Open Access

QGATE: An Educational Environment to Learn and Perform Nuclear Medicine Imaging Simulation with GATE

Xiping Li and Weizhao Zhao*

Department of Biomedical Engineering, University of Miami, Coral Gables, FL 33146, USA

Abstract: Geant4 Application for Tomographic Emission (GATE) is a commonly used platform developed for nuclear medicine imaging simulation. It has been used to investigate imaging system's sensitivity, data acquisition methods, or image reconstruction algorithms. The GATE system, based on the Monte Carlo method, operates relying on user-provided scripts that follow the program definitions in the system's command library. Script-writing is usually time-consuming and frequently error-associated unless one has had certain experience in the simulation field and is familiar with programming. This is obviously an obstacle to students and beginners. The learning curve of script-writing is relatively flat to non-programming background learners. In addition, the GATE system is commonly installed on a high-performance computer, on which the scripts are usually compiled and used by the user only. In order to utilize GATE's potential to be a routine simulation toolkit, we have developed QGATE, a GUI-based GATE system. The QGATE system provides users a graphical environment where a simulation design is achieved by fashions like "drag-and-drop objects" and "choose-and-modify parameters". The scripts will be automatically generated and complied by the system. Parameters belonging to each object have default values so as to avoid errors induced by new users. The QGATE system consists of one or more QGATE *Clients* and one QGATE *Server*. The *Client* can be installed on individual computers so that multiple users can work on their simulation projects simultaneously by using the graphical designing tools. The *Server* is installed on the same computer as the GATE system resides. The *Server* manages the submitted simulation scripts queued by a *job ticket* and returns the *job finish* signal to the corresponding client. The QGATE system is suitable for classroom training and easy to use for students or new users to the field of nuclear medicine imaging simulation.

Keywords: PET, Virtual laboratory, Server-client, GUI.

INTRODUCTION

Position Emission Tomography (PET), a nuclear medicine imaging modality, is commonly used to diagnose or monitor tumor stages in clinical application and research oncology. Radioisotope-labeled pharmaceutical compounds, such as 2-deoxy-2-[^{18}F]fluoro-D-glucose (^{18}FDG), are usually administered to human beings or animals resulting in detectable photons produced by positron-electron annihilation. The detected photons, recorded as "counts" by radiation sensors, ultimately lead to the generation of a 2D/3D map of distribution of the radioisotope-labeled pharmaceutical compounds, the PET image, reconstructed by using various algorithms.

Due to ethical considerations, safety regulations, and/or economic allowance, the clinical protocol or animal experiment of PET imaging is usually investigated through computer simulations prior to its application. However, the simulation is a sophisticated process due to the probabilistic nature of radioactivity in PET imaging as well as issues regarding designs and algorithms in data acquisition, signal processing and image reconstruction. Monte Carlo methods are commonly proposed to deal with the scenario. Among existing simulation software packages, the Geometry and Tracking 4 (Geant4) [1,2] system, has been widely used to simulate applications in many fields, such as medical physics, nuclear physics, and particle physics. It provides well-validated physics models, geometry modeling tools and efficient visualization utilities. Based on the Geant4 libraries and using C++ object-oriented language to achieve a modular, versatile, scripted simulation toolkit, the OpenGATE collaboration [3] developed the Geant4 Application for Tomographic Emission (GATE) [4,5] system, which was dedicated initially to PET and single photon emission computed tomography (SPECT) simulations and has been extended to computed tomography (CT) and optical imaging simulations now.

The Monte Carlo method adopted by the GATE system has been validated [6, 7] through the comparison between the simulated data and the data collected from the commercial systems, such as the ECAT EXACT HR+ system (CPS Innovations, Knoxville, TN, USA) and the dual-headed AXIS SPECT system (Philips Medical System, Cleveland, OH, USA). Simulation results obtained from the GATE system has also been verified by the ^{111}In SPECT acquisitions through energy spectra, spatial resolution and sensitivity comparisons [8]. Data generated by the GATE simulation models were compared with the data collected by the Siemens biograph^{TM} 6 PET scanner (Siemens Medical Solutions, Knoxville, TN, USA). A good agreement between the simulated and the experimental data sets for the scatter fraction and count rate measured at 1 kBq/ml concentration has

*Address correspondence to this author at the Department of Biomedical Engineering, University of Miami, Coral Gables, FL 33146, USA;
Tel: 1-305-284-6763; Fax: 1-305-284-6494;
E-mails: w.zhao1@miami.edu; wzhaoum@gmail.com

1874-3471/11 2011 Bentham Open

been reported [18]. The simulation accuracy, reliability and applicability make the GATE system a validated platform for PET and SPECT applications, such as detector and scanner design [9-11], image reconstruction [12-14], scatter and attenuation correction methods [15,16], protocol optimization [17], and so on.

The GATE system has been frequently used as a simulation toolkit. The popularity of this system has been proven by the fact that more than 11,000 users have registered on the GATE website. However, GATE does not provide users with a graphical user interface (GUI) for simulation-design and performance-administration, which is an obstacle for users with less experience in the field. In addition, in order to fully utilize the capability of GATE, further customization by editing and modifying scripts is required. The work involves user's compiling and programming knowledge under the environments of C++, Roots Object Oriented Technologies (ROOT) [19], or Class Library for High Energy Physics (CLHEP). It is highly desirable to have an instrument that allows a user to construct his/her simulation by graphical tools without involving much programming and to manage the simulation jobs efficiently. An automated or semi-automated solution to the simulations would be appreciated by a number of users, especially those with less experience or non-programming background. Based on the motivation, we designed and implemented the QGATE system. The QGATE system consists of a client component and a server component. The client part provides users with GUI-based tools for preparing and generating scripts, and functions for uploading and monitoring the designed simulation. The server part manages the simulation processes and feeds back the simulation results. Considering that the GATE system is usually installed on a high-performance computer system, our designed QGATE software can be installed separately on a local computer (the client) and on a high-performance computer (the server). Two parts of the QGATE system communicate through Internet connection. Fig. (**1**) shows the configuration of the QGATE system. Following sections describe the QGATE system's design, function, implementation and application results.

The goal of the developed QGATE system is to build an educational GATE environment. The proposed system should have following two features. 1) The system allows less-experienced or non-programming-background users to operate GATE "smoothly" when conducting their designed clinical or research tasks. Meanwhile, the system serves as a training platform to teach users how to use GATE or other similar simulation software. 2) The system allows multiple users to design their simulation tasks independently and simultaneously, and to submit their simulation jobs to the same server where the GATE system is installed. The system also has management functions to save or reuse designed simulations.

METHODS

The GATE simulation software is based on the so-called *script* commands through parameters entered by user in an interactive fashion. An object in the simulation, such as a radiation detector, has a number of property parameters. If any of them had an error, such as a wrong property name, GATE would generate a wrong simulation result or an exception to terminate the simulation. We proposed a GUI client interface to help user design objects (e.g., box, sphere, cylinder, etc.), which can be selected from the *object table* provided by an *ObjectList* window. Each object's property parameters can be selected from the *property table* (e.g., geometric dimensions, material names, etc.). The names of the property parameters are defined in corresponding database files and linked to the dropdown lists so that human errors can be effectively avoided. The client of the system

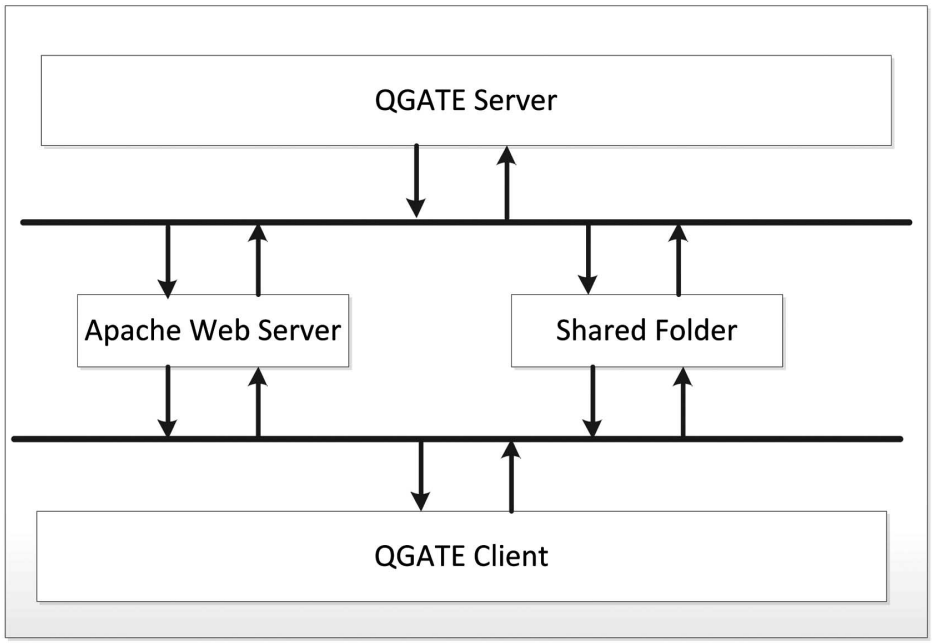

Fig. (1). The QGATE system consists of a server and a client. When the two components are installed on the same computer, they exchange information through shared folders. When the two components are installed separately, they communicate through a secured LAN environment or an Apache web server. The hierarchy of the QGATE system shows the simulation design, processing, and output bi-directionally through these components.

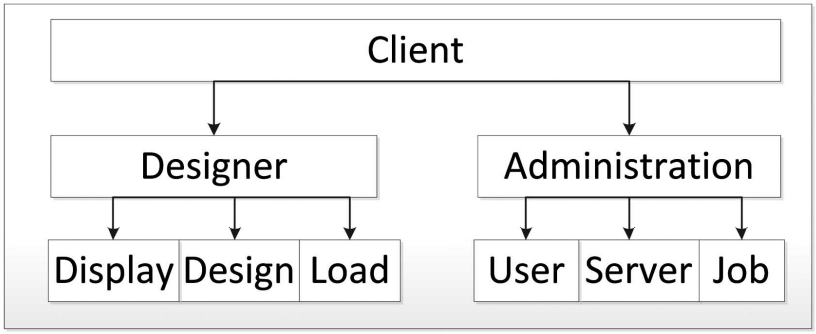

Fig. (2). The QGATE client component has two modules. The Designer module provides users with a GUI interface to facilitate the designing process, interprets the design into the GATE-recognized simulation scripts and uploads the scripts to the server. The designer module also displays the design in a 3D scene. The Administration module manages simulation jobs, configures the server parameters and administrates user accounts.

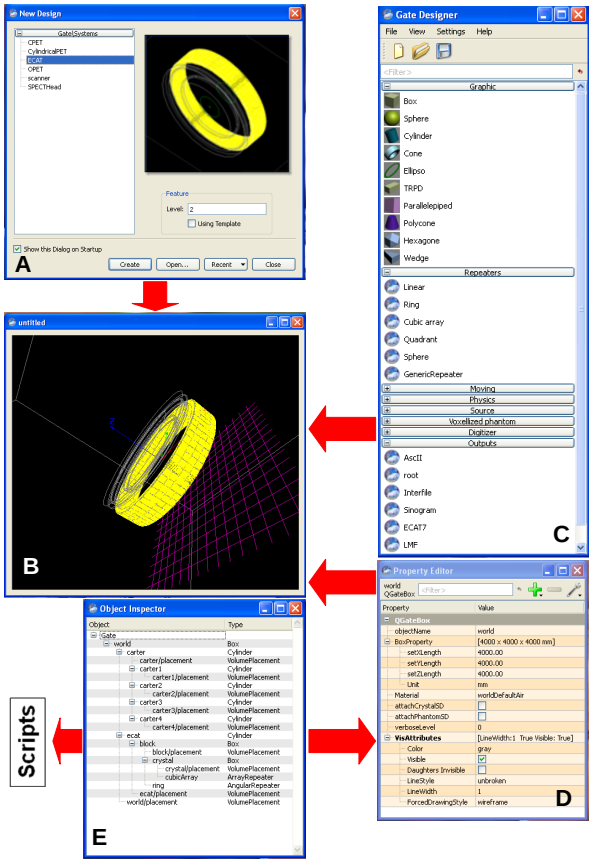

Fig. (3). The graphical designer is the "core" of the QGATE Client, consisting of following components: (**A**) The design platform from which a user can select a pre-designed system (template) as a base to either build a more complicated system or modify the template by manipulating its parameters. This feature particularly benefits beginners by providing a starting point of the design. (**B**) The design display that presents the 3D visual information of graphical objects. A user can move, rotate and zoom the objects on the display. The graphical objects on the display are updated simultaneously when the objects' parameters are modified. (**C**) The designer's Object List provides users with a number of graphical or non-graphical objects to build a simulation system. (**D**) The Property Editor is associated with the object. When an object is selected in the display, its Property Editor appears for users to update its parameters. Every parameter has been assigned a default values before the object is selected. This feature eliminates the error caused by inexperienced users. (**E**) The Object Inspector summaries all objects in the design and creates corresponding scripts. The designer provides a step-by-step "training" fashion for a simulation design. Users can learn not only the design procedures but also the nuclear medicine imaging system at the same time as well.

will automatically generate the scripts for the simulation. Meanwhile, the 3D display of selected objects helps user understand the simulation system intuitively. Under such an environment, the simulated PET/SPECT scanner and the simulated phantom can be directly conveyed to users. Fig. (**2**) shows the functions associated with the client component.

Since the GATE system is usually installed on a high-performance computer [20] and the simulation jobs are usually designed from the user's local computer, we proposed to

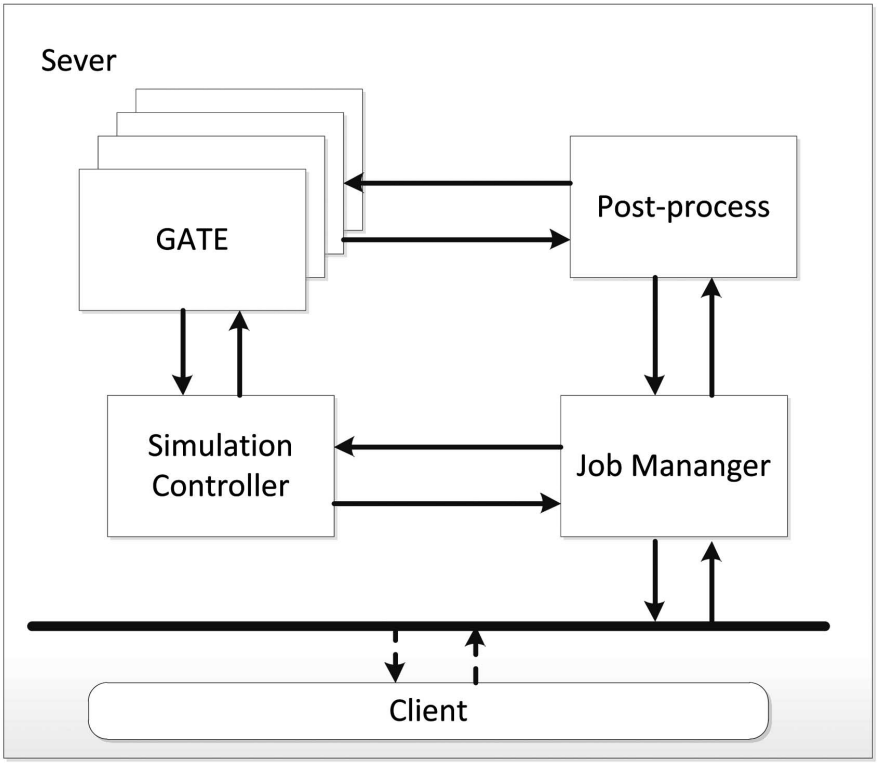

Fig. (4). The QGATE server component has four modules: Job Manager, Simulation Controller, GATE, and Post-Processing. The Job Manager communicates the Administration function in the client to queue simulation jobs. The Simulation Controller imports uploaded scripts to the GATE system. The Post-Processing module collects the simulation results from GATE and exports them back to the Job Manager.

build a server interface to host the GATE system on the high-performance computer. The client interface can remotely access the server interface and deliver the generated scripts to the server. The server manages the scripts to be queued, uploaded, or reused for the GATE system. The server can also return the simulation results to the client. By this way, users can get their simulation jobs done remotely and check their simulation results individually. Fig. (**3**) shows the functions associated with the server component. Based on these design principles, QGATE has been constructed as an integrated solution to simulation design and simulation education under the GATE environment.

Implementation

As a simulation software platform with client and server architecture, the QGATE system is written in C++. Current version of the QGATE Client is developed based on the open source software GATE 6.0, Geant 4.93 and Qt 4.60 [21]. Qt is a cross-platform GUI development toolkit. This single source (C++) compatibility makes the client run on different platforms without additional development. The QGATE Client has been tested under operating systems of Microsoft Windows XP, Microsoft Windows 7 (32 Bit) and Linux Fedora 10-14. QGATE Sever is also written in C++. It wraps GATE 6.0 as the core to interpret the simulation job.

QGATE Client

Based on the design principles, the client component of the QGATE system can be installed on a remote computer (Fig. **1**). The *Designer* module in the client provides users with a GUI interface (Fig. **2**). However, converting a simulation design into the corresponding scripts is the key step to ensure the QGATE system's automation. We took the *open source* advantage of the GATE system and analyzed its workflow and running mechanism. From the source codes, we extracted and created the script templates for different objects and exported them to associate with graphical objects under the GUI environment. The objects' property parameters were linked to the property name database file. The *Designer* guides a user to select an object, to fill out required property parameters (if a parameter is omitted, a default parameter will be used), and to choose/produce its lower/parallel hierarchical objects. The scripts will be generated by the embedded templates.

Fig. (**3**) demonstrates how the QGATE system assists a user to complete a simulation design. The construction of a design starts from choosing a simulation system. A system component, i.e., an object, can be dragged from the *Gate Designer* window's Graphic (object) list and dropped to the *New Design* window. The object properties will be displayed by the *Object Inspector* window. These properties are specified by corresponding parameters for objects of sensor, crystal, digitizer, coincidence, random engine, output and so on. The user can edit the selected object's properties through the *Property Editor* window. A default property will be used if it is not defined by the user. When the user saves the design for the specified object(s), the script(s) will be generated by the QGATE system automatically. Since the GATE system has already built several internal pre-defined systems, such as Cylindrical PET, ECAT, these systems can serve as simulation templates for users. The QGATE Designer allows users to select these templates, modify them, and add new objects to them so as to build their own new designs. The QGATE

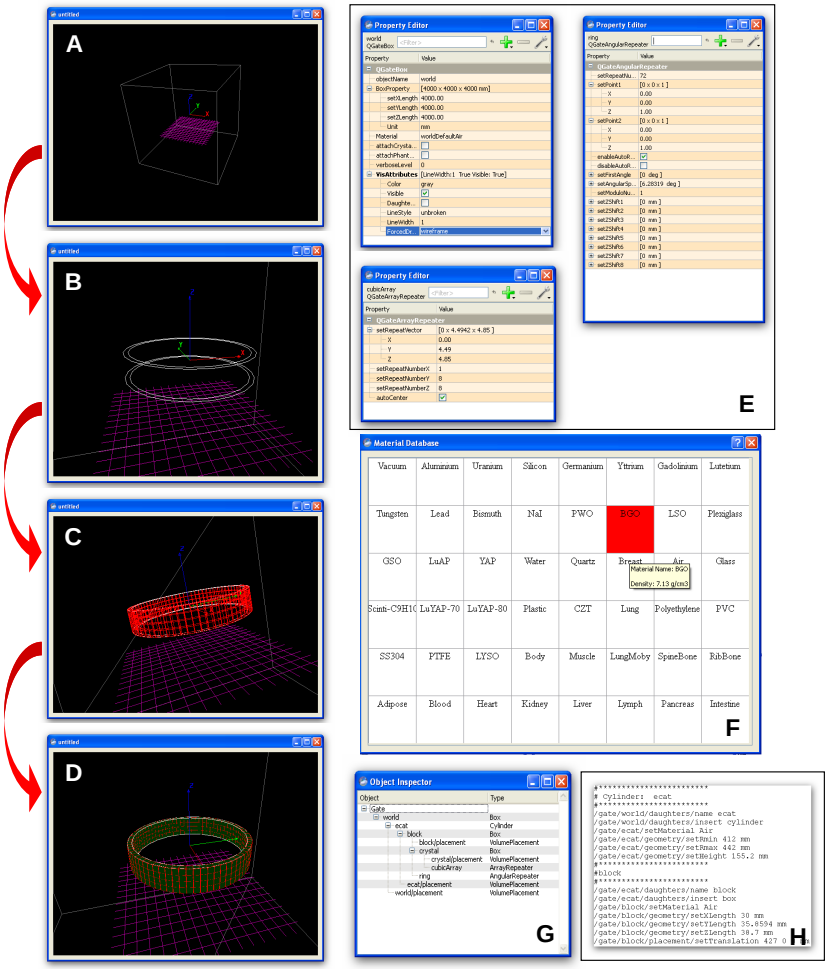

Fig. (5). A design flow chart shows an example by using the QGATE system to build an ECAT family simulation system. **A**) A new design starts from a default "world" (a box object), whose geometric parameters can be modified from the Property Editor (**E**). **B**) A cylinder is dragged from the Object List (Fig. **3C**) and placed onto the world. The cylinder's parameters can also be modified through the Property Editor (E). **C**) A block is added to the cylinder. The block's material (BGO) is selected from the Material Database (**F**). The block's geometry is set through Property Editor. **D**) Linear and cubic array repeater objects are then dragged onto the block object. The cubic array repeater is set to 8 so as to make 8×8 copies in Y and Z direction based on the defined repeater direction. The ring repeater object is associated with the cylinder and the number for the ring repeater is set to 72 so as to fill out the 3D space of the simulator. **F**). The QGATE designer updates all entered objects and generates output scripts (**H**).

system interprets the graphical design to be executable scripts and upload the scripts as outputs of the client. When the graphical elements, i.e., objects, are removed, duplicated or modified during the design, the scripts will also be updated accordingly.

The *Administration* module in the client component plays the *manager* role for the QGATE system. The user-account is initiated by an *administrator* account that can add, remove and modify user accounts so that multiple users can sign on the client to design simulation tasks. If more than one simulation task need to be executed, the *Administration* module will generate a *job ticket* for each task. According to the order of the *job tickets*, the server conducts simulation sequentially. In the QGATE system, each simulation is regarded as a job and each job is listed in one of three job queues: 1) the waiting job queue, 2) the running job queue, or 3) the finished job queue. The server will pick up a job from the waiting job queue and the client will update the queue statuses. The *Administration* module also helps user to communicate with the server, such as uploading simulation scripts and related data files, getting simulation outputs, and checking simulation status.

QGATE Server

The QGATE Server component is a background program under the Linux platform or a service under the windows platform. When the QGATE Server starts, it will broadcast its own information in the local network and listen to the requesting information from a specific port. The QGATE Client, locating in the same network, can identify the Server and communicate the Server through the Apache web server. As illustrated in Fig. (**4**), the QGATE Server component has four modules.

The *Job Manager* module treats every simulation task as a job. Each job is identified by the *job ticket* that contains the information specified by scripts. Each job is submitted by the user to either a specified shared folder in the high-performance computer (if the QGATE Client and the

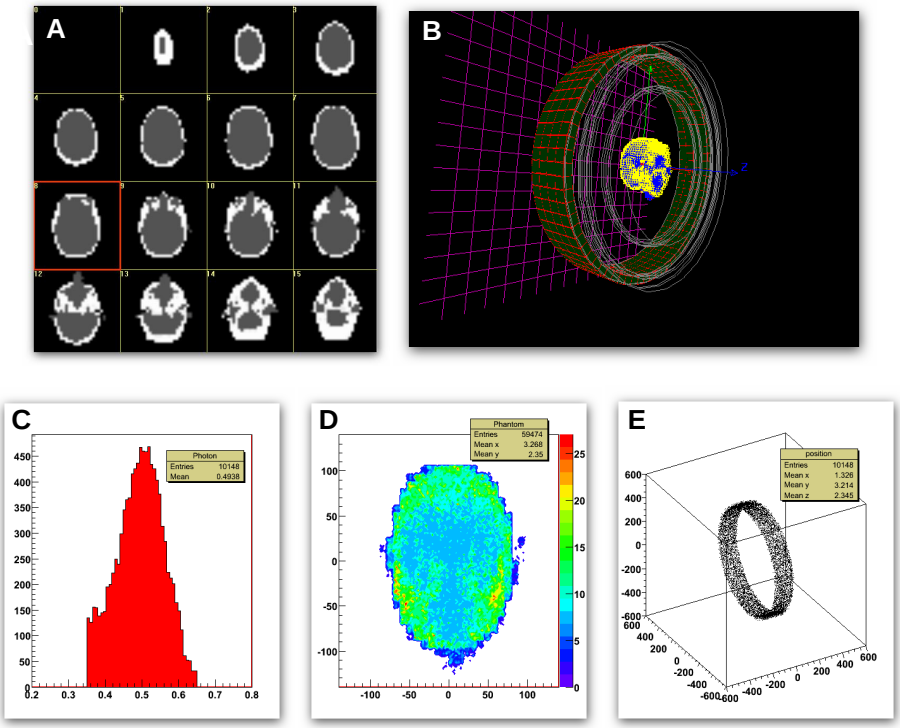

Fig. (6). Data collected through the QGATE system are displayed when the Hoffman digital phantom is placed into the ECAT family simulation design (Fig. **5**). The results were created by the ROOT tools, which can be called by the QGATE server. (**A**) The Hoffman digital phantom is composed by 32×32×16 voxels (X-Y-Z directions). The resolution is 8.8×8.8×11.2 mm^3/voxel. The intensity values corresponds to an active table defined by three subdivisions of 1.0 Bq, 3.0 Bq and 5.0 Bq. (**B**) The Hoffman digital phantom is embedded onto the ECAT family simulation system. (**C**) Distribution histogram of coincidence photon's energy. An energy window is set to [350keV 650keV] to highlight the mean value close to 500kev by the Monte Carlo method through about 10,000 trials. (**D**) 3D distribution on the system (block arrays) by the simulated photons. (**E**) All incident events received by the photons' *source* positions (x,y,z) are projected to the X-Y plane.

QGATE Server locate on the same local area network (LAN)) or Apache web service hosted on the Server side. The Job Manager monitors the shared folder or a port which is used to communicate with the Apache web server constantly. Based on the *job ticket's* description, the Job Manager prepares and forwards all required information to the *Simulation Controller* module to conduct simulation.

Since the QGATE Server can face multiple users through multiple QGATE Clients, the *Job manager* module checks each job-sender's authentication and allocates data space for simulation results. Based on the job status, the *Job Manager* lists the job in the running queue, waiting queue or finished queue. When a job is submitted, a job object will be created. A unique ID will be assigned and the *Job Manager* will store all related information to the ID for backup. The *Job Manager* is also responsible for exporting simulation results processed by the *Post Processing* module. The *Simulation Controller* module monitors the waiting job queue, picks the job that has the highest priority or the first available on the queue, and feeds the required information to the GATE.

Design Example and Results

The ECAT system is a pre-designed example included in the GATE system. Its scripts can be pulled out and modified to conduct specified simulations. However, a user must know all specifications in order to construct scripts. Here we give a design example by using the QGATE designer to build an ECAT family system (Fig. **5**).

When a new design is started, the QGATE *designer* will always create a new default object, a world. The world is a box object with air as default material. The world's geometric dimensions should be large enough so that all objects can be added later. The world's length, width and height can be changed in the *Property Editor* window. Other properties, such as line color, line thickness, or wire-form for display, can also be changed in the window (Fig. **5A** and **E**). The user can then drag a cylinder object from the GATE Designer window's *Object List* (see Fig. **4C**) and drop it onto the world (Fig. **5C**). The cylinder can be named as "ECAT" (or other name) through the cylinder's *Property Editor*. Similar to the world, other properties of the cylinder can also be modified. The user will next add a crystal block to the cylinder. The crystal's material can be chosen from the *Material Database* (Fig. **5F**). The crystal block's dimension is defined by the similar way to the cylinder's. After the crystal block is designed, user can drag a repeater object from the *Object List* (see Fig. **4C**, *Repeater list*) and associate it with the already-selected crystal block. By editing the repeating numbers in x, y and z directions, the user can fill them in the 3D space inside the cylinder (Fig. **5D**). The user can then save the design as scripts and export the file to the server (Fig. **5G** and **H**). A brain phantom can be selected from the *Object List*. Fig. (**6A**) shows the selected Hoffman digital phantom. Its location and radioisotope concentration can also be modified. Fig. (**6B-6E**) show the resultant images returned through the QGATE Server through 100 second simulation by GATE.

The QGATE designer allows every object's properties to be modified graphically by clicking the object. Its associated *Property Editor* will appear. By modifying the visual properties, the result will be updated immediately in the display panel. The QGATE designer will read the *material database* from its system folder. The database includes all materials from defined in the GATE system.

The QGATE Client serves the *interpreter* function for GATE and the QGATE Server plays the *manger* role for GATE. In term of functionality, the QGATE system has so far focused on PET simulations. Objects related to PET simulation in GATE 6.0 have been wrapped in the QGATE Client. In terms of performance, the QGATE system has a very similar performance capability to the GATE 6.0 system. Functions and objects in GATE that are not related to PET simulations have not been included in the QGATE system. Both QGATE Client and QGATE Server are open-source software under GPL. The source codes and related documents can be downloaded from the webpage hosting our medical imaging simulation teaching software, http://mis.eng.miami.edu/qgate/qgate.html.

CONCLUSION AND DISCUSSION

The QGATE system has been developed as an educational or training tool kit to perform nuclear medicine imaging simulation. Considering the complexity of emission tomography, we developed this system aiming to ease the learning process for beginners or medical/biomedical students. The advantages of the system include the user-friendly graphical interface to avoid sophisticated scripts preparation, the multi-user/job management function, and the ability to remote-access the high-performance server.

The GATE system is a powerful simulation tool kit. However, its full potential as a routine simulation system to beginners has not been fully exploited. The major obstacle is the composition of scripts, in terms of syntax, property names and importing/exporting scripts/results. One of our endeavors in this study is the "translation" (or interpretation) from text-based script composition to GUI-based design. Under the GUI environment, beginners in the field can easily build the simulation protocol by visually choosing a predesigned model and editing the objects' properties without struggling on writing program-like commands. Another endeavor in this study is to separate the designer from the high-performance server. The QGATE Client-Server setup enables the designer (client) can be installed on difference local computers due to the fact that the design process does not require much computational resource. Multiple users can do their design individually, such as in a training class setup. The QGATE client provides efficient methods to generate scripts and upload to a remote server. The user can easily design the simulation by using drag and drop tools, and immediately obtain the scripts corresponding to the graphical design. The user can also check the details of each designed object by using zooming, rotating and on/off graphical tools. The QGATE client checks all objects' properties before exporting the scripts to the server. For example, material names are commonly mistyped and entered to the system. This specifically designed feature avoids user's mistake and misunderstanding the GATE commands, such as the command sequence and command name. The QGATE server, installed on the high-performance computer, manages the uploaded jobs and returns the simulation results by the order of job tickets.

The QGATE system is a graphical simulation application that helps the user to use and understand GATE and Geant4. It assists beginning users or students to get training in the field of the emission tomography simulation. We have long been developing the medical imaging teaching software (MITS) and dynamic assessment tracking system (DATS) for medical imaging education [22, 23]. The QGATE system is inherently a part of the MITS/DATS system. Further evaluation for the QGATE system will be assessed through the MITS/DATS application while delivering nuclear medicine imaging courses.

REFERENCES

[1] Agostinelli S**,** Allison J**,** Amako K, *et al*. Geant4 – a simulation toolkit. Nucl. Instum Meth Phys Res A 2003; 506(3): 250-303.

[2] Geometry And Tracking (Geant) transport code available at: http://www.cern.ch/geant4 (06/20/2011)

[3] OpenGATE Collaboration Available at: http://www.opengatecollaboration.org/ (06/20/2011).

[4] Jan S, Santin G, Strul D, *et al.* GATE: A simulation toolkit for PET and SPECT. Phys Med Biol 2004; 49(19):4543–61.

[5] Strul D, Santin G, Lazaro D, Breton V, Morel C. GATE (Geant4 application for tomographic emission): a PET/SPECT general-purpose simulation platform. Nucl Phys B 2003; 125: 75-9.

[6] Assié K, Breton V, Buvat I, *et al*. Monte Carlo simulation in PET and SPECT instrumentation using GATE. Nucl Instrum Meth Phys Res A 2004; 527: 180–9.

[7] Jan S, Comtat C, Strul D, Santin G, Trebossen R. Monte Carlo simulation for the ECAT EXACT HR+ system using GATE. IEEE Trans Nucl Sci 2005; 52: 627-33.

[8] Assié K, Gardin I, Véra P, Buvat I. Validation of the Monte Carlo simulator GATE for indium-111 imaging. Phys Med Biol 2005; 50: 3113–25.

[9] Rannou FR, Kohli V, Prout DL, Chatziioannou AF. Investigation of OPET performance using GATE, a geant4-Based simulation Software. IEEE Trans Nucl Sci 2004; 51: 2713-6.

[10] Simon L, Strul D, Santin G, Krieguer M, Morel C. Simulation of time curves in small animal PET using GATE. Nucl Instrum Meth Phys Res A 2004; 527: 190-4.

[11] Buvat I, Lazaro D. Monte Carlo simulations in emission tomography and GATE: An overview. Nucl Instrum Meth Phys Res A 2006; 569: 323-9.

[12] Santin G, Strul D, Lazaro D, Breton V, Morel C. GATE, a Geant4-based simulation platform for PET and SPECT integrating movement and time management. IEEE Trans Nucl Sci 2003; 50: 1516–1521.

[13] He JF, O'Keefe GJ, Jones G, *et al*. The application of GATE and NCAT to respiratory motion simulation in allegro PET. Proc IEEE Nucl Sci Symp; 2006: M11-202.

[14] Sakellios N, Rubio JL, Karakatsanis N, *et al*. GATE simulations for small animal SPECT/PET using voxelized phantoms and rotating-head detectors. Proc of IEEE Nucl Sci Symp, 2006; M06-206.

[15] Konik A, Madsen MT, Sunderland JJ. GATE simulations of human and small animal PET for determination of scatter fraction as a function of object Size. IEEE Trans Nucl Sci, Orlando, FI 2010; 57: 2558-63.

[16] Yoshida E, Yamaya T, Shibuya K, Nishikido F, Inadama N, Murayama H. Simulation study on sensitivity and count rate characteristics of "OpenPET" Geometries. IEEE Trans Nucl Sci 2010; 57:111-6.

[17] Karakatsanis NA, Loudos G, Nikita KS. A methodology for optimizing the acquisition time of a clinical PET scan using GATE. Proc IEEE Nucl Sci Symp Conf, 2009; M05-358.

[18] Gonias P, Bertsekas N, Karakatsanis N. *et al.* Validation of a GATE model for the simulation of the Siemens biograph 6 PET scanner. Nucl Inst Meth Phys Res A 2007; 571:263-6.

[19] Brun R, Rademakers F. "ROOT - An object oriented data analysis framework", Proc AIHENP Workshop. Nucl Inst Meth Phys Res A 1996; 389:81-6.

[20] De Beenhouwer J, Kruecker D, Staelens SG, Ferrer L, Chatziioannou AF, Rannou FR. Distributed computing platform for PET and SPECT simulations with GATE. Proc. of IEEE Nucl Sci Symp. Conference, 2005; M11-99.

[21] Qt - Cross-platform application and UI framework: Available on: http://qt.nokia.com/ Accessed on; (06/22/2011)

[22] Dikshit A, Wu D, Wu C, Zhao W. An online interactive tutorial to simulate non-invasive medical imaging modalities. Computerized Medical Imaging and Graphics 2005; 29:395-404.

[23] Li X, Manns F, Zhao W. Medical Imaging Education by a Multi-level Module-based Online Teaching and Assessment System. Proceedings of the 40th IEEE Frontiers in Education Conference, Washington DC, October, 2010.

Received: April 24, 2011 Revised: June 20, 2011 Accepted: July 05, 2011

© Li and Zhao; Licensee *Bentham Open*.

This is an open access article licensed under the terms of the Creative Commons Attribution Non-Commercial License (http: //creativecommons.org/licenses/by-nc/3.0/) which permits unrestricted, non-commercial use, distribution and reproduction in any medium, provided the work is properly cited.