Silicon Vertex Tracker's Simulation Status

Mahantesh Halappanavar, Tanest Chinwanawich, Jia Fan, Benjamin Smith, and Amrit Yegneswaran
Physics Division, Thomas Jefferson National Accelerator Facility, Newport News, Va 23606
April 16, 2004

The Silicon Vertex Tracker's (SVT) simulation (SVTSIM) is based on GEANT4 (G4), version 4.6.0. This note presents G4 features and simulation status.

Figure 1 shows SVTSIM's prototype rendering of the conceptual layout of the SVT, proposed detector for CLAS^{++} [1]. Each of the three regions: inner (R1), middle (R2), and outer (R3) around the target indicated in yellow, blue and purple, respectively, has two layers (u, v) that are at an angle of 10^{o} with respect to each other. The pitch and thickness of the silicon strips are 300 μm. In all, the SVT is anticipated to have ~60,000 read-out channels.

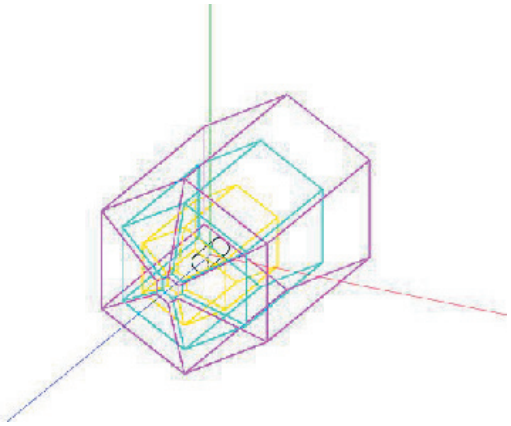

FIG. 1. G4 Geometry of the CLAS detector.

SVTSIM investigates vertex and angular resolution, determines particle-distributions in r, $\emptyset$, and φ, estimates rates, studies acceptance, and evaluates migration-feasibility – from G3 based CLAS simulation, GSIM [2], to G4 [3].

G4, the object-oriented design of which is implemented with C^{++}, offers a variety of physics processes [4] and provides tools – geometry, tracking, detector response, run, event & track management, visualization, and user interface [5,6], for detector simulations.

G4 geometry consists of a hierarchy of volumes, described by shape and physical characteristics contained within the world volume. G4 supports alternate models of solid representation: Constructive solid geometry (CSG) and boundary-represented solids (BREP) [4].

BREP in G4 version 4.5.0 permits imports of CAD drawings; but it was found that automatic description of items required by G4 – logical volumes, material descriptions, and geometric features, by CAD tools is neither complete nor exact. SVTSIM, because of the new options offered, is based on latest version 4.6.0 that unfortunately does not support import of CAD drawings, even for BREP [4]. Since CSG is easier to use and performs better than BREP and SVT geometry is straightforward, SVTSIM's geometries are built with CSG solids [4].

Default geometries are defined in the DetectorConstruction class [7]. Basic shapes are boxes constructed as parameterized volumes defined in ChamberParametrisation. The target is defined as a cylindrical section using the constructor G4Tubs. R1, R2, and R3 are specified for the tracker, which has in all eighteen trapezoidal sections (six per region) that are defined using the G4Trap constructor and eighteen faces (six per region) classified as G4Box. G4SDManager defines sensitive detectors; logical volumes form the sensitive detectors. G4VisAttributes sets visualization attributes such as the different colors in the rendering, Fig. 1.

Commands defined in DetectorMessenger enable target material, chamber material, and magnetic field strength to be specified interactively. G4Element and G4Material, describe properties of elements – atomic number, number of nucleons and atomic mass, and materials – density, state, temperature, and pressure, respectively.

Particles and physics-processes must be defined in the mandatory class named G4VuserPhysicsList.

SVTSIM has: the standard set of particles – baryons, mesons, leptons and bosons. Individual particle properties – name, mass, charge, spin, is defined in G4ParticleDefinition.

Physics in G4 is implemented as processes assigned to particles and comes from the G4VProcess base class. Lists of processes in which particles can interact are defined by G4ProcessManager. Decay processes, extended from G4Decay, are assigned through the ContructGeneral() method of PhysicsList. ConstructEM() method assigns electromagnetic processes to particles. Electromagnetic processes have been defined for SVTSIM

SetCuts() method of G4VuserPhysicsList defines the range threshold values. SVTSIM's default cuts are specified using SetCutsDefault() method.

A uniform magnetic field has been generated by inheriting G4UniformMagField. The value of the field is set by using the SetFieldValue() method. Strength of the field is specified interactively or via input file by the command SetField. Currently, simulations with different field strengths are being studied.

G4TrackingManager handles information interchange between the event, track, and tracking categories. G4Step stores transient information in a step and G4Track provides information on the final status of the particle after completion of the step.

G4 provides classical fourth-order Runge-Kutta stepper by default. Information about G4Step in SVTSIM is generated by implementing the methods Draw() and Print() provided by

the abstract base class G4VHit. Geometrical information is displayed by the Draw() method through a G4VvisManager pointer. Energy deposition and position are displayed by the Print() method using G4coutiostream.

G4SteppingManager handles all messages passing between objects in the different categories relevant to transport a particle. The Stepping() method guides the stepping of particle through the algorithm defined to handle a step. Default handling is used in SVTSIM and an option to provide specific action is offered by extending G4UserSteppingAction.

SVTSIM extends G4SteppingManager to extract verbose information during tracking and outputs information – x, y and z position, kinetic energy, tracking length, volume, and process.

G4VUserPrimaryGeneratorAction has a pure virtual method named generatePrimaries() invoked at the start of each event [6]. Mixing several generators creates complicated primary events.

Run, a sequence of events, is the largest part of the simulation. The beamOn() method via pointer in G4RunManager starts the run. SVTSIM executes either in batch mode with no interactivity and graphics or in an interactive mode with graphics.

G4 provides an abstract framework for persistency of hits, digits, and events and supports different methods for visualization, not only for displaying the simulation and output, but also for debugging the overlapping physical volumes. OpenGL, HepRep/-WIRED, and DAWN are some of the graphic drivers available [6].

OpenGL, generated directly by G4, has photo-realistic images with limited interactive features. WIRED offers interactive features – special projections, display of attributes, and control from hierarchy. Though WIRED does not render photo-realistic images like OpenGL, data is exportable to vector graphic formats like PDF and postscript. DAWN rendered files have the highest technical quality vector postscript files [4]. SVTSIM has both OpenGL and DAWN.

Simulations are performed on a 2.8 GHz P4 1GB RAM Linux-(9.0)-box. SVTSIM has been ported to MAC G5 and a study comparing execution advantages in PC to that in MAC is underway. Figure 2 displays a trial run performed on the box – electron beam (E_e = 500 MeV) on proton target with |B| = 2 T.

Since CLAS^{++} simulation will be demanding in terms of geometry, physics, and number of events that need to be simulated, parallel processing and the grid-computing frameworks with Globus toolkit are being investigated [8, 9].

Simulation result validation is a difficult task. Standard test procedures to isolate potential areas of errors and methods to validate simulation correctness are under development.

Data analysis can be performed with histograms, n-tuples and similar data analysis objects that can be programmed through a common set of standard interfaces provided by abstract interfaces for data analysis (AIDA) [10]. Many C^{++} and Java tools implement AIDA like interfaces – JAIDA, Open Scientist, and Anaphe. SVTSIM's data will be analyzed with A1 and RECSIS.

SVTSIM uses Doxygen [11], a free software that automatically generates HTML, RTF, PDF, and PS documentation from C^{++} programs.

In conclusion, G4 offers several new and powerful features. SVTSIM, a first effort to prototype a CLAS^{++} detector, uses several of these features. SVTSIM has started initial trial runs for different SVT geometries and physics processes.

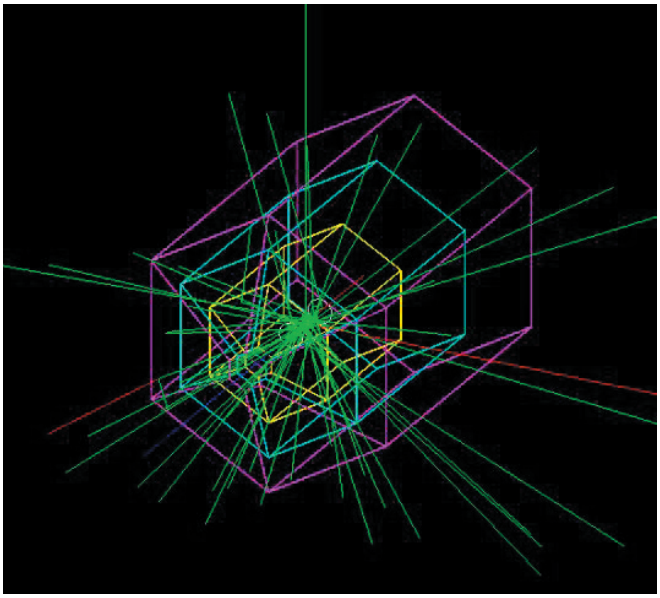

FIG. 2. Geant4 Run of the simulation.

[1] The Hall B 12 GeV Upgrade Pre-Conceptual Design Report, Version 1, October 12, 2002.

[2] CLAS GEANT Simulation (http://www.physics.unh.edu/~maurik/gsim_info.shtml)

[3] GSIM uses FORTRAN. Integrating relevant FORTRAN procedures, like CLAS specific physics processes, into G4 simulation has advantages – elimination of code translation errors and independent development and maintenance of codes.

[4] GEANT 4 User Documents: (http://wwwasd.web.cern.ch/wwwasd/geant4/G4UsersDocuments/Overview/html/index.html)

[5] GEANT4 – a simulation toolkit, GEANT4 Collaboration, SLAC-PUB-9350, August 2002.

[6] GEANT 4 Home page: (http://wwwasd.web.cern.ch/wwwasd/geant4/geant4.html)

[7] Classes are in blue type, methods in red.

[8] The Globus Alliance Home page. (http://globus.org/)

[9] Using TOP-C and AMPIC to Port Large Parallel Applications to the Computational Grid, G. Cooperman, H. Casanova, J. Hayes and T. Witzel. A copy of the paper can be obtained at: (http://citeseer.nj.nec.com/581735.html)

[10] AIDA Home Page (http://aida.freehep.org/)

[11] Doxygen Home page: (http://www.stack.nl/~dimitri/doxygen/index.html