Choosing a Suitable Framework for Flexible Simulation of a Picosecond TOF Device

Shreyas Bhat
Enrico Fermi Institute, University of Chicago
Xiao Han
Department of Radiology and Medical Physics, University of Chicago Hospitals

Abstract

In this paper, we examine the simulations of two different applications of a Picosecond time-of-flight (TOF) detector. The system uses Cherenkov light created in the window of the photomultiplier tube (PMT) as the detecting medium. We examine the possibilities of using GEANT4 [1], and of using the simulation package called Geant4 Application for Tomographic Emission (GATE) [2]. With some simple modifications to the GATE source code, we can use GATE for both Positron Emission Tomography (PET) and HEP applications.

1 Introduction

We have created a working simulation for a Picosecond TOF Device [3]. However, as it stands, the simulation is extremely specific to a particular detector implementation, and is quite inflexible. We aim to create an end-to-end simulation[1] of the TOF Device that is modular and can thus be used for a variety of physical and radiological applications. We want to encompass in the simulation the origins of the particles to be detected, their path through any given geometry and into any defined photomultiplier tube (PMT) or microchannel-plate (MCP) PMT, the charge collection on an anode and the subsequent electronics, for the purpose of obtaining timing resolution.

2 HEP Simulations

In the current system, the HEP simulations are done in three programs: GEANT4, ROOT, and HSPICE. We use a combination of GEANT4 source codes and macros files to define the geometry of the system, the properties of the test particles involved, tracking methods, and to invoke physics processes. The result of this simulation is a text file, which is then routed through a Python script into a series of ROOT scripts. These include material information for different MCP windows, information regarding the quantum efficiency of the MCP (Burle Planacon [3]), and a simulation that produces the location and time of arrival of the resultant pulses. Another ROOT script converts the results into an HSPICE input file. HSPICE is then used to simulate the electronics involved, and to generate pulses. A more detailed set of block diagrams is given in Figures 1- 2.

[1] By an 'end-to-end' simulation, we refer to a simulation that starts from the origin of the particle to be detected, whether in a collision or by decay, and simulates the process of detecting the particle through the time when we register the detection of the particle and its relevant properties.

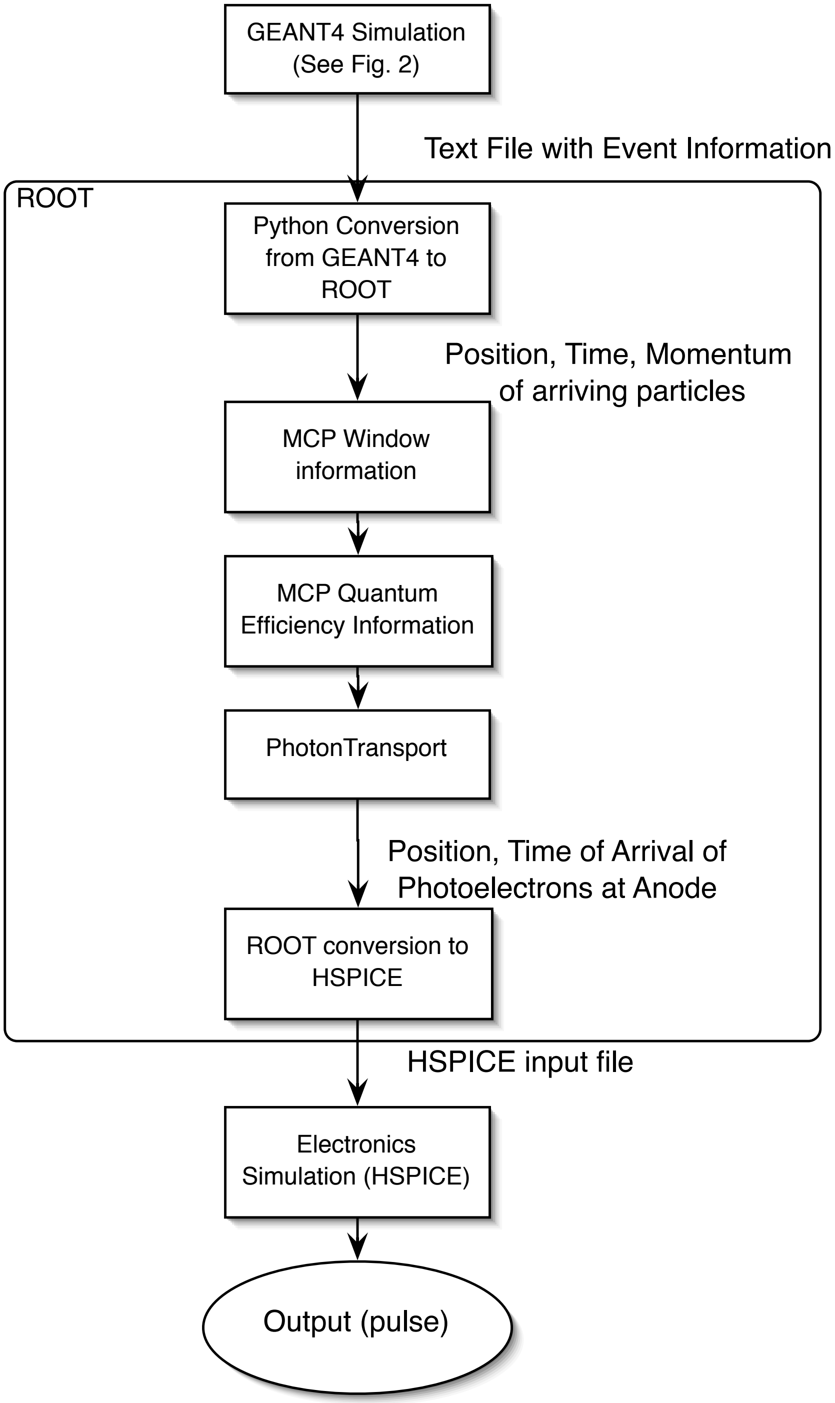
GEANT4 Simulation (See Fig. 2)
Text File with Event Information
ROOT
Python Conversion from GEANT4 to ROOT
Position, Time, Momentum of arriving particles
MCP Window information
MCP Quantum Efficiency Information
PhotonTransport
Position, Time of Arrival of Photoelectrons at Anode
ROOT conversion to HSPICE
HSPICE input file
Electronics Simulation (HSPICE)
Output (pulse)

Figure 1: Block diagram for the current HEP TOF 'end-to-end' simulation.

The current simulation has proved very useful, but has been patched together. It requires the installation of two different programs in order to run (GEANT4 and ROOT) and it is not modular enough to be easily changeable. For example, in order to change the angle of incidence of particles, one must go directly to the source code, whereas particle energy is

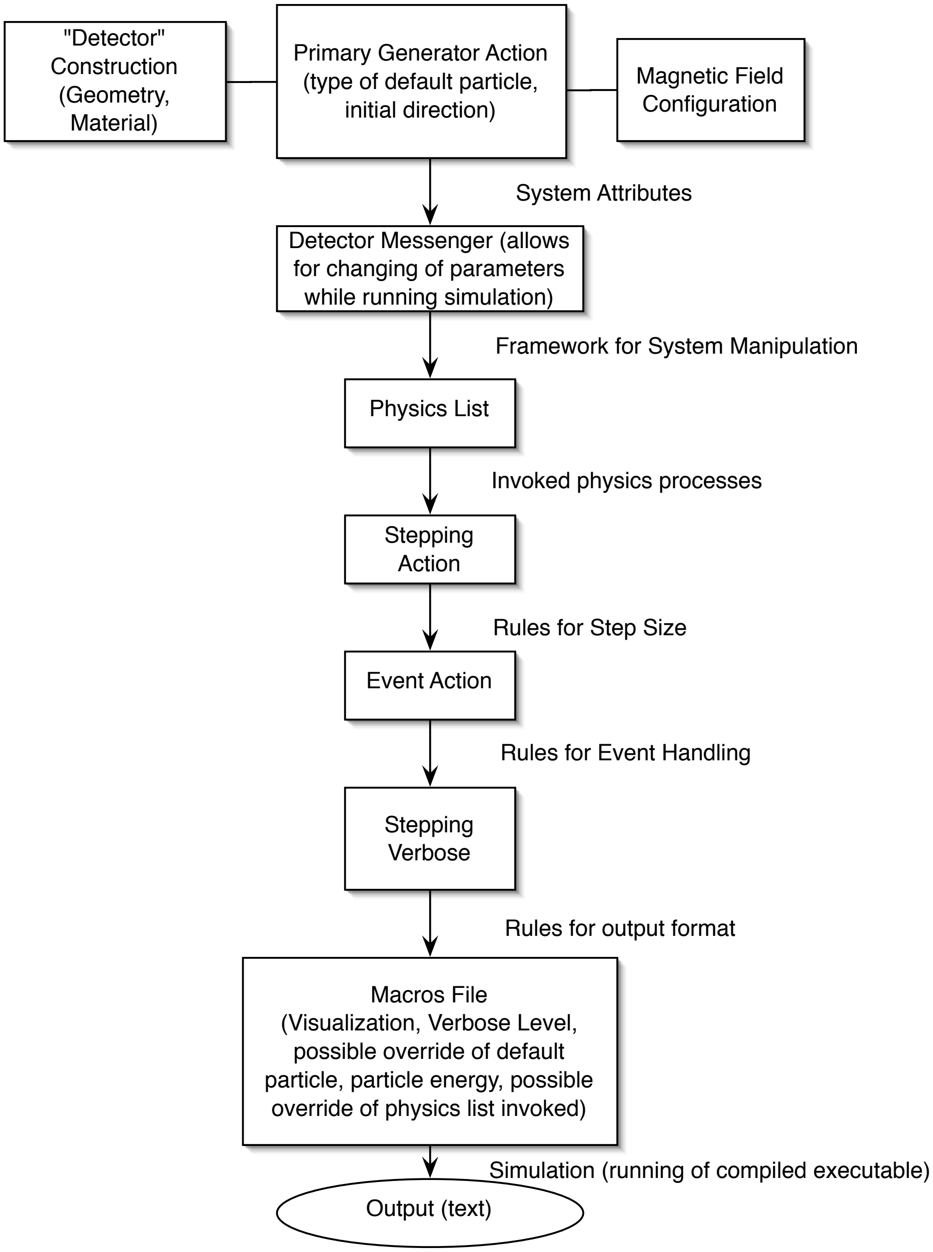
"Detector" Construction (Geometry, Material)
Primary Generator Action (type of default particle, initial direction)
Magnetic Field Configuration
System Attributes
Detector Messenger (allows for changing of parameters while running simulation)
Framework for System Manipulation
Physics List
Invoked physics processes
Stepping Action
Rules for Step Size
Event Action
Rules for Event Handling
Stepping Verbose
Rules for output format
Macros File (Visualization, Verbose Level, possible override of default particle, particle energy, possible override of physics list invoked)
Simulation (running of compiled executable)
Output (text)

Figure 2: Block diagram for the GEANT4 portion of the current simulation. Note that, for example, to change particle attributes, one must either edit the macros file or the source code. Furthermore, there is no flexibility about geometry unless the source code is directly modified.

edited in a macros file. A better framework would eliminate these problems.

There are two possibilities for a new framework. We could continue to use GEANT4 as the simulation program, and rewrite the MCP code for GEANT4. This would allow for more interchangeability within the simulation. However, there are drawbacks. For a pure GEANT4 simulation, we would have to completely rewrite the simulation in order to provide for more modularity. Currently, very few aspects of the simulation are easily changeable, and those are relatively minor. A new simulation would have to allow for different geometries or configurations to be easily implementable to accommodate different applications. Figure 3 shows the layout of an ideal GEANT4-based simulation.

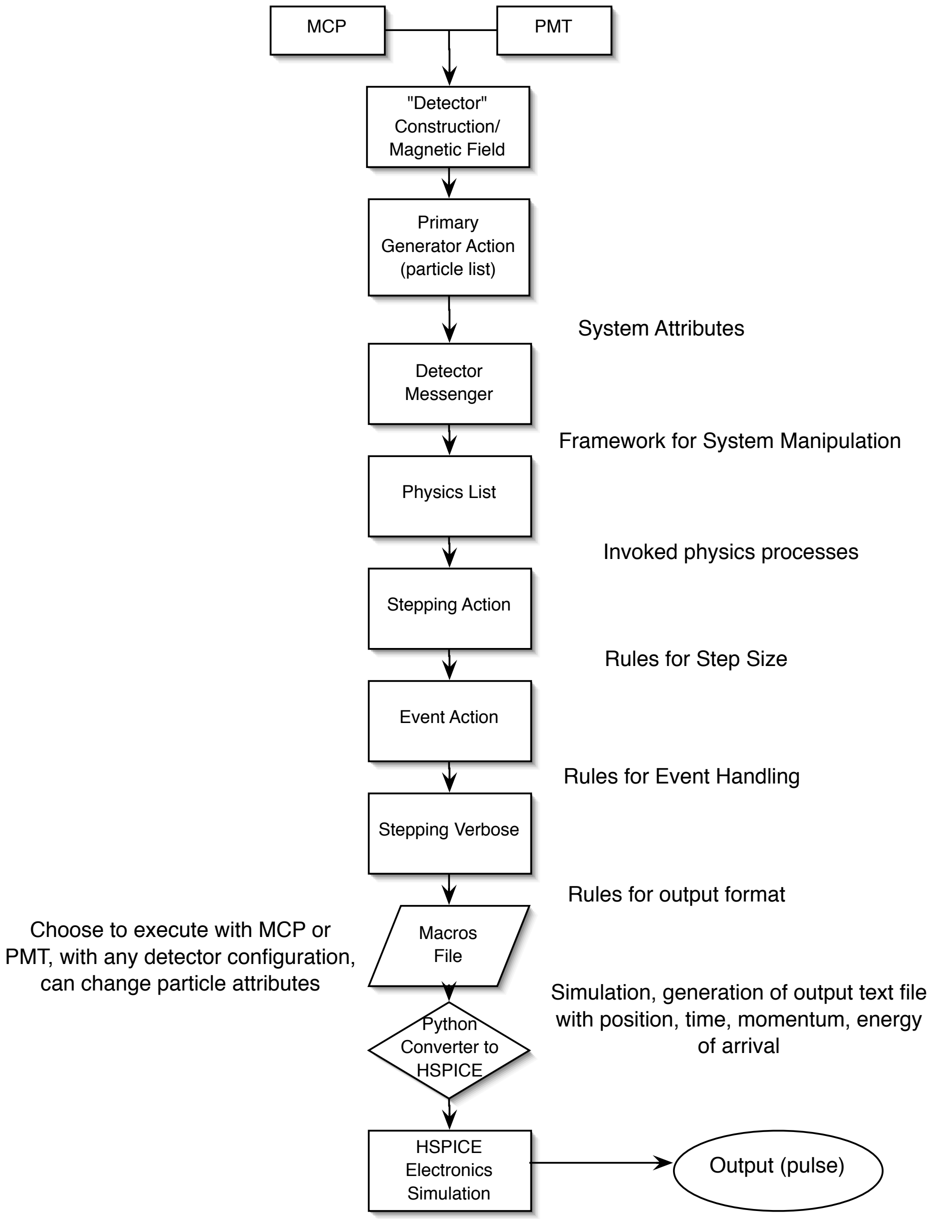
MCP
PMT
"Detector" Construction/ Magnetic Field
Primary Generator Action (particle list)
System Attributes
Detector Messenger
Framework for System Manipulation
Physics List
Invoked physics processes
Stepping Action
Rules for Step Size
Event Action
Rules for Event Handling
Stepping Verbose
Rules for output format
Choose to execute with MCP or PMT, with any detector configuration, can change particle attributes
Macros File
Simulation, generation of output text file with position, time, momentum, energy of arrival
Python Converter to HSPICE
HSPICE Electronics Simulation
Output (pulse)

Figure 3: Block diagram for a "pure" GEANT4 simulation (the electronics are still HSPICE). Depending on the needs of the user, the code can be recompiled or reexecuted using a variety of PMT options as well as geometries.

The second possibility is that we use a modified version of GATE for the entire simulation. In GATE, we would simply input the geometry of the detector, the materials involved, the particle properties, and the verbose level (the detail provided in the output file). A diagram showing the outline of the simulation is given in Figure 4. A more detailed explanation of

GATE is given later on. The PMT or MCP code could be easily swapped with the default "digitization" module in GATE, which would increase the applications for which the code could be used. A simple Python or ROOT script could be used to convert the GATE output into HSPICE input, as was done before.

In a pure GATE scheme, we would have to write a module to simulate a magnetic field, and recompile GATE, since this is not currently supported. We would also have to invoke the Cherenkov radiation process from the GEANT4 library. As the former has already been done for the original GEANT4 simulation, we do not foresee any difficulties for doing the same for GATE. However, providing a framework for a changeable field configuration could pose a challenge.

3 PET Simulations

GATE serves as an assembled toolkit based on Geant4 dedicated for PET and SPECT imaging. In GATE, the user can define the system geometry and set the particle source parameters by editing the macros file. The physics processes are taken from Geant4 and CLHEP libraries. The direct output GATE generates is a file of 'hits', which includes the complete information of the interaction events (time, position, energy deposition, momentum change, type of interaction) between the particle production and the detector. Then a virtual digitizer mimics the electronic read-out by setting detector parameters such as energy resolution, spatial resolution, etc. The latest version of GATE can also simulate the optical photon generation and transport processes inside the scintillators, but the final pulse generation (singles and coincidences in PET) still relies on parameters set by the user (Figure 4).

The most obvious advantage of GATE is its flexibility of changing system geometry and material selection by modifying only the macros file. However, in order to rigorously simulate the electron generation and transport inside the MCP/PMT, we need to replace the digitizer and sorter in Figure 4 by a specific module for each type of PMT.

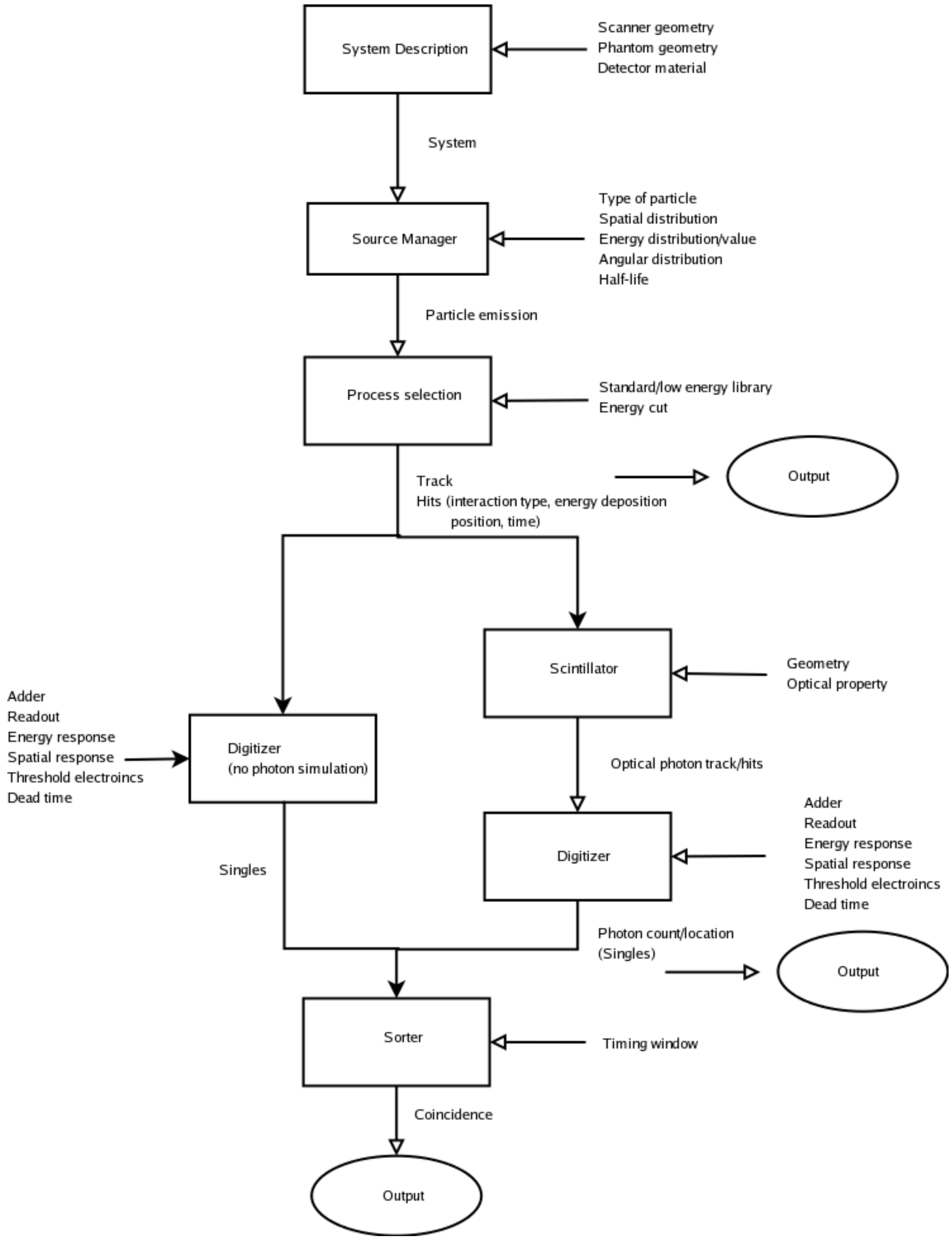

Figure 4: Block diagram for GATE simulation. The latest version of GATE (v. 3) can simulate scintillators, but the PMT - the Digitizer block on the right branch - is still not a rigorous Monte-Carlo simulation.

4 Conclusions

GATE is already a fully functional GEANT4-based program that allows for simulation of tomography. From initial investigations, it appears that we can easily replace the digitizer and sorter modules of the program with specific PMT modules that will more rigorously simulate photoelectron transport. We would ideally like to have a coherent, end-to-end simulation that uses as few different programs as possible so as to simplify modification. Thus, with a few modifications, it appears as if GATE could be used as this flexible framework to simulate both radiological and physical applications.

References

[1] GEANT4 Collaboration, CERN, *GEANT4 Home Page*, http://geant4.cern.ch/

[2] OpenGate Collaboration, *GATE - Geant4 Application for Tomographic Emission*, http://opengatecollaboration.healthgrid.org

[3] S. Bhat, T. Credo, H. Frisch, *Simulation of a Picosecond TOF Device*, EFI Preprint 06-12, 26 Jul 06.

[4] R. Brun and F. Rademakers, *The ROOT System Home Page*, http://root.cern.ch/Package_Overviews/DetSim.html