Table of Contents

New features 3
Update (April 2006) 4
Update (May 2006) 4
Update (September 2006) 4
Update (September 2007) 4
Update (November 2007) 4
Update (April 2009) 4
Installation 5
Compiling the code 5
Defining a new ancillary detector 6
Class definition and instantiation 6
Virtual methods 6
FindMaterials 6
GetDetectorConstruction 7
InitSensitiveDetector 7
ShowStatus 7
WriteHeader 7
GetSegmentNumber 7
GetCrystalType 7
Placement 7
IDs and offsets 8
Plugging in a new event generator 9
BeginOfRun, EndOfRun 9
BeginOfEvent, EndOfEvent 9
GetStatus 9
PrintToFile 9
GetBeginOfEventTag 9
GetEventHeader 9
GetParticleHeader 9
IsStartOfEvent, IsEndOfEvent 9
IsAbortRun 9
GetAbortMessage 9
GetCascadeMult 9
GeneratePrimaries 9
A Brief Agata Simulation Code Tutorial 11
Lesson 1: Set the geometry 12
Lesson 2: Evaluate the response of AGATA 13
Lesson 3: Generate a dataset for Pulse Shape Calculations 14
Lesson 4: Shoot neutrons and gamma rays 15
Lesson 5: Simulate fusion-evaporation events 16
Format of the list-mode output file 17
Header 17
Geometry description 17
Event generator description 21

Beginning of events 23
Events 23
Input Event File Format 26
Emitter line 28
Emitted line 28

New features of the Agata Simulation Code (February 2006)

Several new features are available with the present version of the simulation code, here are the relevant ones:

- Possibility to initialize the seed of the random number generation. If the *-seed* option is chosen, the current system time is chosen as seed of the random number generation.
- Possibility to choose between several ancillary detectors for AGATA without need to recompile the code. The *-a index* option selects the ancillary corresponding to *index* (similar to the choice of the geometry).
- Possibility to start the calculations at a given run number. With the *-run n* option, the number of the first run will be *n*.
- Improved interface to the ancillary detectors, with the possibility to insert new ancillary detectors in an easy way. Examples are also provided to use two or more ancillaries at the same time.
- Improved treatment of polarized photons:
  1. the characteristics of a photon beam are given in terms of a degree of polarization (the fraction of linearly polarized photons) and of an angle (which is related to the possible orthogonal states of linear polarization).
  2. it is possible to express such characteristics in terms of the Stokes parameters.
  3. for the unpolarized fraction of the radiation, one of the two possible states is chosen randomly.
  4. the meaning of the "polarization" parameter in case of input from external file is changed accordingly to the angle between the electric field vector and the "orthogonal" reference direction. It is the user responsibility, in case of polarized photons, to specify the electric field vector direction on an event-by-event basis.
- Possibility to choose the units of length and energy for the output to the list-mode file through interactive commands. The default unit of length is now mm rather than cm.
- Possibility to modify through interactive command the frequency of event number printout (useful in the case of long runs).
- Simplified *AgataSensitiveDetector* class. Only two methods to extract the position of each interaction are now available, namely the default one (0), which correspond to following the electron tracks, and the alternative one (1, formerly known as 2), in which the energy deposition of the electrons is assigned to the point where the primary interaction took place. In case method "2" is selected, automatically it is converted to method "1". The behaviour of method 1 has been deeply modified in the case of pair production events:
  1. whenever a pair production event took place, it is assigned an energy deposition equal to the photon energy minus twice the electron rest mass minus the kinetic energy of the positron at the moment of its annihilation
  2. secondary photons are considered only in case they origin from the point in which the annihilation took place
  3. in case of Compton scattering or annihilation, the energy lost by the photon is deposited in the point where the interaction took place
  4. an additional check is performed at the end of the event to discard events in which the computed deposited energy is larger than the initial energy (because of the "unphysical" way of extracting the information.

  It is strongly recommended to use the default tracking method when the detailed energy depositions are of concern.
- Shorter documentation! A few example macros are presented rather than a full manual. A great effort has been taken to add plenty of comments to the code, which should be clearer to follow. The list of available commands is available interactively through the *help* command, as before; for each command there is a short description, plus the list of required

parameters.

Additional features (April 2006)

- Possibility to plug-in a user-defined native Geant4 event generator with minimal changes to the rest of the simulation code.

Additional features (May 2006)

- Revised interface for the ancillary detectors, with the possibility to load more than one ancillary detector. The usage of the *–a* switch has been modified accordingly:

  $$-a\ N_{anc}\ index_1 \ldots index_N$$

  where N_{anc} is the number of ancillaries which will be loaded, $index_1, \ldots index_N$ are their index in *AgataDetectorAncillary*.
- Revised and improved treatment of the polarized Compton scattering.

Additional features (September 2006)

- The code has been successfully ported to Geant v.4.8.X.

Additional features (September 2007)

- The code has been successfully ported to Geant v.4.9.0. It is now possible to compile the code using the "default" Geant4 libraries, that is without applying the patches.
- It is possible to extend the capabilities of the code, namely to consider the Compton profile, through the optional G4LECS package, which is provided as a separate patch either to the libraries or to the Agata code.

Additional features (November 2007)

- Fixed some bugs with the ancillary detector definition and revised the numbering for the ancillary detectors.
- Change of naming of the compilation flags.

Additional features (April 2009)

- Possibility to activate standard EM interaction separately for gammas and hadrons.
- Improved material definition for neutron detection.
- Revised construction of the geometry to handle new shapes for the detectors.
- Several bugs fixed.

Installation

In order to compile and run the code for AGATA, the CLHEP and Geant4 libraries should be previously installed on your computer. If they are not already installed, the presently recommended versions (April 2009) are CLHEP v.2.0.4.0 and Geant v.4.9.2.p01, which can be downloaded from the corresponding web sites:

- CLHEP: http://wwwasd.web.cern.ch/wwwasd/lhc++/clhep/index.html
- Geant4: http://geant4.web.cern.ch

For the installation of CLHEP and Geant4, please do refer to the corresponding documentation. Please note that the developer cannot test how the code compiles or runs with earlier versions, hence no support is provided for that. Needless to say, a C++ compiler should be installed on your computer. The code is currently being developed with gcc v.4.3.2 on a laptop with Ubuntu 8.10 (standard installation).

Compiling the code

After unpacking the Agata code distribution package, move to the Agata directory. If necessary, change the compilation options in GNUmakefile:

- if Geant4.7.0 or 4.6.X is used, comment the CPPFLAGS += -DG4V47, CPPFLAGS += -DG4V48, CPPFLAGS += -DG4V49 lines
- if Geant4.7.1 is used, comment the CPPFLAGS += -DG4V48, CPPFLAGS += -DG4V49 lines
- if Geant4.8.X is used, comment the CPPFLAGS += -DG4V49 line
- if ancillary detectors should be used, uncomment the CPPFLAGS += -DANCIL line
- if fixed offset range should be used for the ancillary detectors, the line CPPFLAGS += -DFIXED_OFFSET should be uncommented; comment it for dynamical offset assignment
- if the GRETA geometry should be used, uncomment the CPPFLAGS += -DGRETA line
- if output data compatible with the PRISMA simulation are needed, uncomment the CPPFLAGS += -DWRITE_EVNUM line
- if pulse height defect emulation for heavy ion stopping power should be considered, uncomment the CPPFLAGS += -DPULSE_DEFECT line
- in order to flag energy depositions caused by interaction of neutrons (writing out a “negative” energy deposition in the output file), uncomment the CPPFLAGS += -DFLAGGED_NEUTRON line.

After that, load the Geant4 environment variables (with the *env.sh* or *env.csh* scripts generated after the Geant4 installation) and compile with *make*. Forgive the warnings!

Defining a new ancillary detector

The present version of the simulation code includes several examples of ancillary detectors which can be coupled to AGATA. Here the rules to add new ancillary detectors are described in detail. For simplicity, it will be assumed to define a new ancillary detector through an *AgataAncillaryDummy* class (the name obviously might change!!!). See the code for practical implementations. Notice that starting from May 2006 the way of using two or more ancillaries at the same time has been mofified, therefore *AgataAncillaryTwins* is no more included in the distribution.

Class definition and instantiation

The *AgataAncillaryDummy* class handling the construction and placement of the new ancillary detector must inherit from *AgataAncillaryScheme* (and therefore from *AgataDetectorConstructed*). Therefore, all of the pure virtual methods of these classes must be implemented.
In order to be able to instantiate *AgataAncillaryDummy*, the following changes must be made to *AgataDetectorAncillary*:

- the corresponding header file must be included:
  #include "AgataAncillaryDummy.hh"
- in the second constructor of *AgataDetectorAncillary*, accepting three G4String as arguments, a new case must be added to the switch ... case construct, where a pointer to *AgataAncillaryDummy* is instantiated and cast to *AgataAncillaryScheme** and *AgataDetectorConstructed**:

```
AgataAncillaryDummy* theDummy    = NULL;
...
case 5:
   theDummy       = new AgataAncillaryDummy(path,name);
   theAncillary[ii]     = (AgataAncillaryScheme*)   theDummy;
   theConstructed[ii] = (AgataDetectorConstructed*)theDummy;
   break;
```

Notice that the first constructor, accepting a G4int and two G4String as arguments, is still implemented but it is not used.
Two inherited members must be initialized:

- *ancOffset* is a G4int (lowest offset for the instance of *AgataSensitiveDetector* corresponding to AgataAncillaryDummy). In order to get the correct value for ancOffset, please get in touch with Marcin Palacz (palacz@slcj.uw.edu.pl). Do not assign a value at random!
- *ancName* is a G4String (name of *AgataAncillaryDummy*)

Virtual methods

In the following, a brief description of the virtual methods requiring mandatory implementation is given.

FindMaterials

The pointers to the materials composing the detector are retrieved through their names. In case of failure, a non-zero value must be returned. The materials composing the ancillary detector must be declared and defined in the *DefineMaterials* or in the *DefineAdditionalMaterials* methods of *AgataDetectorConstruction*. In order to optimize the execution of the program, in *DefineAdditionalMaterials* only the materials effectively used should be defined. For each material, two members should be defined in *AgataAncillaryDummy*:

a *G4String* giving the name of the material
a *G4Material** where the pointer to the material will be stored

GetDetectorConstruction
This method should not differ from any of the examples. The pointer to the instantiation of the *AgataDetectorConstruction* object is retrieved and stored.

InitSensitiveDetector
If the ancillary detector is an active part of the geometry, the required *AgataSensitiveDetector* object(s) should be initialized and registered in this method. Notice that the offset should be retrieved with a call to *theDetector->GetAncillaryOffset()*, unless the *FIXED_OFFSET* compilation switch has been used. The ancillary detector offsets are handled dynamically. Each time that an *AgataSensitiveDetector* object is initialized and registered, *numAncSd* should be incremented (member inherited from *AgataAncillaryScheme*, which should be set to zero in the constructor of *AgataAncillaryDummy*).

ShowStatus
This method should just prints out to the terminal useful messages to remind the user the characteristics of the ancillary detector which has been placed.

WriteHeader
In the case of the ancillary detectors, this method is normally left empty.

GetSegmentNumber
This method returns the segment number corresponding to a given position, detector number and offset. The position is given in the relative frame of the detector rather than in the laboratory reference frame.

GetCrystalType
This method returns the crystal type corresponding to a given detector number and is meaningful only in the case of germanium detectors. In the case of the ancillary detectors, a dummy value should be returned, for instance -1.

Placement
This method handles the actual construction and placement of the ancillary detector. The world volume must be accessed through the *theDetector->HallLog()* and *theDetector->HallPhys()* methods. Additional methods can be called. See examples for specific details.

Ancillary detectors IDs and offsets

The updated IDs and offsets for the ancillary detectors presently defined within the AGATA simulation code, or under development, are the following:

Ancillary detector	**ID**	**Sensitive detector instances**	**Offsets**
Koeln	1	1	1000
Shell	2 (default)	1	2000
Mcp (DANTE)	3	1	3000
EUCLIDES	4	2	4000, 5000
Brick	6	0	6000
n-Wall	7	1	7000
DIAMANT	8	1	8000
EXOGAM	9	1	9000
HELENA	10	1	10000
RFD	11	1	11000
TRACE	12	2	12000, 13000
CUP	14	1	14000
GASPARD	15	1	15000
CASSANDRA	16	1	16000
AIDA	17	1	17000

In order to get the correct value for ID and offset of new ancillary devices, please get in touch with Marcin Palacz (palacz@slcj.uw.edu.pl). Do not assign a value at random!

Plugging in a new event generator

Any existing event generator written for a Geant4 simulation code can be plugged into the Agata simulation code with minimal changes. The most important rule to follow is that the event generator (more specifically, the implementation of the *UserGeneratorAction*) should inherit from the abstract class *AgataGeneration* rather than from *G4VUserPrimaryGeneratorAction,* which implies that the user should implement the following virtual methods:

BeginOfRun, EndOfRun
These methods are executed at the beginning/end of each run.

BeginOfEvent, EndOfEvent
These methods are executed at the beginning/end of each event.

GetStatus
This method prints out (to screen) some useful information.

PrintToFile
This method prints out (to the output file) the relevant information needed by the tracking codes.

GetBeginOfEventTag
This method returns the begin-of-event tag which is sent to the output file. Users of *mgt* should notice that the presently accepted tags are "-100\n" and "".

GetEventHeader
This method returns a string containing additional information (velocity and position of the source, time of emission) which is used by the tracking codes. Users of *mgt* should refer to the format of the list-mode file where the accepted data format is fully described.

GetParticleHeader
This method returns a string containing the relevant information on the emitted particle.

IsStartOfEvent, IsEndOfEvent
These methods tell whether it is the start (end) of an event (including the multiplicity of the event).

IsAbortRun
This method tells whether the run should be aborted or not.

GetAbortMessage
This method returns a string explaining why the run was aborted.

GetCascadeMult
This method returns the multiplicity of the particle/gamma cascade emitted during the event.

GeneratePrimaries
This method is the main one, where the vertex is created.

The header file for the new event generator should be included into *AgataGeneratorAction.cc* and the event generator should be instantiated as case 2 of the switch … case construct in the constructor of *AgataGeneratorAction,* replacing the instantiation of *AgataAlternativeGenerator* which is provided only as a very simple example. If the default AgataPhysicsList is not suitable,

additional processes can be registered with the *ConstructAdditionalProcesses* method, which the user should implement.

A Brief Agata Simulation Code Tutorial

In the following, a few example macros will be analyzed to show how to use the most important built-in commands for the simulation code. A macro file simply contains the sequence of commands which should be used running interactively. It can be used both from the Agata command prompt and running in batch mode (calling *Agata -b <macro file>*).

Lesson 1: ***Set the geometry to one symmetrical triple cluster and display it in VRML format***
Once the Agata code has been started, the sequence of commands to be executed is the following:

```
/Agata/detector/solidFile ./Symm/Symmsolid.list
/Agata/detector/angleFile ./Symm/Symmeuler.list
/Agata/detector/clustFile ./Symm/Symmclust.list
/Agata/detector/wallsFile ./Symm/Symmwalls.list
/Agata/detector/sliceFile ./Symm/Symmslice.list
/Agata/detector/enableCapsules
/Agata/detector/update
#
/vis/scene/create/vis/open VRML2FILE
/vis/viewer/set/lineSegmentsPerCircle 36
/vis/viewer/set/viewpointThetaPhi 45 45 deg
/vis/viewer/zoom 1.4
/vis/viewer/set/style surface
/vis/viewer/flush
```

It should be reminded that, by default, "naked" non-segmented germanium crystals shaped as in the A180 geometry are built. Therefore, in order to build the symmetrical triple cluster, the system must be notified to load the geometry parameters from different files, namely:

- *Symm/Symmsolid.list*: shape of the germanium crystal(s)
- *Symm/Symmslice.list*: segmentation of the germanium crystal(s)
- *Symm/Symmclust.list*: arrangement of the germanium crystal(s) within a cluster
- *Symm/Symmwalls.list*: shape of the cryostat walls
- *Symm/Symmeuler.list*: placement of the triple cluster in the experimental hall

The format of the geometry files is described in detail in the document which can be found at http://agata.pd.infn.it/documents/simulations/docs/GeometryDescription.html. The encapsulation is built if the `/Agata/detector/enableCapsules` command is executed (the opposite command, `/Agata/detector/disableCapsules` is also implemented). Each series of geometry changes must end with the `/Agata/detector/update` command, which notifies the changes to the geometry manager of the Geant4 kernel.

Since the first part of the macro file is likely to be executed many times, a macro file to set the geometry of AGATA to the symmetrical triple cluster is provided, namely the macros/geomSymm.mac file; the first part of the macro file could be replaced with:

```
/control/execute macros/geomSymm.mac
```

Other similar macros setting the geometry to reduced or full A180 configuration are provided, such as *macros/geom180-Demo.mac* (AGATA Demonstrator), *macros/geom180-1P.mac* (1π of AGATA), *macros/geom180.mac* (full A180 configuration). Notice that the geom180*.mac files differ only for the file loaded with `/Agata/detector/angleFile`. In order to create new arrangement of triple clusters, the user should simply produce a new angle file, containing the rotations and the traslations to place each cluster in the experimental hall. The number of valid lines in such file automatically give the number of clusters which will be placed.

The second part of the macro file is composed of standard Geant4 commands, for which we refer to the Geant4 documentation. Also in this case, a macro file with these commands is provided:

```
/control/execute macros/visVRML.mac
```

Other visualization macro files are provided, namely *macros/visGL.mac* and *macros/visDAWN.mac*.

Lesson 2: ***Evaluate the response of AGATA to monochromatic gammas and to a rotational band***
By default, monochromatic 1 MeV photons are emitted. Interactive commands are provided to change this energy (`/Agata/generator/gamma/energy`) or to select other kind of spectra such as a rotational band; our macro could look like this:

```
/control/execute macros/geom180.mac
/Agata/generator/gamma/energy 1332.5
/Agata/run/beamOn 100000
/Agata/generator/recoil/beta 5.
/Agata/generator/gamma/band 80. 90. 30
/Agata/run/beamOn 100000
```

where the A180 full configuration is selected, then 100000 photons of 1332.5 keV energy are fired and finally 1000000 rotational cascades are fired (a rotational cascade is a sequence of gammas equally spaced in energy) from a nucleus moving with velocity 5% of the speed of light. The lowest energy within the rotational cascade is 80 keV; the cascade is composed of 30 gammas, with a 90 keV energy spacing.

If you execute this macro, you will realize that the program did not produce anything (save for a few printouts on terminal!). In fact, by default no output is produced, meaning that the output should be enabled explicitly and that the correct macro should be:

```
/Agata/file/enableLM
/Agata/file/verbose 1
/control/execute macros/geom180.mac
/Agata/generator/gamma/energy 1332.5
/Agata/run/beamOn 100000
/Agata/generator/recoil/beta 5.
/Agata/generator/gamma/band 80. 90. 30
/Agata/run/beamOn 100000
```

Running the proper version of the macro, the program will produce two files: *GammaEvents.0000* and *GammaEvents.0001*. At each new run, if the output is enabled, a *GammaEvents.XXXX* will be written, where XXXX is the run number. No check on the existence of these files is provided, thus take care of saving your files somewhere else before the code overwrites them! Notice also that the Geant4 code does not take care of the tracking algorithms, therefore, in order to get meaningful results, you should process the output files with some tracking code (not included in the Agata Simulation Code distribution).

Lesson 3: ***Generate a dataset for Pulse Shape Calculations***

Let us assume that we want to provide the PSA Team with a set of interaction points corresponding to the interaction of 661 keV photons within the A180 geometry (a 137Cs source placed in the geometrical centre of the array). For this kind of calculations, the point coordinates should be given relative to the crystal frame rather than to the laboratory frame as is the default for the Agata code. Interactive commands are provided to set the output format to that needed by the tracking codes or by the pulse shape calculation codes, respectively `/Agata/file/info/trackingData` and /Agata/file/info/pulseShapeData. Our macro should look like the following:

```
/Agata/file/enableLM
/Agata/file/verbose 1
/Agata/file/info/pulseShapeData
/Agata/file/unitLength 1.
/Agata/file/unitEnergy 1.
/Agata/generator/gamma/energy 661.
/Agata/run/beamOn 100000
```

where the lengths will be written out in mm and the energies in keV.

Lesson 4: ***Shoot neutrons and gamma rays***

Let us assume that we want to evaluate the response of 1π of AGATA coupled to the n-Wall when a cascade of two evaporative neutrons and a band of 10 gammas is fired from a moving source. In this case, the main program should be started as *Agata -n -a 1 4* so that the hadronic cross sections are loaded and the ancillary detector is the neutron wall; the germanium detectors should not be positioned in the same place as the scintillators; the event generation should be chosen accordingly.

Concerning the geometry, it should be reminded that the geometry files for the partial configurations of AGATA assume that the detectors are positioned in the forward direction, where the neutron detectors should lie also. An interactive command is provided to rotate the germanium detector array to a new position, `/Agata/detector/rotateArray`. Therefore, the construction of our geometry should look like this:

```
/Agata/detector/enableAncillary
/Agata/detector/rotateArray 180. 0.
/control/execute macros/geom180-1P.mac
```

The gamma emission is properly set through the following command:

```
/Agata/generator/gamma/band 80. 90. 10
```

Concerning the neutrons, we have instead:

```
/Agata/generator/neutron/fileSpectrum specNeutrons.spc
/Agata/generator/neutron/gunType 4
/Agata/generator/neutron/multiplicity 2
```

where the shape of the centre-of-mass spectrum is read from *specNeutrons.spc*. Notice that the commands relevant for the neutrons are the same as for the gammas, where “gamma” is replaced by “neutron”. The full macro should finally look like this:

```
/Agata/file/enableLM
/Agata/file/verbose 1
/Agata/file/packingDistance 1.
/Agata/detector/enableAncillary
/Agata/detector/rotateArray 180. 0.
/control/execute macros/geom180-1P.mac
/Agata/generator/gamma/band 80. 90. 10
/Agata/generator/neutron/fileSpectrum specNeutrons.spc
/Agata/generator/neutron/gunType 4
/Agata/generator/neutron/multiplicity 2
/Agata/generator/recoil/beta 5.
/Agata/run/beamOn 1000000
```

When hadrons interact with the detectors, too many interaction points are found because of the larger length of the ionization tracks and of the larger number of secondary particles, which could be a problem for the tracking codes; in this cases the `/Agata/file/packingDistance` command is used to reduce the size of the output file.

Lesson 5: ***Simulate fusion-evaporation events with AGATA+EUCLIDES***

In this case, we choose to generate "realistic" events with an external program and use the Agata code just to evaluate the response function of the array. The program should start as *Agata -a 1 4 -n -Ext* to load the proper ancillary geometry and the hadronic cross sections.

As usual, the parameters of the geometry should be trimmed to the specific case. The ancillary detector should be enabled:

```
/Agata/detector/enableAncillary
```

The proper absorber foils for EUCLIDES should be chosen, as well as the scattering chamber:

```
/Agata/detector/ancillary/Euclides/layerNumber 2
/Agata/detector/ancillary/Euclides/layerMaterial 1 Aluminium
/Agata/detector/ancillary/Euclides/layerMaterial 2 Aluminium
/Agata/detector/ancillary/Euclides/layerThick 1 12.
/Agata/detector/ancillary/Euclides/layerThick 2 12.
/Agata/detector/ancillary/Euclides/layerRange 2 15. 60.
/Agata/detector/chamberMaterial Aluminium
/Agata/detector/chamberRadius 115.
```

The geometry is completed by loading the full A180 configuration:

```
/control/execute macros/geom180.mac
```

In this case, the details of the events are stored in an external formatted file, therefore the only thing to do is to select that specific file:

```
/Agata/generator/emitter/eventFile /d01/recchia/cascade-gsort/28Si/28Si28Si-50milb.event
```

In this case, the file was obviously */d01/recchia/cascade-gsort/28Si/28Si28Si-50milb.event*. The full macro file will look like this:

```
/Agata/file/enableLM
/Agata/file/verbose 1
/Agata/file/packingDistance 1.
/Agata/generator/emitter/eventFile /d01/recchia/cascade-gsort/28Si/28Si28Si-50milb.event
/Agata/detector/ancillary/Euclides/layerNumber 2
/Agata/detector/ancillary/Euclides/layerMaterial 1 Aluminium
/Agata/detector/ancillary/Euclides/layerMaterial 2 Aluminium
/Agata/detector/ancillary/Euclides/layerThick 1 12.
/Agata/detector/ancillary/Euclides/layerThick 2 12.
/Agata/detector/ancillary/Euclides/layerRange 2 15. 60.
/Agata/detector/chamberMaterial Aluminium
/Agata/detector/chamberRadius 115.
/Agata/detector/enableAncillary
/control/execute macros/geom180-1P.mac
/Agata/run/beamOn 15000000
```

Format of the list-mode output file

Version 7.4.0, April 7th 2009

This document describes the format of the list-mode files produced by the Agata simulation program. In most cases, the user will only be interested in the actual events (starting after the '$' symbol). In this case, please do feel free to skip the first part of this document. The header contains useful information about the way the file was generated (geometry and event generation).

The list-mode file is composed of two main sections:

- header
  1. geometry description
  2. event generator description
  3. read out geometry
  4. beginning of events
- events

Header

At present, the header begins with the name of the program which generated the file, the version of the data format, a short description of the event format and the tracking method used in the simulation:

AGATA version	begin of file, version of the format
OUTPUT_MASK xxxxxxxx	specifies which parameters are actually written (see events section) 0 --> not written, 1 --> written
DISTFACTOR x	Specifies the unit of length (in mm), according to which the distances are written out.
ENERFACTOR	Specifies the unit of energy (in keV), according to which the energies are written out.
G4TRACKING method	refers to the method used to analyze the tracks in the AgataSensitiveDetector class
DATE day week DD time year	date and time of the opening of the run

Geometry description

The geometry section always begins with the keyword GEOMETRY:

GEOMETRY	geometry section begins

The geometry description section changes with the implementation of the actual arrangement of the detectors, the present possibilities are:

a) full array

The files used to generate this geometry are copied into the list-mode file.

AGATA	beginning of geometry description

AGATA	beginning of geometry description
SUMMARY Rmin Rmax nDets nSolids nSlice_1 nSector_1 ... nSlice_nSolids nSector_nSolids	inner, outer radius of an equivalent shell (in terms of the unit of length specified above), array composed of nDets detectors with nSolids different shapes, each of them divided into nSlice_i slices and nSector_i sectors
PASSIVE x	specifies whether the passivated areas have been used (1) or not (0) in the calculation
CAPSULES x	specifies whether the Aluminium capsules have been used (1) or not (0) in the calculation
TRANSFORMATION x y z theta phi	Specifies the amount by which the array has been shifted (x, y, z) and rotated (theta, phi in degrees) with respect to the original geometry files.

In case ancillary detectors have been defined, the following information is written out for each of them:

ANCIL ancName numSD offset_1 … offset_numSD	ancName is the name of the ancillary detector, which registers numSD instances of AgataSensitiveDetector. The offsets corresponding to each of them are written out.

The following information is written only when the level of verbosity is greater than zero (that is, 1 or 2):

SOLID solidFile	solidFile contains the vertexes of the polyhedra
<content>	if solidFile exists, its content is copied here.
ENDSOLID	
CLUSTER clustFile	clustFile contains the description of the clusters (which solids, their positions and rotations)
<content>	if clustFile exists, its content is copied here.
ENDCLUSTER	
WALLS wallsFile	wallsFile contains the vertexes of the walls polyhedra relative to the clusters.
<content>	if wallsFile exists, its content is copied here.
ENDWALLS	
EULER eulerFile	eulerFile contains the angles and the positions used to place the clusters.
<content>	if eulerFile exists, its content is copied here.

ENDEULER	
SLICES sliceFile	sliceFile contains the distances of the planes used to generate the readout segments. This part is written only when the read out geometry is generated (that is, if sliceFile exists).
<content>	if sliceFile exists, its content is copied here.
ENDSLICES	
CRYSTAL_LUT	The lookup table assigning the crystal type (as defined in solidFile) to each detector number is included here with the following format:
detNum crystType	
ENDCRYSTAL_LUT	
PLANAR_LUT	The lookup table specifying whether each crystal is planar (1) or coaxial (0) is included here with the following format:
detNum planar	
ENDPLANAR_LUT	

The following information is written only when the level of verbosity is greater than 1 (that is, 2 or 3):

POSITION_CLUSTERS	the rotation matrices transforming from the world reference frame to the cluster reference frame and the positions of the centres of the clusters are included here with the following format:
nEul 0 Px Py Pz	nEul is the number of the cluster 0: position (Px, Py, Pz) in cm
1 xx xy xz	1: first row of a rotation matrix RotMxx, RotMxy, RotMxz. RotM is the rotation matrix from the cluster to the world reference frame.
2 yx yy yz	2: second row of a rotation matrix RotMyx, RotMyy, RotMyz
3 zx zy zz	3: third row of a rotation matrix RotMzx, RotMzy, RotMzz
ENDPOSITION_CLUSTERS	end of the section
POSITION_CRYSTALS	The positions of the centre of the crystals in cm are included here.
nCryst 0 Px Py Pz	nCryst is the number of the detector 0: position (Px, Py, Pz) in cm

1 xx xy xz	1: first row of a rotation matrix RotMxx, RotMxy, RotMxz. RotM is the rotation matrix from the detector to the world reference frame.
2 yx yy yz	2: second row of a rotation matrix RotMyx, RotMyy, RotMyz
3 zx zy zz	3: third row of a rotation matrix RotMzx, RotMzy, RotMzz
ENDPOSITION_CRYSTALS	end of the section
POSITION_SEGMENTS	The positions of the centre of the segments (in cm) and the volume of the segments (in cm3) are included here.
nCryst nSlice nSector Px Py Pz Vol	nCryst is the number of the detector; nSlice, nSector are the number of slice/sector; (Px, Py, Pz) is the position of the centre; Vol is the volume of the segment
ENDPOSITION_SEGMENTS	end of the section

The following information is written only when the level of verbosity is greater than 2 (that is, 3):

TRANSFORMATION_CRYSTALS	the transformations which should be applied to place each crystal in the world reference frame are written out in terms of their components (a 3x4 matrix). The format is the following:
nCryst tCryst 0 Rxx Rxy Rxz Tx	nCryst is the crystal number, tCryst the crystal type (as in the CRYSTAL_LUT above), Rxx, Rxy, Rxz the components of the rotation matrix, Tx the component of the traslation in terms of the unit of length.
1 Ryx Ryy Ryz Ty	Ryx, Ryy, Ryz the components of the rotation matrix, Ty the component of the traslation in terms of the unit of length.
2 Rzx Rzy Rzz Tz	Rzx, Rzy, Rzz the components of the rotation matrix, Tz the component of the traslation in terms of the unit of length.
ENDTRANSFORMATION_CRYSTALS	end of the section

b) germanium shell

SHELL	beginning of geometry description
Rmin Rmax	inner, outer radius of the shell (in terms of the unit of length)

c) single coaxial crystal

No details on the dimensions of the detector are given:

CBAR	beginning of geometry description

d) empty geometry

NONE	beginning of geometry description

The geometry description is always ended by the following line:

ENDGEOMETRY	end of geometry description

Event generator description

In this section of the list-mode file the relevant information on the source and on the radiation (generally gammas) are collected.

GENERATOR type	Type of event generator used; 0 corresponding to events produced by the built-in generator, 1 to events read from an external file

1) Built-in generation

In this case, a cascade of Mn neutrons, Ma alphas, Mp protons, Me- electrons, Me+ positrons and Mg gammas is assumed to happen at each event. The vector velocity of the recoil is not modified by the emission of the particles.

a) Dummy keyword

ENDGENERATOR	Dummy keyword for consistence with the External generation case

b) Source

The velocity (module and direction) and the position of the source are given:

RECOIL beta sigma Dx Dy Dz ang	verage recoil speed (in % of c), dispersion (fraction of beta), average direction of the recoil, half-opening angle (degrees) of the recoil cone
SOURCE type movement Px Py Pz	type, movement (movement=1-->source moves during the emission of the cascade) and position of the source (central position of the source in case of diffused source)
TARGET Sx Sy Sz	size (in terms of the unit of length) of the target. Written only in case of diffused source.

c) Particles

For each particle with multiplicity greater than zero, the characteristics of the energy spectrum and the multiplicity are given. The possible cases are (for alphas, protons, neutrons, electrons and positrons substitute respectively ALPHA, PROTON, NEUTRON, E-, E+ to the keyword GAMMA):

- monochromatic particles

GAMMA 1	
egamma	energy of the particle (in terms of the unit of energy)

- rotational band

GAMMA mult	multiplicity
egamma_1	energy of the 1st particle (in terms of the unit of energy)
...	
egamma_mult	energy of the mult-th particle (in terms of the unit of energy)

- discrete energies from file

GAMMA mult	multiplicity
egamma_1	energy of the 1st particle (in terms of the unit of energy)
...	
egamma_mult	energy of the mult-th particle (in terms of the unit of energy)

- flat distribution

GAMMA -1	
Emin Emax	limits of the distribution (in terms of the unit of energy)

- continuous spectrum (from file)

GAMMA -2	

- discrete energies from file (with intensities)

GAMMA mult	multiplicity
egamma_1	energy of the 1st particle (in terms of the unit of energy)
...	
egamma_mult	energy of the mult-th particle (in terms of the unit of energy)

Obviously, the hadronic part is written only if the hadronic part has been loaded at the start of the program.

2) External generation

In this case, the event structure is read from an external file. The vector velocity of the recoil is modified by the emission of the particles. The cascade does not repeat at each event.

EVENTFILE eventFile	file from which the events are read

EMITTED type nEmi emi_0 ... emi_N	type is the format of the information on the emitted particles in eventFile; nEmi is the number of emitted particles defined in eventFile, emi_0, ... emi_N are the emitted particles encoded as following: 1 -> gamma 2 -> neutron 3 -> proton 4 -> deuteron 5 -> triton 6 -> 3He 7 -> alpha 8 -> generic ion 97 -> electron 98 -> positron 99 -> geantino
EMITTER type	type is the format of the information on the recoils in eventFile; it might be followed by additional information when part of the data are produced by the built-in generator
REACTION zT aT zB aB eB [sigma Dx Dy Dz angle]	reaction (target Z, A, beam Z, A, energy) plus additional information when the recoil direction and energy are produced by the built-in generator: sigma is the fractional dispersion in energy, (Dx, Dy, Dz) is the average recoil direction, angle is the half-opening angle of the recoils
SOURCE type movement Px Py Pz	type, movement (movement=1-->source moves during the emission of the cascade) and position of the source (central position of the source in case of diffused source)
TARGET Sx Sy Sz	size (in terms of the unit of length) of the target. Written only in case of diffused source.
ENDGENERATOR	end of section

Beginning of events

The beginning of the events section is marked by a $ sign:

$	beginning of events

Events

The information on the fired particle and on its interactions with the active elements is listed with the following format:

-100	begin of event (only in the case of events read from file)
-101 beta Dx Dy Dz	module and direction of the source velocity (beta: fraction of c). For the internal generation this is written only in case of variable velocity (event-by-event); in case of a cascade of more particles this line is written only at the beginning of the cascade. For the events read from file this is written before each particle unless suppressed by the user with the specific command.
-102 Px Py Pz	position (Px, Py, Pz) of the source in terms of the unit of length. For the internal generation this is written only when the source position can change event-by-event, that is in case of diffused source or in case of transitions of non-zero lifetime and non-zero recoil velocity. The information is written at the beginning of the cascade and before any change along the cascade (due to transitions of non-zero lifetime and non-zero recoil velocity). For the events read from file this is written before each particle unless suppressed by the user with the specific command.
-103 time	time of emission in ns. This parameter is written when it is non-zero for the first transition or when it changes along the cascade(due to transitions of non-zero lifetime and non-zero recoil velocity).
-type energy Dx Dy Dz evNum	type and energy of the particle (in terms of the unit of energy) fired in direction (Dx, Dy, Dz) at event# evNum. type gives the kind of particle: 1 -> gamma 2 -> neutron 3 -> proton 4 -> deuteron 5 -> triton 6 -> 3He 7 -> alpha 8 -> generic ion 97 -> electron 98 -> positron 99 -> geantino

From format version 4.0.0 above, it is possible to choose which parameters relevant to the interaction points can be written to file, as specified by the OUTPUT_MASK line in the header section of the file.
When interactions have occured, the parameters are written in the following order:

ndet	detector number in which the interaction occured
edep	energy (in terms of the unit of energy) released in the interaction

Px Py Pz	absolute position (laboratory reference frame, in terms of the unit of length) in which the interaction occured
P'x P'y P'z	position (relative to the crystal, in terms of the unit of length) in which the interaction occurred (mgs reference frame)
P”x P”y P”z	position (relative to the crystal, in terms of the unit of length) in which the interaction occurred (mgs reference frame)
nseg	segment number (encoded as 10*slice number + sector number)
time	time (in ns) from the start of the event in which the interaction occurred
nInt	interaction which generated the interaction point. The interactions are listed as following: 1 --> Compton scattering 2 --> Photoelectric absorption 3 --> Pair production 4 --> Rayleigh scattering 5 --> transportation 99 --> other In the default case of G4TRACKING 0, the interaction is directly the interaction which generated the interaction point if the particle is primary (i.e. the particle shoot with G4ParticleGun in AgataGeneratorAction.cc); in case of secondary particles, it is the interaction which generated the secondary particle. In the case of G4TRACKING 1, only the primary particles are tracked and the correspondance listed above holds valid.

Input Event File Format

Version 7.2.0, November 3rd 2006

This document describes the format of the input event file in case the Agata simulation program is started with the *-Ext* flag. Short sample files can be found in the ./events subdirectory.

Differently from the built-in generator, here nuclear reactions are assumed. By specifying Beam+Target, energy and momentum conservation can be properly taken care of on an event-by-event basis. By "event" here we mean a number of heavy ions recoiling and emitting light particles and gamma rays. In principle, with this definition, several kinds of reactions can be treated in a "realistic" way, such as fusion-evaporation (one recoiling nucleus), transfer reactions (two nuclei) or fragmentation reactions (more nuclei). In most cases, however, the velocity of each fragment should be specified event-by-event since the value deduced by the Beam+Target kinematics would be a poor approximation.

In order to fire a particle, the simulation code needs the following information:
- Energy in the laboratory frame
- Direction in the laboratory frame
- Position of emission in the laboratory frame

Optional information is:
Polarization of the particle
Time of emission from the actual begin of event.

The values in the laboratory frame can be deduced from the corresponding centre-of-mass values once some information on the emitting nucleus is available:
- Emitting nucleus vector velocity
- Emitting nucleus Z, A

The general philosophy of the program is to use the built-in generator to generate the relevant part of the information which is not directly read from the input file. This is specified by two flags (emitterType emittedType) which must be provided at the very beginning of the event file with the following line:

FORMAT emitterType emittedType

where emitterType = 0, 1, 2, 3, 4 and emittedType = 0, 1, 2, 3, 4. If emitterType and/or emittedType are missing, the program resets to the default values 0, 0 corresponding to the case in which the maximum level of information is read from the input file. The higher values correspond to the minimum information decoded from file (and therefore to the maximum information generated with the built-in generator).

Lines starting with '#' are skipped as comments.

By default the simulation code assumes a proton beam, a 12C target and 0.0 MeV beam energy. These values are used to calculate the centre-of-mass velocity and they can be overridden using the built-in commands:

/Agata/generator/emitter/Beam
/Agata/generator/emitter/energy

/Agata/generator/emitter/Target

Alternatively, the beam can be defined via the following instructions:

ZBEAM ***zBeam***
ABEAM ***aBeam***
EBEAM ***eBeam[MeV]***

The change will be effective only if both zBeam and aBeam are defined. The instructions for the target are quite similar:

ZTARG ***zTarg***
ATARG ***aTarg***

Another possibility is to use the following syntax:

REACTION ***zBeam aBeam zTarget aTarget eBeam[Mev]***

The values effectively used, in case the REACTION line specifies different values from ZBEAM, ABEAM, EBEAM, ZTARG, ATARG, or in case some values have been provided with the interactive commands, will be the last ones written in the input file.

A line specifying the number and type of emitted particles must be present:

EMITTED ***nEmitted emi_0 ... emi_N***

where the type of emitted particles is encoded as following:
1 -> gamma
2 -> neutron
3 -> proton
4 -> deuteron
5 -> triton
6 -> 3He
7 -> alpha
8 -> generic ion
97 -> electron
98 -> positron
99 -> geantino

Starting from September 2006, it is possible to limit the angular range in which the emitted particles are shot, using the following instruction:

RANGE ***type thetaMin thetaMax phiMin phiMax***

where type is the type of emitted particle as defined above and thetaMin, thetaMax, phiMin, phiMax are in degrees.

Each event must begin with a '$' tag; each event should include at least an emitter line; the first line after a '$' tag should be an emitter line.
After an emitter line, any number of emitted lines may appear; for each emitted particle, momentum conservation is applied, resetting after an emitter or after a begin of event tag.

The structure of the “emitter” and “emitted” lines will depend respectively on the values of emitterType and emittedType. The possible values are:

Emitter line

EmitterType	Line
0	-101 zEmi aEmi eEmi Dx Dy Dz Sx Sy Sz
1	-101 zEmi aEmi eEmi Dx Dy Dz
2	-101 zEmi aEmi eEmi
3	-101 zEmi aEmi
4	-101

Where:
zEmi, aEmi are the atomic and the mass number of the nucleus emitting the particle;
eEmi is the energy in the laboratory frame (in MeV) of the nucleus emitting the particle;
(Dx, Dy, Dz) is the direction in the laboratory frame of the nucleus emitting the particle;
(Sx, Sy, Sz) is the position of emission of the particle in the laboratory frame (in cm).

Emitted line

EmittedType	Line
0	type Elab Dx Dy Dz Sx Sy Sz [t P]
1	type ECM D'x D'y D'z Sx Sy Sz [t P]
2	type Elab Dx Dy Dz [t P]
3	type ECM D'x D'y D'z [t P]
4	type ECM [t P]

Where:
type is the type of emitted particle
Elab is the laboratory energy (in keV)
ECM is the centre-of-mass energy (in keV)
(Dx, Dy, Dz) is the direction in the laboratory frame
(D'x D'y D'z) is the direction in the centre-of-mass frame
(Sx, Sy, Sz) is the position of emission (in cm)

Optional parameters (from November 2006):
t is the time of emission (in ns)
P is the angle (in degrees) between the polarization (electric field) vector and the “orthogonal” state, that is the vector orthogonal to the scattering plane defined by the recoil nucleus velocity and the gamma direction. In case of polarized photons, it is the user responsibility to specify such direction on an event-by-event basis.