PET UJ Report 2020

J-PET Monte Carlo simulations in Geant4 toolkit
User Manual for version 4.0

J-PET MC group

Instytut Fizyki, Uniwersytet Jagiellonski, Poland

November 2020

Contents

1 Introduction **4**

2 Physics and geometry reference manual **5**
2.1 Detector geometry . . . 5
2.1.1 Layers of the scintillators . . . 5
2.1.2 Detector frame . . . 5
2.1.3 Annihilation chambers . . . 6
2.2 Physics . . . 8
2.2.1 Events types . . . 8
2.2.2 Physical processes . . . 10
2.2.3 How hits are created? . . . 11
2.3 What can be simulated? . . . 11

3 User's guide: results of the simulation **13**
3.1 Structure of the generated information . . . 13
3.2 Event types registered in scintillator . . . 13
3.3 DecayTree . . . 14
3.4 Control histograms . . . 14
3.5 Files available at servers . . . 15

4 Using the simulation output with J-PET Framework **16**
4.1 How to run? and what to expect . . . 16
4.2 JPetGeantParser in Framework . . . 16
4.3 Timescale in JPetMC and Framework . . . 16
4.4 How to change detector resolution in simulation? . . . 16

5 Developer's guide: Installation and usage **19**
5.1 Source code . . . 19
5.2 Requirements . . . 19
5.3 Compilation procedure . . . 19
5.4 How to run? . . . 20
5.5 Examples . . . 20
5.5.1 Ps decays in the chamber . . . 20
5.5.2 Beam of the photons . . . 20
5.5.3 Cylindrical isotope . . . 20
5.5.4 NEMA sources . . . 21
5.6 Interaction with the simulation . . . 21
5.6.1 Detector construction messenger . . . 21
5.6.2 Material Extention messenger . . . 22
5.6.3 Event messenger . . . 22

5.6.4 Primary Generator Action messenger for run parameters 22

6 Documentation & support **23**
6.1 Useful links 23
6.2 Doxygen 23
6.3 Posting bugs and getting support 23
6.4 Contacts 23

1. Introduction

J-PET-Geant4 is a Monte Carlo simulation program designed for the J-PET detector created in Geant4 toolkit. Program output is prepared in a Root tree format and can be directly loaded into J-PET Framework software.

From the user point of view, the files containing MC-generated events have to be processed in the same manner as collected data. For this purpose a dedicated module is created as a part of the J-PET Analysis Framework. As an input it uses structures created by the Geant4 software with generated information about gamma quanta interactions. In further steps of the analysis chain, those hits are processed in the same manner as experimental data. A scheme of the data flow is shown in Figure 1.1.

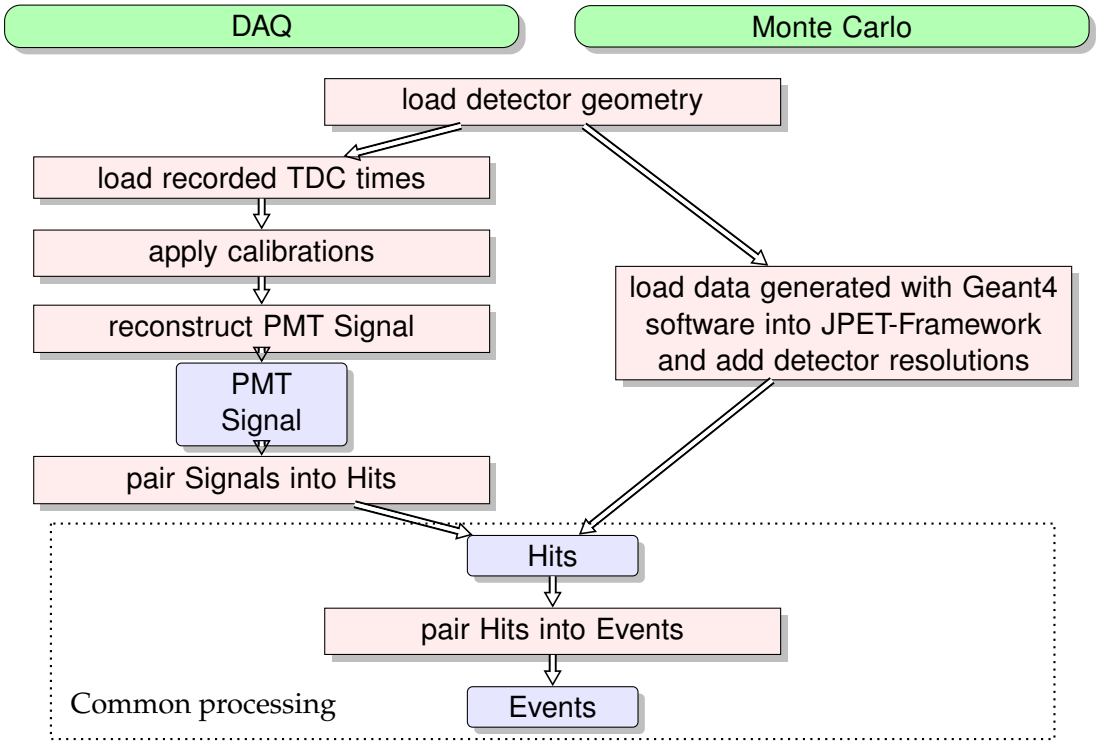

Figure 1.1: Scheme of data processing by the J-PET Framework software with implemented module dedicated for Monte Carlo simulation processing. Left and right columns correspond to the input taken from DAQ or Monte Carlo simulation, respectively. Violet rectangles represent reconstructed physical information obtained from dedicated analysis modules (pink rectangles). Reconstructed hits, both from DAQ system or Monte Carlo simulation, pass in the further steps through the same reconstruction algorithms.

2. Physics and geometry reference manual

This chapter provides description of detector geometry and explanations of the physics implemented in the simulation.

2.1. Detector geometry

Detector geometry consists of following components: layers of the scintillators, detector frame and sources (annihilation chambers with source or bare radioactive sources). Example is given in Figure 2.1.

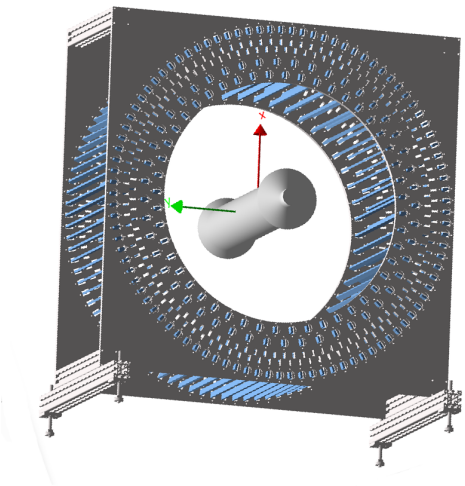

Figure 2.1: Visualization of the simulated detector. The detection strips (blue) are organized into three cylindrical layers and mounted between two detector frames (grey). In the center of the detector is visible an annihilation chamber.

2.1.1 Layers of the scintillators

Simulations can be preformed for the geometry that corresponds to the full scale prototype which consists of three layers of EJ-230 plastic scintillator strips with dimensions of $7 \times 19 \times 500$ mm^3. The innermost and middle layers consist of 48 strips each, while the outermost layer - 96 strips. In program, as in reality, each scintillator strip is covered by kapton foil.

Simulations allows also to work with a new modular solution (see Figure 2.2). A new detection layer consists of 24 modules, each of them contains of 13 plastic scintillator strips.

2.1.2 Detector frame

Detector frame is loaded from technical drawings in a form of a CAD file. This solution allows to keep detailed description of the frame shape during the simulations.

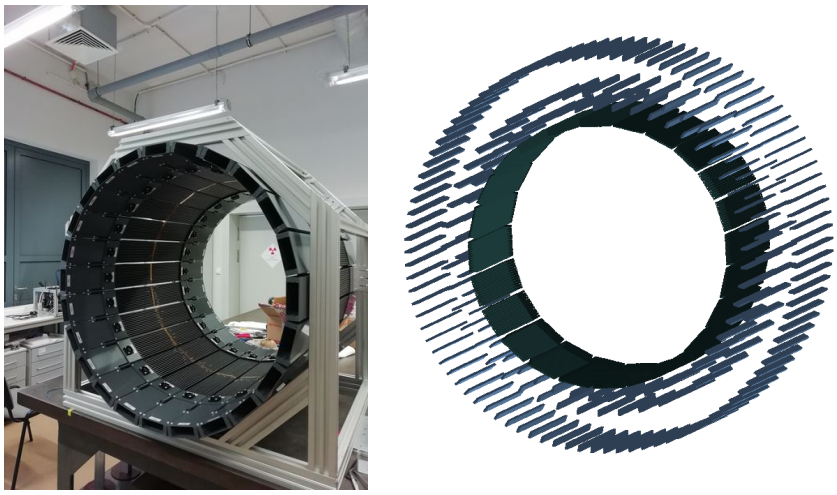

Figure 2.2: Left: photo of a modular layer. Right: corresponding layer included in J-PET-Geant4 simulation. For clarity purposes detector frames are not shown.

2.1.3 Annihilation chambers

Until now, during the measurements with the J-PET detector, the two types of annihilation chambers were used, main difference being in the distribution of the porous material. In such kind of the material o-Ps$\rightarrow 3\gamma$ bound states are forming more frequently than in the other parts of the detector and where the annihilations take place. In the case of the small chamber, the radioactive source is covered with the porous material (see Figure 2.3, left panel), while in the case of the large chamber the inner walls are coated with it (Figure 2.3, right panel).

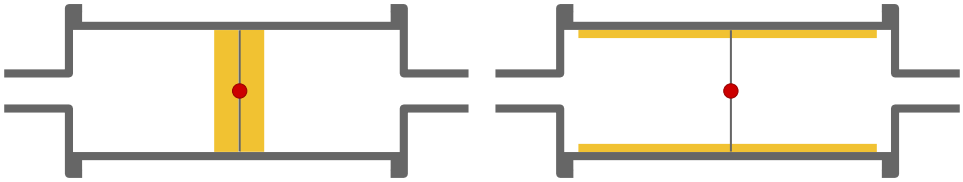

Figure 2.3: Left: Scheme of small chamber where the porous material (yellow band) is located around the radioactive source (red dot). Right: Scheme of large chamber, its walls are coated on the inner side with a porous material. Chambers dimensions are not to scale.

During the simulations of the decays of o-Ps$\rightarrow 3\gamma$ particles it is only necessary to generate the products of the decay - three photons, which are dubbed as the primary particles in this reaction. The position of the decay - vertex - is predefined and depends on the geometry and the type of the material.

Small chamber

In case of small chamber the porous material surrounds the radioactive source and in first approach all annihilations will take place within the effective radius (see Figure 2.6). This parameter can be set in the user macro - see section 5.6.4).

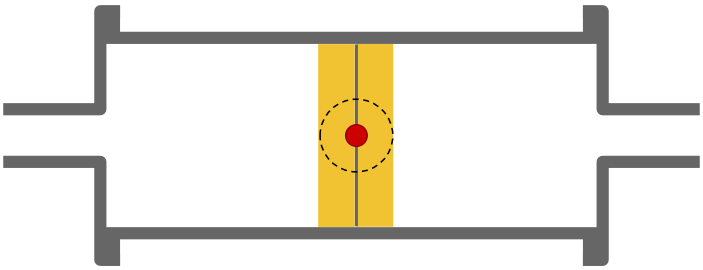

Figure 2.4: Scheme of small chamber (compare with Figure 2.3, left). The dashed circle represents the radius of sphere up to which annihilations took place.

Large chamber

Events in the large chamber are simulated mostly on the walls of the chamber, assuming the isotropic emission from the radioactive source.

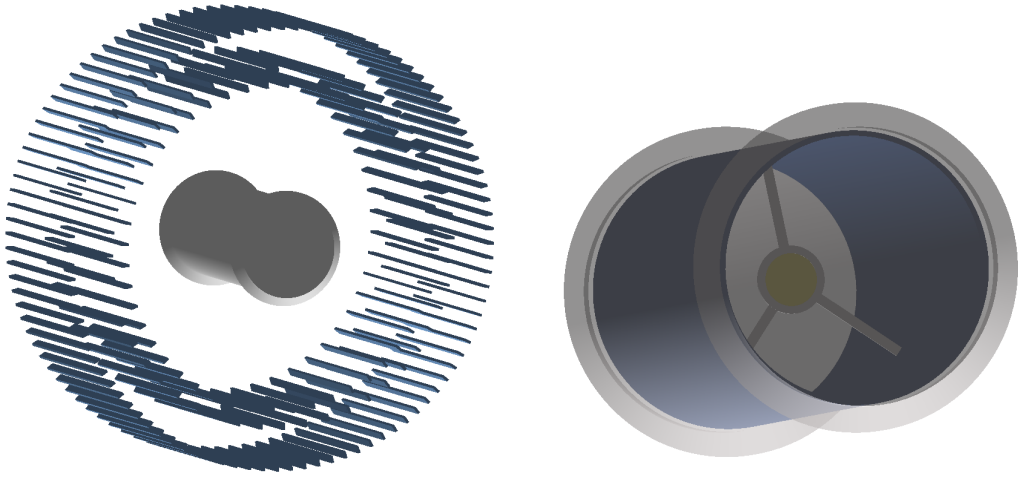

Figure 2.5: Left: Visualization of the larger chamber located at the center of the detector. Right: Detailed view of chamber's inside. The radioactive source is located between two kapton foils (yellow). Structure is stabilized by the holder. Blue cylinder represents the porous material.

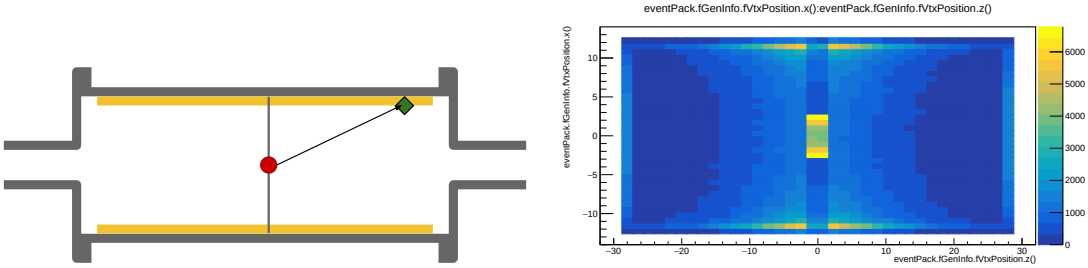

Figure 2.6: Left: Scheme of big chamber (compare with Figure 2.3, right). The place of the annihilations place is denoted by green rhombus. Events are simulated assuming isotropic emission from the center of the chamber. Right: distribution of the vertices in $x - z$ plane obtained from simulation. The major part of the simulated events originate in the source and on the inner walls of the chamber, while less number of decays originate in the part of the chamber shadowed by the holder of the source.

2.2. Physics

2.2.1 Events types

For predefined geometries user can expect following kinds of events:

- prompt gamma quanta
- 2γ originating from direct annihilation or p-Ps state
- 3γ originating from direct annihilation (fraction $1/372$)
- 3γ originating from o-Ps annihilation (lifetime depends on the material)
- 2γ from pick-off process (see Figure 2.7)

Each fraction of events can be adjusted during execution of the simulation. For details see 5.6.2.

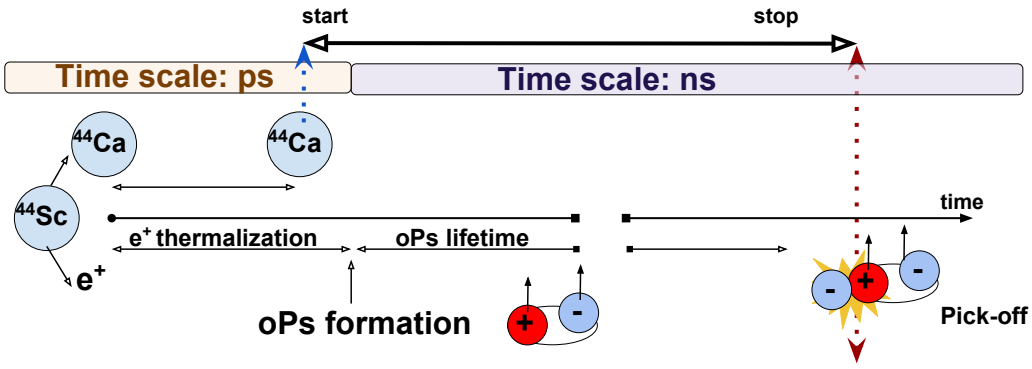

Figure 2.7: Scheme of the time sequence used in simulations. The radioactive decay denotes starts of the time for specific event. The prompt photon is simulated according to the exponential decay (order of ps). Parallelly, the positron in matter undergoes either the direct annihilation into 2γ or forms a bound state.

Back-to-back events

Due to spin and parity conservation back-to-back events annihilate into two entangled photons:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|RR\rangle - |LL\rangle) \tag{2.1}$$

where R and L stands for right and left handed circular polarization. We can rewrite this in linear polarization basis:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|HV\rangle + |VH\rangle) \tag{2.2}$$

which shows that linear polarizations vectors should be orthogonal.

In our simulations we are only able to simulate separable state, that is: $|HV\rangle$. We are generating the momentum direction of both photons. Polarization of first one is set to be a random vector orthogonal to the momentum direction. Polarization of the second photon is a polarization of first rotated 90° around momentum direction of first photon. In that way we made sure that polarizations are orthogonal to each other and to both momenta vectors.

Ortho-positronium annihilation

Positronium is the lightest purely leptonic system consisting of an electron and positron. Its triplet state - ortho-positronium annihilates into $(2n+1)$ gamma quanta. The most common decay of o-Ps$\rightarrow$ 3γ into three photons is implemented in the J-PET-Geant4, as decays of this particle are not included in Geant4 library by default. These gamma quanta are co-planar in the Center of Mass (CM) frame due to momentum conservation and are generated according to the double differential cross section expressed as a function of photons' energies E_1 and E_2:

$$\frac{d^2\sigma}{dE_1 dE_2} = \frac{1}{6}\frac{8e^6}{v m_e^2}\left[\left(\frac{m_e - E_3}{E_1 E_2}\right)^2 + \left(\frac{m_e - E_2}{E_1 E_3}\right)^2 + \left(\frac{m_e - E_1}{E_2 E_3}\right)^2\right], \tag{2.3}$$

where e and m_e are electron charge and mass, respectively, v denotes electron-positron relative velocity, E_i (i=1,2,3) are gamma photon energies, and $E_1 + E_2 + E_3 = 2m_e$ follows from energy conservation. In above formula, the conservation of four-momentum results in the characteristic energy distribution of gamma quanta (see Figures 2.8).

Please notice that no specific polarization is assumed.

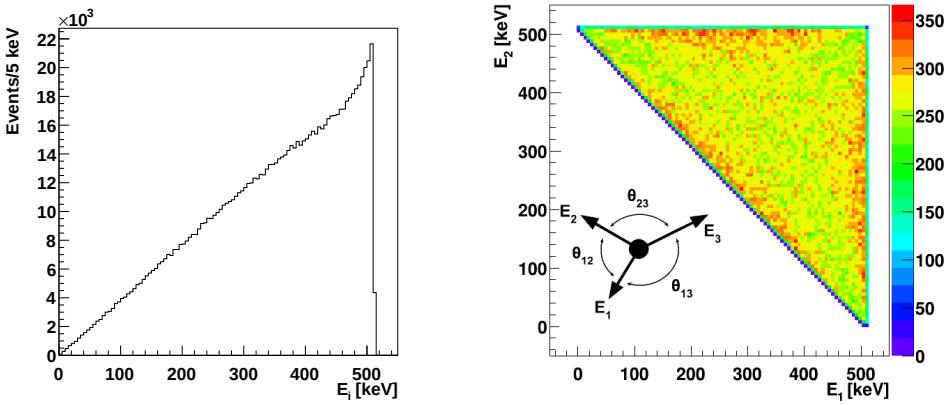

Figure 2.8: Left: Energy spectrum of photons originating from three-photon annihilation of an electron and a positron. Right: Dalitz plot for the o-Ps$\rightarrow 3\gamma$ annihilation. In both figures the non-homogeneity of the density distribution is due to the energy dependence of the o-Ps$\rightarrow 3\gamma$ transition amplitude (see Eq. 2.3) which was taken into account in simulations according to the predictions based on quantum electrodynamics.

2.2.2 Physical processes

Used physics package in Geant4 is LivermorePolarizedPhysics. For details visit Geant4 physics website. Physics process included for different particles (status for GEANT 10.3):

- gamma
  - PhotoElectricEffect
  - LivermorePolarizedPhotoElectricModel
  - ComptonScattering
  - LivermorePolarizedComptonModel
  - GammaConversion
  - LivermorePolarizedGammaConversionModel
  - RayleighScattering
  - LivermorePolarizedRayleighModel
- e+-
  - eMultipleScattering
  - UniversalFluctuation
  - eIonisation
  - LivermoreIonisationModel
  - eBremsstrahlung
  - LivermoreBremsstrahlungModel
- e+
  - eplusAnnihilation

- mu+-
  - MuMultipleScattering
  - MuIonisation
  - MuBremsstrahlung
  - MuPairProduction
  - MuBremsstrahlungModel
  - MuPairProductionModel
  - hBremsstrahlungModel
  - hPairProductionModel
- hadrons
  - hMultipleScattering
  - MscStepLimitType
  - hBremsstrahlung
  - hPairProduction
  - hIonisation
  - ionIonisation
  - alphaIonisation
  - IonParametrisedLossModel
  - NuclearStopping
- msc models
  - UrbanMscModel
  - WentzelVIModel
  - GoudsmitSaundersonMscModel
  - CoulombScattering
  - eCoulombScatteringModel

2.2.3 How hits are created?

The schematic view of Compton scattering is shown in Figure 2.9. The hit position and time registered by photomultipliers consists of many small contributions. Therefore final position and time are weighted sums of all contributions with energy as weight.

2.3. What can be simulated?

In general, user can modify the source code and incorporate in a easy way any type of needed geometry. Till now a few predefined setups were created:

- J-PET runs (contains Ps annihilation in the chamber)
- beams of photons
- cylindrical isotopes
- NEMA sources

Details are given in section 5.5.

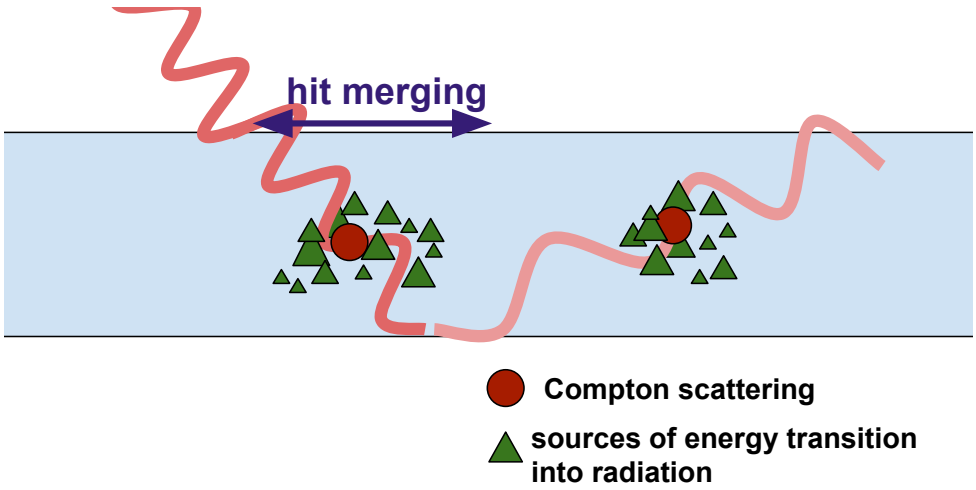

Figure 2.9: Gamma quanta enters the scintillator (blue) and undergoes through Compton scattering (red dot). Absorbed energy in scintillator is emitted via fluorescence through many intermediate processes (green triangles). Light registered by the photomultipliers is assigned to the single energy deposition if its components are close in time. Therefore, in simulation the hit merging time is introduced as a tunable parameter (default value: 5 ns). This also mean, that two energy depositions, ex. from multiple scatterings in single scintillator, may be merged into single entry.

3. User's guide: results of the simulation

3.1. Structure of the generated information

In MC files events are kept in structure called JPetGeantEventPack, which stores following classes:

- JPetGeantEventInformation - information about primary photons,
- JPetGeantScinHits - hits registered in scintillators,
- JPetGeantDecayTree - (not implemented yet) full chain of tracks and vertices

In principle, the JPetGeantEventInformation allows to obtain full information about initial state. Summary is given in Table 3.1.

Table 3.1: Functions and their description available in JPetGeantEventInformation

Function	Return type	Return description
GetVtxPosition()	TVector3	position of the $2\gamma/3\gamma$ vertex
GetVtxPromptPosition()	TVector3	position of the prompt gamma quanta
GetPromptGammaGen()	bool	check if prompt gamma is generated
GetTwoGammaGen()	bool	check if 2γ event is generated
GetThreeGammaGen()	bool	check if 3γ event is generated
GetLifetime()	double	annihilation time for $2\gamma/3\gamma$ event
GetPromptLifetime()	double	emission time for prompt gamma
GetMomentumGamma(index) index=0: prompt γ index=(1-3): $2/3\gamma$	TVector3	momentum vector of gamma quanta

3.2. Event types registered in scintillator

Generated gamma quanta have associated value called multiplicity (in structures its named: fGenGammaMultiplicity). User can access it in JPetGeantScinHits. Accessing it from the reconstructed hit (JPetHit) requires first identify the generated event (getMCindex()) and then checking its multiplicity flag (GetGenGammaMultiplicity()). The numbering scheme given during photons generation is given in Table 3.2.

This variable allows also to track further behaviour of the gamma quanta:

- 0 - Rayleigh scatterings
- +10 - number added after each scattering in the non-sensitive part of the detector (chamber, phantom, construction, foil)
- +100 - number added for each scattering in the sensitive part of the detector (scintillators)

Table 3.2: Numbering scheme for primary gamma quanta (fGenGammaMultiplicity).

Number	Gamma quanta origin
1	prompt photon
2	photon from 2γ annihilation
3	photon from 3γ annihilation

- $\times 10$ - if there was a secondary particle created during simulations. Particle multiplicity is then equal to the photon multiplicity from which this particle was created multiplied by 10.

Figure 3.1 shows example for prompt photon (initial multiplicity = 1) with sequential scattering in scintillators.

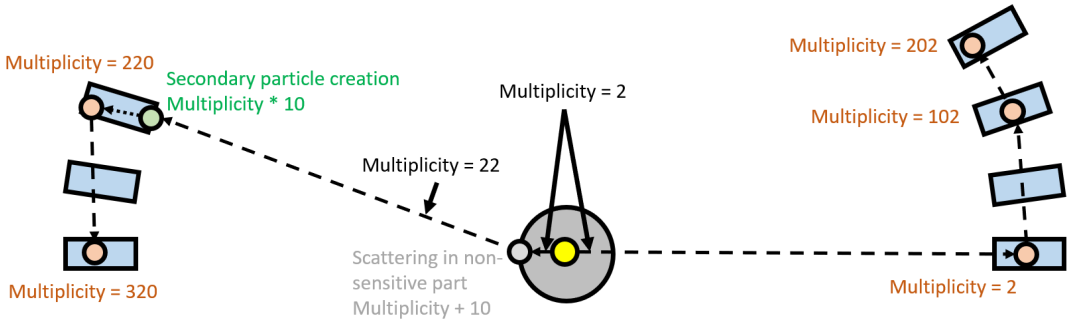

Figure 3.1: Example of the back-to-back gamma quanta emission (black dashed arrows, from the source (yellow circle)). One of the quantum (left) scatters in the chamber (grey circle, non-sensitive part of the detector) resulting in the increase of the multiplicity of the photon by 10. Next, photon produces a secondary particle (green circle), which gets a multiplicity of the previous photon multiplied by 10 (220). Secondary particle after some time emits a photon, which inherits its multiplicity from the secondary particle. Created photon scatters in the other scintillator. Left path will produce hit with multiplicity (320). The second primary photon (multiplicity 2) scatters two times in different scintilators, in each act increasing its multiplicity by 100, resulting in creation of hits with multiplicity 2, 102 and 202.

User can also access all basics information about incident gamma quanta. Summary is given in Table 3.3.

3.3. DecayTree

In this version the first implementation of the Decay Tree structure is introduced. Decay Tree will be storing information about whole process of the generation different hits and to extract more informations at the level of the analysis. Whole structure probably will be available in the new version.

3.4. Control histograms

In `mcGeant.root` files, user will find control histograms containing the information about types of generated events (gamma multiplicity), annihilation position, deposited energy etc.

Table 3.3: Summary of functions available in JPetGeantScinHits class.

Function	Return type	Return description
GetHitPosition()	TVector3	coordinated of hit in the scintillator
GetPolarizationIn()	TVector3	polarization of incident γ
GetPolarizationOut()	TVector3	polarization of scattered γ
GetMomentumIn()	TVector3	momentum of incident γ
GetMomentumOut()	TVector3	momentum of scattered γ
GetGenGammaMultiplicity()	int	multiplicity flag
GetEvtID()	int	sequential number of generated event
GetScinID()	int	number of scintillator
GetTrackPDG()	int	particles encoding number from PDG
GetEneDepos()	float	value of deposited energy
GetTime()	float	interaction time

3.5. Files available at servers

At the J-PET servers there will be a dedicated common space for some simulated data that could be used for some tests. User can access them from their private accounts: /data/4/users/mc

4. Using the simulation output with J-PET Framework

4.1. How to run? and what to expect

For general information please read section 1, which contains general idea of data flow. Simulation code can be downloaded from official repository (see section 5) and installed in user system. This path is recommended only for software developers. For users are available generated MC files located at J-PET servers. Current file location is given at dedicated Wiki page.

Files from simulation contain only a generated information about simulated events (so called 'true' Monte Carlo). The detector resolution are applied while processing through the J-PET Framework module called `JPetGeantParser` (see next section). Therefore, in order to use the Monte Carlo simulation, the user should be familiarized with the J-PET Framework.

Files are processed automatically based on file extension. Example of usage:

```
./MyAnalysis.x -t mcGeant -f name.mcGeant.root  -u userParams.json
                                                -l setup.json -i runNr
```

4.2. JPetGeantParser in Framework

JPetGeantParser is a module taking generated MC files (`name.mcGeant.root`), applying detector response and filling the JPetHit structure with reconstructed MC information.

In simulation each radioactive decay and further processes, e.g. 3γ annihilation with prompt photon, is treated as a separate event. However, the acquisition of the experimental data in the detector is performed in a continuous manner, digital data is written in parametrized intervals, which are called time window - they can contain registered data from many physical events. To mimic this acquisition scheme, generated Monte Carlo events are adjusted in time window according to the Poisson distribution, depending on source activity and length of the time window.

During filling the `JPetHit` structures, each registered event have applied smearing functions (see section 4.4) and requirements for registration (now only energy threshold). Please notice, that not all of generated events will have reconstructed equivalent. In `JPetHit` class function `getMCindex()` allows to obtain reference to the generated MC hit.

4.3. Timescale in JPetMC and Framework

In Geant4 simulation the start of the time is given by β decay. Events have to be adjusted to the time window scheme given by Data Acquisition System (DAQ). Please notice that time scheme in Geant4 is in a range $[0, \infty)$, while in Framework times are measured in range $[-T_{window\ length}, 0]$. Scheme is shown in Figure 4.1.

4.4. How to change detector resolution in simulation?

In Framework v8 the following smearing functions are applied:

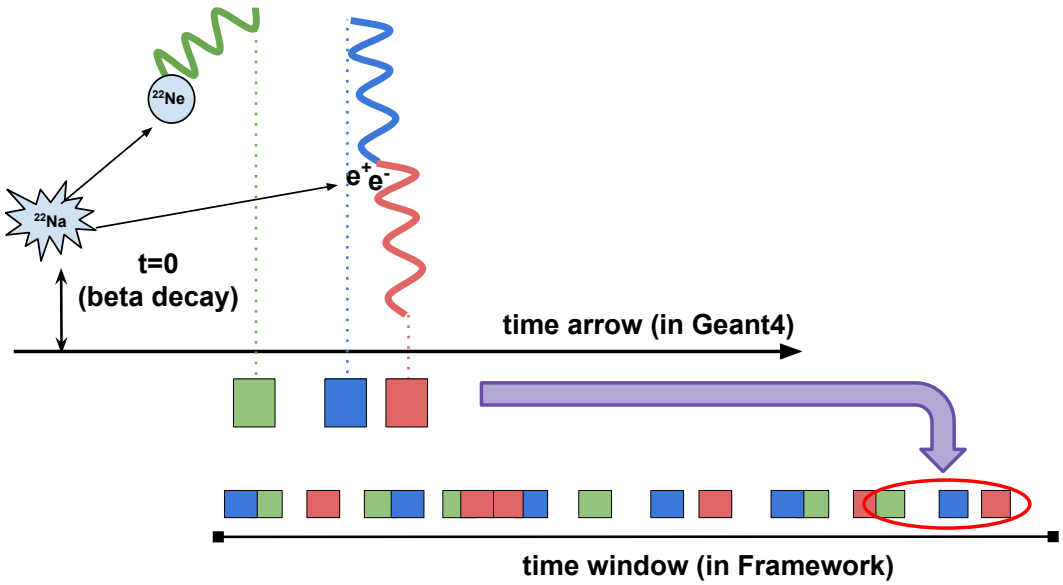

Figure 4.1: The β decay of the radioactive isotope gives the start time in Geant4 simulations. Then the arrival times of the gamma quanta in the detector (green-blue-red squares) are stored. Each event in simulation consists of single β^+ decay. During processing of the MC files events have to be arranged in the DAQ structure called 'time window'. Therefore, generated events are adjusted into time window structure (see bottom part of the figure) with requirement that β^+ decay have to reproduce the Poisson distribution.

- 'z' position of the hit:
  $\sigma(z) = G(z[cm], 0.976\ cm)$
- hit-time:
  $E_{hit} > 200$ kev: $\sigma(t_{hit}) = G(t_{hit}, 80\ ps)$
  $E_{hit} \leq 200$ kev: $\sigma(t_{hit}) = G(t_{hit}, 80\ ps)/\sqrt{E_{dep}[keV]/270}$
- hit-energy:
  $\sigma(E)/E = 0.44/\sqrt{E[MeV]}$

where $G(c, \sigma)$ is normalized Gaussian function centered at c and with variance σ^2.

Gathered till now indications suggest that more tuning will be needed. Therefore, during the reconstruction user can apply his own smearing function. It will require creating a new class that will contain smearing functions with predefined scheme:

```
class NewSmearingFunctions
{
  public:
    double newHitZSmearing(double *x, double *p);
    double newHitEnergySmearing(double *x, double *p);
    double newHitTimeSmearing(double *x, double *p);
};
```

where x parameter should not be overwritten, and p contains following values:
Therefore, z smearing function can be expressed following:

p[0]	scintillator ID
p[1]	z value of hit
p[2]	E_{dep} deposited energy
p[3]	(only for time smearing) interaction time

```
double newHitZSmearing(double *x, double *p){
 double zIn = p[1];
 double sigma = 0.976;
 return exp(-0.5*pow((x[0]-zIn)/sigma,2))/(sqrt(2*M_PI)*sigma);
}
```

Please notice that user should provide the distribution. Taking the random value for specific event is provided by the JPetHitSmearingFunctions in Framework. Afterwords, function should be implemented in user program:

```
NewSmearingFunctions* sf = new NewSmearingFunctions();
TF1* fun = new TF1("newZSmearing",sf,
      &NewSmearingFunctions::newHitZSmearing,
      -200., 200.,3,"NewSmearingFunctions","newHitZSmearing");
JPetSmearingFunctions::getSmearingFunctions().setFunZHitSmearing(fun);
```

Analogous functions can be created and registered for energy (setFunEnergySmearing) and time (setFunTimeHitSmearing) smearing functions.

5. Developer's guide: Installation and usage

5.1. Source code

Monte Carlo software is maintained within public repositories on GitHub:

- https://github.com/JPETTomography/J-PET-geant4

Easiest way to obtain Monte Carlo code, is with Git version control system. If you are not acquainted with Git, there are many useful tutorials for beginner users of version control systems, i.e. http://rogerdudler.github.io/git-guide/. To obtain source code just type in terminal the command:
`git clone https://github.com/JPETTomography/J-PET-geant4.git`

5.2. Requirements

J-PET Monte Carlo can be installed at your own computer or at J-PET server. For installation one need:

- Linux operating system - tested and used with versions from 14.04 to 18.10.
- g++ compiler - version supporting C++11 standard
- ROOT Data Analysis Framework - version at least 6.0 (https://root.cern.ch/)
- Geant4 - simulation toolkit in version at least 4.10.04 (https://geant4.web.cern.ch/)
- CADMesh - External library which allows to import CAD models into Geant4. Source code can be found at https://github.com/christopherpoole/CADMesh

Installation instructions of the required software can be found under the given links and won't be covered in this manual.

5.3. Compilation procedure

To compile Monte Carlo code go to the directory with the source code. Then type following commands:

```
mkdir build
cd build
cmake .. -DCMAKE_PREFIX_PATH=[pathToCADMESHFile].cmake
make
```

After successful compilation in build directory there is new directory: bin inside which you can find executable to run simulations: jpet_mc

5.4. How to run?

In `build/bin` folder user can find executable `jpet_mc`. It can be run in visual mode (type in terminal: `./jpet_mc`) or call one of the macro (`./jpet_mc macro.mac`). Examples are summarized in section 5.5.

5.5. Examples

Each example is prepared in a macro form and can be executed from the command line:

```
./jpet_mc macro.mac
```

5.5.1 Ps decays in the chamber

For selected J-PET run N a whole setup, scintillator configuration and annihilation chamber, can be called by single command:

```
/jpetmc/detector/loadJPetBasicGeom
/jpetmc/detector/loadTargetForRun N
```

Examples are given in macros: *run5.mac*, *run6.mac*, *run7.mac*.

5.5.2 Beam of the photons

In order to simulate beam of photons user needs to use following command:

```
/jpetmc/source/setType beam
```

Such a source have following properties that can be set:

- `/jpetmc/source/gammaBeam/setEnergy` **`X unit`**
  This command sets energy of a photons in a beam. Default value: **511 keV**
- `/jpetmc/source/gammaBeam/setPosition` **`X Y Z unit`**
  Sets vertex position of the gamma quanta beam. Default value: **0 0 0 cm**
- `/jpetmc/source/gammaBeam/setMomentum` **`X Y Z unit`**
  Sets momentum direction of the gamma quanta beam (length is not important since we defined energy of photons using `setEnergy` command). Default value: **0 0 0 keV**

Example is given in macro *singleBeam.mac*.

5.5.3 Cylindrical isotope

An extended cylindrical source can be defined in following way:

```
/jpetmc/source/setType isotope
```

Such a source have following properties that can be set:

- `/jpetmc/source/isotope/setShape` **`cylinder`**
  Currently only cylinder option is supported
- `/jpetmc/source/isotope/setNGamma` **`X`**
  Give number of gamma quanta to generate 1 / 2 / 3 photons

- `/jpetmc/source/isotope/setShape/cylinderRadius` **X units**
  Set radius of the cylinder. Default value: **10 cm**
- `/jpetmc/source/isotope/setShape/cylinderZ` **X units**
  Set half of Z dimention of a cylinder. Defaault value: **10 cm**
- `/jpetmc/source/isotope/setPosition` **X Y Z units**
  Set position of the center of source. Default value: **0 0 0 cm**

Example is given in macro *extendedSource.mac*.

5.5.4 NEMA sources

In the plane comprising the central detector axis the 0.1-mm cylindrical ^{22}Na sources are simulated. Their positions were chosen according to the NEMA NU 2-2012 norm and coordinates are adjusted to the laboratory measurement preformed in laboratory. Simulation is executed by command:

```
/jpetmc/source/setType nema
/jpetmc/source/nema P
```

where *P* denotes source position (1-6). Translation between position and coordinates is given in Table 5.1.
Example is given in macro *nema.mac*.

Table 5.1: Coordinates of simulated "point-like" sources positioned according to the NEMA norm.

Position	Coordinates [cm]
1	(1, 0, 0)
2	(10, 0, 0)
3	(20, 0, 0)
4	(1, 0, -18.75)
5	(10, 0, -18.75)
6	(20, 0, -18.75)

5.6. Interaction with the simulation

5.6.1 Detector construction messenger

- `/jpetmc/detector/loadOnlyScintillators`
  Using this option will load only scintillators
- `/jpetmc/detector/loadJPetBasicGeom`
  This option will load scintillators with frames
- `/jpetmc/detector/loadModularLayer` **X**
- `/jpetmc/detector/hitMergingTime` **X**
  Use this option if you want to change default time window for merging hits substituting desired value in place of *X*. If there was more than one hit in single scintillator in an event, and their hit times difference is lower than specified time window they are merged to a single hit. Hit merging was explained in: section 2.2.3 If you don't want to merge hits, set this to **0.0**. Default value: **5 ns**.

5.6.2 Material Extention messenger

- `/jpetmc/material/threeGammaOnly`
  Only 3 gamma events will be generated
- `/jpetmc/material/twoGammaOnly`
  Only 2 gamma events will be generated
- `/jpetmc/material/pickOffOnly`
  Only 2 gamma events from pick-off/conversion will be generated (lifetime as for 3g)
- `/jpetmc/material/oPslifetime` **`Lifetime[ns] Probability[%]`**
  Adds lifetime component for oPs. Exemplary value: **10 5**
- `/jpetmc/material/pPslifetime` **`Lifetime[ns] Fraction[0-1]`**
  Adds lifetime component for pPs. Exemplary value: **0.125 0.25**
- `/jpetmc/material/directComponent` **`Lifetime[ns] Probability[%]`**
  Adds lifetime component for direct annihilation. Exemplary value: **0.5 50**
- `/jpetmc/material/reloadMaterials` **`Material`**
  Reloads the newly set lifetime components. Material that can be reloaded: xad4, kapton, aluminium, plexiglass, pa6

5.6.3 Event messenger

- `/jpetmc/event/saveEvtsDetAcc`
  This option allows to save generated events based on its multiplicity. Options below allow to modify standard behaviour (save multiplicity 0,2-10).
- `/jpetmc/event/minRegMulti`
  change lower value of saved multiplicity (0)
- `/jpetmc/event/maxRegMulti`
  change upper value of saved multiplicity (10)
- `/jpetmc/event/excludedMulti`
  change excluded value of multiplicity (1)
- `/jpetmc/event/printEvtStat` *X*
  Set *X* to true to show how many events were generated. Default value: **false**
- `/jpetmc/event/printEvtFactor` *X*
  Set *X* to natural number to print number of generated events every 10^X events. Default value: **10**
- `/jpetmc/event/ShowProgress` *X*
  Set *X* to true to show how many events were generated (in %). Default value: **false**

5.6.4 Primary Generator Action messenger for run parameters

- `/jpetmc/run/setChamberCenter` **`X Y Z unit`**
  This command sets the position of center of the annihilation chamber. Default value is set to **0 0 0 cm**
- `/jpetmc/run/setEffectivePositronRange` **`X unit`**
  This command sets effective range of positrons to a desired value. Default value: **0.5 cm**

6. Documentation & support

6.1. Useful links

- J-PET Experiment homepage
  http://koza.if.uj.edu.pl/pet/
- PetWiki MC meetings
  http://koza.if.uj.edu.pl/petwiki/index.php/MC_2019
- Redmine - bug tracking, task management and discussion forum for Framework developers and users
  http://sphinx.if.uj.edu.pl/redmine/projects/gate-geant-mc-simulations/issues

6.2. Doxygen

Documentation can be created via Doxygen software. In build folder type in terminal:

```
make doc
```

Output files can be viewed in your web browser: *f*irefox doc/html/index.html

6.3. Posting bugs and getting support

Please report discovered bugs or required features on Redmine forum: http://sphinx.if.uj.edu.pl/redmine/projects/gate-geant-mc-simulations/issues. Include as much detail as possible. At minimum: software version (JPetMC, Geant and Root) and any error messages. Is also helps to list the sequence of commands so we can try to reproduce the issue.

6.4. Contacts

In case of problems with simulations (installation, running, etc.) one can contact following experts:

- **Kamil Dulski** (kamil.dulski@gmail.com) main developer of J-PET Monte Carlo simulations project
- **Krzysztof Kacprzak** (k.kacprzak@alumni.uj.edu.pl) developer of the J-PET Monte Carlo simulations project
- **Sushil Sharma** (sushil.sharma.uj@googlemail.com) preparation of CAD files
- **Nikodem Krawczyk** (nikodem.krawczyk@gmail.com)
- **Juhi Raj** (juhi.raj@doctoral.uj.edu.pl) MC-Data verification: run 4/5

- **Aleksander Gajos** (aleksander.gajos@uj.edu.pl) J-PET Framework development
- **Wojciech Krzemień** (wojciech.krzemien@if.uj.edu.pl) J-PET Framework development
- **Eryk Czerwiński** (eryk.czerwinski@uj.edu.pl) administration of servers and data access/ management
- **Daria Kisielewska** (dk.dariakisielewska@gmail.com) - former main developer of J-PET Monte Carlo simulations project