter.tar.gz
$ cd jobsplitter
$ gmake

13.4.2 Installation of the file merger

To install, unpack the file merger source code into its own directory and compile (bash example):

$ tar -zxvf filemerger.tar.gz
$ cd filemerger
$ gmake

The file merger executable is located in the current directory.
### 13.4.3 Preparing your macro

The cluster software should be able to handle all GATE macros. However, only ROOT is currently supported as an output format. Please disable other output formats as they cannot yet be merged. If an isotope with a short half life compared to 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 load will be approximately equal for each CPU.

### 13.4.4 Using the job splitter

In order to view information on general usage, just run the job splitter executable without any options:

```
$ ./bin/Linux-g++/gjs
```

```
gjs [-options] your_file.mac
```

**options (in any order):**

- `-a value alias` : use any alias
- `-numberofsplits n` : the number of job splits; default=1
- `-noseeds` : do not create random seed files in the .Gate/job directory
- `-clusterplatform name` : the cluster platform, name is one of the following:
  - openmosix - condor - openPBS - xgrid
  - `openPBSscript` : template for an openPBS script  
  - `condorscript` : template for a condor submit file

**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
- `optional GC_PBS_SCRIPT` : the openPBS template script

**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`

**WARNING:** just like in a normal GATE macro you do not use filename extensions for output formats (ROOT,INTERFILE,...)

The supported platforms are currently: openMosix, openPBS, Condor and Xgrid. Let’s take openMosix as an example:
13.4 Cluster mode

$ ./bin/Linux-g++/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 placed under 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:

ROOT output is enabled
ASCII output is disabled
INTER output is disabled
LMF output is disabled
ECAT output enabled if ECAT + sinogram selected
  but no filename is given; using a default one
SINO output enabled if ECAT system selected
  but no filename is given; using a default one

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 .Gate directory now has a subdirectory called macro, that contains:

macro1.mac
macro2.mac
macro3.mac
macro4.mac
macro5.mac
seed1.rndm
seed2.rndm
seed3.rndm
seed4.rndm
seed5.rndm
macro.split

The 5 macros are listed as well as 5 random seeds to initialize the random engine. The .split file contains information about the splitted simulation and can be used to merge the data after the simulation. The current directory, from which the jobsplitter was called, now contains the cluster submit file. In order to run the splitted simulation on the cluster, one only needs to execute this file: $ chmod +x macro.submit
$ ./macro.submit

The .Gate directory supports automatic numbering. If the same macro is used repeatedly, then the subsequent directories will be numbered.

13.4.5 Using the file merger

The file merger can be run with either the original macro or the split file as input. In order to view information on general usage, just run the file merger executable without any options:
13.4 Cluster mode

$ /gjm

gjm [-options] your_file.mac or your_file.split
You may either give the name of your gate macro file or the name of a split file created by gjs.

options:
- `raw`: says to the gjm to merge Raw output files. Can not be used with the fastMerge option.
- `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
- `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

In order to merge the output files into a single file, just supply the macro file or the split file to the file merger. The output file can be used as a usual single CPU output file. For some older ROOT versions it may be necessary to use the -maxRoot option to limit the ouput file size in MB:

$ /gjm macro.split or /gjm -maxRoot 1900 macro.mac

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

Figure 13.1: Example ROOT file with added cluster eventIDs

eventIDs in the ouputfiles are corrected locally. Figure 13.1 shows the newly created tree in each ROOT file. 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):

```c
{
```
13.5 How to launch DigiGate

```c
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")
```

13.4.6 What about errors?

If something went wrong during a simulation and a ROOT file is corrupt 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 easily be 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.

13.5 How to launch DigiGate

GATE offers an operation mode dedicated to digitizer optimization, known as DigiGate (see chapter8). DigiGate works by re-reading a previously generated ROOT hit-file and changing according to new values.

It 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 re-analysis is done by using the gate.root file as an input file for the program DigiGate and this is achieved by launching Gate with a `-d` option.
13.6 How to separate the phantom and detector tracking - Phase space approach

Gate < MacroTest.mac

→ a root output file is produced with *Hits* informations
→ 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.

### 13.6 How to separate the phantom and detector tracking - Phase space approach

In the way to speed-up the simulation, it’s possible to split and separate the particle tracking. This is a pure 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.

#### 13.6.1 Use Gate in the tracker mode: phantom tracking

Basically, as it is illustrated in the folder example_TrackerDetector, 3 major command lines are availables to use the phantom tracker mode:

```
# To select the mode
#gate/stepping/SetMode Tracker

# To set the policy regarding the tracker mode
#gate/stepping/SetPolicy Option1
/gate/stepping/SetPolicy Option2
```

With the variable `Option1` which could be chosen in the following list:

- *StopOnPhantomBoundary* *(Default)*: the particles are tracked until the last hit before the phantom boundary;
- *StopAfterPhantomBoundary*: the particles are tracked until the first hit after phantom boundary;
- *KillTrackAndSecondaries*: *StopOnPhantomBoundary* + no secondary production.

And the variable `Option2` 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
13.6 How to separate the phantom and detector tracking - Phase space approach

Two more command line options are also availables:

/gate/stepping/SetEnergyThreshold aThreshold keV

Only particles reaching the phantom boundary with an energy greater than *aThreshold* keV will be stored.

/gate/stepping/SetTextOutput status

With status flag set as *On* or *Off*. This is a command to print 'Tracks' informations 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 choosed by user with the classical output command line to set the file name:

/gate/output/root/setFileName OutPutRoot

### 13.6.2 Use Gate in the detector mode: detector tracking

As indicated before, 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 tracker mode. All this will be done by the 2 following command lines:

/gate/stepping/SetMode Detector
/gate/stepping/SetNumberOfTrackerDataFiles N
13.6 How to separate the phantom and detector tracking - Phase space approach
Bibliography


[16] Biggs F, and Lighthill R 1990 Preprint Sandia Laboratory, SAND 87-0070


ROOT website : http://root.cern.ch/


[23] *Geant4 physics reference manual*

[24] *Geant4 User’s Guide For Application Developers*


List of Figures

1 Structure of GATE ................................................................. vii

1.1 World volume ................................................................. 7
1.2 Cylindrical scanner ........................................................... 8
1.3 first level of the scanner ................................................... 8
1.4 crystal, daughter of the block ............................................ 9
1.5 matrix of crystals ............................................................ 9
1.6 complete ring of 30 block detectors .................................. 10
1.7 cylindrical phantom .......................................................... 11
1.8 Simulation is started ........................................................ 15

3.1 Description of the default parameters associated with the world volume ................ 22
3.2 Three-axis system defined in GATE. The red, green and blue axes are the X, Y and Z axes respectively. ................................................................. 23
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. ............. 30
3.4 When a wedge is inserted, it is oriented as shown in this figure. ....................... 31
3.5 Illustration of the application of the linear repeater .................................... 32
3.6 Illustration of the application of the auto-rotation option ............................... 34
3.7 Illustration of the application of the ring repeater when the auto-rotation option is disabled ................................................................. 34
3.8 Illustration of the application of the ring repeater ....................................... 35
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. .................................................... 36
3.10 Illustration of the application of the cubic array repeater ............................... 37
3.11 Illustration of the application of the cubic array repeater ............................... 38
3.12 Illustration of the application of the sphere repeater .................................... 39
3.13 Illustration of the translation placement .......................................... 40
3.14 Illustration of the rotation placement ............................................. 41
3.15 Center of wedge. .................................................................. 42
3.16 A block approximating a true circular geometry. ....................................... 42
3.17 The OPET scanner. ................................................................ 43

4.1 Picture of a phantom and a cylindrical PET system composed of 5 sectors, 4 modules (repeated along Z axis), 3 submodules (repeated along Y axis), 64 crystals (8 x 8) and 2 layers (red and yellow) ......................... 49
4.2 Illustration of the scanner system. The different volumes, in particular the sensitive one, can be of any shape, here cylindrical sectors crystals, instead of box in other systems. Scanner cylinder is draw in magenta, whereas one of the sector hierarchy: Level1, Level2, Level3, Level4 is overdrawn in yellow, blue, green, red, respectively. “Detector” volume of cylindrical sector shapes are shown in plain red.

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

4.4 One NaI crystal with a curved shape.

4.5 After the ring repeater, 6 NaI crystals are describing the scanner.

4.6 Definition of the base.

4.7 Definition of the block.

4.8 Definition of the crystal.

4.9 The OPET scanner.

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

5.1 Particle interactions in a sensitive detector.

7.1 Properties of radioactive sources.

7.2 A simple voxellized phantom a) without transparency, b) with transparency.

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

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 observables seen from the detector.

8.2 Actions of the adder and readout modules.

8.3 Setting the readout depth in a CylindricalPET system.

8.4 For 7 incoming particles and a fixed dead-time \( \tau \), 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 plain ones are the accepted singles).

8.5 Comparison between the two methods of coincidences finding, for a given succession of singles. In the first one (in the upper side), the S2 single don’t 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. In the other side, with the second method, where all the singles are opening their coincidence window, one find, with the same single order, 2 different coincidences.

8.6 Comparison of the behavior of all the available multiple treatment policy, for various multiple coincidence situation. In the figure, the stars represents the singles detected in the event. The size of the star, as well as the number next to it, indicate the energy level of the single (ie. the single n°1 has more energy than the single n°2, which is itself more energetics than the single n°3). The lines represents the possibles good coincidences (ie. with a sector difference higher or equal to the minSectorDifference of the coincidence sorter). In the table, a minus(-) sign indicate that the event is killed (ie. no coincidence is formed). The ★ sign indicate that all the singles are kept into a unique multicoincidence, which will not be written to disk, but which could participate to data loss via dead time or bandwidth occupancy. In the other cases, the list of pairs which will be written to disk (unless being removed thereafter, by a possible filter applied to the coincidences) is indicated.

8.7 Readout scheme produced by the listing from the sections 8.3 and 8.5.
8.8  Readout scheme produced by the listing in section 8.7. The disk icons represents the data written to the GATE output files. .................................................. 146

9.1  GATE simulation architecture ...................................................... 162

10.1 The ROOT Object Browser when opening the GATE output file, containing : 4 histograms and 4 trees (GATE, COINCIDENCES, HITS and SINGLEs) in which several variables are stored. .................................................. 168

10.2 The Online plotter ................................................................. 170

10.3 Michelogram for a 16 crystal-ring scanner. .................................. 176

10.4 Increase of the radial bin size due to the D.O.I. effect. .................... 177

10.5 Size of informations to be stored in LMF. .................................. 178

13.1 Example ROOT file with added cluster eventIDs ......................... 208