PARIS Documentation

Marc Labiche

STFC Daresbury Laboratory,
marc.labiche@stfc.ac.uk

February 3, 2011

Contents

1 Introduction

The PARIS project is a new γ-ray array which aims at addressing topical nuclear physics research programs with SPIRAL2 beams. We describe this project here as it is developed within the simulation GEANT4-based NPTool framework. The design and choice of the scintillator is still matter of discussion within the collaboration, and several and independent simulation frameworks are being used to help converging towards a definitive solution. In NPTool, we consider an array of phoswich detectors consisting of $LaBr_3$ and NaI (originally CsI) scintillators, surrounded by NaI (originally CsI) shields, in an EXOGAM-like spherical configuration. The modular aspect of the NPTool framework allows for the PARIS array to be modelled either as a standalone detector or coupled to other NPTool modelled detectors like GASPARD and MUST2.The general structure of the PARIS program is based on the structure of the GASPARD program, thus many details described here are similar to the GASPARD document.

2 NPSimulation

2.1 Specificity of Paris

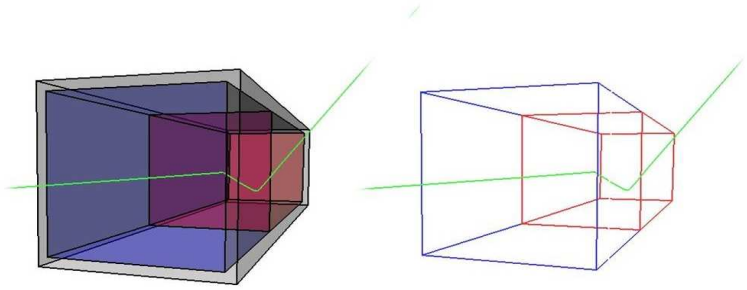

Figure 1: Two drawings of a single phoswich module. The $LaBr_3$ scintillator is in blue and the NaI scintillator is in red

In this released of NPTool, the PARIS γ-ray array consists of phoswich detectors with squared cross section of 2"$\times$2". The phoswich first layer is a 2" long $LaBr_3$ scintillator. The second layer is a 6" long scintillator of NaI scintillator (see fig. 1). Originally, CsI scintillators were considered and, for

that unique reason, many comments and variable names in the program still point at CsI scintillators. The user should be aware that the CsI material has been replaced by NaI following prototype testing results. All the comments and variable names will be corrected in future a release.

The phoswich modules are grouped in 3×3 clusters (see fig. 2) or used as single modules (see fig. 1) . All are positioned to form a spherical configuration of an adjustable inner radius between 20.8 and 23.5 cm, as show in fig. 3. A shield made of extra NaI scintillators surrounds the cluster and single phoswich to fill the gaps in between, and give a spherical geometry to the full array (see fig. 4). The clusters and single phoswich modules of the Paris detector are described in the Paris class defined in the Paris.{hh,cc} files. The geometry of the cluster is defined in the ParisCluster.{hh,cc} files, and the geometry of a single phoswich module is defined in the ParisPhoswich.{hh,cc} files. The shields of CsI scintillators are described in a different class called Shield, and defined in the Shield.{hh,cc} files. Different shield geometries are used for the clusters and single phoswich modules. The geometry of the cluster shield is defined in the files ShieldClParis.{hh,cc}. The geometry of the single phoswich module shield is defined in the files ShieldPhParis.{hh,cc}.

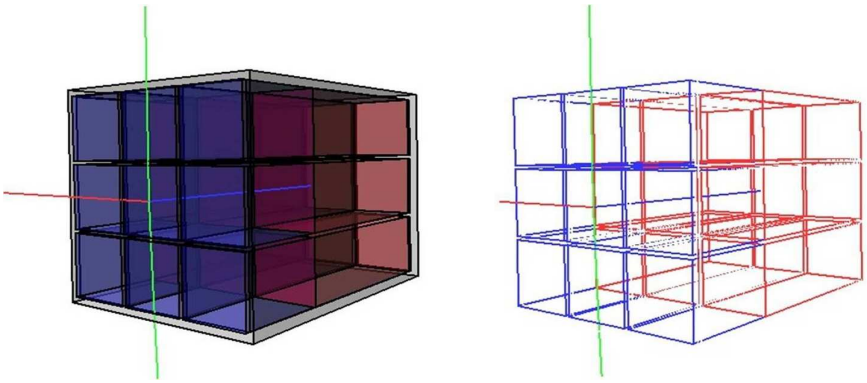

Figure 2: Two drawings of a 3×3 cluster. The $LaBr_3$ scintillator is in blue and the NaI scintillator is in red

Since the Paris detector and the NaI shield are registered in the DetectorConstructor.cc file ,they are available for NPSimulation.

As for other NPTool modelled detectors, in order to manage the different detector shapes (cluster, single module, cluster shield, single module shield) of the Paris array, the Paris and Shield classes hold a vector of ParisModule and ShieldModule objects, respectively, from which are deriving all the different shapes (ParisCluster, ParisPhoswich,ShieldClParis, and ShieldPhParis

classes).

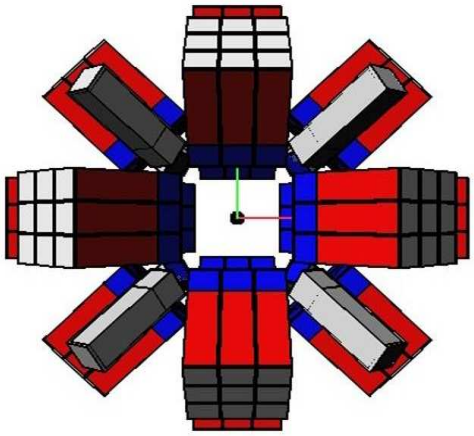

Figure 3: Spherical configuration of Paris, without shielding.

2.2 Running the simulation

As for other NPTool simulations, to run Paris simulations the following command line should be executed:

```
Simulation -D yyy.detector -E  xxx.reaction
```

where xxx.reaction is an input file describing the event generator and yyy.detector is an input file describing the detector geometry. All these input files are based on keywords and can be found in the $NPTool/Inputs subdirectories.

2.2.1 Event Generators

In the distributed version, a source of γ-ray either at rest or in motion along the beam axis can be used. For an isotropic or a collimated source at rest, the event generator file isotropic.source in the Inputs/EventGenerator is available.An example is given in the file gamma.source. For a source in the move, one can use the example given in the source.reaction file.

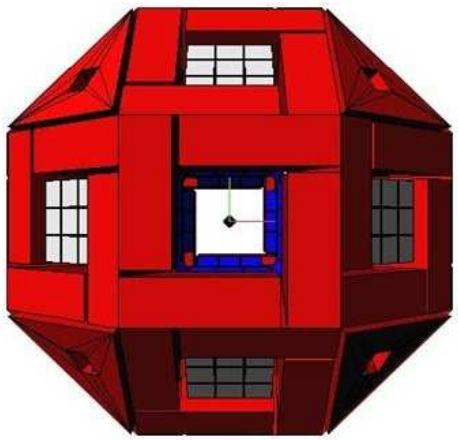

Figure 4: Spherical configuration of Paris, with shielding.

2.2.2 Detector Configurations

The keywords associated to the detector geometry file are different for each detector. In case of the Paris detector, an example with all the kind of detector shapes available at the moment is given below:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Paris
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ParisCluster
  X1_Y1=       -84.5 -208   84.5
  X1_Y128=      84.5 -208   84.5
  X128_Y1=     -84.5 -208  -84.5
  X128_Y128=    84.5 -208  -84.5
  VIS=all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ParisPhoswich
  X1_Y1=      -128.6063166   143.3590149   -88.301235
  X1_Y128=    -151.8764696    96.81870993 -111.571389
  X128_Y1=     -88.30122956  143.3590149  -128.606323
  X128_Y128=  -111.5713826    96.81870993  -151.876476
  VIS= all
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Shield
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ShieldClParis
  X1_Y1=       151.375  153.75   364
  X1_Y128=     151.375   91.5    364
  X128_Y1=     -84.5    153.75   364
  X128_Y128=   -84.5     91.5    364
  VIS= all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
ShieldPhParis
  THETA= 54.73
  PHI=   45.02
  R=    248
  BETA=   0 0 -30
  VIS= all
```

The Paris and shield detectors are independent. In other words, one can use Paris detectors with or without the NaI shields.

In order to declare a Paris detector in NPSimulation, the key word *Paris* should be specified in the geometry file. It should then be followed by other keywords concerning the different detectors present. Such keywords available at the moment are:

- ParisCluster
- ParisPhoswich

Each keyword corresponds to a detector shape which has its own set of keywords which is used to position the detector in the world volume. For the Shield detectors, the keyword *Shield* should be specified in the geometry file, followed by other keywords associated to the different detector shield for cluster (Cl) and single phoswich (Ph):

- ShieldClParis
- ShieldPhParis

In principle, to position the detectors two possibilities exist. Either the Cartesian coordinates (x,y,z) of each detector's corner are specified with the keywords *X1_ Y1*, *X1_ Y128*, *X128_ Y1* and *X128_ Y128*(case of ParisCluster, ParisPhoswich and ShieldClParis), either the spherical coordinates of the

detector's centre are specified with the keywords *R*, *THETA* and *PHI* (case of ShieldPhParis). While the first solution is very helpful when working with the mechanical engineers, the second solution is useful when investigating new geometries. However, up to now, only one solution for each type of Paris and Shield detectors has been tested, and this is the solution given above as an example.

2.2.3 Physics list

The electromagnetic physics process used by default in NPTool is the standard GEANT4 physics list. The user can however choose a different physics list amongst the Standard, Low Energy and Penelope physics list by modifying the file: $NPTool/NPSimulation/src/PhysicList.cc. As an example,in order to choose the low energy physics list (recommended), open the PhysicList.cc file and comment and uncomment the relevant lines as follow:

```
if (particleName == "gamma") {
  // gamma
  //standard Geant4
  //pmanager->AddDiscreteProcess(new G4PhotoElectricEffect);
  //pmanager->AddDiscreteProcess(new G4ComptonScattering);
  //pmanager->AddDiscreteProcess(new G4GammaConversion);

  //Low energy
  //pmanager->AddDiscreteProcess(new G4LowEnergyPhotoElectric);
  pmanager->AddDiscreteProcess(new G4LowEnergyCompton);
  G4LowEnergyPhotoElectric* LePeprocess = new G4LowEnergyPhotoElectric();
  LePeprocess->ActivateAuger(true);
  LePeprocess->SetCutForLowEnSecPhotons(0.250*keV);
  LePeprocess->SetCutForLowEnSecElectrons(0.250*keV);
  pmanager->AddDiscreteProcess(LePeprocess);
  pmanager->AddDiscreteProcess(new G4LowEnergyGammaConversion);
  pmanager->AddDiscreteProcess(new G4LowEnergyRayleigh);
  pmanager->AddProcess(new G4StepLimiter(),-1,-1,3);

  // Penelope
  //pmanager->AddDiscreteProcess(new G4PenelopePhotoElectric);
  //pmanager->AddDiscreteProcess(new G4PenelopeCompton);
  //pmanager->AddDiscreteProcess(new G4PenelopeGammaConversion);
  //pmanager->AddDiscreteProcess(new G4PenelopeRayleigh);

} else if (particleName == "e-") {
  //electron
  pmanager->AddProcess(new G4MultipleScattering, -1,  1, 1);
  //standard geant4:
  //pmanager->AddProcess(new G4eIonisation, -1,  2, 2);
  //pmanager->AddProcess(new G4eBremsstrahlung, -1, -1, 3);

  // Low energy:
  G4LowEnergyIonisation* LeIoprocess = new G4LowEnergyIonisation("IONI");
  LeIoprocess->ActivateAuger(true);
  LeIoprocess->SetCutForLowEnSecPhotons(0.1*keV);
  LeIoprocess->SetCutForLowEnSecElectrons(0.1*keV);
  pmanager->AddProcess(LeIoprocess,-1,2,2);
  //pmanager->AddProcess(new G4LowEnergyIonisation, -1,  2, 2);
```

```
    G4LowEnergyBremsstrahlung* LeBrprocess = new G4LowEnergyBremsstrahlung();
    pmanager->AddProcess(LeBrprocess, -1, -1, 3);
    pmanager->AddProcess(new G4StepLimiter,-1,-1,3);
    //pmanager->AddProcess(new G4LowEnergyBremsstrahlung, -1, -1, 3);

    // Penelope:
    // pmanager->AddProcess(new G4PenelopeIonisation, -1, 2, 2);
    // pmanager->AddProcess(new G4PenelopeBremsstrahlung, -1, -1, 3);

  } else if (particleName == "e+") {
    //positron
    pmanager->AddProcess(new G4MultipleScattering, -1,  1, 1 );
    // standard Geant4 and Low energy
    pmanager->AddProcess(new G4eIonisation, -1,  2, 2 );
    pmanager->AddProcess(new G4eBremsstrahlung, -1, -1, 3 );
    pmanager->AddProcess(new G4eplusAnnihilation,  0, -1, 4 );
    pmanager->AddProcess(new G4StepLimiter(),  -1, -1, 3 );
    //Penelope:
    //pmanager->AddProcess(new G4PenelopeIonisation , -1,  2, 2 );
    //pmanager->AddProcess(new G4PenelopeBremsstrahlung, -1, -1, 3 );
    //pmanager->AddProcess(new G4PenelopeAnnihilation,  0, -1, 4 );
```

2.3 Simulation output

Again, as in other NPTool simulations, the results of the simulations are in the ROOT format and the output file is stored in the $NPTool/Output/Simulation directory. If the PARIS geometry input file includes the NaI Shield, the output ROOT file contains four classes:

- TInitialConditions: This class records all the information concerning the event generator such as the vertex of interaction, the angles of emitted particles in the center of mass and laboratory frames...
- TInteractionCoordinates: Although this class appears in the tree, it is not in use with the PARIS array alone. Please, see GASPARD documentation for more details.
- TParisData: This class stores the results of the simulation for the cluster and single phoswich detector. Independently of the number and shape of the detectors involved in the geometry, only *one* class is created for the whole PARIS detector. For each event, the energy for each layer of scintillators is recorded. The time, detector number and crystal number information will be added in future release. At the moment, this is equivalent of having a perfect add-back algorithm between crystals within each layer.
- TShieldData: This class stores the results of the simulation for Shield detector. Independently of the number and shape of the detectors involved in the geometry, only *one* class is created for the whole shield

detector. For each event, the energy is recorded. The time, detector number and crystal number information will be added in future release. At the moment, this is equivalent of having a perfect add-back algorithm between the shield crystals.

3 NPAnalysis

3.1 General

Only a very preliminary analysis code is provided with this release.

3.2 Paris

The analysis code for the Paris array is in the $NPTool/NPAnalysis/Paris directory. For the moment the main feature is the photopeak efficiency as function of the γ-ray incoming energy. The photopeak efficiency in each layer of the array is calculated as well as for the NaI sheild. The full photopeak efficiencies, with and without add-back correction between the different layers, are also determined.

3.2.1 Running the analysis

To run NPAnalysis the following command line should be executed in the $NPTool/NPAnalysis/Paris directory:

```
./Analysis -D yyy.detector -E xxx.reaction -R RunToTreat.txt
```

where xxx.reaction is the input file describing the event generator used in NPSimulation and yyy.detector is the input file describing the detector geometry used in NPSimulation. All these input files are based on keywords and can be found in the $NPTool/Inputs subdirectories. The RunToTreat.txt file contains the name of the files (either from NPSimulation or from real experiment) which should be analysed. The name of the tree should also be specified. An example of such a file is given here:

```
TTreeName
        SimulatedTree
RootFileName
       ../../Outputs/Simulation/myResult.root
%       ../../Outputs/Simulation/mySimul.root
```

3.2.2 Results of the analysis

The results of the analysis are stored in a ROOT file in the $NPTool/Output/Analysis directory.

3.2.3 Structure of the analysis

******* to be documented *********