DETECT2000: An Improved Monte-Carlo Simulator for the Computer Aided Design of Photon Sensing Devices

François Cayouette[1,2], Denis Laurendeau[3] and Christian Moisan[3]
[1]Montreal Neurological Institute, McGill University, Montreal, Canada
[2]Biomedical Engineering Department, McGill University, Montreal, Canada
[3]Electrical and Computer Engineering Department, Laval University, Quebec City, Canada

ABSTRACT

We introduce a new version of DETECT. DETECT is a Monte-Carlo simulator developed for the Computer Aided Design (CAD) of optical photon sensing devices. The simulator generates individual emission photons in specified locations of a photon-emitting device and tracks their passage and interactions in active and passive components of the system. Extensive options are available in the simulator to model the geometry of the photon sensing device, to account for the time and wavelength distribution of emission photons, to track their interactions with surfaces, to account for their possible absorption and re-emission by a wave-shifting components and to model their detection by pixelated photomultipliers or photodiodes. DETECT2000 is a very significant upgrade of DETECT97, which has long been established in the nuclear medicine instrumentation community for its accuracy to model the performances of high resolution energy and position sensitive gamma-ray detectors. The 2000 version of DETECT offers an accelerated version of the simulator which has been redesigned in the object-oriented C++ language. New features such as the tracking of the time and wavelength history of individual optical photons have been added.

Keywords: DETECT, Monte-Carlo, light photon transport, Computer Aided Design, Photon Sensing Devices

1. INTRODUCTION

Testing the theoretical performance of a photon sensing device before it is actually constructed can be of great help. With the appropriate Monte-Carlo simulator, it would be possible to study the light photon transport inside the device and give important information that could be used to improve the efficiency of the device. DETECT2000 is a Monte-Carlo simulation model of the light photon behaviour inside an optical device. DETECT2000's general geometric syntax can create any complex by joining together four basic geometrical forms: planes, cones, cylinders and spheres. Also, DETECT2000 comes with a variety of possible surface finishes. Some of the finishes are idealized surface model that can be used to estimate the best and worst possible performance of a device, as it was done for a scintillation crystal used in positron emission tomography (PET)[1]. Also, there is one surface finish type that can be customized to create any kind of surface roughness with any kind of possible reflection. This allows for very accurate simulations.

DETECT2000 is an important upgrade from DETECT97[2], one of the successors of DETECT[3]. DETECT2000 has been entirely rewritten from the C language to the object oriented C++ language in order to simplify code maintenance and integration of further simulation features. Also, the rewriting was aimed at improving the portability of the source code therefore insuring an easier distribution. The source code of the second release of DETECT2000 is now available for download on the web since November 2001.

2. DETECT2000 SOFTWARE DESIGN

The Unified Modeling Language (UML) class diagram of the new architecture is shown in figure 1. The empty arrow ended links define an inheritance relation, an "is a" relation, between two class. The links starting with a diamond define an aggregation relationship, a "have a" relation, between the two classes. The differences between the full and empty diamonds are subtle and are beyond the scope of this paper. Some classes have been defined as abstract classes in order to define the general behaviour of the sub-classes that inherits from them. One such class is the Geometric class that

defines the minimal behaviours for geometric objects which will compose the different components of the model. In a normal UML diagram, the name of the variables and the name of functions are written inside the class box. Doing so would make the diagram unreadable and do not give much additional information about the software design of DETECT2000. DETECT2000 has been coded in such a way that it should be relatively easy to add new features to the existing code. These features can be added either as new classes in the model or as new functions in the current classes.

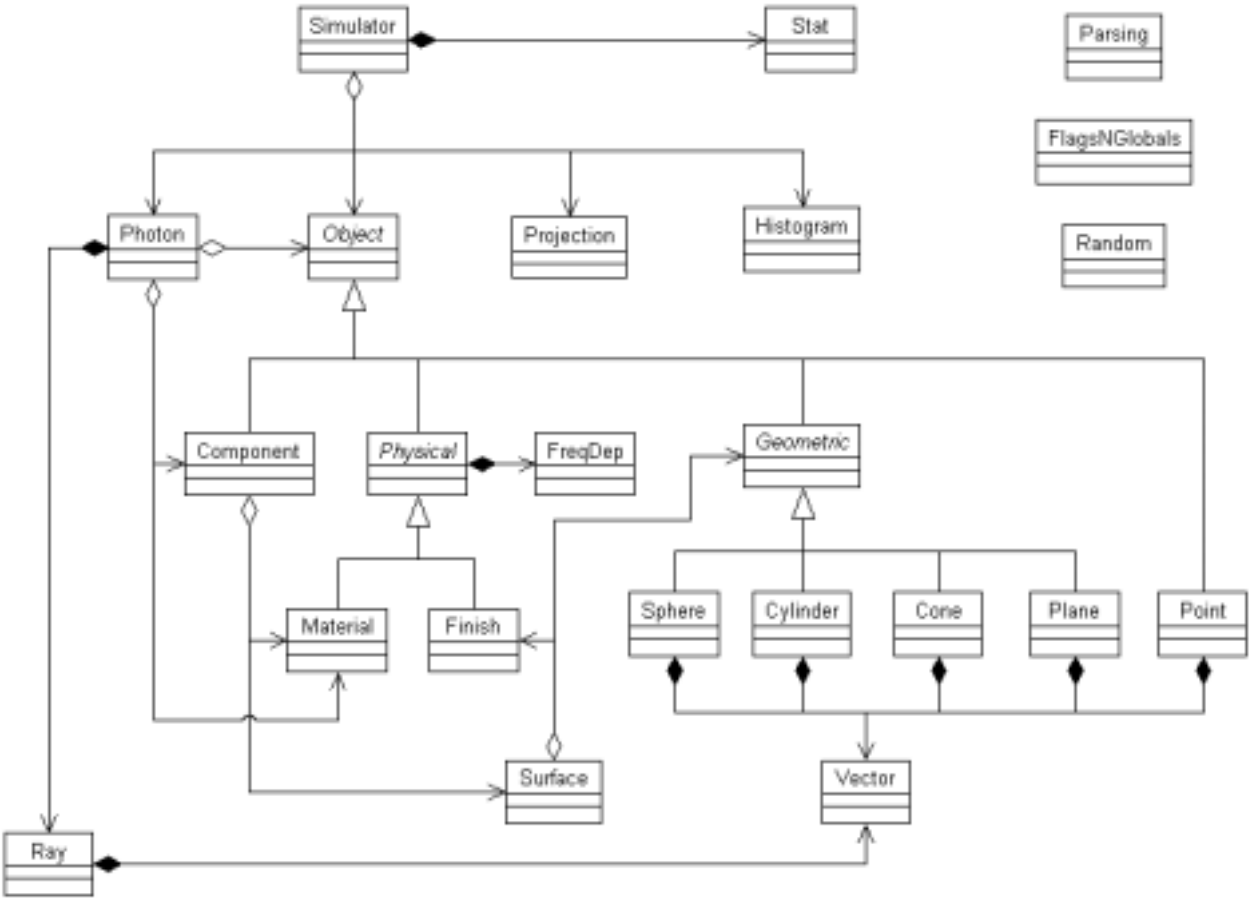

Fig. 1 UML class diagram of DETECT2000

1. New architecture

Some of the classes in DETECT2000 have been coded according to specific design patterns[4]. The design patterns provide standard solutions for commonly arising problems in software engineering. The first design pattern used is the Singleton that had been implemented in the FlagsNGlobals class. The values that are stored in this class are used in many parts of the program. By storing the values in the class, we are reducing the namespace pollution and this will reduce the probabilities of problems if another program tries to interface with DETECT2000's code. The second design pattern used in DETECT2000 is the facade in class Simulator. By doing so, the Simulator class is in charge of the entire simulation process and creates an easy interface for any code to start DETECT simulations.

During the rewriting process, special care was taken in order to have maximal portability. DETECT2000 has been tested under various operating systems (OS) such as: Sun OS, Windows (9x, 2000, ME and XP) and Red Hat Linux. Other

Unix-like OS are very likely to be able to run DETECT2000 without any change to the source code. Also, under Windows, DETECT2000 can be compiled and executed using Cygwin, a text based Linux emulator.

3. IMPROVEMENTS UPON EARLIER RELEASES

1. Streamlined code

While the code was translated from C to C++, special attention was given to the algorithms used. When it was possible, the algorithm implementation was modified to have it run faster without changing the overall behaviour. Also, some redundant checks that were done in both the simulation and the model checking phase were removed. Even if the C++ language is known to be slower than the C language, the DETECT2000 implementation is typically 10-15% faster than the earlier implementation for an equivalent simulation model simulation.

Other algorithmic changes yield improved results only in particular cases. The most important of such improvements is the photon generation algorithm. Now, if the generation volume is a single point or completely enclosed inside one model component, the photons will be generated much faster. This is done by looking at the component in which the last photon was generated instead of checking the entire model. This speed improvement cannot be quantified because it is dependent on the model size and the number of simulated photons. The only inconvenience of this approach occurs when photons are generated in a large volume, there is a very small decrease in the simulation performance. The speed decrease is equivalent to having an additional component in the simulation model. Since the generation of photon is relatively rare, the penalty cost is negligible for most users.

2. Better random number generator

The conversion process, it became clear that the random number generation method was not sufficient when it came to doing very long simulations. In very long simulations, the total number of random numbers that can be generated can reach a significant portion of the length of the generator period. In order to avoid random number generation problems, the random number generator has been changed to a generator with a very long period: The L'Ecuyer generator with Bays-Durham shuffle[5]. This generator has a random number sequence of more than $2x10^{18}$ numbers. With current computers, the entire period cannot be reached unless the simulation runs for decades. Although this random generator is slower than the previous one, the gain of having a random number sequence that has no risks of repeating itself is a bigger gain.

The random number generation method change signifies that a simulation that was run under a previous version of DETECT may not have the exact same results. However, the differing results should not be statistically significant. Thorough testing was done after the conversion process to insure the statistical validity of the simulations.

3. Thorough model checking

In the previous DETECT version, some significant errors in the model definition could pass the model checking algorithm. Previously, it was possible to define a component with a surface without a finish and that surface was interfacing with the surrounding void. In normal conditions, a surface without a finish is use to connect components of the same material together in order to create more complex volume geometry. Since there is no check for a change of refractive index when a photon hits a surface without a finish, the simulation was simulating photons that left the simulation model with a direction change or possibility of Fresnel reflection. Now, in order to keep a correct simulation model, defining a component with a surface without a surface finish generate a check for refractive index of the connecting component. If the refractive index values are not exactly the same for each component, or equal to 1 if the component interface with the void, then the model is considered to be invalid.

With the advent of the wavelength dependent coefficient, discussed in section 4.2 of this paper, many new possible interactions can occur in the model. This requires a very stringent checking method for all the coefficient values and equally stringent model definition. Most of the coefficients on a given model component are independent from each

other and can be individually checked, but some are dependent on others. In this case, the dependent coefficients must be all defined for the same wavelengths. If they are not defined in this way, the normalization of the coefficients when a photon is wave-shifted could create abnormal behaviours in the long run. DETECT2000 can now find model definitions that, although correct when first defined, would create problematic behaviour after long simulations.

4. Limitless model size

The rewriting process has removed the model limitations on the various component types. In the previous version of DETECT, the different model components had a limit because they were stored in static arrays. The static arrays were changed to dynamically allocated arrays. The use of dynamic structures now allows for a very efficient memory usage and the actual memory allocation is a function of the size of the simulation. As an example, a simulation involving about 2000 components requires about 2.5MB of physical memory. With dynamic memory allocation, the number of model components is only limited by the physical memory of the system on which the simulations are running or by the maximal value of the system integer, which is more than 2 billion for a 32 bits system.

In order to increase further the memory efficiency of DETECT2000, all the model objects are also dynamically created. The model is in fact stored in dynamically allocated arrays of pointers that are instantiated only when needed. This approach does not have any memory overhead because dynamically allocated objects do not need an instantiated flag inside their structure.

Since the model no longer has limits, some of the definition lines can be very long. This is especially true for the definition of some component's surfaces. The connection definition can take many lines of the text. Thus, there is no limit on the size of the input string because it is now able to grow dynamically in order to fit any simulation file.

All these modifications were done to remove the need to recompile DETECT2000's source code every time a model limitation is reached. Now, one compilation of the source code creates a DETECT2000 executable that will accept any valid simulation file.

Along with the model size limitations, the limitation on the suffix of the FATE file possibilities has been removed. From ten possibilities, the FATE can take any positive integer value as a suffix. Also, one simulation file is no longer limited to one FATE file. At any time in the simulation, the FATE file can be closed and replaced by a new one. This new feature can help separate different events that are simulated inside one simulation file.

4. NEW DETECT2000 FEATURES

1. Time delay generation

In previous versions of DETECT, scintillation photons were generated and detected "instantaneously". In reality, this is never the case. For each material and photon-detecting surface, it is now possible to specify the parameters of a normalized probability distribution function (PDF) from which the delay is randomly sampled. Leo [6], Birks [7] and Moszynski [8] discuss these distributions at length, typically under the heading of "light pulse shape" and "pulse shape discrimination". Decay constants, relative weights and other distribution characteristics found in the literature are typically obtained using a "delayed-coincidence method" described by Bollinger and Thomas [9].

In DETECT2000, the generation time and detection time of each photon are written into the FATE files of the simulation. By extracting these values in conjunction with the flight time, it is now possible to have the entire elapsed time for each photon. DESTINY, a small program provided in the DETECT2000 suite is able to parse the FATE files in order to extract time data. DESTINY then creates a histogram of the desired time value such as generation time, detection time, flight time, etc. If no time delay is defined for a given material or detection surface, then no delay is applied for that component. There are three different ways to define a delay with DETECT2000: the sum of *n* decreasing exponentials, the sum of one rising and *n* decreasing exponentials and the convolution of a gaussian and the sum of *n* decreasing exponentials. For each case, the decreasing exponentials can have different weights to create a more realistic delay.

Since DETECT2000 can have very long simulations, it is important to reduce the time needed to compute the delay time. Taking the inverse of the delay function would take, in the long run, a significant part of the simulation time. To avoid this problem, we are creating a table of time related to the PDF when the delay time is specified. The PDF is divided into 1024 different parts and for each part, a mean value is computed. Since having only 1024 values is not sufficient to create a good time distribution, we use interpolation. Each time DETECT2000 generates a delay time, it produces a uniform distribution random number. This number is then multiplied by 1024 to choose which value will be used. The floating point remainder of the multiplication is used to linearly interpolate the delay value using the next value in the table. This method yields a very good approximation of the true PDF as shown in figures 2 and 3.

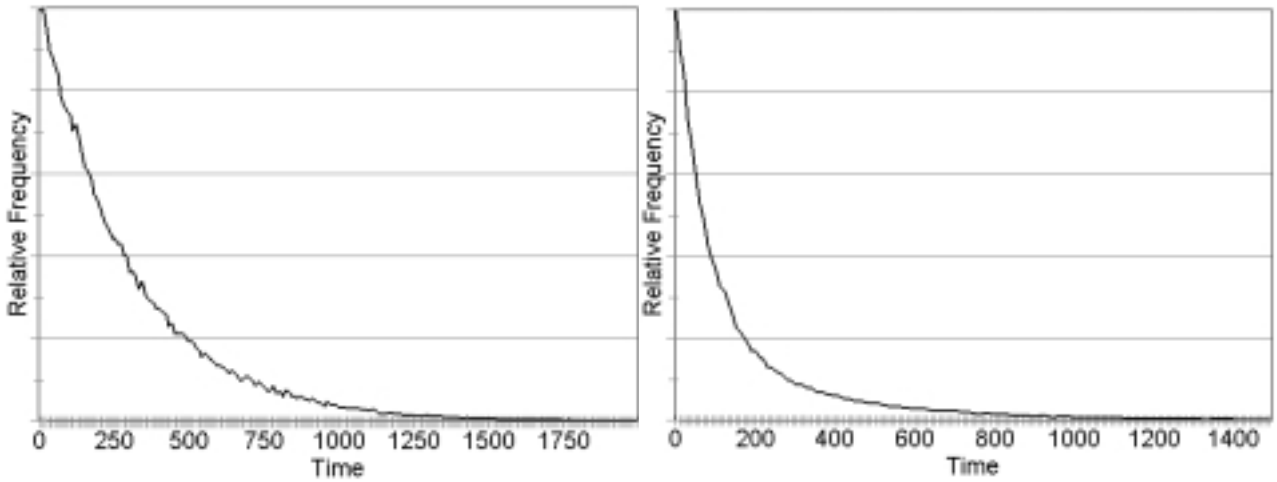

Fig. 2 Generated time dependencies. A falling exponential PDF of 300ns on the left and a PDF composed of two equally weighted falling exponentials of 75ns and 300ns on the right.

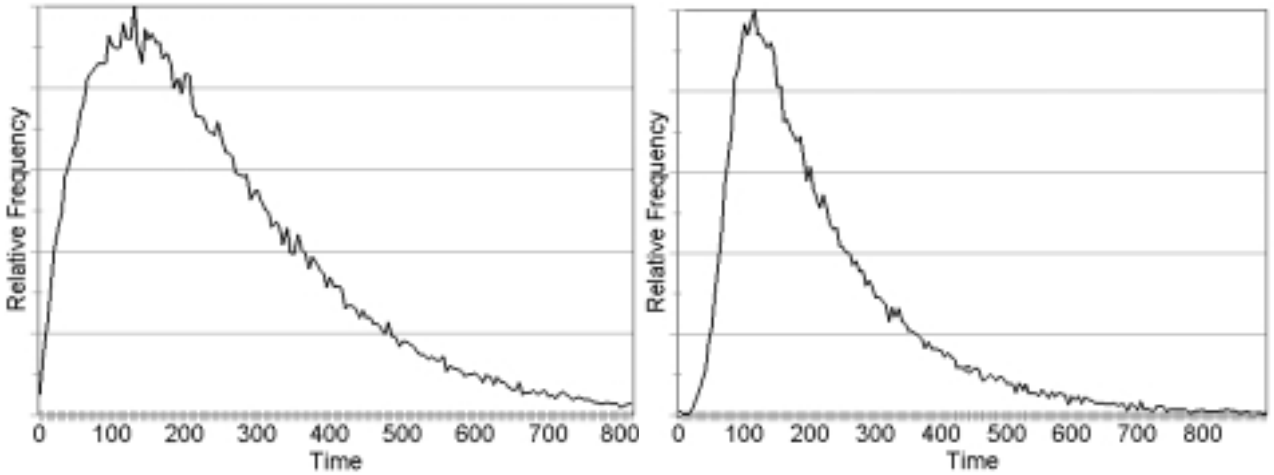

Fig. 3 Generated time dependencies. A rising and a falling exponential PDF of 100ns and 150ns respectively on the left and a falling exponential of 150ns convoluted with a gaussian of 20ns standard deviation on the right.

2. Wavelength dependent coefficients

Material and surface finish definitions can now call files in which a relevant coefficient or, in some cases, end result of an event is specified as a function of scintillation photon wavelength in nm. Interaction processes involving a wavelength dependent coefficient then occurs as a function of the current wavelength of the photon being tracked

throughout its lifetime. This is a very powerful new feature of DETECT2000 which accounts for spectral dependencies that are known to the user.

In previous versions of DETECT, the only wavelength dependencies were done by specifying two values for a coefficient. The first value was used for the wavelength of the generated photon and the second value was used when the photon had been wavelength shifted. This approach greatly reduce the number of possibilities. Now, it is possible to specify a generation spectrum and specify the wavelength dependent coefficients where they are needed. With DETECT2000, it is not necessary to specify a value for every possible generation wavelength. The simulation program will interpolate the correct value given the specified coefficient distribution.

3. Fast Statistical Counting

In DETECT2000 a new method, the fast statistical counting (FSC), as been implemented to help reduce the simulation run time. This method can also be called fast quantum efficiency counting. This method is used to reduce the number of photons that need to be simulated. Depending on the simulation model, FSC can reduce the simulation time by a very significant amount. However, using FSC does not guarantee similar results as a simulation done without FSC. When this feature is used, it is recommended to compare the FSC simulation runs with normal simulation to find out discrepancies. As a safeguard, this feature is disabled by default. FSC has already been used with success to study a position-encoding PET detector. In that case, only a single detection surface type was used for multiple PMTs and the authors were primarily interested in relative number of photons incident on each PMT[2][10].

The FSC algorithm uses the quantum efficiency of the PMT to increase the weight of each simulated photon. The quantum efficiency determines the probability of the detection surface to correctly detect the photon. When a photon hits a detection surface, the inverse of the PMT's quantum efficiency at the photon's wavelength becomes the weight of the current and subsequent non-detected photons. Since the quantum efficiency can only be defined in the [0,1] range, the new weights reduce the number of actually simulated photons.

4. Position sensitive surfaces

Another new feature that has been developed in DETECT2000 is the position sensitive detection surfaces, which simulates the behaviour of position sensitive PMTs (PS-PMT). This kind of detection surface is in fact an optional feature that can be activated for any detection surface in the model. When this is activated, the position of each detected light photon is recorded inside the surface. At the end of each simulation, the centroid of all the detected photons is written along with the simulation results. Since the data is stored inside the surface finish definition, it is possible to merge many detection surfaces results into one centroid data. However, many surface definitions with the same coefficient must be defined in order to have different centroid data if more than one similar PS-PMT is used in the model. The only negative impact of this approach is the slightly longer model definition file and a very slight increase in the memory usage of the software.

Before this feature was implemented, the user had to parse the entire FATE file in order to compute the centroid of detected light photons. Since the FATE file of a simulation can be more than more a few megabytes, using the position sensitive surface has the advantage of avoiding the creation of big intermediate files.

5. Projections of detected photons

Previously, if one was interested in knowing the location of the detected light photons, it would be necessary to parse the entire FATE file with a custom program. Now, a new feature has been integrated to DETECT2000 that yields very good results without the use of a specialized program. It is now possible to define projections of detected light photons. Since detection surfaces are not necessarily planar, the Projection object allows a very simple way to determine the distribution of detected photons.

A Projection object is defined by a plane center, a projection axis, a height axis and a plane width and height. When a photon is detected by any of the detection surfaces in the model, the final location of the photon is kept in memory. Then

this location is projected on each of the Projection objects. When the projection of the final location falls outside the plane boundaries, the event is ignored for that projection plane. This allows the definition of multiple projection planes which may each be in charge of one PMT. This also provides overall information of the photon distribution inside the crystal.

After each simulation, the Projection object outputs its results in the form a 2D histogram. The number of bins in each axis of the histogram can be defined by the user when the Projection object is created. This allows for optimal resolution for the simulated model. Each of the Projection objects writes its data into a separate file that can be later used for analysis.

5. DISCUSSIONS AND CONCLUSION

Using DETECT2000, we have done Monte-Carlo simulations on a multi-layered PET scintillation crystal in order to assess its performance[1]. The scintillation crystal is composed of two arrays of independent crystals that are all connected by a central layer. The scintillation block geometry has been modeled as an aggregation of more than 2200 individual components. For a simulation model of this size, the required memory is less than 3 megabytes of RAM under the Windows 2000 OS. When the most computing intensive surface finish is used for that simulation model, the average time to simulate 10000 photons without FSC was about 30 seconds on a Pentium 733MHz. However, this average simulation time should be taken only as an indication because the simulation is dependent on many things such as the size of the model, the actual location of the generated photons and the model's surfaces type.

DETECT2000 is enjoying a wide user base in both academia and industry. This is mostly due to the accessibility of the source code and the comprehensive license that comes with it. Also, it is possible to suggest modifications to the source code that would remove bugs or add features. A significant number of suggestions have been used to add features for the second release of DETECT2000. New features are currently implemented in a further release of DETECT2000.

The source code of DETECT2000 and of the other programs in the DETECT2000 suite this available for download at the following URL:
http://www.gel.ulaval.ca/detect/
Online documentation, user's guides and papers relating to DETECT can also be found on this website.

6. ACKNOWLEDGEMENTS

F. Cayouette's work is supported by a scholarship from the Natural Science and Engineering Research Council of Canada (NSERC Grant no. PGS A - 231976 – 2000)

REFERENCES

1. F. Cayouette, C. Moisan and C. J. Thomson, “Monte-Carlo Modeling of Scintillation Crystal Performance for Stratified PET Detectors using DETECT2000", IEEE 2001Conference Record, M13A-21, (2001).
2. G. Tsang, C. Moisan and J.G. Rogers, "A Simulation to Model Position Encoding Multicrystal PET Detectors", IEEE Trans. Nucl. Sci., NS-42 p. 2236 (1995).
3. G. F. Knoll, T. F. Knoll and T. M. Henderson, “Light Collection Scintillation Detector Composites for Neutron Detection", IEEE Trans. Nucl. Sci., NS-35, p. 872 (1988).
4. E. Gamma, R. Helm, R. Johnson, J Vlissides. Design Patterns: Element of Reusable Object-Oriented Software, Addison-Wesley (1994).
5. P. L’Ecuyer, Communications of the ACM, vol. 31, pp. 742-774 (1988)
6. W. R. Leo. Techniques for Nuclear and Particle Physics Experiments: A How-To Approach, Springer-Verlag: New York, (1987).
7. J. B. Birks. The Theory and Practice of Scintillation Counting. The MacMillan Co.: New York, (1964).
8. M. Moszynski and B. Bengtson. “Light Pulse Shapes from Plastic Scintillators", Nucl. Instr. and Meth., 142 pp. 417-434 (1977).
9. L. M. Bollinger and G. E. Thomas, “Measurement of the time dependence of scintillation intensity by a delayed-coincidence method", Rev. Sci. Inst., 32, pp. 1044-1050, (1961).
10. D. Vozza, C. Moisan and M. Loope, "Simulating the Performances of an LSO Based Position Encoding Detector for PET", IEEE Trans. Nucl. Sc., NS-44 p. 2450 (1997).